VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ALGORITMY PRO DOPŘEDNÉ A ZPĚTNÉ PLÁNOVÁNÍ
ALGORITHMS FOR FORWARD AND BACKWARD PLANNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Filip Sluka
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Simeon Simeonov, CSc.
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Filip Sluka**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Simeon Simeonov, CSc.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Algoritmy pro dopřední a zpětné plánování

Stručná charakteristika problematiky úkolu:

Analýza algoritmů pro dopřední a zpětné plánování. Návrh programového vybavení algoritmů dopředního a zpětného plánování.Návrh symetrických algoritmů. Naprogramování algoritmů dopředního a zpětného plánování.

Cíle diplomové práce:

Analýza algoritmů pro dopředné a zpětné plánování.
Návrh programového vybavení algoritmů dopředního a zpětného plánování.
Návrh symetrických algoritmů.
Naprogramování algoritmů dopředního a zpětného plánování.

Seznam doporučené literatury:

PROUD, John F. Master scheduling: a practical guide to competitive manufacturing. 3rd ed. Hoboken, N.J.: John Wiley, c2007. ISBN 978-0-471-75727-6.

BRANDIMARTE, Paolo. a A. VILLA. Modeling manufacturing systems: from aggregate planning to real-time control. New York: Springer, c1999. ISBN 35-406-5500-X.

HARRISON, David K. a David J. PETTY. Systems for planning and control in manufacturing: systems and management for competitive manufacture. Boston: Newnes, 2002. ISBN 07-506-4977-1.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Práce se zabývá plánováním výroby. Obsahuje teoretický popis metod používaných pro plánování a optimalizaci výroby. Popisuje problematiku úzkých míst ve výrobě. Nabízí přehled způsobů jejich identifikace a analýzu jejich vlivů na efektivitu výrobního procesu. Navrhuje odstranění úzkých míst s využitím různých druhů algoritmů. Teoretické poznatky z oblasti optimalizace a teorie grafů aplikuje při vytvoření programu určeného k minimalizaci zpoždění zakázek a doby přeseřízení strojů. Program implementuje genetický algoritmus.

ABSTRACT

The thesis deals with production planning. It contains theoretical description of methods used for production planning and optimizing. Thesis describes bottleneck problems in production. It offers overview of ways to identify and analyze bottleneck influence to manufacturing process efficiency. Thesis proposes ways to eliminate bottlenecks using various algorithm types. It applies theoretical knowledges from optimization and graph theory to program creation that is focused on order delay and readjustment time minimizing. The program implements genetic algorithm.

KLÍČOVÁ SLOVA

Plánování výroby, úzká místo, optimalizace, genetický algoritmus, Ganttův diagram, ROP, MRP, MRP II, MPS, TSP, přeseřízení, zpoždění, vývoj softwaru

KEYWORDS

Manufacturing planning, bottleneck, optimization, genetic, algorithm, Gantt diagram, ROP, MRP, MRP II, MPS, TSP, readjustment, delay, software development

BIBLIOGRAFICKÁ CITACE

SLUKA, Filip. *Algoritmy pro dopřední a zpětné plánování* [online]. Brno, 2019 [cit. 2019-05-22]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/113167. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce doc. Ing. Simeon Simeonov, CSc.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Simeona Simeonova, CSc. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 22. 5. 2019 ……

Bc. Filip Sluka

Obsah

1 Úvod

Strojní inženýrství se zabývá širokým okruhem teoretických i praktických otázek a úkolů, které přináší výrobní praxe, a poskytuje průmyslové výrobě všestrannou podporu. Kromě řešení technologických problémů a uplatnění nových technologií se výrobní závody pochopitelně zabývají také ekonomickou stránkou výroby. Z toho důvodu mají stále významnější místo mezi inženýrskými úkoly požadavky na lepší ekonomickou efektivnost výroby. Té lze dosáhnout více způsoby – snižováním nákladů, úsporou zdrojů a energií, zrychlením výrobního procesu a řadou dalších, které se obvykle využívají ve vzájemné kombinaci.

Jedna ze žádaných a oblíbených metod je úspora nákladů v důsledku lepší organizace výroby. Její velkou výhodou je, že je zcela nebo z významné části bezinvestiční, tj. není zapotřebí nakupovat nové výkonnější stroje, dopravníky, vytvářet další sklady atd. Účinek zlepšené organizace se může projevit ihned, protože se zakládá na vyloučení neúčelných prostojů, zbytečných manipulací s materiálem a s výrobky, zbytečně častých změn v nastavení strojů, linek apod.

Díky současnému stavu teoretických poznatků a možnostem informačních technologií se nabízí možnost modelovat vybrané části výrobního procesu matematicky a řešit jejich fungování se zohledněním vnitřních vztahů a podmínek jako teoretický problém. Kromě složitých úkolů ve výrobních procesech, s nimiž si dokáže poradit umělá inteligence, jsou i jednodušší, řešitelné běžnou výpočetní technikou. Do této kategorie patří některá zadání z organizační oblasti. Typickým představitelem úkolů směřujících ke zlepšení organizace výroby je hledání *účelnějších způsobů plánování* této výroby. Do posledně zmíněné kategorie patří téma, s nímž pracuje tato diplomová práce.

Tuto práci lze rozčlenit na několik celků. V prvním se věnuji samotným základům rozvrhování výroby. Je zde nastíněna problematika plánování ve výrobních procesech a jaké jsou přístupy k jejímu řešení. V této části je také možné nalézt metody a postupy, které byly vytvořeny pro zlepšení efektivity výrobního procesu. Poskytnut je historický průřez metodami od poměrně primitivních až po moderní komplexní systémy, které pronikají do všech oblastí souvisejících s řízením podniku.

V další části lze nalézt rozbor toho, jak konkrétně vypadá práce člověka, který se snaží získat dostatek informací o výrobním procesu, aby mohl následně tyto znalosti použít pro upravení a optimalizaci systému. Jsou zde rozebrány pojmy, které se ve výrobním procesu objevují, jakým způsobem je programové vybavení reprezentuje a jak s nimi může uživatel pracovat. Vše je demonstrováno na databázi, která obsahuje data standardní pro výrobní procesy.

Jednou z nejpodstatnějších částí této práce je ovšem problematika úzkých míst. V této práci je nejdříve rozebráno, co to vlastně úzké místo je, jakým způsobem ho lze identifikovat

a jaký má vliv na výrobní proces. Cílem je dosáhnout zlepšení v této oblasti a proto jsou nejdříve položeny teoretické základy optimalizačních problémů a postupů (algoritmů), které se k jejich řešení využívají. Na uvedených základech je poté postupně vystavěno řešení tohoto problému.

Jedním z kýžených výstupů diplomové práce je vytvoření programového vybavení, které implementuje algoritmus schopný provést optimalizaci úzkého místa. Pro vytvoření tohoto programu jsem se rozhodl použít programovací jazyk Visual Basic .NET. Program byl tvořen na základě poznatků získaných z teorie optimalizace. Pro řešení jsem postupně použil několik algoritmů a stěžejní částí tvorby programu je implementace genetického algoritmu. Postupně jsem prováděl ověřování a testování programů a na závěr je z těchto poznatků vytvořeno shrnutí a celkové zhodnocení funkčnosti a použitelnosti algoritmů a celého programu.

2 Plánování výroby

Výroba je proces, při kterém dochází k přeměně surového materiálu na produkt, který se poté následně prodává. Zlomovým okamžikem v historii byla průmyslová revoluce. Kolem roku 1850 už se většina výroby přemístila do továren. Výrobní proces probíhající v továrnách lze rozdělit na několik na sebe navazujících procesů. Samotný výrobní proces je jeden z článků většího řetězce. Celý systém, který přeměňuje surové materiály na finální výrobky, se označuje jako dodavatelský řetězec (Supply Chain). Tento řetězec má několik oblastí, které plní určitou funkci. Mezi tyto oblasti patří například prodej, který je určován poptávkou. Dalšími oblastmi jsou účetnictví, které se věnuje financím podniku, nákup, který vybírá dodavatele a zadává příkazy k nákupu. Dále jsou součástí dodavatelského řetězce také vývojová oblast, výrobní inženýrství (Production engineering) a výrobní plánování. Vývoj určuje jaký produkt se bude vyrábět, výrobní inženýrství má na starost to, jak se bude produkt vyrábět. A konečně výrobní plánování určuje, kdy se bude vyrábět a v jakém množství.

Základním cílem drtivé většiny firem a podniků je dosažení zisku. Z čistě elementárního pohledu tak lze zisku dosáhnout dvěma způsoby: zvyšováním výnosů nebo snižování nákladů. Zvyšování výnosů je spíše problematika pro ekonomy. Každopádně této oblasti v této práci nebude věnována pozornost, i když ekonomicky „zdravý“ podnik musí být synergií optimalizace výnosů a nákladů. Tato práce se bude především zabývat problematikou snižování nákladů.

Existuje mnoho způsobů, kterými lze dosáhnout snižování nákladů. Může se jednat například o úpravu konstrukce vyráběných součástek, které jsou tak schopné plnit požadované účely při nižších výrobních nákladech. Může se jednat také o nové technologie, které urychlí výrobní proces. Jednou z otázek, které můžou vyvstat je, zda je výhodnější použít postup, který je rychlejší, ale více nákladný nebo postup delší a méně nákladný. Tuto otázku a jim podobné je schopen zodpovědět proces, který se nazývá plánování.

2.1 Plánování

Plánování je proces umožňující nejefektivnější využití zdrojů tak, aby bylo dosaženo splnění úkolu ve stanoveném čase [1]. Pomocí plánování se rozvrhuje využití zařízení a vybavení a rozvržení lidských činností. Existují dva základní přístupy, které umožňují plánovat a rozvrhovat výrobu. Nazývají se plánování dopředné a zpětné.

2.1.1 Dopředné plánování

Pomocí dopředného plánování je možné určit nejkratší čas, během kterého je možné dokončit zakázku. Při dopředném plánování se využívá metoda kritické cesty. Začíná se procesy, které nemají žádné předchůdce – tzn. procesy, které ke svému dokončení nepotřebují, aby byly dokončeny procesy jiné. Těmto procesům se přidělí zadaný start, kdy se má na zakázce začít pracovat. Poté se u jednotlivých procesů přičte časový úsek, který je k provedení těchto akcí potřeba. Tím se dosáhne koncového času procesu. Tento čas se pak použije jako startovní čas pro proces nebo procesy následující. Toto se opakuje do doby, než je dopočítán koncový stav

všech procesů, které už nemají další následovníky. Největší ze všech koncových časů je doba za kterou lze dokončit zakázku v co nejkratším čase [2].

Podle této metody procesy začínají okamžitě poté, co jsou předešlé procesy dokončeny a využívá tedy zdroje hned jak se uvolní [1]. To může způsobit předčasné dokončení, které má negativní dopady, protože tím mohou vzniknout další náklady, např. náklady na skladování, než může být zásilka expedována. Další problém představuje dokončení dílčích akcí, které jsou potřeba pro finální montáž, v rozdílných časech.

2.1.2 Zpětné plánování

Jako vstup pro zpětné plánování se použijí dva údaje. Předpokládaný začátek práce na zakázce a doba, kdy má být zakázka dokončena a připravena. Termín dokončení se následně použije k vypočítaní nejzazších termínů začátků práce na jednotlivých procesech. Opět se využívá metoda kritické cesty. To tedy znamená, že pro koncové procesy se vezme termín dokončení a doba potřebná k provedení daného procesu. Z těchto hodnot se vypočítá čas, kdy je potřeba proces zahájit. Tento čas se označuje jako pozdní start. Ten se použije jako vstup pro procesy, které danému procesu předchází. Toto se stále opakuje, než jsou spočítány pozdní starty všech procesů. Poté se spočítá plánovaný začátek práce na projektu. Ten se získá tak, že se vezme minimum z množiny, která obsahuje předpokládaný začátek a všechny pozdní starty. Z plánovaného začátku se pomocí dopředného plánování vypočítá brzký start a konec všech procesů. Brzké a pozdní začátky a konce tak určují interval, ve kterém je možné daný proces provést. Projektový manažer může podle potřeby přiřadit procesu čas z daného intervalu [3].

U zpětného plánování může nastat situace, kdy vypočítané datum plánovaného startu už uplynulo. V tom případě je nutné upravit délku procesů tak, aby bylo možné zakázku ve stanovené době stihnout nebo je nutné termín dokončení odsunout [3].

Popis k obrázku č. 1

ES – early start, brzký start, v němž lze nejdříve zahájit úlohu označenou ID

EF – early finish, brzký konec, v němž lze nejdříve dokončit úlohu ID

LS – late start, pozdní start, okamžik nejpozdějšího zahájení úlohy ID

LF – late finish, pozdní konec, okamžik nejpozdějšího ukončení úlohy ID

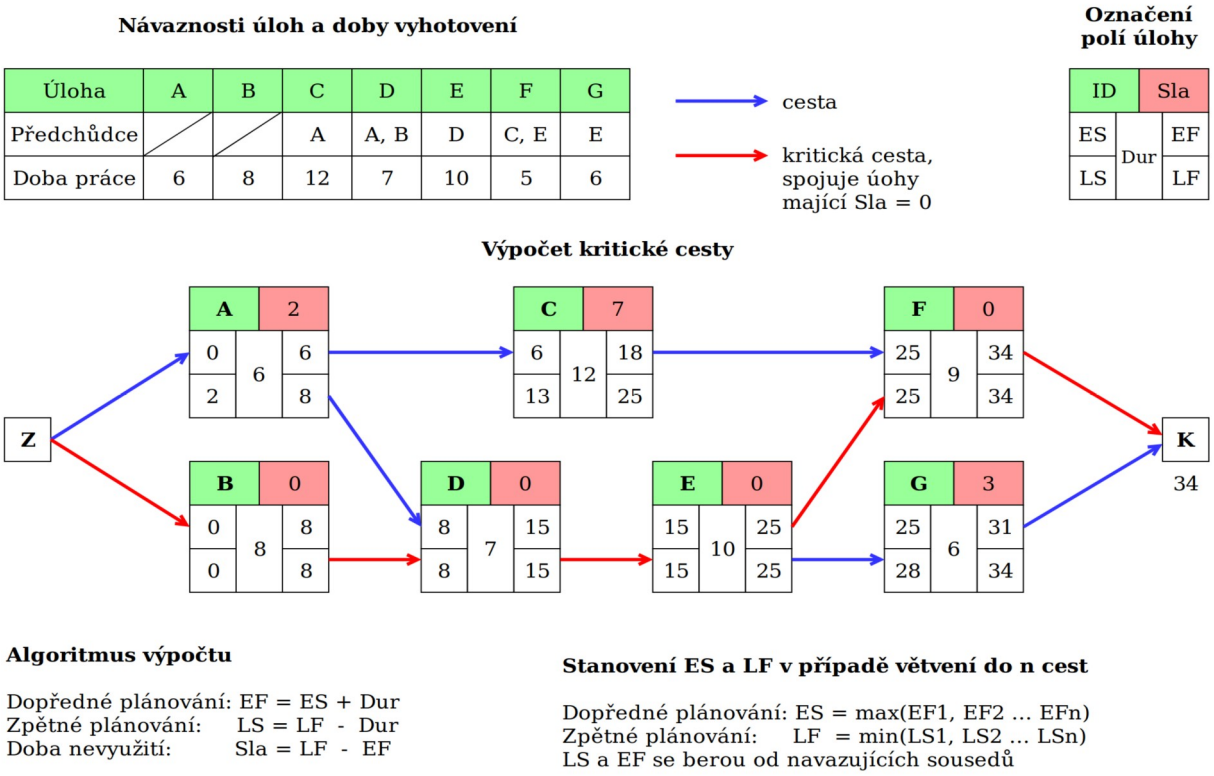
Dur – duration, doba trvání úlohy

ID – identifikátor úlohy, její označení

Sla – slack, doba nevytížení

Z – začátek

K – konec



Obrázek 2.1: Dopředné a zpětné plánování na příkladu

3 Metody pokročilého plánování výroby

Továrny a výrobní podniky existovaly po stovky let, ale teprve až ve 20. století se podařilo vyvinout systémy, které byly položeny na vědeckých základech a výpočtech a které pomáhaly určovat dobu, kdy se má objednat další zboží, tak aby nedošlo k přerušení výroby. První systémy které to dokázaly se souhrnně nazývají systémy s otevřenou smyčkou (Open-Loop Control Systems). Je potřeba nejdříve definovat 2 základní pojmy: Materiál a kusovník. Materiál (v angl. literatuře označováno jako Item) je jak finální produkt, tak všechny součásti ze kterých se skládá, ať už se jedná o polotovary, které firma nakupuje, nebo komponenty, která firma vyrábí. Kusovník (BOM – z angl. Bill of Materials) udává všechny komponenty ze kterých se daná součástka sestavuje nebo vyrábí. Položkou v kusovníku může být i další součástka, která má vlastní výkres a kusovník. Pomocí kusovníku tak lze získat pro každý konečný produkt všechny základní materiály ze kterých se skládá a které je potřeba pro výrobu naskladnit.

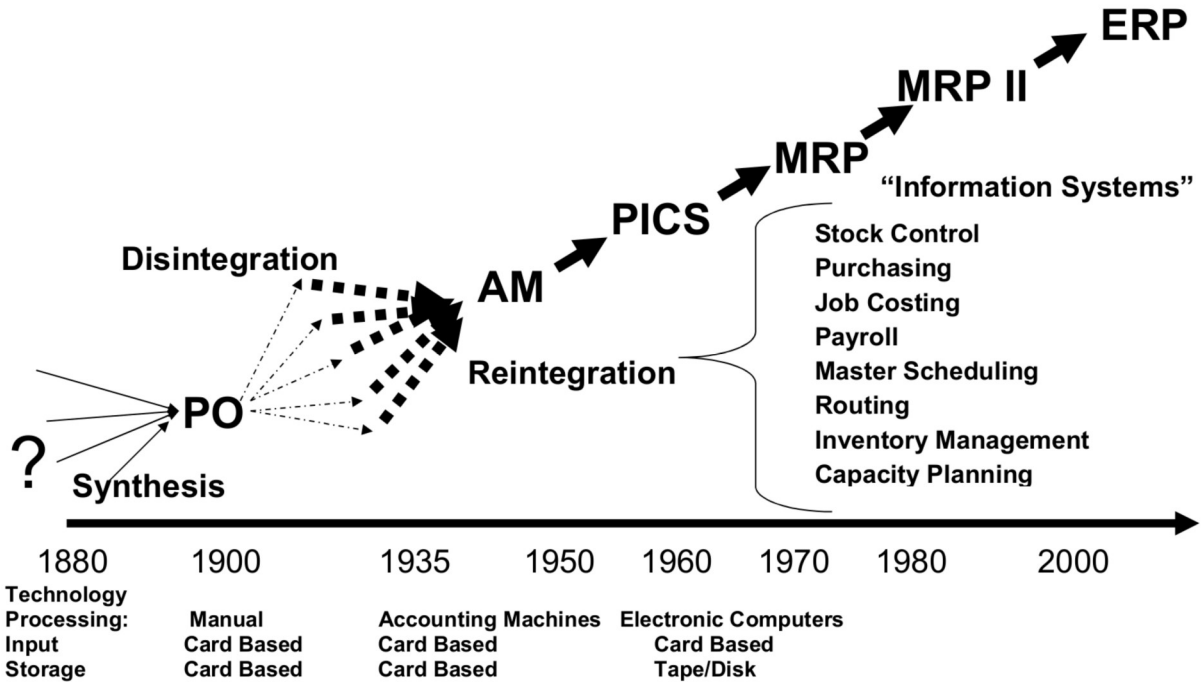
Systémy plánování výroby pomáhají určovat, kdy je potřeba provést nějakou akci (např. objednat zboží). Systémy toho docílí pomocí tzv. příkazů (v angl. literatuře Orders). Existují tři typy těchto příkazů. Prvním je příkaz k prodeji. Tento příkaz obsahuje údaje o zákazníkovi, údaje o zboží, které má být dodána (množství, typ,...) a datum, kdy má zboží být dodáno. Druhý typ příkazu je příkaz k nákupu. datum dodání. Posledním typem příkazu je pracovní příkaz. Pracovní příkaz určuje, že se má provést nějaká výrobní činnost, neboli přeměna surového materiálu na jiný materiál. Tento příkaz se skládá z informací o produktu a jeho množství a o údajích o době, kdy se má práce začít a skončit.

3.1 ROP

Už před rozšířením výpočetní techniky se používalo metod, které se snažily do jisté míry optimalizovat objednávání a nákup, aby nedocházelo ke zbytečnému naskladnění. Potřebné výpočty se prováděly ručně. Jednou z nejvýznamnějších metod je metoda ROP (Reorder Point). Teorie ROP se objevila v 50. letech 20. století a jejím účelem bylo zjistit, kdy se má začít práce na zakázce a kdy je vhodné objednat potřebné materiály. ROP má tři základní omezení:

- Objednávka dorazí pozdě – Toto je slabina všech plánovacích přístupů
- Nepravidelnost ve spotřebě materiálu
- Průměrná poptávka nekoresponduje s budoucí spotřebou, nelze tak použít průměrnou poptávku v uplynulém období jako přesný odhad budoucí poptávky

Kvůli těmto úskalím se využívá pojistné zásoby. Samotný systém ROP nenabízí dostatek možností a proto byl nahrazen komplexnějším systémem MRP. Časový vývoj je na obrázku 3.1.



Obrázek 3.1: Evoluce produkčního plánování podle [7]

3.2 MRP

V polovině 20. století dochází k zavádění pokročilejší výpočetní techniky do řízení výroby. K zadávání dat se používaly děrné štítky. Velký rozvoj přinesla 70. léta a 80. léta, kdy v mnoha velkých podnicích došlo k zavedení systému MRP [6]. Za jednoho z hlavních zakladatelů MRP se považuje Joseph Orlicky [7]. Zavedení MRP znamenalo velký pokrok a zlepšení výrobního procesu. MRP byl nakonec pohlcen pokročilejšími systémy ERP [5] [6].

V systémech MRP se pracuje s několika pojmy: Kusovník (BOM – angl. Bill of Material) a hlavní plán výroby (MPS – z angl. Master Production Schedule). Jak u finálního produktu, tak u jednotlivých součástek je nezbytnou součástí výkresu kusovník. Hlavní plán výroby udává co, kdy a v jakém množství se bude vyrábět. Požadavky hlavního plánu výroby jsou zadávány ručně a určuje je např. manažer výroby. Požadavky MPS se označují jako nezávislé. Naopak požadavky na materiály přímo souvisí s požadavkem po finálních produktech, který je určen MPS. Požadavky na materiály se proto označují jako závislé. Závislé požadavky jsou „schované“ uvnitř MRP. Jsou vypočítávány samotným systémem a není potřeba do nich jakýmkoliv způsobem zasahovat. Cílem je, aby se materiály se závislými požadavky naskladnily až těsně před tím, než jsou potřeba [4] [7].

Hlavním rozdílem mezi ROP a MRP je ten, že ROP se zajímá o každý materiál zvlášť. MRP bere v úvahu, že při výrobě jsou na sobě jednotlivé materiály a součástky závislé a že je potřeba chápat závislosti mezi nimi. Přístup ke každé součástce jako k samostatné nezávislé entitě není správný. Proto MRP zavádí pojmy závislé a nezávislé požadavky a přístup k nim se liší.

Kromě kusovníků a hlavního plánu výroby jsou ještě vstupem do MRP systému údaje o stavu zásob (v [7] označované jako IS z angl. Inventory Status) a plánovací parametry. Mezi plánovací parametry patří například procento odpadu, pojistná zásoba a průběžné doby [8]. Existují dva způsoby jakými se systémy aktualizují. V prvním z nich se shromažďují informace o změnách v rozvrhu nebo příkazech. V určitých časových okamžicích jsou informace z předešlého intervalu najednou systému předány. Tento přístup se nazývá dávkový. V druhém přístupu jsou informace o změnách systému předávány průběžně, tak jak se vyskytnou. Je to tzv. Průběžný přístup. Dávkový přístup je vhodný pro stabilní systémy, kde nevadí delší odezva na změny. Naopak průběžný přístup se využije pro méně stabilní výrobní systémy, kde je nutné rychle reagovat na změny [4] [7] [9].

Výstupy z MRP lze rozdělit na primární a sekundární. Primární výstupy dávají data k samotnému plánování výroby a řízení výrobních zásob. Mezi tyto výstupy patří pracovní příkazy, přejímky příkazů a změny plánovaných příkazů. Sekundární výstupy poskytují další data, která lze využít například pro kontrolu kvality provedených prací. Do těchto výstupů spadají výrobní přehledy a plánovací přehledy [9].

Dosud se hovořilo o MRP a o systémech s otevřenou smyčkou, existuje však propracovanější varianta MRP, která pracuje se systémem s uzavřenou smyčkou (angl. Closed-Loop Control System). Tato varianta MRP totiž sleduje aktuální činnost systému. Práce obvodu MRP se skládá z těchto kroků. Na začátku se provede plánování. Systém určí, kdy se mají provést jaké příkazy. Poté následuje provedení naplánovaných operací. Třetím krokem je zpětná vazby, která informuje systém o změnách. Podle dat ze zpětné vazby v posledním kroku systém doporučuje opatření týkající se těchto změn. Tato verze MRP tak oproti klasickému MRP bere v úvahu, že výrobní proces není statický, ale že podmínky se stále mění. Například běžně dochází ke zpoždění dodávek objednaného materiálu. Také může dojít ke změně parametrů zakázky od zákazníka nebo se může změnit technologický proces výroby. Oproti klasickému MRP používá systém s uzavřenou smyčkou několik vylepšení. Nabízí možnost kapacitního plánování, které pomáhá zjistit, zda jsou výstupy plánovacího procesu reálné. Druhým prostředkem je už zmíněná zpětná vazba určená ke sledování toho, zda jsou jednotlivé procesy plněny podle plánu. Posledním zlepšením je možnost generování zpráv o plánovacích akcích. Smyslem kapacitního plánování je zajistit, že příkazy generované systémem jsou reálné. Kapacitní propočty se skládají ze tří částí. První je SOP (z angl. Sales and Operations Planning). Jedná se o manuální proces, kdy vedení kontroluje požadavky a přiděluje potřebnou kapacitu. Další částí je MPS (z angl. Master Production Scheduling). MPS mechanismus, který převádí prodeje na výrobní plány. Poslední částí je CRP (z angl. Capacity Requirements Planning). CRP kontroluje, zda výstup z plánovacího procesu má k dispozici dostatečné kapacity a je tak reálný.

3.3 MRP II

Material Resource Planning (označované také jako MRP II) je do značné míry podobné systému MRP s uzavřenou smyčkou. Rozdíl je v tom, že MRP II používá kromě podrobnějšího rozvrhování výroby a kapacitních propočtů také to, že obsahuje schopnost simulace a vazbu na řízení prodeje a finanční plánování obecně [9]. Výstupem z MRP II jsou

dvě zprávy: Výrobní příkazy a příkazy k nákupu. Systém MRP II je značně složitý a s jeho implementací vzniká mnoho rizik. S řešením těchto nebezpečí pomáhají další výstupy systému MRP II. Jsou to zprávy o jednotlivých položkách nebo objednávkách výrobního procesu. Tyto zprávy tak upozorňují třeba na to, že komponenta je přidělena více projektům nebo že poptávka po nějaké komponentě převyšuje zásoby. Jednou z hlavních slabin MRP II systému je požadavek na vysokou přesnost vstupních dat. I menší nepřesnosti na vstupu mohou způsobit, že na výstupu jsou zcela chybná data [10].

MRP II se většinou dodává v podobě skupiny modulů. Množství a typy modulů se odvíjí od specifických požadavků jednotlivých zákazníků. Jednotlivé moduly souvisí s jednou z několika hlavních oblastí, které obsahuje MRP II. Moduly z oblasti distribuce se věnují vztahům mezi podnikem a dodavateli a zákazníky. Tyto moduly jsou například schopné předpovídat poptávku po výrobcích, sledovat stav zásob a zpracovávat data pro prodej (SOP – Sales Order Processing) a nákup (POP – Purchase Order Processing). Další oblastí je výroba a moduly sem patřící jsou například moduly MPS (Master Production Scheduling), které umožňují vytvářet hlavní plán výroby. Další moduly jsou schopné vytvářet kusovníky (BOM) nebo provádět kapacitní plánování (CRP). Důležitým modulem je samotný klasický MRP, který vytvoří plánovací příkazy. Třetí oblastí jsou finance. Moduly například vedou účetní knihy pro prodeje i nákupy. Finální systém MRP II je složeninou několika modulů, které jsou vzájemně propojené a komunikují spolu. Data, která do jednotlivých modulů vstupují jsou výstupem z jednoho nebo více jiných modulů. V dnešní době se používá jako součást ERP systémů [12].

3.4 ERP (Plánování podnikových zdrojů)

Už od 60. let se v mnoha podnicích objevovaly snahy o zavedení informačních systémů. Vznikly tak systémy pro sledování a řízení zásob a skladů. V 70. letech poté vznikl systém MRP, který byl jedním z prvních ucelených a široce nasazovaných systémů. V 80. letech vznikl systém MRP II, který obsahoval navíc ještě výrobní činnosti podniku. V devadesátých letech byl na základě MRP a MRP II vyvinut systém ERP (z angl. Enterprise Resource Planning), který integroval součásti výrobního procesu jako například výrobu, distribuci, lidské zdroje a řízení zásob. ERP se dál vyvíjel a v roce 2000 byl rozšířen o moduly a funkce pro lepší propojení s dalšími články podnikového hodnotového řetězce. Takto vylepšený systém se někdy označuje jako ERP II. Dalším zlomovým rokem se stal rok 2004, kdy byl uveden standard SOA (Services Oriented Architecture). Cílem SOA je umožnit komunikaci a spolupráci ERP systému od různých výrobců. V následujících letech stále pokračují snahy ERP systémy rozšiřovat a zapracovávat do nich další oblasti, které ještě nejsou jeho součástí (např. Management znalostí) [11].

ERP je systém, který je určen pro podniky a který pomáhá při jejich správě a řízení. Systémy ERP jsou dodávány v podobě modulů. Každý modul plní určitou funkci a pokud je potřeba komunikuje s dalšími moduly. Ve většině případů si podnik podle svých potřeb pořídí „balík“ modulů, které plní funkce, které daný podnik potřebuje. ERP má tři hlavní oblasti ve kterých pracuje. Tyto oblasti jsou Ekonomika, Logistika a skladové hospodářství a lidské

zdroje. Do oblasti ekonomiky patří správa účetnictví, faktur a majetku. Do logistické části spadají činnosti spojené s výrobou, jako například nákupy a prodeje a správa skladu [12].

Existuje několik způsobů realizace ERP systémů, které se liší typem a architekturou.

3.4.1 Dvouúrovňová architektura (klient/server)

Původní ERP systém byl vyvinut na principu klient/server. Klient umožňuje uživatelům zobrazit srozumitelná data. Server data zpracovává a ukládá. Nevýhodou této architektury je pokles výkonu s rostoucím počtem klientů. Další slabinou je nemožnost přidat změnu databáze. Tato architektura byla využívána společnostmi jako NetSUite, OpenBravo, SAP, Microsoft [11].

3.4.2 Třístupňová architektura

Třístupňová architektura nabízí řešení problémů spojených s dvouúrovňovou architekturou. Skládá se ze tří vrstev. První vrstva je vrstva prezentační (GUI – grafické rozhraní), druhou vrstvou je aplikační. Aplikační vrstva zajišťuje přenos zpráv pro aplikační servery a také v této vrstvě dochází ke zpracování zpráv. Tzn. k samotnému rozvrhování a řízení (bussiness logic). Aplikační vrstva také zprostředkovává komunikaci mezi první vrstvou a třetí, kterou je databázová vrstva. Databázová vrstva se stará o ukládání, přidávání, odebírání a aktualizaci dat. V této architektuře se využívá databázových serverů, které už umožňují měnit data přímo na serveru. Výhodou této architektury je lepší flexibilita a spolehlivost. Za nevýhodu lze označit větší složitost a vyšší cenu [11].

3.4.3 Architektura založená na architektuře Webu

Rozšíření internetu a vznik Webu značně změnil svět informačních technologií a ovlivnil směr vývoje na roky následující. Systém ERP samozřejmě nezůstal pozadu a došlo ke vzniku architektury, která využívá principů a architektury na kterých je vystaven internet a samotný Web. Typ komunikace klient/server zůstal zachován, ale s tím rozdílem, že funkce klienta plní stanice uživatele a roli serveru plní webový server. Tato architektura opět využívá tří vrstev. První vrstvou jsou již zmíněné uživatelské stanice. Druhou vrstvu tvoří webový server, který zprostředkovává komunikaci mezi první vrstvou a třetí (aplikační) vrstvou. Webový server tedy zaručuje, že informace z aplikační vrstvy jsou přenášené uživatelům ve formě HTML a ASP. Uživatel tak na svém počítači pracuje s grafickým rozhraním, které funguje ve webovém prohlížeči. Třetí aplikační a datová vrstva zodpovídá za zpracování a ukládání dat. Tato architektura umožňuje snadnou modifikaci dat a rychlý přenos dat. Největším nebezpečím je bezpečnost, protože data jsou přístupná přes internet [11].

3.4.4 MPS (Master Production Scheduling)

MPS je jeden z modulů systémů MRP II a jeho funkce je velice podstatná. Jeho účel se může na první pohled zdát nesmyslný nebo dokonce kontraproduktivní. Cílem MPS je totiž oddělit prodeje od výroby. Tyto dvě oblasti spolu značně souvisí a je naprosto logický přístup, který říká, že množství zboží, které se má vyrobit se rovná množství zboží, které je poptáváno.

Tento přístup však nebere v potaz některé skutečnosti. Pokud například vedení podniku ví, že kvůli poptávce jsou schopni prodat 500 kusů nějakého výrobků je potřeba nejdříve zjistit, zda je podnik takové množství schopen včas vyrobit. Pokud se totiž daný výrobek sestavuje ze tří dílů a pro výrobu 500 kusů sestav je potřeba 500 kusů od každého druhu dílu, může nastat problém, že z omezení daných výrobními nebo skladovacími kapacitami nemůže být dodáno včas dostatek dílů. Pokud přesto podnik naplánuje výrobu 500 kusů výrobku, dojde k tomu, že nebude tento požadavek splněn, ale ještě navíc bude vyrobeno nadbytečné množství některých druhů dílů. Pokud je na výrobu jednoho finálního výrobky potřeba jeden díl A, jeden díl B a jeden díl C a podnik je schopen vyrobit celkově jen 1200 kusů jednotlivých dílů, nastává problém. K výrobě 500 kusů finálního produktu je celkem potřeba 1500 kusů dílů. Podnik je tedy schopen dodat jen 400 výrobků. Tato úvaha ovšem musí být provedena už na začátku, protože jinak se v průběhu výroby zjistí, že není možné získat dostatečné množství dílů a zatímco některý druh dílů chybí, dílů jiného druhu je nadbytek. Toto je jeden z důvodů, proč se používá MPS.

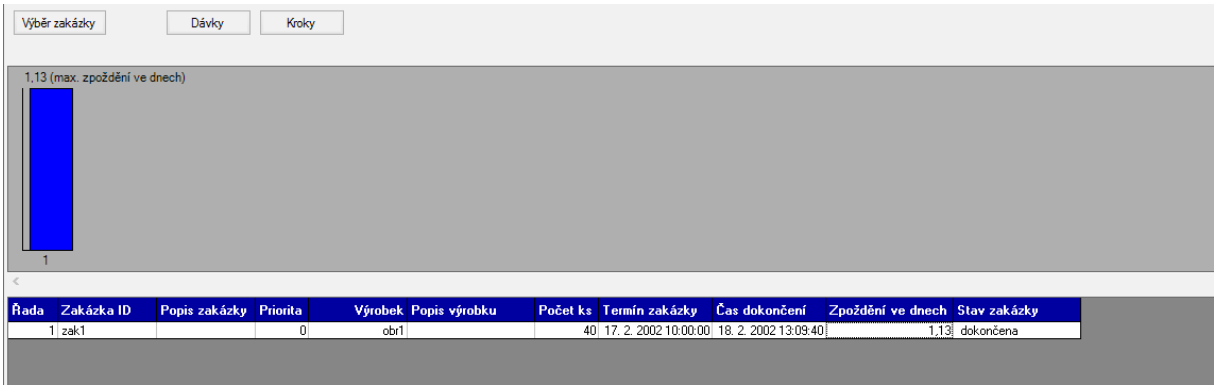
MPS je mechanismus, který pomáhá převést prodeje na výrobní plány a zároveň vyvažuje poptávku a zásoby. Existuje několik důvodů proč se MPS používá. Většina z nich skončí výše popsaným případem, kdy podnik není schopen včas vyrobit požadované množství. K tomuto dochází z několika důvodů. Obecně lze říci, že hlavním důvodem je nepravidelnost poptávky. Typickým příkladem nepravidelné poptávky může být například sezónní poptávka, kdy některé zboží zákazník potřebuje jen v určité části roku. V mnoha oborech je také poptávka obecně vyšší na konci měsíce. Nepravidelnou poptávku je potřeba předpovídat a vyrábět předem na sklad v období, kdy jsou výrobní kapacity méně vytížené. Samozřejmě může přijít velká zakázky, kterou nebylo možné předvídat nebo se ji nepodařilo přesně odhadnout. Může tak dojít k přetížení výrobních kapacit. V takovém případě je potřeba prioritizovat a méně důležité práce provést, až se kapacity uvolní.

4 Rozhraní pro rozvrhování výroby

V následující části bude na konkrétní databázi demonstrováno, ze kterých částí se skládá výrobní proces, které pojmy a objekty v něm vystupují a jakým způsobem jsou informace ukládány a prezentovány. Budou zde představeny vztahy a principy, které se ve výrobních procesech vyskytují. Přehledný systém, který zobrazuje jednotlivé objekty a vztahy mezi nimi je velice potřebný, protože umožňuje pochopit vazby jednotlivých prvků systému a stejně tak je potřebná dobrá prezentace dat, která dovolí znázornit a lépe pochopit, jakým způsobem dochází k pohybu materiálu a jeho přeměně. Tato část je také velmi významná jak pro identifikaci úzkých míst, tak i potenciálně rizikových části systému. Souhrnně řečeno, když jsou systémová data přehledně uspořádána a názorně prezentována, lze je použít pro hlubší porozumění systému, nalezení míst, jejichž vylepšením se významně přispěje k souhrnné optimalizaci a bude tak dosažen cíl – zlepšení v celkových parametrech systému. To například znamená snížení nákladů, zrychlení výrobního procesu a maximalizaci zisku.

4.1 Zakázka

Na základě zakázek dochází k inicializaci výroby. Zakázka je žádost na výrobu určitého množství nějakého výrobku. Jinak řečeno, je to požadavek, aby stanovený materiál byl určitým množstvím předem daných operací přeměněn na nějaký finální produkt. Příklad jednoduché zakázky je na obr. 4.1. Jedním z parametrů, které zakázka obsahuje, je označení zakázky identifikátorem. V tomto případě se jedná o název zak1. Dalším parametrem je priorita. Vyšší číslo znamená vyšší prioritu. Zakázka, která má vyšší prioritu dostává přednost před zakázkami s nižší prioritou. Důvodů, proč některé zakázky musí být dokončeny přednostně, bývá několik. Uvedu tři příklady: Jedná se, o důležitého zákazníka, ve smlouvě je stanovena penalizace při zpoždění zakázky, zákazník si si připlatil za přednostní zhotovení. V případě zachyceném na obrázku č. 4.1 je priorita 0, nejedná se tedy o zakázku, která musí být dokončena přednostně. Dalším údajem obsaženým v definici zakázky je jaký výrobek je požadován a v jakém množství. Zde je potřeba 40 kusů obrobku, který je označen jako obr1.



Obrázek 4.1: Ukázka dat, která obsahuje zakázka

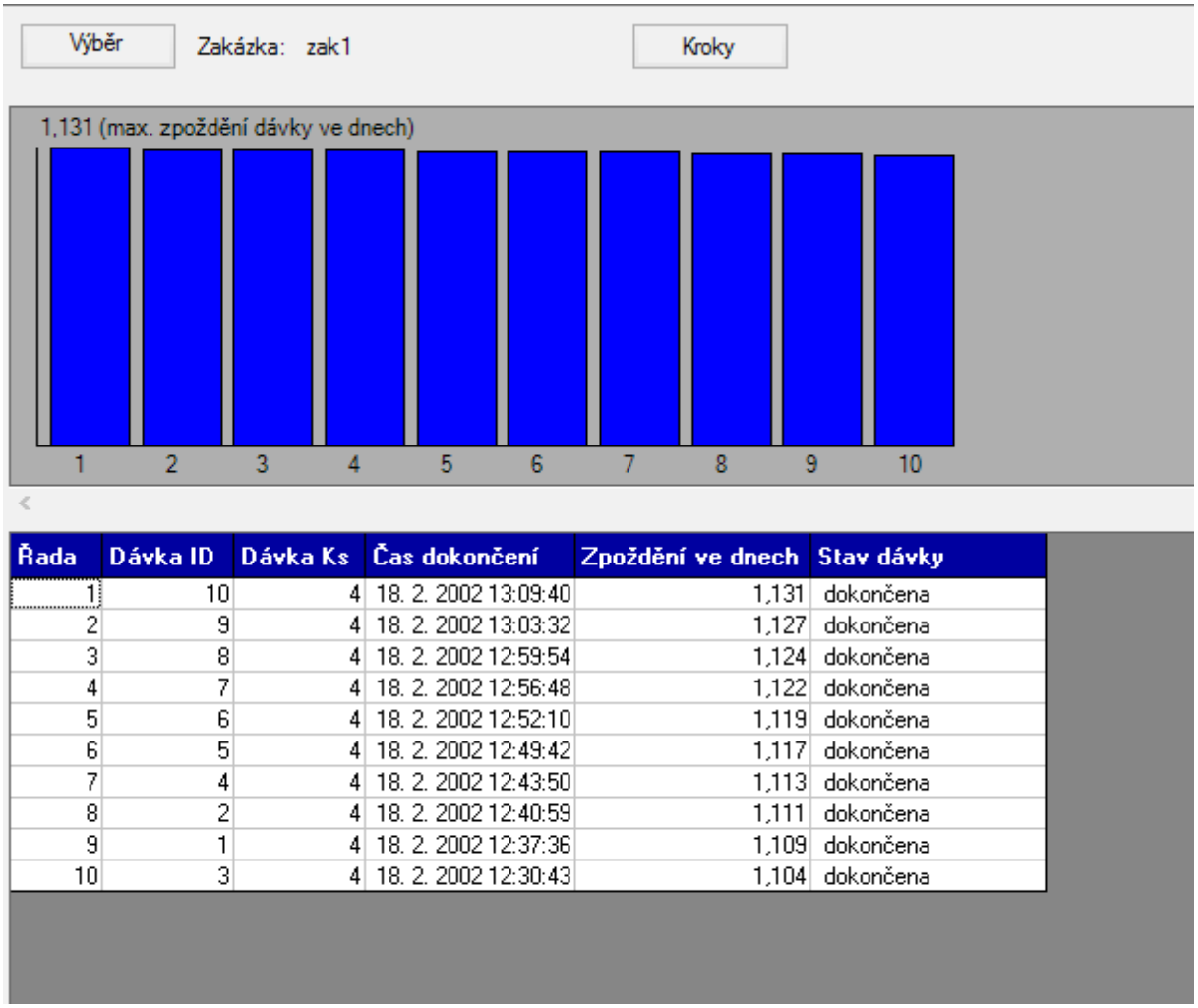
Následující údaj je termín zakázky. Termín udává, kdy má být zakázka hotova. Zakázka na obrázku už byla dokončena a obsahuje tedy i údaj o času dokončení. Protože zakázka nebyla dokončena včas, je pro ni známo i zpoždění, které je 1,13 dní.

4.1.1 Zakázka a výrobní dávky

Zakázka je rozdělena na výrobní dávky. Výrobní dávka je množství určitého výrobku, které vystupuje jako celek, tedy jako celek je evidováno, jako celek vstupuje do jednotlivých výrobních operací a jako celek z něj vystupuje. Kusy v dávce jsou zpracovávány na jednom pracovišti buď současně nebo hned po sobě. Velikost výrobní dávky se určuje jako kompromis. Na jedné straně je snaha nemít zakázky příliš malé, ty totiž ztěžují řízení výroby a snižují produktivitu práce. Na druhou stranu není dobré mít výrobní dávky ani příliš velké. Velké dávky se negativně promítnou do nákladů na skladování, snižují možnost reagovat na změny nebo poruchy při výrobě a prodlužují průběžnou dobu. [9]

Rozdělení zakázky na výrobní dávky umožňuje větší flexibilitu, kdy více dávek z jedné zakázky je možné přidělit na několik výrobních strojů provádějících stejnou operaci a tak tedy vyrábět na několika strojích výrobky pro jednu zakázku současně.

Na obrázku 4.2 jsou uvedeny údaje pro výrobní dávky u zakázky zak1. Tato zakázka obsahuje 40 kusů výrobku obr1. Zde je tato zakázka rozdělena na 10 výrobních dávek, každá po čtyrech kusech. Každá dávka má své specifické označení. Všechny výrobní dávky už byly dokončeny a ta dávka, která je dokončena jako poslední (tzn. dávka s ID 10) určuje čas dokončení celé zakázky. Zakázka je hotová, až jsou dokončeny všechny výrobní dávky, ze kterých se zakázka skládá. Čas dokončení a zpoždění poslední dávky odpovídá dokončení a zpoždění celé zakázky.

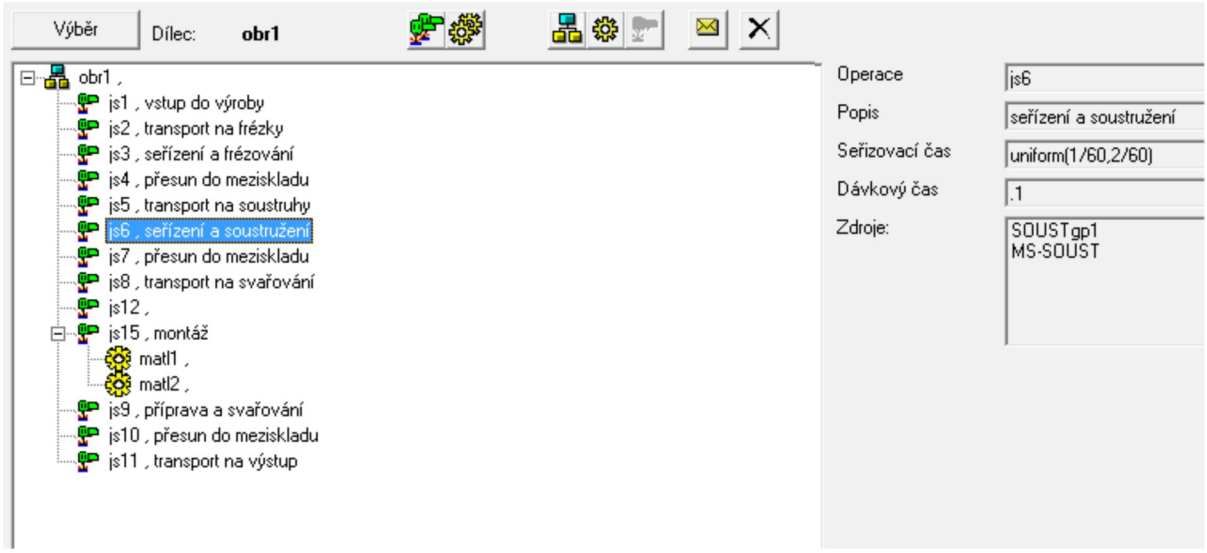


Řada	Dávka ID	Dávka Ks	Čas dokončení	Zpoždění ve dnech	Stav dávky
1	10	4	18. 2. 2002 13:09:40	1,131	dokončena
2	9	4	18. 2. 2002 13:03:32	1,127	dokončena
3	8	4	18. 2. 2002 12:59:54	1,124	dokončena
4	7	4	18. 2. 2002 12:56:48	1,122	dokončena
5	6	4	18. 2. 2002 12:52:10	1,119	dokončena
6	5	4	18. 2. 2002 12:49:42	1,117	dokončena
7	4	4	18. 2. 2002 12:43:50	1,113	dokončena
8	2	4	18. 2. 2002 12:40:59	1,111	dokončena
9	1	4	18. 2. 2002 12:37:36	1,109	dokončena
10	3	4	18. 2. 2002 12:30:43	1,104	dokončena

Obrázek 4.2: Výrobní dávky zakázky zak1

4.1.2 Kusovník

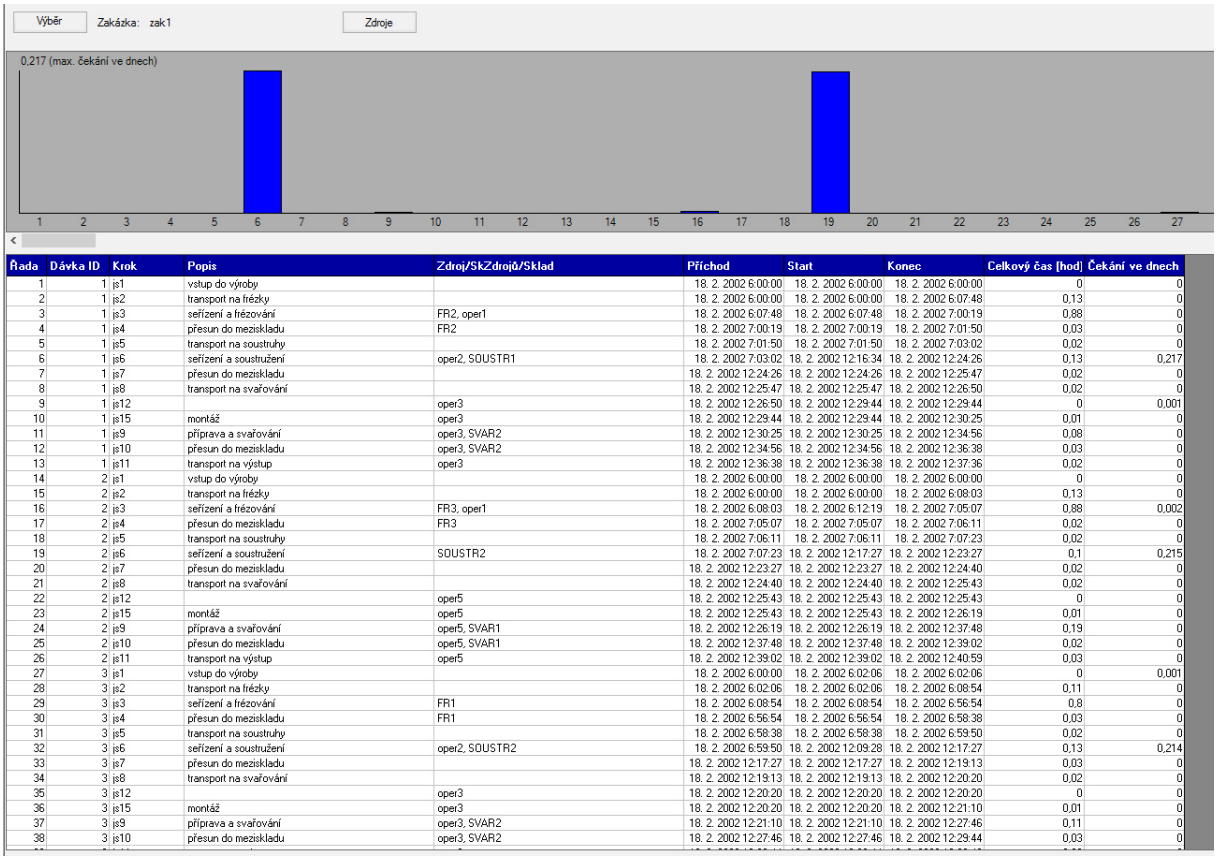
V předešlé části se rozebírala zakázka zak1, ve které se vyrábí výrobek obr1. Každý výrobek musí mít svůj kusovník. Na obr. 4.3 je uveden kusovník pro výrobek obr1. Kusovník (angl. BOM – Bill of Materials) typicky obsahuje údaje o tom, z jakých částí se výrobek skládá. V tomto případě se jedná o kusovník, který obsahuje jednotlivé kroky, které je potřeba provést pro získání výrobku. Operace se týkají pohybu materiálu nebo výrobních operací. Pro každou operaci jsou uvedeny zdroje nebo skupiny zdrojů na kterých se provádí. V kroku js15 se provádí montáž a do procesu vstupují dva nakupované materiály matl1 a matl2.



Obrázek 4.3: Kusovník pro výrobek obr1

4.1.3 Procesní kroky

Když je známé rozdělení na výrobní dávky, pomocí kusovníku se provede rozvržení na procesní kroky pro výrobek obsažený v zakázce (viz obr. 4.4). Každý procesní krok obsahuje jednu operaci pro jednu výrobní dávku. Popis kroku obsahuje označení dávky, označení kroku podle kusovníku, popřípadě zdroj, kterému je operace přiřazena. Dále obsahuje termín začátku a konce kroku a celkový čas. Dalším údajem je doba čekání. Když je při výrobním procesu dokončen jeden krok, v ideálním případě hned následuje další krok a tok materiálu se pak nezastavuje. Během výrobního procesu neustále probíhají výrobní operace nebo další činnosti, jako jsou přesuny do meziskladu. Výrobní proces je ale značně rozsáhlý a komplexní systém. Dochází v něm k vzájemnému ovlivňování jednotlivých dílčích procesů. Proto nastávají situace, kdy je dokončen jeden procesní krok, ale další nemůže začít, protože se musí například čekat na transport materiálu ze skladu. V procesních krocích se tak může vyskytnout stav, kdy se pouze čeká. Příchod materiálu na dané stanoviště a začátek práce na daném kroku se tak liší a naopak je nulová doba mezi startem a koncem kroku. To je patrné na obr. 4.4, viz řádek 9 .



Řada	Dávka ID	Krok	Popis	Zdroj/SkZdrojů/Sklad	Příchod	Start	Konec	Celkový čas [hod]	Čekání ve dnech
1	1	js1	vstup do výroby		18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	0	0
2	1	js2	transport na frézky		18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	18. 2. 2002 6:07:48	0,13	0
3	1	js3	seřízení a frézování	FR2, oper1	18. 2. 2002 6:07:48	18. 2. 2002 6:07:48	18. 2. 2002 7:00:19	0,88	0
4	1	js4	přesun do meziskladu	FR2	18. 2. 2002 7:00:19	18. 2. 2002 7:00:19	18. 2. 2002 7:01:50	0,03	0
5	1	js5	transport na soustruhy		18. 2. 2002 7:01:50	18. 2. 2002 7:01:50	18. 2. 2002 7:03:02	0,02	0
6	1	js6	seřízení a soustružení	oper2, SOUSTR1	18. 2. 2002 7:03:02	18. 2. 2002 12:16:34	18. 2. 2002 12:24:26	0,13	0,217
7	1	js7	přesun do meziskladu		18. 2. 2002 12:24:26	18. 2. 2002 12:24:26	18. 2. 2002 12:25:47	0,02	0
8	1	js8	transport na svařování		18. 2. 2002 12:25:47	18. 2. 2002 12:25:47	18. 2. 2002 12:26:50	0,02	0
9	1	js12		oper3	18. 2. 2002 12:26:50	18. 2. 2002 12:29:44	18. 2. 2002 12:29:44	0	0,001
10	1	js15	montáž	oper3	18. 2. 2002 12:29:44	18. 2. 2002 12:29:44	18. 2. 2002 12:30:25	0,01	0
11	1	js9	příprava a svařování	oper3, SVAR2	18. 2. 2002 12:30:25	18. 2. 2002 12:30:25	18. 2. 2002 12:34:56	0,08	0
12	1	js10	přesun do meziskladu	oper3, SVAR2	18. 2. 2002 12:34:56	18. 2. 2002 12:34:56	18. 2. 2002 12:36:38	0,03	0
13	1	js11	transport na výstup	oper3	18. 2. 2002 12:36:38	18. 2. 2002 12:36:38	18. 2. 2002 12:37:36	0,02	0
14	2	js1	vstup do výroby		18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	0	0
15	2	js2	transport na frézky		18. 2. 2002 6:00:00	18. 2. 2002 6:00:00	18. 2. 2002 6:08:03	0,13	0
16	2	js3	seřízení a frézování	FR3, oper1	18. 2. 2002 6:08:03	18. 2. 2002 6:12:19	18. 2. 2002 7:05:07	0,88	0,002
17	2	js4	přesun do meziskladu	FR3	18. 2. 2002 7:05:07	18. 2. 2002 7:05:07	18. 2. 2002 7:06:11	0,02	0
18	2	js5	transport na soustruhy		18. 2. 2002 7:06:11	18. 2. 2002 7:06:11	18. 2. 2002 7:07:23	0,02	0
19	2	js6	seřízení a soustružení	SOUSTR2	18. 2. 2002 7:07:23	18. 2. 2002 12:17:27	18. 2. 2002 12:23:27	0,1	0,215
20	2	js7	přesun do meziskladu		18. 2. 2002 12:23:27	18. 2. 2002 12:23:27	18. 2. 2002 12:24:40	0,02	0
21	2	js8	transport na svařování		18. 2. 2002 12:24:40	18. 2. 2002 12:24:40	18. 2. 2002 12:25:43	0,02	0
22	2	js12		oper5	18. 2. 2002 12:25:43	18. 2. 2002 12:25:43	18. 2. 2002 12:25:43	0	0
23	2	js15	montáž	oper5	18. 2. 2002 12:25:43	18. 2. 2002 12:25:43	18. 2. 2002 12:26:19	0,01	0
24	2	js9	příprava a svařování	oper5, SVAR1	18. 2. 2002 12:26:19	18. 2. 2002 12:26:19	18. 2. 2002 12:37:48	0,19	0
25	2	js10	přesun do meziskladu	oper5, SVAR1	18. 2. 2002 12:37:48	18. 2. 2002 12:37:48	18. 2. 2002 12:39:02	0,02	0
26	2	js11	transport na výstup	oper5	18. 2. 2002 12:39:02	18. 2. 2002 12:39:02	18. 2. 2002 12:40:59	0,03	0
27	3	js1	vstup do výroby		18. 2. 2002 6:00:00	18. 2. 2002 6:02:06	18. 2. 2002 6:02:06	0	0,001
28	3	js2	transport na frézky		18. 2. 2002 6:02:06	18. 2. 2002 6:02:06	18. 2. 2002 6:08:54	0,11	0
29	3	js3	seřízení a frézování	FR1	18. 2. 2002 6:08:54	18. 2. 2002 6:08:54	18. 2. 2002 6:56:54	0,8	0
30	3	js4	přesun do meziskladu	FR1	18. 2. 2002 6:56:54	18. 2. 2002 6:56:54	18. 2. 2002 6:58:38	0,03	0
31	3	js5	transport na soustruhy		18. 2. 2002 6:58:38	18. 2. 2002 6:58:38	18. 2. 2002 6:59:50	0,02	0
32	3	js6	seřízení a soustružení	oper2, SOUSTR2	18. 2. 2002 6:59:50	18. 2. 2002 12:09:28	18. 2. 2002 12:17:27	0,13	0,214
33	3	js7	přesun do meziskladu		18. 2. 2002 12:17:27	18. 2. 2002 12:17:27	18. 2. 2002 12:19:13	0,03	0
34	3	js8	transport na svařování		18. 2. 2002 12:19:13	18. 2. 2002 12:19:13	18. 2. 2002 12:20:20	0,02	0
35	3	js12		oper3	18. 2. 2002 12:20:20	18. 2. 2002 12:20:20	18. 2. 2002 12:20:20	0	0
36	3	js15	montáž	oper3	18. 2. 2002 12:20:20	18. 2. 2002 12:20:20	18. 2. 2002 12:21:10	0,01	0
37	3	js9	příprava a svařování	oper3, SVAR2	18. 2. 2002 12:21:10	18. 2. 2002 12:21:10	18. 2. 2002 12:27:46	0,11	0
38	3	js10	přesun do meziskladu	oper3, SVAR2	18. 2. 2002 12:27:46	18. 2. 2002 12:27:46	18. 2. 2002 12:29:44	0,03	0

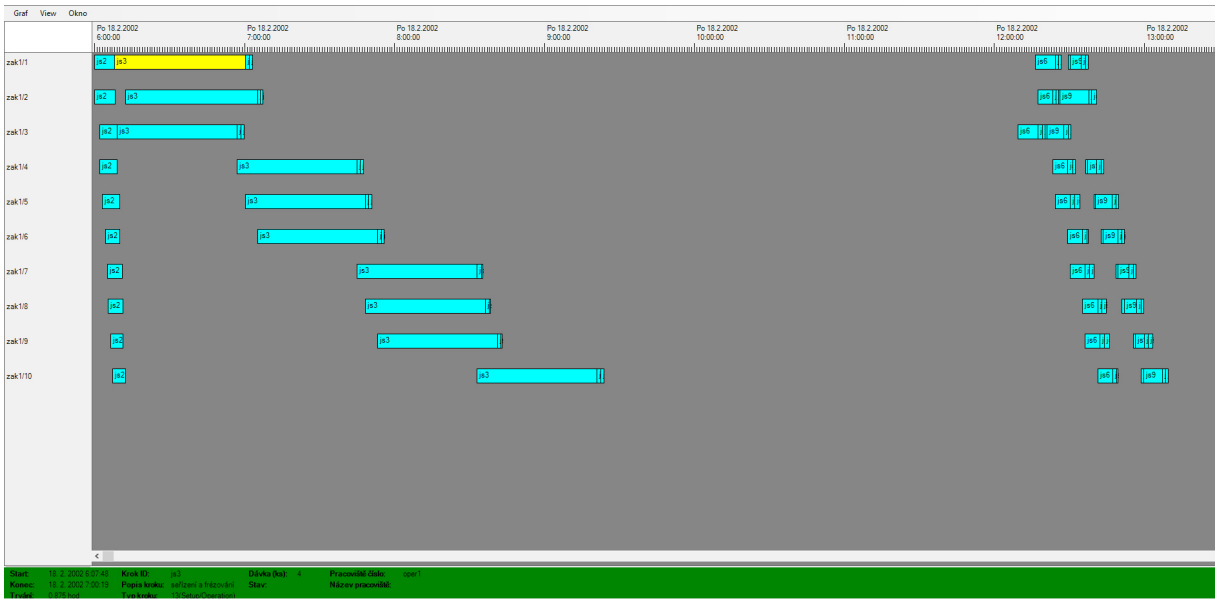
Obrázek 4.4: Procesní kroky zakázky zak1

4.1.4 Ganttův diagram pro zakázky

Než bude zodpovězena otázka co je Ganttův diagram, je vhodné vysvětlit okolnosti, které vedly k jeho vzniku a použití. Jak už bylo řečeno, výrobní systém je poměrně složitý a vystupuje v něm mnoho zdrojů, zakázek, druhů materiálu apod. Celý výrobní proces, který určuje využití výrobního systému, probíhá v čase, a proto je třeba brát v úvahu posloupnost jednotlivých operací. Pokud v systému figurují desítky zakázek, což může znamenat stovky kroků a operací, pokaždé s přiřazeným zdrojem a stanovenou dobou začátku a konce, je velice nesnadné udržet si přehled. Pro správné naplánování výrobního procesu je ale nezbytné, aby nedocházelo ke stavům, kdy například jednomu stroji je v jeden časový okamžik přiřazeno několik různých výrobních dávek na jednou. Vzniká tedy potřeba vhodné prezentace či vizualizace operací vzhledem k časové ose a řešením je právě zavedení Ganttových diagramů. Ganttův diagram je grafické znázornění posloupnosti činností v čase. S jeho pomocí lze snadno získat přehled o činnostech ve formě jakéhosi rozvrhu. Při plánování výroby jsou Ganttovy diagramy naprosto nezbytným pomocníkem. Ganttův diagram umožňuje zobrazit přehled činností, z něhož je hned patrné, že v nějakém bodě je naplánováno více činností než je možné provést. Pomocí Ganttových diagramů tak lze okamžitě možné získat přehled o tom, zda je možné dané činnosti skutečně provést tak, jak je to naplánováno.

Ganttovy diagramy je možné použít pro zdroje a zobrazit tak pro každý jednotlivý zdroj posloupnost činností, které na něm budou probíhat. O tom bude řeč v další kapitole věnované zdrojům.

Popsané diagramy je možné také použít pro zakázky a výrobní dávky. Na obrázku 4.5 je vyobrazen Ganttův diagram pro zakázku zak1, která je rozdělena na 10 výrobních dávek. U každé výrobní dávky jsou vidět jednotlivé procesní kroky a kdy v čase probíhají. Je možné zobrazit podrobnosti jednotlivých kroků a zjistit k jakému zdroji jsou přiřazeny a také je možné zobrazit informace o dané výrobní dávce. Lze zjistit, kdy skončí poslední krok a tedy, kdy bude dokončena výrobní dávka. Především diagramy umožní srozumitelně zobrazit operace a takto nabyté vědomosti mohou být nadále využity. U první dávky zakázky zak1 (označení zak1/1) je vidět velká časová prodleva před krokem js6. Tato mezera je způsobena tím, že zdroj pro krok js6 je v daném okamžiku vytížen a nemá dostupnou kapacitu. Také je z diagramu jednoznačně patrné, že některé kroky jsou velice krátké a jejich délka je oproti jiným téměř zanedbatelná.



Obrázek 4.5: Ganttův diagram zakázky zak1

4.2 Zdroje

Zdroje jsou prostředky, na kterých probíhají jednotlivé operace. Typickým představitelem zdroje ve výrobním procesu je stroj, například frézka. Tato frézka má určitou výrobní kapacitu, to znamená, že je schopná provádět určité množství výrobních operací určitou rychlostí. Na zdrojích dochází k přetváření materiálu s cílem dosáhnout přeměny na nějaký finální výrobek. Tento výrobek může být následně prodán a generuje tak společnosti zisk. Zdroje tak svou činností vytváří nějakou hodnotu. Zdroje umožňují dosáhnout výnosů, ale projevují se také v nákladech.

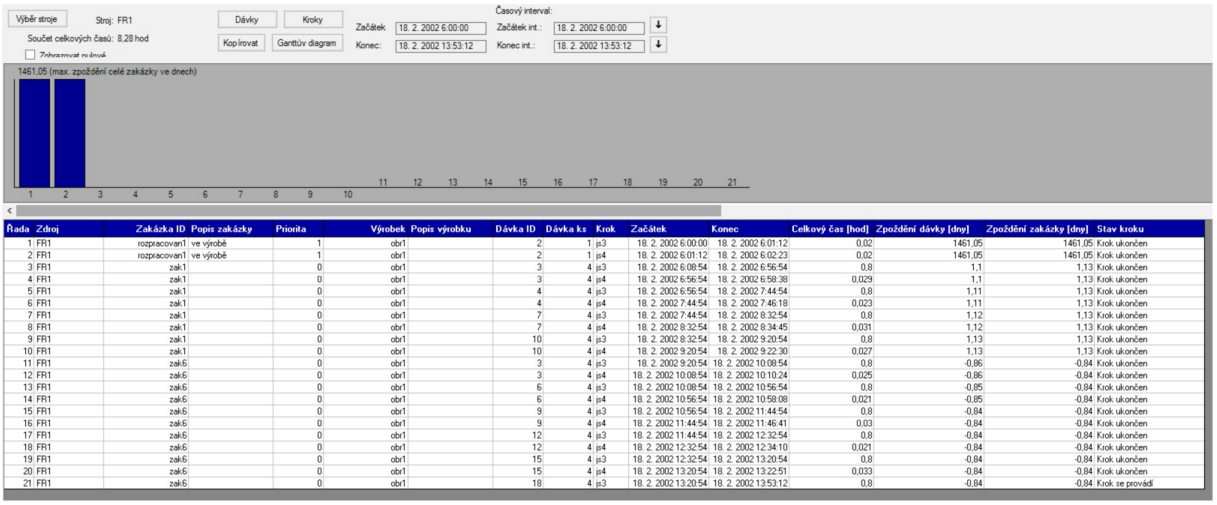
Každý stroj má nějakou pořizovací cenu, k tomu přistupují tzv. provozní náklady, vynaložené na provoz a údržbu. Každý stroj také stárne a ztrácí hodnotu jak opotřebením fyzickým, závislým na počtu výrobků a počtu provozních hodin, tak opotřebením morálním, na těchto parametrech nezávislým. Podnik pro maximalizaci ekonomických ukazatelů musí propočítat, kolik strojů bude pro zajištění výroby potřebovat. Pravidla ekonomické efektivity počítají vhodnost vytvoření a navýšení výrobní kapacity podle různých kritérií. Jeden typ investiční optimalizace přihlíží k odhadované poptávce a jejím výkyvům. Zjišťuje např., jestli se kvůli velké, ale krátkodobé zakázce vyplatí pořídit další stroje a případně kolik, když pak po skončení dlouhodobě zůstanou nevyužité.

Jiná optimalizace řeší vztah mezi cenou, kvalitou a provozními náklady různých značek a typů stroje pro tutéž výrobní činnost. Obvykle platí, že stroje investičně drahé mají levnější provoz a delší životnost. Výpočtem je možno pro každý typ nalézt minimum funkce udávající závislost souhrnných nákladů stroje na době, po kterou stroj bude v provozu používán. Od nákladového minima se směrem ke kratšímu času celkové doby použití stroje uplatní více vliv drahé málo amortizované investice, směrem k delšímu času zase vliv zvyšujících se provozních nákladů. Každý stroj má minimum jinde, levné typy mají minimum u kratších dob a jejich provozní náklady stoupají strměji, u kvalitních drahých strojů je tomu naopak. S celou optimalizací pak významně hýbou okolnosti jako leasing místo investování, výhodný předčasný prodej stroje před koncem jeho doby života a pořízení stroje v podobě služby hrazené stálým pravidelným platem poskytovateli. Ale protože investiční plánování se jak metodami, tak i dlouhodobým časovým rámcem vymyká zadání této práce, nadále se jím nebudu zabývat.

V ideálním případě má každý výrobní podnik takové množství strojů, že jsou všechny zdroje vždy dostatečně využité a zároveň nedochází ke stavům, kdy by byly natolik využité, že by nestíhaly dokončovat zakázky včas.

4.2.1 Rozvrh zdroje

Rozvrh zdroje je rozpis činností, pro určité časové období, které se budou provádět na určitém zdroji. Na obrázku 4.6 je ukázka rozvrhu pro zdroj, kterým je frézka s označením FR1. Každý údaj udává jednu operaci. Každá operace je tvořena výrobní dávkou určité zakázky. V rozvrhu je možné nalézt prioritu operace, která závisí na prioritě zakázky. Je patrné, že operace prioritní zakázky se provádí jako první. Dalším údajem je identifikace výrobku, který je předmětem výrobní operace. Kromě identifikace zakázky rozvrh obsahuje samozřejmě i identifikaci dávky a množství výrobků v dávce obsažených. Nezbytným údajem rozvrhu je samozřejmě datum a čas začátku a ukončení operace. Podle termínu dokončení se zjistí zpoždění dávky. Rozvrh obsahuje také dobu trvání dané operace.



Řada	Zdroj	Zakázka ID	Popis zakázky	Priorita	Výrobek	Popis výrobku	Dávka ID	Dávka ks	Krok	Začátek	Konec	Celkový čas [hod]	Zpoždění dávky [dny]	Zpoždění zakázky [dny]	Stav kroku
1	FR1	rozpracovan1	ve výrobě	1	obr1		2	1	jt3	18. 2. 2002 6:00:00	18. 2. 2002 6:01:12	0,02	1461,05	1461,05	Krok ukončen
2	FR1	rozpracovan1	ve výrobě	1	obr1		2	1	jt4	18. 2. 2002 6:01:12	18. 2. 2002 6:02:23	0,02	1461,05	1461,05	Krok ukončen
3	FR1	zak1		0	obr1		3	4	jt3	18. 2. 2002 6:08:54	18. 2. 2002 6:56:54	0,8	1,1	1,13	Krok ukončen
4	FR1	zak1		0	obr1		3	4	jt4	18. 2. 2002 6:56:54	18. 2. 2002 6:58:38	0,029	1,1	1,13	Krok ukončen
5	FR1	zak1		0	obr1		4	4	jt3	18. 2. 2002 6:56:54	18. 2. 2002 7:44:54	0,8	1,11	1,13	Krok ukončen
6	FR1	zak1		0	obr1		4	4	jt4	18. 2. 2002 7:44:54	18. 2. 2002 7:46:18	0,023	1,11	1,13	Krok ukončen
7	FR1	zak1		0	obr1		7	4	jt3	18. 2. 2002 7:44:54	18. 2. 2002 8:32:54	0,8	1,12	1,13	Krok ukončen
8	FR1	zak1		0	obr1		7	4	jt4	18. 2. 2002 8:32:54	18. 2. 2002 8:34:45	0,031	1,12	1,13	Krok ukončen
9	FR1	zak1		0	obr1		10	4	jt3	18. 2. 2002 8:32:54	18. 2. 2002 9:20:54	0,8	1,13	1,13	Krok ukončen
10	FR1	zak1		0	obr1		10	4	jt4	18. 2. 2002 9:20:54	18. 2. 2002 9:22:30	0,027	1,13	1,13	Krok ukončen
11	FR1	zak6		0	obr1		3	4	jt3	18. 2. 2002 9:20:54	18. 2. 2002 10:08:54	0,8	-0,86	-0,84	Krok ukončen
12	FR1	zak6		0	obr1		3	4	jt4	18. 2. 2002 10:08:54	18. 2. 2002 10:10:24	0,025	-0,86	-0,84	Krok ukončen
13	FR1	zak6		0	obr1		6	4	jt3	18. 2. 2002 10:08:54	18. 2. 2002 10:56:54	0,8	-0,85	-0,84	Krok ukončen
14	FR1	zak6		0	obr1		6	4	jt4	18. 2. 2002 10:56:54	18. 2. 2002 10:58:08	0,021	-0,85	-0,84	Krok ukončen
15	FR1	zak6		0	obr1		9	4	jt3	18. 2. 2002 10:56:54	18. 2. 2002 11:44:54	0,8	-0,84	-0,84	Krok ukončen
16	FR1	zak6		0	obr1		9	4	jt4	18. 2. 2002 11:44:54	18. 2. 2002 11:46:41	0,03	-0,84	-0,84	Krok ukončen
17	FR1	zak6		0	obr1		12	4	jt3	18. 2. 2002 11:44:54	18. 2. 2002 12:32:54	0,8	-0,84	-0,84	Krok ukončen
18	FR1	zak6		0	obr1		12	4	jt4	18. 2. 2002 12:32:54	18. 2. 2002 12:34:10	0,021	-0,84	-0,84	Krok ukončen
19	FR1	zak6		0	obr1		15	4	jt3	18. 2. 2002 12:32:54	18. 2. 2002 13:20:54	0,8	-0,84	-0,84	Krok ukončen
20	FR1	zak6		0	obr1		15	4	jt4	18. 2. 2002 13:20:54	18. 2. 2002 13:22:51	0,033	-0,84	-0,84	Krok ukončen
21	FR1	zak6		0	obr1		18	4	jt3	18. 2. 2002 13:20:54	18. 2. 2002 13:53:12	0,8	-0,84	-0,84	Krok se provádí

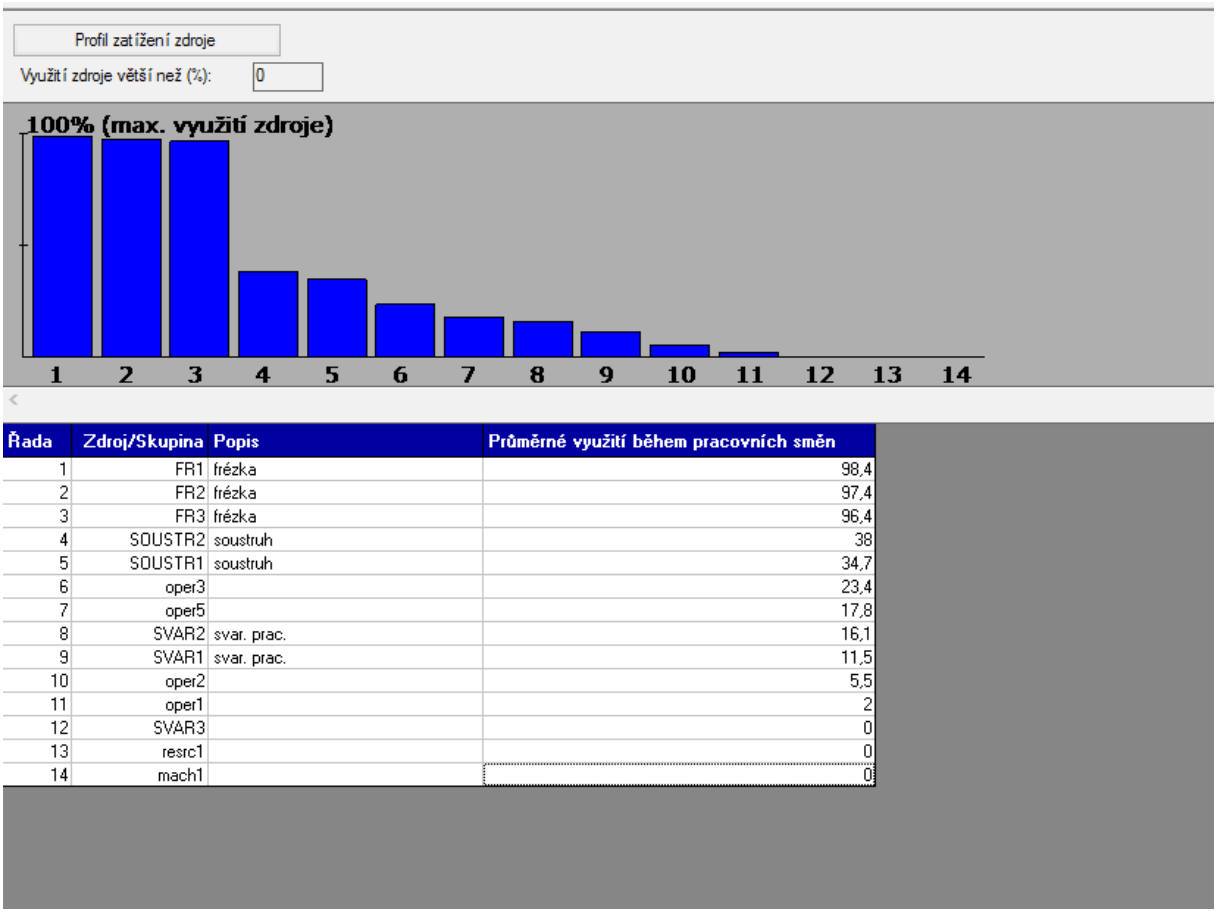
Obrázek 4.6: Rozvrh zdroje FR1

4.2.2 Úzká místa

Důležitým parametrem zdroje je jeho využití. Využití zdroje v určitém časovém úseku udává jaké množství času z tohoto úseku zdroj skutečně vyrábí, tj. nenachází se ve stavu prostoje. Je to poměr mezi čistou dobou produkce a celkovou dobou. Ta je tvořena souhrnem doby výroby i doby prostojů. Pokud je například využití stroje v nějakém časovém období 90 %, znamená to, že v tomto období stroj skutečně vyrábí po 90 % stanoveného času. Příliš nízké využití stroje může být problémové, protože to znamená, že stroj málo produkuje, ale přesto vznikají náklady na provoz a údržbu. V určitých situacích je tento stav nevyhnutelný. Naopak druhým rizikem je, když má stroj vytížení příliš vysoké. Znamená to totiž, že stroj nemá žádnou rezervu a v podstatě neustále vyrábí.

Vysoká vytíženost značí, že se před zdrojem tvoří fronty zakázek a stroj jakmile jednu zakázku dokončí, začíná hned práci na zakázce další, která je ve frontě. Teoreticky by to mohlo znamenat, že stroji přesně v okamžik, kdy dokončí zakázku přijde zakázka další a nevznikne tak žádná čekací doba. Tento stav je ale téměř nemožný. Pokud je stroj velice vytížený, značí to že se před ním hromadí nedokončené zakázky, které čekají ve frontě.

Na obrázku 4.7 je znázorněno využití několika zdrojů. Tři z nich mají velice podobné průměrné využití. Jsou to zdroje FR1, FR2 a FR3. Tyto zdroje patří do stejné skupiny zdrojů a to znamená, že jsou navzájem shodné a jsou jim přidělovány stejné výrobní operace. To kterému z těch tří zdrojů bude přiřazena výrobní dávka závisí na tom, který z nich je právě volný. V tomto případě není takové množství výrobních dávek, aby bylo možné je rovnoměrně rozdělit mezi zdroje a proto se využití těchto zdrojů mírně liší. Velice podstatné je, že se jejich využití pohybuje okolo 96 %. Před uvedenými třemi zdroji se tedy budou tvořit fronty a budou zpomalovat celý výrobní proces. Proto je možné tyto stroje identifikovat jako úzká místa. Pokud je potřeba vylepšit výrobní proces, právě tyto zdroje jsou tím, na co je nutné se zaměřit.



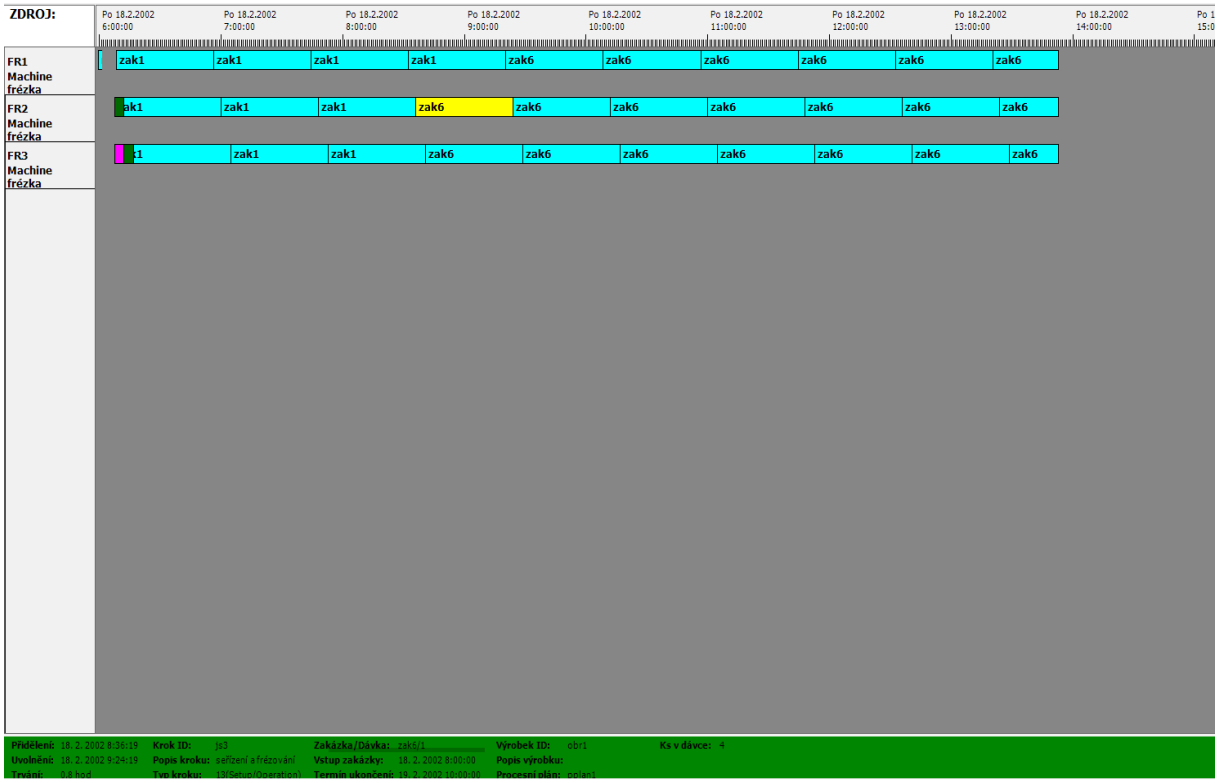
Řada	Zdroj/Skupina	Popis	Průměrné využití během pracovních směn
1	FR1	frézka	98,4
2	FR2	frézka	97,4
3	FR3	frézka	96,4
4	SOUSTR2	soustruh	38
5	SOUSTR1	soustruh	34,7
6	oper3		23,4
7	oper5		17,8
8	SVAR2	svar. prac.	16,1
9	SVAR1	svar. prac.	11,5
10	oper2		5,5
11	oper1		2
12	SVAR3		0
13	resrc1		0
14	mach1		0

Obrázek 4.7: Využití zdrojů a úzká místa

4.2.3 Ganttovy diagramy zdrojů

Co to je Ganttův diagram a k čemu se používá už bylo popsáno v části věnované zakázkám (viz podkapitola 4.1.4). Konkrétně u zdrojů je Ganttův diagram velice užitečný, protože umožňuje přehledně zobrazit, jak je zdroj využíván a v jakém pořadí jsou mu přiřazovány zakázky. Na obrázku 4.8 jsou vykresleny Ganttovy diagramy pro zdroje s nejvyšším využitím FR1, FR2 a FR3 (viz obr. 4.7). U těchto úzkých míst je využití okolo 96% a na Ganttově diagramu (obr. 4.8) lze vidět, že na zdroji následují výrobní dávky těsně po sobě. Jediným okamžikem, kdy se nevyrábí, je krátký okamžik na začátku. Všechny výrobní dávky jsou složené ze stejných výrobků a stejných procesních kroků. Zdroj tedy provádí stejnou operaci pro všechny na stejném výrobku, pouze s tím rozdílem, že výrobky jsou součástí jiné výrobní dávky. Vyvstává otázka, jak lze problém příliš vysokého využití řešit. Jedním způsobem by mohlo být přidání dalšího stejného zdroje. Nevýhodou je, že úzká místa nejsou stálá a závisí na aktuálních zakázkách a proto další měsíc můžou být úzkým místem jiné zdroje. Dalším způsobem je dosáhnout ještě vyššího využití zdroje. V tomto případě už ovšem není příliš

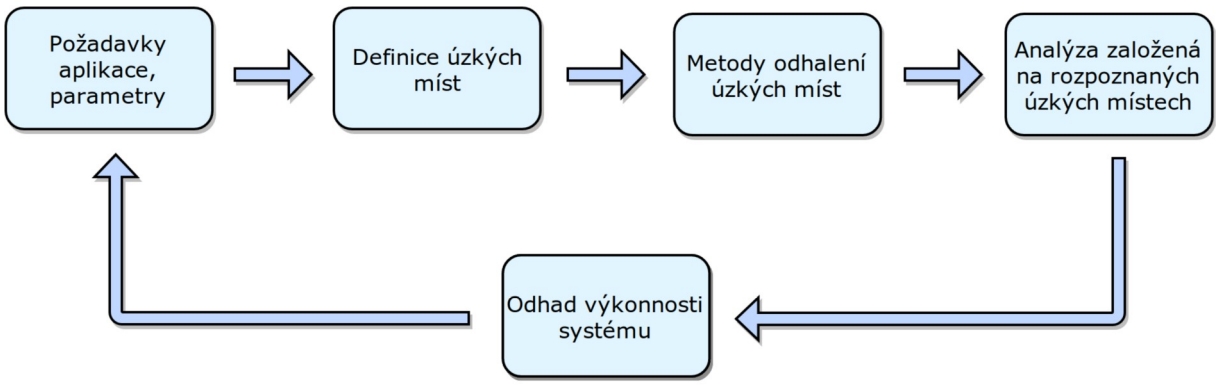
veliký prostor pro zvýšení využití. Poslední možností, která zbývá, je alespoň přeuspořádat pořadí zakázek tak, aby byly dodrženy všechny nebo co nejvíce termínů. Této problematice se věnují další části diplomové práce.



Obrázek 4.8: Ganttovy diagramy skupiny zdrojů

5 Úzká místa

Doba za kterou výrobní proces přemění surový materiál na potřebné množství konečných výrobků je závislá na mnoha faktorech. Problémovým faktorem je kapacita strojů a dostupnost materiálů. V praxi to znamená, že se v nějakém bodě systému začne hromadit velké množství součástek a naopak za tímto bodem je nedostatek potřebných materiálů a výroba často „stojí“. Důvodem k tomuto stavu může být to, že v daném bodě je nedostatečná kapacita výrobní operace. Toto stanoviště i přes dostatek potřebných materiálů není schopné vyprodukovat v potřebné době takové množství materiálu, které by následující procesy byly schopny přijmout. Dalším důvodem ke zpomalení systému v nějakém bodě může být nedostatek všech potřebných materiálů. Například pokud se jedná o operaci při které dochází k sestavování produktu ze tří různých součástek a jeden druh součástek není dodáván v dostatečném množství, zatímco ostatní druhy jsou, proces se zpomalí a začnou se hromadit součástky, kterých je dostatek. Body systému, ve kterých dochází ke zpomalení a kde se hromadí přebytky, se nazývají úzká místa (angl. bottlenecks). Typicky se jedná o jeden nebo dva stroje nebo materiály, které zpomalují celý řetězec. Pro vytvoření dobře pracujícího systému je nezbytné úzká místa systému identifikovat a následně vylepšit jejich propustnost [13].



Obrázek 5.1: Iterace zlepšování úzkých míst podle [13]

Rozpoznání a vylepšení propustnosti úzkých míst není jednoduchá úloha. Je k tomu zapotřebí množství provozních dat z výrobního procesu nebo záznamů (logů) vytvářených zařízeními během jejich činnosti. Iterační zlepšování propustnosti úzkých míst znázorňuje obrázek 5.1. Úzká místa jsou definována na základě požadavků aplikace, následně jsou nalezena a označena s využitím různých detekčních metod. Pak se použijí aproximační a asymptotické metody pro úzká místa k odhadu výkonnosti systému a k návrhu zlepšení propustnosti. Toho se následně dosáhne úpravou systémových parametrů [13].

Jednotlivé systémy se mohou vzájemně značně lišit a to stejné platí i pro úzká místa. Úzká místa mohou být statická a vyskytovat se po celou dobu trvání výrobního procesu, nebo se mohou vyskytovat pouze dočasně. Parametry a omezení systému mohou být rozdílné a uvedená definice úzkého místa by nemusela pro všechny systémy obstát. Je proto nutné

uvést ještě obecnější definici, která vyhoví všem systémům a aplikacím. Úzké místo je něco, co negativně ovlivňuje míru produkce. Ovšem to, jak přesně definovat úzké místo, záleží také na úhlu pohledu. Definice se rozdělují na dvě skupiny. První skupina se nazývá PIP z anglického Performance In Processing (výkonnost zpracovávání). Při definici založené na PIP se hledí na úzká místa z hlediska výkonu systému a úzká místa se definují na základě měření průměrných čekacích dob a kapacitní vytíženosti. Druhým způsobem jak definovat úzká místa je dívat se na ně z pohledu citlivosti. V tomto přístupu se zkoumá, u kterých strojů má jejich propustnost největší vliv na propustnost sytému. Rozhodujícím faktorem je citlivost výkonu systému na změnách parametrů jednotlivých strojů [13].

5.1 Definice úzkého místa založené na přístupu PIP

Prvním způsobem jak zjistit úzké místo je pomocí měření průměrné čekací doby. Stroj, u kterého byla naměřena nejvyšší průměrná čekací doba se považuje za úzké místo. Pokud je naměřena stejná maximální čekací doba u více strojů, tento přístup není schopen rozhodnout, který ze strojů má největší vliv, a určit tak jediné úzké místo. Takový způsob určení úzkého místa je vhodný pro systémy, které obsahují neomezený buffer. Buffer je v tomto případu chápán jako zásoba, která zajišťuje, že při nepravidelné dodávce je výroba stále plynulá.

U druhého přístupu se u jednotlivých strojů vypočítá poměr vytíženosti ku nečinnosti. Stroj, který má tento poměr nejvyšší se označí jako úzké místo. Problém se u této metody vyskytuje v tom, že více strojů má podobnou pracovní zátěž a výstupem tak zpravidla je několik úzkých míst.

5.2 Definice úzkého místa založené na citlivostním přístupu

Odlišně než PIP definuje a hledá úzká místa citlivostní přístup. Úzkým místem je podle něj stroj, který má největší vliv na celkovou propustnost systému. I zde se uplatňují různé pohledy, a to podle toho, která veličina se sleduje – zda produkce výroby nebo ziskovost. V závislosti na tom se pak vyhodnocuje úzké místo produkční nebo ekonomické.

5.2.1 Produkční úzké místo

Definic produkčních úzkých míst podle [13] je několik. V dalším textu pro ně použiji označení zavedené v literatuře [13] [18]:

- *UT-BN* … úzké místo pro dobu činnosti stroje, uptime bottleneck,
- *DT-BN* … úzké místo pro dobu prostoje stroje, downtime bottleneck,
- *c-BN* … úzké místo typu *c*, související s pracovním cyklem stroje.

Při popisu úzkých míst se pracuje s veličinou rychlost výroby ***PR*** (Production Rate), která je definována jako počet kusů vyrobených za jednotku času posledním strojem. Někdy bývá do češtiny překládána také jako produkční výkon. Je funkcí všech strojů a jejich zásobníků. Nechť množina strojů má M členů. Je-li $i \in \{1, \cdots, M\}$, pak pro i-tý stroj m_i jsou stanoveny tyto proměnné: velikost zásobníku N_i, čas pracovního cyklu c_i, doba činnosti stroje (tzv. uptime) p_i a doba prostoje stroje (downtime) r_i.

S použitím uvedených veličin se dá podle [13] zapsat rychlost výroby jako vektorová funkce takto:

$$\boldsymbol{PM} = \boldsymbol{PM}\left(p_1, r_1, \cdots, p_M, r_M, N_1, \cdots, N_{M-1}\right) \tag{5.1}$$

Nechť doba činnosti T_{up} a doba prostoje T_{down} jsou:

$$T_{up_i} = \frac{1}{p_i}; \qquad T_{down_i} = \frac{1}{r_i}; \qquad i \in \{1, \cdots, M\} \tag{5.2}$$

Pak stroj m_i je úzkým místem typu *UT-BN*, když platí

$$\frac{\partial \boldsymbol{PR}}{\partial T_{up_i}} > \frac{\partial \boldsymbol{PR}}{\partial T_{up_j}}, \qquad \forall\, j \neq i \tag{5.3}$$

a je úzkým místem typu *DT-BN*, když platí

$$\left|\frac{\partial \boldsymbol{PR}}{\partial T_{down_i}}\right| > \left|\frac{\partial \boldsymbol{PR}}{\partial T_{down_j}}\right|, \qquad \forall\, j \neq i \tag{5.4}$$

V nerovnici (5.4) je nutné použít absolutní hodnotu, protože pravá strana nerovnice je záporné číslo. Zvýšení doby prostoje totiž vede ke snížení rychlosti výroby a derivace je záporná.

Použijí-li se výše uvedené výrazy, je možno rozhodnout o vlivu daného stroje na výslednou rychlost výroby. **Stroj je úzkým místem výroby právě tehdy, když je současně jak úzkým místem typu *UT-BN*, tak i typu *DT-BN*.**

Pro některé analýzy se používají ještě další typy úzkých míst. Jedním z příkladů je úzké místo *c-BN* závislé na citlivosti stroje vzhledem k jeho pracovnímu cyklu. Stroj je úzkým místem, když platí

$$\frac{\partial \boldsymbol{PR}}{\partial c_i} > \frac{\partial \boldsymbol{PR}}{\partial c_j}, \qquad \forall\, j \neq i \tag{5.5}$$

Poznámka: V [13] jsem našel řadu formálních chyb v zápisu matematických výrazů. Chybí značení vektorových veličin v některých nerovnicích, došlo k záměně významů *m* a *M*, chybí popisy a definice několika veličin, s nimiž se pracuje apod.

5.2.2 Ekonomické úzké místo

Na rozdíl od výše uvedených produkčních úzkých míst se vyhodnocují ekonomická úzká místa podle vlivu na rentabilitu.

Podle [13] se předpokládá, že funkce nákladů vzniklých v důsledku zahlcení *F(μ)* je úměrná délce fronty u každého pracoviště. Tato funkce je definována jako dlouhodobý tok nákladů za jednotku času:

$$F(\mu)=\sum_{k=1}^{m}\frac{F_k\gamma_k}{\mu_k-\gamma_k} \tag{5.6}$$

kde

F_k ..... jednotková cena nákladů v důsledku stojící práce k-tého pracoviště nebo stroje,

γ_k ..... frekvence v počtu kusů za jednotku času na vstupu pracoviště

μ_k ..... frekvence v počtu kusů za jednotku času pracovištěm zpracovaných, tzv. kapacita

V rovnici (5.6) udává výraz $\frac{F_k\gamma_k}{\mu_k-\gamma_k}$ délku fronty *k*-tého pracoviště.

Jakmile je kterékoliv pracoviště zahlceno, okamžitě se začnou zvětšovat náklady způsobené zahlcením a děje se to úměrně délce fronty.

Ekonomické úzké místo je definováno jako pracoviště, u něhož při nepodstatném zvýšení výrobní kapacity dojde k velkému snížení nákladů způsobených zahlcením. Obecně je úzkým místem to k-té pracoviště, pro nějž platí:

$$\left|\frac{\partial F(\mu)}{\partial \mu_k}\right| > \left|\frac{\partial F(\mu)}{\partial \mu_j}\right|, \qquad \forall\, j\neq k \tag{5.7}$$

Opět je nutné v nerovnici (5.7) pracovat s absolutními hodnotami, protože poměr $F(\mu)/\mu_k$ je záporný. Připomíná to formálně podobnou situaci v nerovnici (5.4).

Úzké místo lze pak definovat jako *j*-té pracoviště, které splňuje podmínku

$$-\frac{F_k\gamma_k}{(\mu_k-\gamma_k)^2} < -\frac{F_j\gamma_j}{(\mu_j-\gamma_j)^2}, \qquad \forall\, j\neq k \tag{5.8}$$

6 Algoritmy pro řešení optimalizačního problému

Předtím než je možné začít s vytvářením algoritmů a postupů řešících problém optimalizace úzkého místa, je potřeba nejdříve položit teoretické základy, na kterých bude tvorba řešení vystavěna. V této kapitole je uveden souhrn algoritmů a principy jejich fungování. Použití těchto algoritmů je ilustrováno na problému obchodního cestujícího. V kapitole 7 je konkrétně popsáno, jak se dospěje k tomuto optimalizačnímu problému. Pro optimalizaci úzkého místa se využívá několik variant problému obchodního cestujícího a proto je potřeba nejdříve detailně popsat a vysvětlit tento problém v jeho základní podobě.

Problém obchodního cestujícího je jednou z typických úloh, která se vyskytuje při optimalizaci plánování výroby. Označuje se zkratkou TSP z anglického názvu Travelling Salesman Problem. Spočívá v hledání nejkratší cesty, která prochází všemi zadanými body v mapě.

K popisu a zkoumání úloh tohoto typu se používá teorie grafů. Obecným grafem se zde myslí uspořádaná dvojice množin vrcholů (uzlů) V a hran E.

Poznámka: Některé prameny, např. [14] definují graf jako uspořádanou *trojici* množin vrcholů V, hran E a incidenčního zobrazení A. Incidence přiřazuje každé hraně uspořádanou dvojici uzlů, každá hrana je určena právě dvěma vrcholy, které nemusejí být různé. Incidenční zobrazení je ze všech tří množin nejdůležitější.

6.1 Terminologické poznámky k teorii grafů

Teorie grafů používá řadu základních pojmů. Narazil jsem na to, že někdy bývá v literatuře použit stejný název pro pojmy s odlišným významem (např. cyklus) nebo naopak tentýž pojem bývá popsán různými názvy (uzel, vrchol). Různost v použití pojmů dokládají např. prameny [14] a [15]. Z toho pak mohou vzniknout pochybnosti a nejasnosti. Abych jim předešel, uvedu slovní označení a význam několika základních pojmů tak, jak budou použity v této práci.

Graf – zjednodušené abstraktní vyjádření situace z reálného světa, kdy zkoumaný problém se znázorní pomocí bodů (vrcholů grafu) a čar (hran grafu), které tyto body spojují. Pro matematickou definici grafu viz rovnici (6.1).

Vrchol grafu, zkráceně *vrchol* – je to bod v grafu tvořící místo, z nějž vycházejí a / nebo v němž končí hrany grafu. V grafu mohou existovat i izolované vrcholy, které nejsou spojené se žádnou hranou, např. po odebrání poslední hrany daného uzlu. Vrchol je synonymum pro *uzel*, v práci jsou použity oba pojmy.

Hrana grafu, zkráceně *hrana* – je to spojnice vrcholů v grafu. Hrana bez přívlastku je „symetrická“, tj. nemá určen směr. Hranu mezi body A a B lze chápat jako spojnici vycházející z A a končící v B, ale také naopak, vycházející z B a končící v A, přičemž žádný z obou naznačených významů není přednostní.

Orientovaná hrana – hrana vedoucí jen jedním směrem, tj. např. z A do B, ale nikoliv z B do A. V reálu může být orientovanou hranou jednosměrná ulice mezi dvěma křižovatkami, kdy křižovatky představují vrcholy grafu. Jako synonymum se v literatuře o grafech občas vyskytuje slovo *šipka*, které ale zde v práci pro pojem orientované hrany nebudu používat.

Orientovaný graf – zjednodušeně řečeno je to graf, jehož hrany jsou orientované. Nevylučuje se tím možnost, že mezi body A a B existuje obousměrná spojnice. Ta je pak vyjádřena jako dvě orientované hrany, jedna z A do B a druhá z B do A. Matice souslednosti orientovaného grafu je nesymetrická, tím se jednoznačně a na první pohled liší od symetrické matice grafu neorientovaného.

Matice souslednosti – je to čtvercová matice, jejíž prvky nabývají hodnot 0 nebo 1. Hodnota 1 je pouze v prvcích, jejichž souřadnice vyjadřují dvojici uzlů propojených hranou. Viz matematický zápis ve výrazu (6.2).

Cesta – je to konečná posloupnost vrcholů splňující podmínku, že daný vrchol je vždy spojen hranou s následujícím vrcholem v posloupnosti, při tom se žádné dva vrcholy ani hrany v posloupnosti neopakují.

Tah – je to cesta, v níž se mohou opakovat vrcholy a hrany.

Kružnice – je tvořena uzavřeným tahem, v němž počáteční vrchol je totožný s koncovým a všechny ostatní uzly jsou vzájemně různé. Jiná definice podle [14] dávající stejný výsledek říká, že kružnice je cesta, která připouští rovnost počátečního a koncového uzlu.

Cyklus – je definován shodně jako kružnice s tím rozdílem, že cyklus je orientovaný, tj. jsou orientované jeho hrany.

Hamiltonovská cesta – je to cesta procházející všemi uzly grafu. Podobně *hamiltonovská kružnice* je kružnice procházející všemi uzly grafu, na rozdíl od hamilt. cesty je uzavřená.

Hamiltonovský cyklus – je hamiltonovskou kružnicí, která je orientovaná.

6.2 Aplikace teorie grafů

Matematická definice grafu podle teorie grafů:

$$G=(V,E) \tag{6.1}$$

kde

G ..... graf

V ..... množina vrcholů grafu, *V(G)*

E ..... množina hran grafu, *E(G)*

Počet vrcholů množiny *V* je $|V|$ a počet hran množiny *E* je $|E|$.

Množina hran *E* je množinou dvojic vrcholů příslušných jednotlivým hranám. Podle toho, jestli jsou hrany neorientované nebo orientované, je i graf neorientovaný nebo orientovaný.

Konkrétní význam uzlů a hran je určen až popisem problému, jehož řešení se s pomocí teorie grafů provádí. Např. uzly představují křižovatky ulic na mapě, jimiž musí obchodní cestující projít nebo projet. Hrany jsou spojovací cesty mezi uzly, každá z hran může být charakterizována dalšími parametry – *směrem* (pak je to hrana orientovaná) a *cenou* (váhou). Cena vyjadřuje např. délku cesty nebo náklady na přepravu po této cestě.

Matematicky s využitím zmíněné teorie grafů lze úlohu TSP popsat jako hledání *hamiltonovské cesty* v orientovaném grafu s *n* uzly, z nichž počáteční je uzel v_A a konečný uzel v_B.

Uzavřená hamiltonovská cesta se nazývá *hamiltonovská kružnice;* je-li současně s tím i orientovaná, používá se pro ni název *hamiltonovský cyklus*. Pro TSP je to případ, kdy se má cestující nakonec vrátit do výchozího bodu. Řešením TSP je stanovení minima ceny hamiltonovského cyklu v hranově ohodnoceném grafu. Ovšem to je tzv. NP-úplný problém (NPC), čas potřebný k jeho optimálnímu řešení exponenciálně roste s počtem uzlů. Problém je tedy obecně neřešitelný a jako NPC patří TSP v rámci nedeterminstických polynomiálních problémů (NP) k těm obtížnějším, jejichž případné vyřešení by současně vyřešilo celou třídu podřízených problémů typu NP.

Matice souslednosti má jednotkové prvky pro existující hrany mezi uzly v_i a v_j, jinak má prvky nulové. Pro neorientovaný graf je symetrická, pro orientovaný graf nesymetrická. Má nulové prvky v hlavní diagonále.

$$X_G = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} \quad \text{kde} \quad x_{ij} = \begin{cases} 1 & \text{pro } (\mathrm{v_i}, \mathrm{v_j}) \in \mathrm{E} \\ 0 & \text{pro ostatní případy} \end{cases} \tag{6.2}$$

Matematický model úlohy TSP má několik možných vyjádření lišících se omezujícími podmínkami. Jeden z možných zápisů:

$$z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{6.3}$$

za podmínek

$$\sum_{i=1}^{n=1} x_{ij} = 1 \qquad j = 1,\ 2, \ldots,\ n \tag{6.4}$$

$$\sum_{j=1}^{n=1} x_{ij} = 1 \qquad i = 1,\ 2, \ldots,\ n \tag{6.5}$$

$$x_{ij} \in \{0,1\} \qquad i, j = 1,\ 2, \ldots,\ n \tag{6.6}$$

$$v_i - v_j + n x_{ij} \leq n - 1, \qquad i = 1,\ 2, \ldots,\ n, \quad j = 2,\ 3, \ldots,\ n, \quad \forall\ i \neq j \tag{6.7}$$

kde

- z je nákladová funkce vyjadřující cenu nebo délku cesty, její minimum se hledá;
- x_{ij} je proměnná nabývající dvou hodnot 0 a 1, hodnotu 1 má, když je uzel v_i spojen s uzlem v_j a hodnotu 0 v ostatních případech;
- d_{ij} je součinitel vyjadřující vzdálenost mezi uzlem v_i a v_j;
- v_i je pomocná proměnná obsahující číslo přiřazené i-tému uzlu
- n je počet uzlů.

Rovnice (6.4) a (6.5) stanovují podmínku, že cesta musí procházet každým uzlem právě jednou. Výraz (6.6) definuje proměnnou x jako dvoustavovou veličinu. Konečně nerovnost (6.7) zamezuje vzniku dílčích cyklů a zajišťuje jediný hlavní cyklus procházení uzlů. Jsou to tzv. Tuckerovy podmínky [16].

Nárůst obtížnosti řešení je možno doložit jednoduchým výpočtem. Pro n uzlů v orientovaném grafu, které mají být navštíveny, existuje obecně $n!$ orientovaných cest navzájem mezi nimi. Pro nalezení optima nejnižších nákladů cestovatele je třeba spočítat náklady každé z možných cest složených z hran mezi uzly. Uzlům odpovídají místa v mapě. Pro 20 uzlů je 20! přibližně $2{,}4 \cdot 10^{18}$ možných cest, ale pro 50 uzlů je to už přibližně $3 \cdot 10^{64}$. Poznámka: Orientovaný graf jsem k vysvětlení počtu kombinací použil proto, že je obecnější a více odpovídá situaci řešené dále v práci, kdy doby přeseřízení strojů „tam“ a „zpět“ nejsou symetrické a tedy ani hrany modelového grafu nemají stejnou délku pro směr tam a zpět.

Místo časově náročného hledání optima procházením všech možností se proto v praxi používají heuristiky a nedeterministické algoritmy. Ty sice nenajdou cestu s absolutně nejnižší cenou, ale zato k řešení dospějí v rozumném čase.

6.3 Prakticky používané metody řešení TSP

Metoda spočtení a porovnání všech variant cest se někdy nazývá metoda hrubé síly a jak už bylo naznačeno, lze ji použít jen pro velmi malý počet uzlů.

Existuje celá řada heuristických metod, které se v praxi využívají pro hledání suboptimálních řešení problému TSP. Nalezená řešení sice nejsou nejlepší, ale dávají dobrý výsledek blížící se optimu a metody k němu dospějí za přijatelnou dobu. Heuristika poskytuje buď způsob *nalezení* cesty nebo způsob jejího *vylepšení*. Toto rozdělení se používá proto, že některé algoritmy umožní v krátkém výpočetním čase cestu zkrátit, ale jsou velmi neefektivní, kdyby byly použity k prvotnímu vyhledání cesty. Příkladem „vylepšovacího“ typu je algoritmus prohození (křížení) hran.

V následujícím popisu vybraných algoritmů bude používán pojem hamiltonovská kružnice. Pro některé praktické úlohy se však hledá hamiltonovská (neuzavřená) cesta a nikoliv kružnice. Algoritmus lze použít stejný, jen se ve výsledku u kružnice vyškrtne ze seznamu hran ta nejdelší a tím bude převedena kružnice na cestu. Tato obecně použitelná úprava měnící algoritmus nalezení kružnice na algoritmus nalezení cesty nebude už v popisu metod zvlášť zmiňována, protože je jednoduchá a dá se použít bez dalšího vysvětlování.

6.3.1 Metoda nejbližšího souseda

Postup při použití je:

- Zvolí se výchozí uzel (vrchol) ze seznamu dosud nenavštívených uzlů. Na počátku tento seznam obsahuje všechny uzly.
- Pokračuje se k uzlu, který je nejbližší, tj. má nejnižší cenu hrany, a který dosud nebyl navštíven, tj. zatím je obsažen v seznamu. Hrana je přidána jako nový úsek cesty, nově navštívený uzel je odebrán ze seznamu.
- Opakuje se předchozí krok, cesta se prodlouží o další hranu a seznam se zmenší o další uzel. Skončí se, když je seznam uzlů prázdný. Výsledná cesta je určena navazujícími hranami v pořadí proběhlých kroků a tvoří hamiltonovskou kružnici, případně hamiltonovský cyklus, byl-li použit orientovaný graf.

Metoda je rychlá, jednoduchá, ale málo účinná co se týká kvality (délky, ceny) nalezené cesty. Zlepšení je možné opakovaným spuštěním pro jiný výchozí bod atd. až do vyčerpání všech bodů jako výchozích. Z celkového počtu *n* takto nalezených cest se vybere nejkratší (nejlevnější) [16].

6.3.2 Hladový algoritmus

Metoda pracuje s hledáním lokálních minim, vybírá v každém kroku to nejlepší. Existuje možnost, že nalezené minimum je globální. Zpracuje hodnoty všech hran grafu (na rozdíl od metody nejbližšího souseda, který pracuje jen s hranami k sousedům), zvolí nejkratší z nich a kontroluje splnění podmínek pro přidání hrany do grafu – eliminuje možnost vzniku krátkého cyklu a větvení. Algoritmus pak pokračuje další nejkratší hranou až do postupného vytvoření hamiltonovské kružnice.

6.3.3 Metoda minimální kostry

Metoda spočívá v těchto základních krocích:

- Nalezne se minimální kostra grafu.
- Všechny hrany minimální kostry se zdvojí, tím vznikne Eulerův graf. Takový graf lze nakreslit jedním tahem (Eulerův tah).
- Eulerův tah se postupně převádí na hamiltonovskou kružnici. Prochází se postupně celý Eulerův tah, dokud žádný uzel nebyl navštíven dvakrát. Má-li být uzel navštíven

podruhé, zvolí se místo něj jiný dosud nenavštívený nejbližší uzel. Tím dochází k eliminací uzlů, zkrácení celé cesty ve srovnání s původní zdvojenou kostrou a postupně vznikne hamiltonovská kružnice [16].

6.3.4 Horolezecký algoritmus

Vyznačuje se tím, že v něm jsou povoleny i kroky, které mají za následek zhoršení aktuálního řešení. V každém kroku se vybírají ty nejlepší možnosti pro sousedy podobně jako u metody nejbližšího souseda, ale rozhodnutí o ukončení algoritmu není vázáno na nalezení zlepšujícího řešení. Místo toho se kontroluje počet provedených iterací. Metoda je vhodná pro problémy s neúplným nebo chybějícím popisem [19].

7 Optimalizace úzkého místa

Problematika úzkého místa při rozvrhování výroby byla popsána v předešlých kapitolách. Tam byla pozornost věnována způsobům, jakými se identifikuje úzké místo, co to vlastně úzké místo je a co způsobuje (viz kapitola 5 a podkapitola 4.2.2). Následující část se zabývá procesem optimalizace úzkého místa a popisem programového vybavení, které jsem pro řešení této problematiky vytvořil. Popsán je zde i postup použitý při vytváření programu v programovacím jazyce Visual Basic .NET.

7.1 Úvod do problému

Jak už bylo řečeno v předešlých částech, úzkým místem výrobního procesu se rozumí bod, který negativně ovlivňuje ostatní části a celý výrobní proces. Způsobuje zpoždění, nedodržení termínů a vede k ekonomickým ztrátám. Ve většině případů je úzkým místem stroj nebo skupina strojů. Tyto stroje nestíhají produkovat dostatečné množství výrobků v požadovaném čase. Prvním krokem je určit, který stroj/stroje lze označit za úzké místo. Této problematice je věnována kapitola 5. Pokud existuje snaha výrobní proces nějakým způsobem vylepšit, je nezbytné se pokusit nějakým způsobem zlepšit právě úzké místo. Jedním z řešení může být zlepšení produktivity tím, že je zvýšena výrobní kapacita. To například znamená pořízení dalšího stroje, který bude vykonávat stejnou operaci. Nebo je také možné přiřadit na dané pracoviště více lidí, pokud je produktivita stroje závislá na počtu operátorů. Negativním dopadem těchto řešení je ovšem zvýšení nákladů na výrobu, které ve výsledku mohou způsobit pokles finančního zisku. Podrobnější rozbor rizik je popsán v podkapitolách 4.2.2 a 4.3.3 a v kapitole 5. Proto je mnohem lepší snažit se docílit optimalizace úzkého místa pouze pomocí jednoduchých změn ve výrobním procesu (např. uspořádáním pořadí zakázek). Existuje několik parametrů stroje, které lze použít pro proces optimalizace úzkého místo. Těmto parametrům a práci s nim se věnují další podkapitoly.

7.2 Matice přeseřízení

Při výrobním procesu je snaha, aby všechny stroje byly dostatečně vytížené, protože jejich nákup, provoz a údržba jsou finančně nákladné. Stroj, který by většinu času vůbec nevyráběl, je plýtváním a výrobní společnost by si jej za obvyklých podmínek vůbec nepořídila.

Většina strojů je univerzálních a to znamená, že jsou schopny vyrábět více druhů výrobků. Typický stroj tedy při výrobním procesu produkuje několik druhů výrobků. Je samozřejmě možné, že vyrábí stále tentýž výrobek, ale v takovém případě je optimalizace výroby z hlediska přseřizovacího času u takovéhoto stroje poměrně obtížná. Z tohoto všeho plyne, že úzkým místem bude ve většině případů stroj, který vyrábí několik odlišných výrobků, které jsou součástí několika zakázek.

Ve všech následujících podkapitolách se za zakázku bude považovat nějaký požadavek na zhotovení určitého množství jednoho typu výrobku. Zakázka bude určena dobou, která je

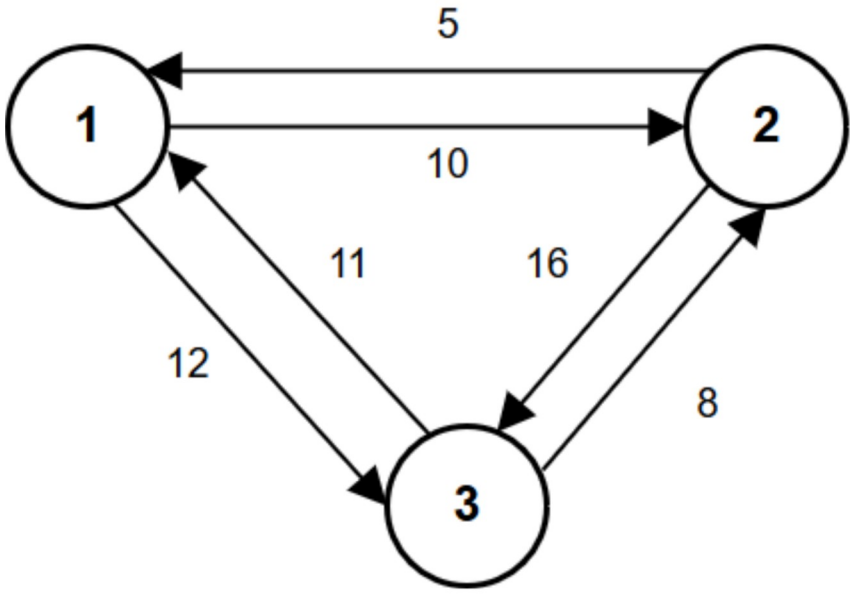
potřebná k jejímu splnění (tzn. dobou výroby) a termínem, kdy musí být dodána. Tím se budou podrobně zabývat až další podkapitoly.

Uvažujeme tedy úzké místo, kterým je stroj vyrábějící *n* typů výrobků. Úskalím ovšem je to, že není možné hned po skončení práce na jednom typu výrobků začít práci na jiném typu. Je potřeba provést úpravu nebo přenastavení stroje, což je operace, která může trvat několik minut, ale také třeba několik hodin. Tato doba upravování stroje se označuje jako doba přeseřízení. Pro každý stroj vyrábějící *n* typů výrobků je tak možné vytvořit tzv. matici přeseřízení o velikosti $n \times n$, která obsahuje všechny přeseřizovací časy mezi jednotlivými typy výrobků. Příklad takové matice je možné nalézt v tabulce 7.1, kde jednotlivé časy jsou v minutách. Ta tedy udává, že například přeseřídit stroj na kterém se vyrábí první typ výrobku na třetí typ výrobku trvá 12 minut (1 řádek, 3 sloupec). Naopak z třetího na první to bude trvat 11 minut.

Tabulka 7.1: Matice přeseřízení 3x3

Doba přeseřízení [min]	na výrobek 1. typu	na výrobek 2. typu	na výrobek 3. typu
z výrobku 1. typu	——	10	12
z výrobku 2. typu	5	——	16
z výrobku 3. typu	11	8	——

Dalším způsob, jakým lze vyjádřit matici přeseřízení nebo obecně všechny přeseřizovací časy, je orientovaný graf. Stejná matice jako v předchozí tabulce, ale znázorněná orientovaným grafem, je na obr. 7.1. Jednotlivé typy výrobků jsou reprezentovány pomocí uzlů grafu,



Obrázek 7.1: Matice přeseřízení jako orientovaný graf

přeseřizovací časy jsou představeny jako hrany spojující uzly a přeseřizovací čas je délka těchto hran.

Přeseřizovací časy mezi jednotlivými typy výrobků se liší a to znamená, že záleží na tom, v jakém pořadí se budou jednotlivé výrobky vyrábět. Úpravou pořadí, v jakém se budou vyrábět jednotlivé výrobky, lze dosáhnout snížení výrobní doby a zvýšení využití stroje. Prvním způsobem zlepšení úzkého místa je tak upravení pořadí zakázek, tím dosažení nižšího celkového přeseřizovacího času a ve výsledku zkrácení výrobní doby. Cílem je tedy najít takové pořadí výrobků, u kterého se dosáhne nejmenšího součtu všech přeseřizovacích časů. Na příkladu orientovaného grafu z obr. 7.1 se dá ukázat nalezení minimálního celkového přeseřizovacího času. Protože čas odpovídá délce hran, znamená to najít nejkratší cestu grafem, kdy každý uzel je navštíven právě jednou. Uvažuje se tak, že se vyrobí všechny potřebné kusy jednoho typu výrobku a teprve poté se začne vyrábět typ jiný. Pro orientovaný graf to znamená, že není možné navštívit uzel více než jedenkrát.

Jedna z možností jak nalézt nejkratší cestu je najít všechny možnosti a vybrat z nich tu nejkratší. Z matematického hlediska se jedná o permutaci bez opakování a to znamená, že počet možností je roven $n!$ (n – faktorial), jak už bylo uvedeno teoretické části v kapitole 6.2, zabývající se řešením problému obchodního cestujícího (TSP). Pro uvedený příklad, kdy počet typů výrobku je 3, existuje $3! = 6$ možností jak projít grafem. V příkladu uvedeném v tabulce 7.1 a obr. 7.1 je nejlepší možnost 13 minut a lze ji dosáhnout při výrobě v pořadí 3. typ, 2. typ, 1. typ. Avšak pokud nastane situace, kdy je typů výrobků mnohem více, je hledání nejlepšího řešení výpočtem a porovnáním všech možností nereálné.

Naprogramoval jsem algoritmus, který nalezne všechny možnosti a spočítá pro ně celkový přeseřizovací čas. Nalézt nejlepší řešení tímto způsobem v přiměřeném reálném čase (tzn. v řádu sekund) bylo možné pouze když byl počet typů výrobků nanejvýš 7. Pro větší přeseřizovací matice je nutné použít jiný postup.

Vše výše uvedené jednoznačně evokuje, že takto formulovaný problém vede na problém obchodního cestujícího, který byl podrobně popsán v kapitole 6. Zde popsaný problém povede na variantu TSP, kde se hledá hamiltonovská cesta. V klasickém TSP se většinou hledá hamiltonovská kružnice, to je v tomto případě nepoužitelné. Znamenalo by to, že se stroj seřídí zpět na typ výrobku, který už byl vyráběn. Pouze hledání nejkratší hamiltonovké cesty ovšem nestačí, protože problematika úzkého místa u výrobního stroje je totiž významně složitější a vstupují do ní další parametry. Popisu a řešení jsou věnovány další podkapitoly.

7.3 Termíny zakázek

Jak již bylo zmíněno v předešlé podkapitole, není možné spokojit se pouze s minimalizací přeseřizovacího času a toto řešení posléze prohlásit za nejlepší možnost pro vyřešení problému úzkého místa. Důležitý je totiž i další parametr práce stroje, a to je doba výroby. Snahou je provést optimalizaci úzkého místa, kterým je stroj, v určitém časovém úseku. V tomto časovém úseku se vyrábí několik typů výrobků. Každý typ má stanovené množství,

které je potřeba v daném úseku vyrobit. Místo množství se v celé kapitole a také v programu používá jako parametr doba výroby. Doba výroby udává čas, po který se jeden typ výrobků musí v tom daném časovém úseku vyrábět. Jaké množství se během této doby stihne vyrobit, je irelevantní, protože pro potřeby výpočtu a možnost srovnání se musí brát údaje ve stejných jednotkách. Znamená to, že množství musí být vyjádřeno jako čas, tj. dobou výroby a ne počtem kusů.

Každý typ výrobků lze označit za zakázku. Zakázka tedy sestává z požadavku na výrobu jednoho typu výrobku v určitém množství a toto množství je vyjádřeno jako doba, která je pro výrobu tohoto množství potřeba. Zde do problému vstupuje další důležitý parametr. Tímto parametrem je termín zakázky. Každá zakázka má stanovenou dobu dokdy musí být vyrobena a odeslána. Nesplnění zakázky má negativní dopad na celý výrobní proces.

Je nutné poznamenat, že v následujících příkladech bude zjednodušeně předpokládáno, že každá zakázka se skládá z právě jednoho specifického typu výrobku. Zakázka 1 z tabulky 7.2 obsahuje výrobky 1. typu podle tabulky 7.1. Zakázka 2 z tabulky 7.2 obsahuje výrobky 2. typu z tabulky 7.1 atd. Pokud se tedy vezme stroj a jeho matice přeseřízení uvedená v tab. 7.1 a doplní se o doby výroby a termíny podle tabulky 7.2, stává se problém složitějším. Pokud by se použilo stejného přístupu jako v předešlé podkapitole a za nejlepší řešení by se označilo uspořádání 3.→2.→1., kdy celkový přeseřizovací čas je 13 min, vyskytne se problém týkající se termínů zakázek. Při popsaném uspořádání dojde k tomu, že 3. zakázka je dokončena jako první za 1 hodinu, to je 2 hodiny před termínem. Druhá zakázka je vyrobena za 2 hodiny, ale musí se přičíst přeseřizovací čas z 3. zakázky na 2. a také je potřeba přičíst 1 hodinu, kterou stroj strávil výrobou zakázky 3. Příklad je pouze pro ilustraci, proto lze pro jednoduchost zanedbat přeseřizovací časy, které jsou kratší než doby výroby. Zakázka 2 je dokončena za 3 hodiny a tedy jednu hodinu před termínem. Zakázka jedna bude vyrobena za 1 hodinu, ke které se přičte čas, který stroj strávil výrobou předešlých zakázek. Zakázka jedna bude dokončena za 4 hodiny, což znamená 3 hodiny po stanoveném termínu. Uspořádáním s optimalizovaným přeseřizovacím časem se sice ušetřil čas při přeseřizování, ale došlo k velkému zpoždění zakázek. Pokud se použije uspořádání zakázek 1.→3.→2., přeseřizovací čas bude delší, ale všechny zakázky budou dokončeny v termínu.

Tabulka 7.2: Parametry zakázek

	Výroba zakázky [h] (doba za kterou se stihne zakázka vyrobit)	**Termín zakázky [h]** (doba za kterou musí být zakázka hotová)
Zakázka 1 (1. typ výrobků)	1	1
Zakázka 2 (2. typ výrobků)	2	4
Zakázka 3 (3. typ výrobků)	1	3

Na příkladu bylo ilustrováno, že optimalizace úzkého místa nezávisí pouze na jednom parametru, kterým je celkový přeseřizovací čas, ale také na tom, jaké jsou výrobní doby a termíny jednotlivých zakázek. Nejlepším řešením bude kompromis mezi těmito parametry.

7.4 Dynamická mezera

Pro řešení optimalizačního procesu bude dobré definovat několik proměnných a pojmů:

n ..... Počet zakázek

t_i ..... Termín i-té zakázky ($i = 1,..., n$) [h]

v_i ..... Doba výroby i-té zakázky ($i = 1,..., n$) [h]

b_i ..... Doba startu práce na i-té zakázce [h]

p_{ij} ..... Přeseřizovací čas z i-té na j-tou zakázku ($i \neq j \quad i = 1,\ldots,n \quad j = 1,\ldots,n$) [h]

d_i ..... Dynamická mezera ($i = 1,\ldots, n$) [h]

Dynamickou mezeru lze definovat následovně:

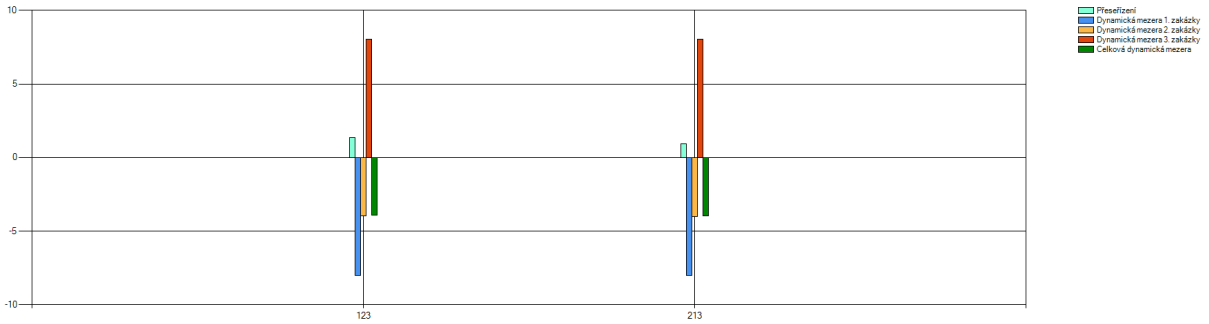
$$d_i = b_i + v_i - t_i \tag{7.1}$$

A celkovou dynamickou mezeru jako:

$$d_{celk} = \sum_{i=1}^{n} d_i \tag{7.2}$$

Dynamická mezera je parametr, který udává rozdíl mezi dobou dokončení práce na zakázce a termínem zakázky. Tak, jak je zde definována, znamená, že pokud je dynamická mezera pro zakázku i kladná, zakázka bude dokončena až po termínu – bude zpožděna. Pokud je dynamická mezera záporná, zakázka bude dokončena v předstihu a pokud je dynamická mezera rovna nule, zakázka bude dokončena přesně v termínu.

Celková dynamická mezera spočtená podle rovnice (7.2) se jeví jako parametr, který by se mohl použít pro optimalizaci úzkého místa. Ovšem u dynamické mezery nastává problém, který je patrný z obrázku 7.2. Zatímco pro obě uspořádání je celková dynamická mezera záporná, dynamická mezera 3. zakázky je kladná a to znamená, že dojde ke zpoždění 3. zakázky. I když podle celkové dynamické mezery se jeví obě kombinace jako dobré, dochází k velkým zpožděním zakázek. Bude proto třeba použít jiný parametr.



Obrázek 7.2: Graf přeseřízení a dynamických mezer

7.5 Zpoždění zakázky

Nechť je definován nový parametr z_i:

z_i ..... Zpoždění i-té zakázky ($i = 1, ..., n$) [h]

$$z_i = \begin{cases} d_i & \text{pro } d_i \geq 0 \\ 0 & \text{pro } d_i < 0 \end{cases} \tag{7.3}$$

A celkové zpoždění je definováno jako:

$$z_{celk} = \sum_{i=1}^{n} z_i \tag{7.4}$$

Takto definovaný parametr celkové zpoždění se už jeví jako lepší varianta. Zpoždění se projeví pouze v případě, že skutečně dojde ke zpoždění nějaké zakázky. Celkové zpoždění nabývá vždy pouze nezáporných hodnot. Má-li celkové zpoždění kladnou hodnotu, znamená to, že dochází ke zpožďování nějaké zakázky.

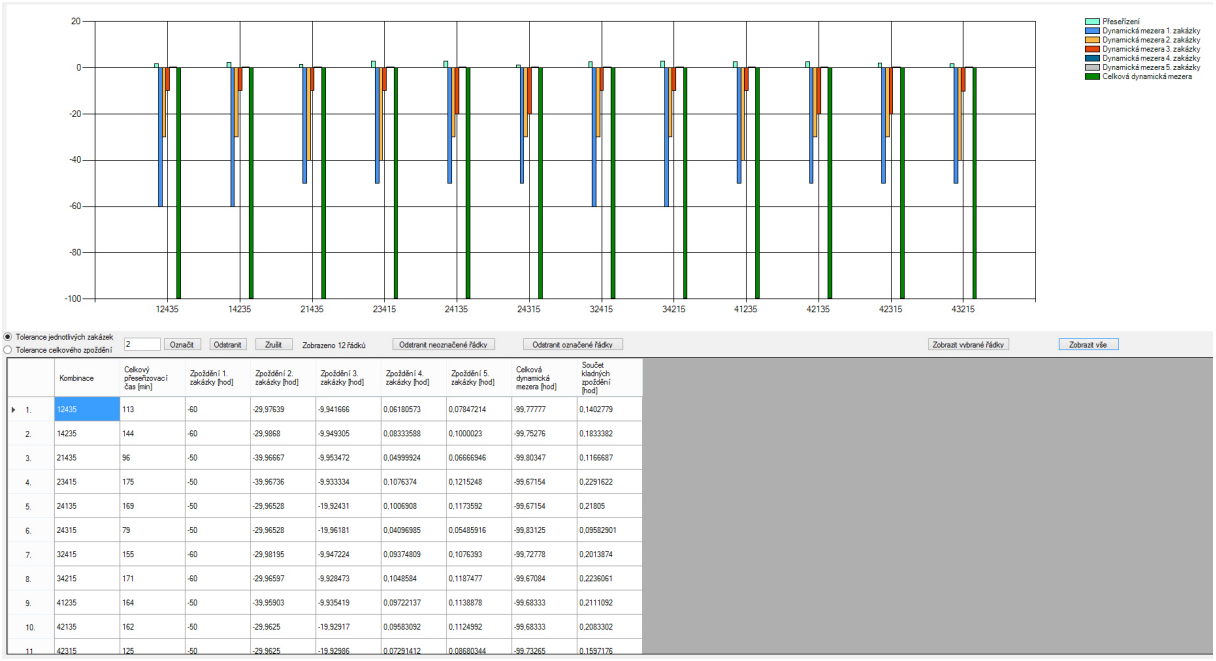
7.6 Výběr parametrů kriteriální funkce

Předtím, než je možné stanovit kriteriální funkci a začít s optimalizací, je nejdříve potřeba si definovat co je vlastně dobré řešení. V případě, kdy se zkoumal pouze přeseřizovací čas, bylo už na první pohled patrné jak dosáhnout nejlepšího řešení. Byl pouze jeden požadavek – aby celkový přeseřizovací čas byl co nejmenší. Čím menší přeseřizovací čas byl, tím bylo řešení lepší. Pokud bylo nutné porovnat dvě řešení, stačilo se podívat na dvě hodnoty a bylo možné

s jistotou rozhodnout o kvalitě řešení. Ale když se pracuje navíc také s termíny a dobou výroby zakázky, už není možné okamžitě a s jistotou rozhodnout.

Postup, který jsem se rozhodl zvolit při řešení tohoto problému je následující. Nejdříve je potřeba zkoumat jednoduché příklady, kdy je možné projít všechny možnosti, najít mezi nimi řešení, která se zdají být nejlepší, odvodit z těchto poznatků kriteriální funkci a vytvořit algoritmus, který bude tuto kriteriální funkci optimalizovat. Poté zpětně tento algoritmus použít na jednodušší případy a ověřit, zda skutečně bude vracet kvalitní řešení. Jak už bylo v předešlých podkapitolách předesláno, při řešení tohoto problému je možné vypočítat všechny hodnoty pouze při menších počtech zakázek. Nejvyšší možný počet zakázek, pro které je možné získat všechna řešení a pracovat s nimi, je roven 6. Počet všech možných uspořádání je pak 720. Následuje postup, kdy se snažím od zcela empirických zjištění dopracovat až ke konkrétním hodnotám a kriteriálním funkcím.

Prvním krokem je vygenerovat jednodušší příklad. Pokud počet zakázek je 5, program vygeneruje všech 120 možností a jejich hodnoty vloží do tabulky. Vykreslovat jednotlivé kombinace graficky při takovém množství nemá smysl. Pro lepší práci první krok musí být odstranění kombinací, které jsou jednoznačně příliš špatné. První kritérium podle, kterého je možné odfiltrovat hodnoty, je určit maximální hodnotu zpoždění a odstranit tak všechny kombinace, kde alespoň u jedné zakázky zpoždění přesáhne maximální stanovenou hodnotu. Při zvolené hodnotě tolerance zůstane už jen 12 řádků. Nyní je potřeba rozhodnout, podle kterého parametru se bude určovat, které řešení je nejlepší. V tomto případě kombinace 24315 má nejnižší celkový čas přeseřízení i celkové zpoždění (viz obr. 7.3) a rozhodnutí je tak jednoznačné. V ostatních případech je dobré se opět rozhodovat podle opět podle zpoždění, ale to pouze tehdy, pokud rozdíl mezi jednotlivými kombinacemi je velký. V příkladu se celkové zpoždění liší v řádech desítek minut a proto rozhodovacím kritériem by měl být spíš celkový přeseřizovací čas.



	Kombinace	Celkový přeseřizovací čas [min]	Zpoždění 1. zakázky [hod]	Zpoždění 2. zakázky [hod]	Zpoždění 3. zakázky [hod]	Zpoždění 4. zakázky [hod]	Zpoždění 5. zakázky [hod]	Celková dynamická mezera [hod]	Součet kladných zpoždění [hod]
1.	12435	113	-60	-29,97639	-9,941666	0,06180573	0,07847214	-99,77777	0,1402779
2.	14235	144	-60	-29,9868	-9,949305	0,08333588	0,1000023	-99,75276	0,1833382
3.	21435	96	-50	-39,96667	-9,953472	0,04999524	0,06666946	-99,80347	0,1166687
4.	23415	175	-50	-39,96736	-9,933334	0,1076374	0,1215248	-99,67154	0,2291622
5.	24135	169	-50	-29,96528	-19,92431	0,1006908	0,1173592	-99,67154	0,21805
6.	24315	79	-50	-29,96528	-19,96181	0,04096985	0,05485916	-99,83125	0,09582901
7.	32415	155	-60	-29,98195	-9,947224	0,09374809	0,1076393	-99,72778	0,2013874
8.	34215	171	-60	-29,96597	-9,928473	0,1048584	0,1187477	-99,67084	0,2236061
9.	41235	164	-50	-39,95903	-9,935419	0,09722137	0,1138878	-99,68333	0,2111092
10.	42135	162	-50	-29,9625	-19,92917	0,09583092	0,1124992	-99,68333	0,2083302
11.	42315	125	-50	-29,9625	-19,92986	0,07291412	0,08680344	-99,73265	0,1597176

Obrázek 7.3: Ukázka výběru nejlepšího řešení u jednoduchého příkladu

Z tohoto příkladu lze odvodit několik věcí. Prvním a nejdůležitějším parametrem by mělo být zpoždění. Použití celkového zpoždění je postačující a v mnoha ohledech lepší než jednotlivé zpoždění. Druhým parametrem, podle kterého posuzovat kvalitu řešení, je celkový přeseřizovací čas. Ten je však co se týče výsledné kvality řešení mnohem méně významný. Pokud by nastal případ, kdy přeseřizovací časy by byly značné, tak se to ve výsledku opět projeví na celkovém zpoždění. Když nastane situace, že je přeseřizovací čas velikostí srovnatelný s termíny, tak se v uspořádání, negativně projeví dlouhé přeseřízení tím, že zvětšuje jednotlivá zpoždění. Tyto poznatky lze shrnout tak, že v kriteriální funkci se bude pracovat se dvěma parametry: celkovým přeseřizovacím časem a celkovým zpožděním, přičemž celkové zpoždění má výrazně vyšší vliv na kvalitu řešení.

Celkové zpoždění je problematickým parametrem. Velká obtíž při použití tohoto parametru je, že v něm vystupuje proměnná b_i. Ta udává, kdy začne výroba i-té zakázky, a vypočítává se postupně. Pro první zakázku v pořadí se rovná nule. Pro druhou zakázku v pořadí se rovná součtu výrobní doby první zakázky a přeseřizovacímu času z první zakázky na druhou. Problémem je, že k získání hodnoty startu práce na zakázce je nutné vzít konkrétní kombinaci uspořádání a pro ní spočítat přeseřizovací časy atd. Pokud by se přistupovalo k tomuto problému stejně jako při hledání minimálního přeseřizovacího času a zpoždění jednotlivých zakázek se bralo jako ohodnocení uzlů grafu, nebylo by možné dospět k řešení. Důvod je, že na rozdíl od hledání minimálního přeseřizovacího času nejsou známá všechna ohodnocení už předem. Konkrétní jednotlivé zpoždění je možné získat pouze tehdy, pokud se zvolí konkrétní cesta a všechny parametry se vypočítají.

7.7 Sestavení kriteriální funkce

Následující kriteriální funkce a omezení částečně vychází z [20].

$$a_p \cdot \sum_{i \in N} \sum_{j \in N} s_{ij} \cdot x_{ij} + a_z \cdot \sum_{i \in N} z_i(b_i) \rightarrow min \tag{7.5}$$

kde:

$$\begin{gathered} \sum_{j \in N} x_{ij} = 1 \quad i \neq j \quad j = 1, \ldots, n \\ \sum_{i \in N} x_{ij} = 1 \quad i \neq j \quad i = 1, \ldots, n \\ d_i = b_i + v_i - t_i \\ z_i(b_i) = d_i \quad pro\, d_i \geq 0 \quad i = 1, \ldots, n \\ z_i(b_i) = 0 \quad pro\, d_i < 0 \quad i = 1, \ldots, n \\ b_i \geq 0 \quad i = 1, \ldots, n \\ x_{ij} \in \{0,1\} \quad i \in N \quad j \in N \end{gathered} \tag{7.6}$$

n ..... Počet zakázek [-]

t_i ..... Termín i-té zakázky ($i = 1,..., n$) [h]

v_i ..... Doba výroby i-té zakázky ($i = 1,..., n$) [h]

b_i ..... Doba startu práce na i-té zakázce [h]

s_{ij} ..... Přeseřizovací čas z i-té na j-tou zakázku ($i \neq j \quad i = 1,\ldots,n \quad j = 1,\ldots,n$) [h]

d_i ..... Dynamická mezera (i = 1,…, n) [h]

z_i ..... Zpoždění zakázky i-té zakázky (i = 1,…, n) [h]

N ..... Množina všech uzlů [-]

a_p ..... Váha přeseřízení [-]

a_z ..... Váha zpoždění [-]

7.8 Náročnost algoritmů, prostorová a časová složitost

Důležitým parametrem algoritmů použitých pro počítačové řešení úloh je jejich složitost. Je to vyjádření nároku na počet elementárních operací, které musí být provedeny nad množinou vstupních dat k vykonání algoritmu. Pro vyhodnocení složitosti algoritmu není zajímavý absolutní počet operací, ale relativní zvýšení jejich počtu při zvýšení objemu vstupních dat.

Ze složitosti algoritmu pak lze odvodit, jestli bude algoritmus na reálném počítači pracovat rychle nebo pomalu ve srovnání s algoritmem s jinou složitostí. Složitost je také mírou

efektivity vykonávání algoritmu. Při srovnání dvou různých algoritmů k řešení shodného úkolu je algoritmus spotřebovávající více operací méně efektivní.

Počet nárokovaných operací algoritmu vzhledem ke změně množství dat je obecná funkce *f(x)*, která může být lineární, ale zdaleka ne vždy tomu tak je.

Protože porovnání dvou různých průběhů složitosti vyjádřených navzájem různými obecnými funkcemi *f(x), g(x)* je obtížné, byl zaveden pojem asymptotické složitosti. Jak je zřejmé z názvu, hledá se asymptota k dané funkci, přesněji asymptotické chování funkce, a porovnávají se pak asymptoty pro velké hodnoty vstupních dat *x*. Porovnání dovolí jednoznačně rozhodnout, která z funkcí má pro velká vstupní data vyšší funkční hodnotu. Výsledek porovnání pak platí pro algoritmy, jejichž funkce se takto srovnávají. Konvence pro značení vstupních dat a zápis asymptotického chování funkce, který se používá při stanovení a porovnání složitosti, je následující:

N ..... vstupní data

O(f(N)) ..... asymptotická funkce k funkci *f(N)*

O(N) ..... zjednodušený zápis asymptotické funkce k funkci f(N)

Pro vyhodnocení algoritmů se často porovnává ne jedna, ale několik typů složitosti.

- Jeden typ určuje nárok na počet elementárních operací dané úlohy a tedy souvisí s časem potřebným k vykonání úlohy. Obvykle se označuje pojmenováním *časová složitost*.
- Druhý typ udává, jak velký nárok na objem dat algoritmus při své činnosti má. Souvisí s velikostí obsazené paměti. Označuje se jako *prostorová složitost*.
- Další hodnotí nároky na komunikaci algoritmu a používá se pro situace, kdy je algoritmus distribuovaný. Obvyklý název je *komunikační složitost*. Tou se dále nebudu zabývat, protože navržená a použitá řešení nejsou distribuovaná.

Pro asymptotickou složitost se používají tzv. třídy. Několik příkladů tříd s ustálenými názvy, seřazených od nejnižší složitosti k nejvyšší:

O(1).....konstantní

O(N).....lineární

$O(N^2)$….kvadratická

$O(K^N)$….exponenciální

K porovnání složitosti vybraných algoritmů používaných pro řešení TSP jsem sestavil tabulku s využitím údajů z [19]:

Tabulka 7.3: Přehled asymptotických složitostí jednotlivých algoritmů

Algoritmus	Časová složitost	Paměťová složitost
Genetický	$O(K^2)$	$O(2N)$
Lokální hledání	$O(K^2)$	$O(N)$
Iterativní metoda	$O(N^2)$	$O(N)$
Hladový	$O(M \log N)$	$O(N)$
Horolezecký	$O(M \log N)$	$O(N)$
Metoda zakázaného prohledávání	$O(N)$	$O(N)$
Simulované žíhání	$O(2N)$	$O(N)$
Gradientní	$O(N)$	$O(N)$

N ..... počet uzlů
M …. počet hran
K ..... konstanta

7.9 Algoritmus pro řešení optimalizačního problému

Jak už bylo zmíněno výše, pro hledání řešení při optimalizaci je možno s výhodou využít teorii související s problémem obchodního cestujícího. Uvedená optimalizace úzkého místa má vůči klasickému problému obchodního cestujícího několik specifik. Jedno z nich je, že nedochází k navrácení se do výchozího bodu, protože to by znamenalo, že jeden uzel bude navštíven více než jednou. Dalším a poměrně podstatným rozdílem je, že se při průchodu grafem využívá dvojích cen. Přeseřizovací časy jsou fixní ceny. Druhou variabilní cenou je zpoždění, které je závislé na době uplynulé od startu z výchozí pozice. Existuje několik algoritmů nebo metod, které jsou schopny tento problém řešit. Nejdříve jsem se pokusil použít k řešení jednoduchý algoritmus. Začíná se v náhodném uzlu, spočítá se hodnota kriteriální funkce ve všech ostatních uzlech a pokračuje se do uzlu, který má nejnižší hodnotu účelové funkce. Poté se opět spočítají hodnoty účelové funkce ještě nenavštívených uzlů, vybere se nejmenší a do něj se pokračuje. Toto se opakuje dokud, nebudou navštíveny všechny uzly. Jedná se tedy o algoritmus, který je popsán v podkapitole 6.4.1. Malou úpravou tohoto algoritmu je to, že se nezačíná v náhodném uzlu, ale v tom, kde je nejmenší rozdíl mezi dobou výroby a termínem zakázky. Ale takový algoritmus neposkytuje příliš dobré výsledky. Jeho slabina se projevuje až ke konci běhu. Ze začátku jsou hodnoty kriteriální funkce nízké, ale

jak ubývá nenavštívených uzlů a algoritmus má čím dál méně na výběr, hodnoty účelové funkce se začnou rapidně zhoršovat. Celkové zpoždění je pak sice vysoké, ale většina jeho hodnoty se skládá ze zpoždění několika posledních zakázek. Ještě jsem se pokusil vylepšit algoritmus zavedením náhody. S během algoritmu je pak stále klesající pravděpodobnost, že nebude vybrán uzel s nejnižší hodnotou účelové funkce. I přesto se nepodařilo dosáhnout příliš kvalitních řešení.

Nakonec jsem se rozhodl pro genetický algoritmus. Vybral jsem ho, protože s ním mám už mnoho zkušeností, vím, že poskytuje dostatečnou rychlost a k této aplikaci by měl být postačující. Hlavní slabinou genetického algoritmu, obzvláště u problému obchodního cestujícího, je, že předčasně konverguje a je velice závislý na počáteční populaci [21]. To se i potvrdilo při testování algoritmu na jednodušších příkladech. Řešením této nevýhody je zajistit kvalitní počáteční populaci. Jak ji získat je ovšem neméně obtížné. Populace, která má dobré hodnoty kriteriální funkce, může způsobit, že se algoritmus dostane do lokálního extrému. Konkrétně v tomto případě by bylo možné vybrat na základě nějakých primitivních algoritmů počáteční populaci třeba tak, že by se přihlédlo k tomu, jaké termíny a doby výroby mají jednotlivé zakázky. Ale protože i pro vysoké množství zakázek s rozumně velikou populací (např. 30) algoritmus skončí do několika vteřin, je jednodušší počáteční populaci generovat zcela náhodně a celý běh algoritmu několikrát po sobě opakovat. Tím se lze vyhnout riziku, kdy by algoritmus pro výběr počáteční populace mohl poskytovat nekvalitní hodnoty vedoucí do lokálního extrému. Náhodným výběrem populace se nezatíží hledání řešení dalšími postupy, které sice mohou zlepšit kvalitu genetického algoritmu, ale také mu mohou zabránit v nalézání kvalitních řešení.

7.10 Genetický algoritmus

Genetický algoritmus využívá principů evoluční biologie pro hledání řešení problémů. Každá implementace genetického algoritmu se liší od jiných a tento můj algoritmus není výjimkou. Skládá se z několika kroků:

7.10.1 Vytvoření počáteční generace

Prvním krokem genetického algoritmu je výběr populace na které budou prováděny evoluční mechanismy. V mém případě je vytvoření počáteční generace zcela náhodné. Uživatel zadá, jak velká populace má být, a program vygeneruje korespondující množství posloupností, kde je každá zakázka reprezentována právě jedním číslem. Pro každý prvek počáteční generace jsou spočítány hodnoty kriteriální funkce.

7.10.2 Výběr potomků

Poté, co jsou prvky v populaci seřazeny podle kriteriální funkce, následuje vytvoření potomků, kteří budou tvořit další generaci. Několik nejlepších jedinců je přímo zkopírováno

do další generace. Pak, když jsou spočítány hodnoty kriteriální funkce všech členů populace, dojde k uspořádání podle této hodnoty. Několik nejlepších prvků se vezme a vloží se nezměněné do další generace. Pokud by byly pro vytvoření nových prvků pozměněny, mohlo by dojít ke ztrátě dobrého řešení.

7.10.3 Křížení

Z několika nejlepších jedinců jsou vytvořeny páry rodičů, ze kterých se tvoří jejich potomci. U genetických algoritmů probíhá křížení tak, že si oba rodiče vzájemně vymění svoje části. U problému, který je zde řešen, není možné libovolně měnit jednotlivé části dvou rodičů. Prvek populace je posloupnost čísel, která představuje posloupnost zakázek. Proto pokud by došlo k libovolné výměně, mohl by vzniknout prvek, který bude obsahovat některé zakázky vícekrát a jiné vůbec. Způsob, jakým probíhá křížení, je následující. Jedná se o upravený postup z [22].

Vezmou se dva rodiče:

$R_1 = 3\ 4\ 6\ 2\ 1\ 5$
$R_2 = 6\ 2\ 1\ 5\ 4\ 3$

Oba rodiče jsou rozděleni na tři části pomocí dvou řezů.

$R_1 = 3\ 4 \mid 6\ 2\ 1 \mid 5$
$R_2 = 6\ 2 \mid 1\ 5\ 4 \mid 3$

Části ohraničené dvěma řezy se vymění a zbylé prvky se, pokud je to možné, doplní z rodičů.

$P_1 = 3\ \mathrm{X} \mid 1\ 5\ 4 \mid \mathrm{X}$
$P_2 = \mathrm{X}\ \mathrm{X} \mid 6\ 2\ 1 \mid 3$

Nevyplněné prvky se vyplní zbývající možnými prvky.

$P_1 = 3\ 2\ 1\ 5\ 4\ 6$
$P_2 = 4\ 5\ 6\ 2\ 1\ 3$

7.10.4 Mutace

U genetického algoritmu, stejně jako u mnoha dalších, se vyskytuje riziko, že řešení uvízne v lokálním extrému. Proto je potřeba zachovat v populaci rozmanitost. Prostředek, kterým se toho docílí, je u genetického algoritmu mutace. Mutace znamená, že se vezme prvek a pro každou pozici (zakázku) se podle náhody určuje, jestli dojde k jeho změně. Změna se děje vždy tak, aby na konci vznikl prvek, který splňuje všechny náležitosti (tzn. obsahuje každou zakázku právě jednou).

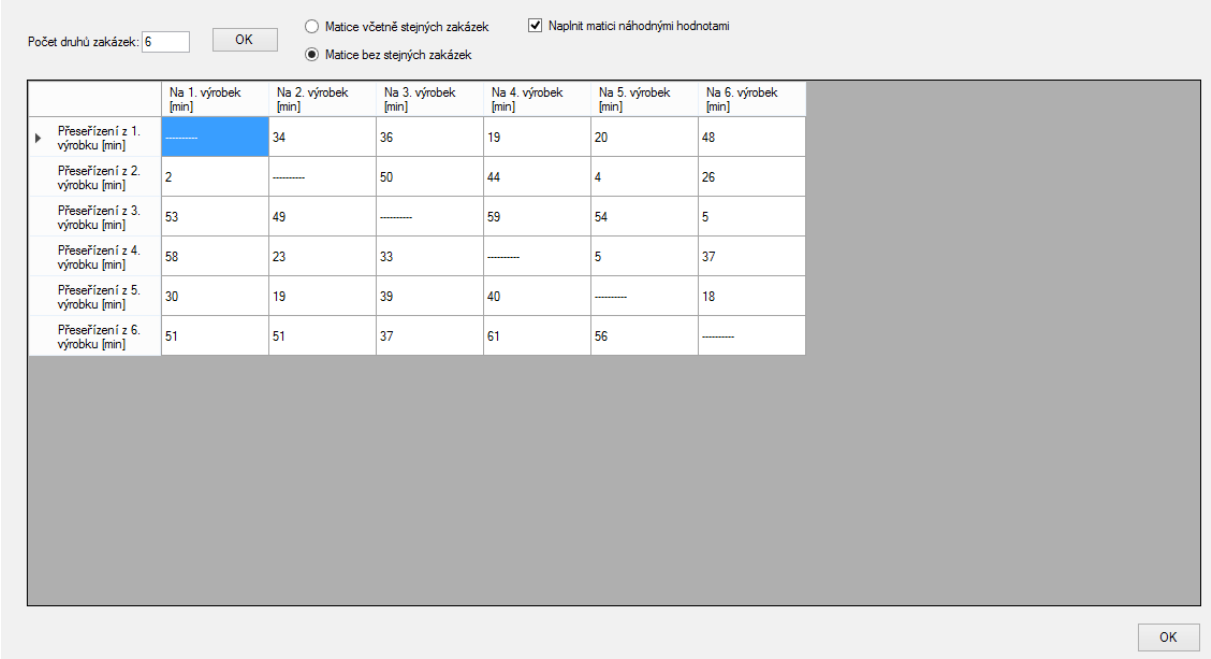
7.10.5 Běh algoritmu

Běh genetického algoritmu začíná výběrem počáteční populace. Podle hodnoty kriteriální funkce se prvky populace seřadí. Několik nejlepších prvků, tzn. prvky s nejnižší hodnotou kriteriální funkce, se nezměněné přenesou do další generace. Z části nejlepších prvků se vytvoří dvojice, které se vzájemně kříží. U zbylých prvků se provede mutace, která zajišťuje rozmanitost a pomáhá, aby algoritmus neskončil v lokálním extrému. Tímto postupem se vytvoří nová generace prvků. Tyto prvky jsou opět uspořádány a celý postup se opakuje. Zastavení genetického algoritmu proběhne tak, že pokud má nejlepší prvek nové generace stejnou hodnotu účelové funkce jako nejlepší prvek generace předešlé, hodnota čítače se zvýší o 1. Pokud se podaří najít lepší prvek, čítač se vynuluje. Maximální hodnota čítače je nastavena před během algoritmu. Pokud se hodnota kriteriální funkce nejlepšího prvku nemění a hodnota čítače dosáhne maximální hodnoty, algoritmus se zastaví.

7.10.6 Test algoritmu

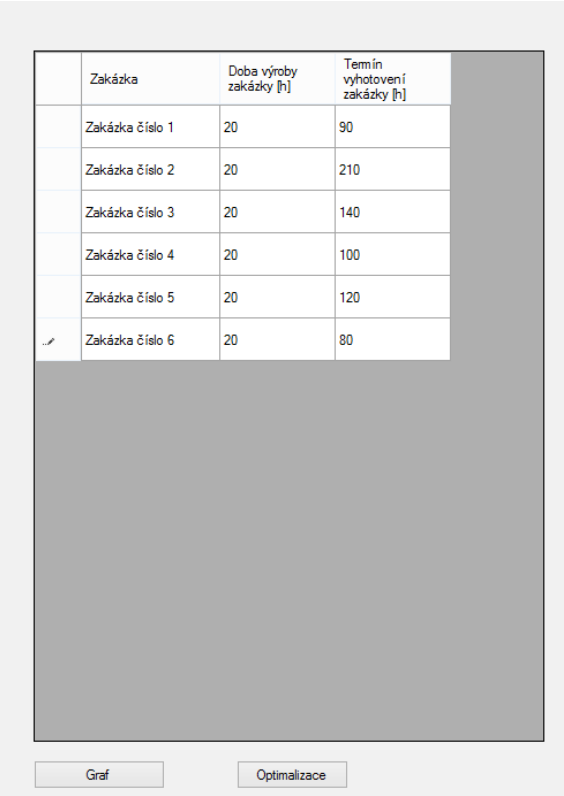
Následujícím krokem je kontrola, zda je genetický algoritmus schopný dostatečně rychle nalézt dostatečně dobré řešení. Jak už bylo napsáno výše, prozkoumat všechny možnosti pro dané zadání je reálné pokud je maximálně 6 typů výrobků. Při takto nízkém počtu bude nezbytné, aby genetický algoritmus nalezl optimální řešení.

Hodnoty přeseřízení pro zkušební příklad byly vygenerovány náhodně a jsou v obr. 7.4. Doba výroby a termíny jsou v obrázku 7.5.



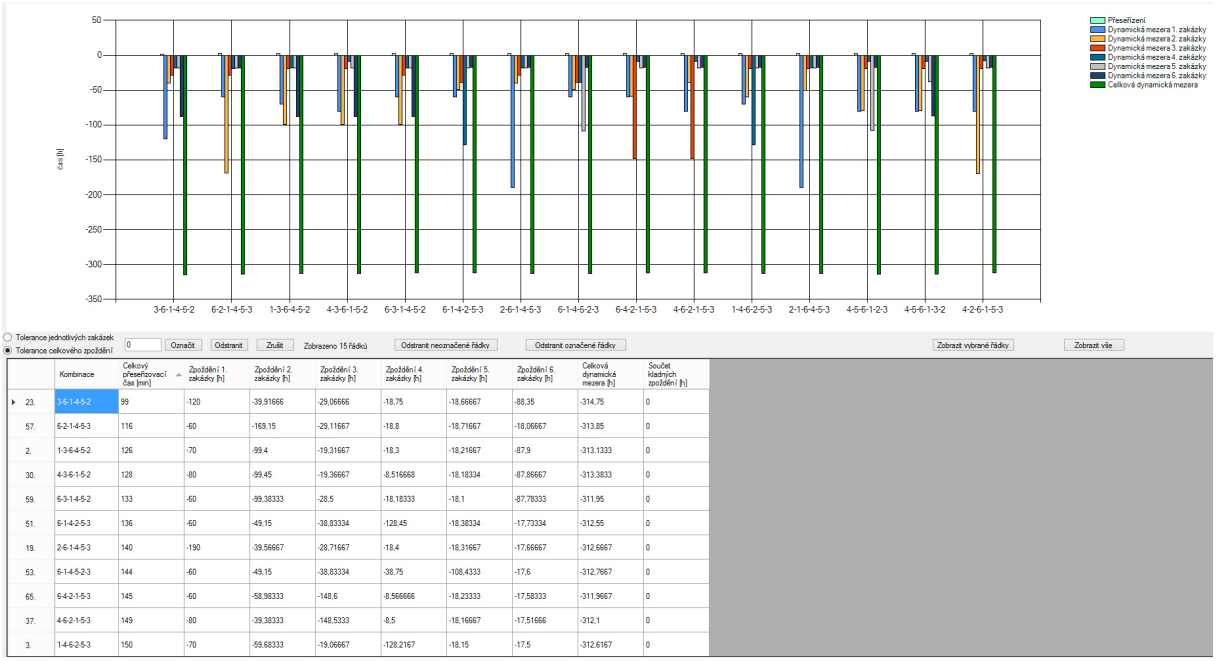
	Na 1. výrobek [min]	Na 2. výrobek [min]	Na 3. výrobek [min]	Na 4. výrobek [min]	Na 5. výrobek [min]	Na 6. výrobek [min]
Přeseřízení z 1. výrobku [min]	---------	34	36	19	20	48
Přeseřízení z 2. výrobku [min]	2	---------	50	44	4	26
Přeseřízení z 3. výrobku [min]	53	49	---------	59	54	5
Přeseřízení z 4. výrobku [min]	58	23	33	---------	5	37
Přeseřízení z 5. výrobku [min]	30	19	39	40	---------	18
Přeseřízení z 6. výrobku [min]	51	51	37	61	56	---------

Obrázek 7.4: Matice přeseřízení ukázkového příkladu



Obrázek 7.5: Doby výroby a termíny pro ukázkový příkladu

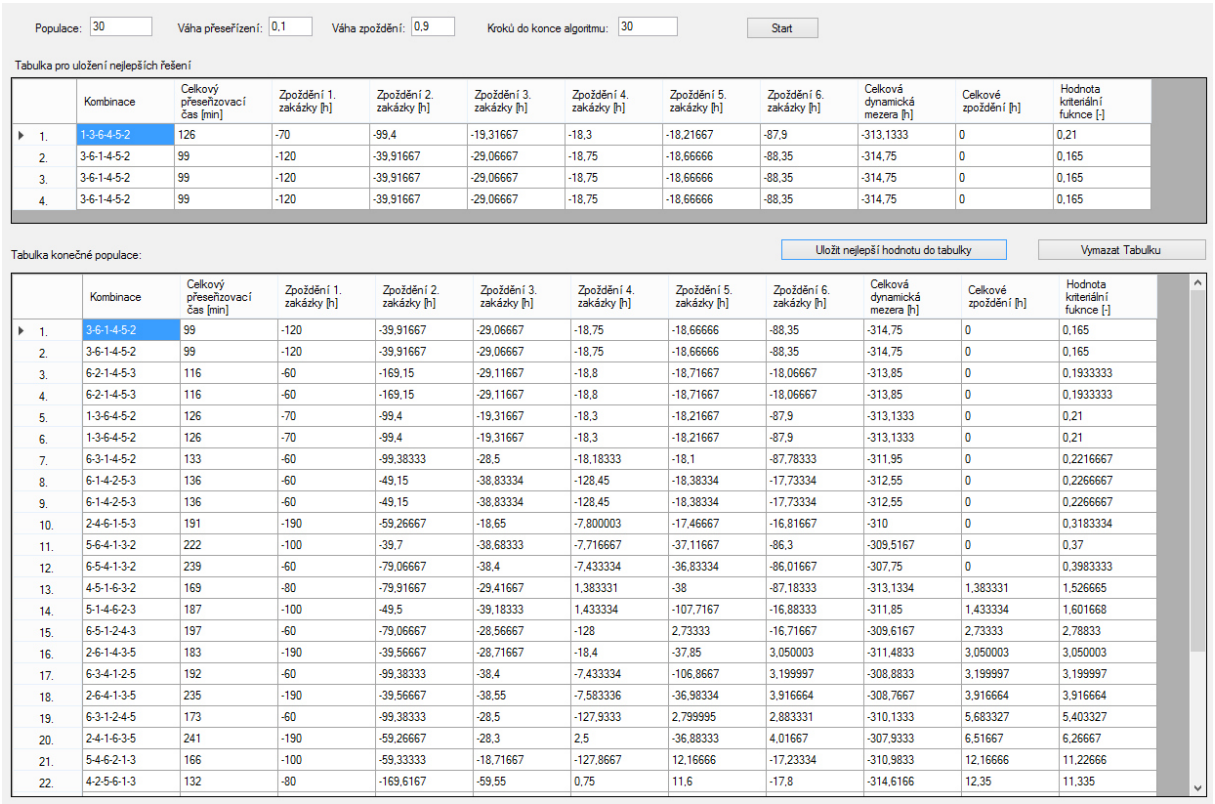
U takto zadaných hodnot existuje celkem 720 možností, jak lze jednotlivé zakázky uspořádat. Aby bylo možné určit optimální řešení, je potřeba nejdříve odfiltrovat řešení, která jednoznačně nevyhovují. Prvním kritériem je zpoždění. Je třeba uspořádat zakázky tak, aby nedocházelo ke zpoždění a žádná zakázka tak nebyla dokončena pozdě. Kombinací zakázek, u kterých nedochází ke zpoždění, je celkem 72. Druhým kritériem je celkový přeseřizovací čas. Všech zbylých 72 možností se seřadí podle velikosti celkového přeseřizovacího času. Kombinace, která má nejnižší celkový přeseřizovací čas je optimálním řešením. Optimálním řešením je 3-6-1-4-5-2, kdy celkový přeseřizovací čas je 99 minut. Nejlepších 15 řešení je zobrazeno na obr. 7.6.



	Kombinace	Celkový přeseřizovací čas [min]	Zpoždění 1. zakázky [h]	Zpoždění 2. zakázky [h]	Zpoždění 3. zakázky [h]	Zpoždění 4. zakázky [h]	Zpoždění 5. zakázky [h]	Zpoždění 6. zakázky [h]	Celková dynamická mezera [h]	Součet kladných zpoždění [h]
23.	3-6-1-4-5-2	99	-120	-39,91666	-29,06666	-18,75	-18,66667	-88,35	-314,75	0
57.	6-2-1-4-5-3	116	-60	-169,15	-29,11667	-18,8	-18,71667	-18,06667	-313,85	0
2.	1-3-6-4-5-2	126	-70	-99,4	-19,31667	-18,3	-18,21667	-87,9	-313,1333	0
30.	4-3-6-1-5-2	128	-80	-99,45	-19,36667	-8,516668	-18,18334	-87,86667	-313,3833	0
59.	6-3-1-4-5-2	133	-60	-99,38333	-28,5	-18,18333	-18,1	-87,78333	-311,95	0
51.	6-1-4-2-5-3	136	-60	-49,15	-38,83334	-128,45	-18,38334	-17,73334	-312,55	0
19.	2-6-1-4-5-3	140	-190	-39,56667	-28,71667	-18,4	-18,31667	-17,66667	-312,6667	0
53.	6-1-4-5-2-3	144	-60	-49,15	-38,83334	-38,75	-108,4333	-17,6	-312,7667	0
65.	6-4-2-1-5-3	145	-60	-58,98333	-148,6	-8,566666	-18,23333	-17,58333	-311,9667	0
37.	4-6-2-1-5-3	149	-80	-39,38333	-148,5333	-8,5	-18,16667	-17,51666	-312,1	0
3.	1-4-6-2-5-3	150	-70	-59,68333	-19,06667	-128,2167	-18,15	-17,5	-312,6167	0

Obrázek 7.6: Grafické zobrazení nejlepších řešení ukázkového příkladu

Dalším krokem je použití genetického algoritmu. Počáteční populaci genetického algoritmu jsem nastavil na 30. Pokud algoritmus po 30-ti po sobě jdoucích krocích nenalezne lepší řešení, zastaví se. Důležitým krokem je nastavení vah. To, jak jsou nastaveny, určí, jak velký vliv má který parametr na optimální řešení. Váha přeseřízení $a_p = 0,1$ a váha zpoždění $a_z = 0,9$. Mnohem větší důležitost tedy bude přikládána tomu, aby zpoždění bylo co nejmenší. Takto nastavený algoritmus byl spuštěn čtyřikrát. Počáteční populace je vybírána zcela náhodně a tento výběr ovlivní konečné řešení. Proto je lepší spustit algoritmus opakovaně. Ze čtyř pokusů algoritmus našel optimální řešení třikrát. Vše je patrné z obr. 7.7. V horní tabulce jsou uloženy nejlepší hodnoty každého pokusu. Spodní tabulka obsahuje všechny členy konečné populace jednoho z pokusů. Algoritmus jsem spustil ještě dvacetkrát a optimální řešení bylo nalezeno v 10 případech. Při upravených váhách $a_p = 0,3$ a $a_z = 0,7$ algoritmus nalezl optimální řešení v 15 z 20 pokusů. Aby bylo možné říct, že druhé nastavení vah je lepší, bylo by potřeba provést mnohem více pokusů. Každopádně nastavení vah ovlivňuje konečné řešení. Při nastavení váhy zpoždění na nulu lze genetický algoritmus použít pro hledání minimálního přeseřizovacího času a řešit tak jednodušší variantu problému obchodního cestujícího. Schopnost genetického algoritmu nalézt optimální řešení závisí samozřejmě také na dalších parametrech, jako je velikost populace nebo nastavení hranice pro zastavení algoritmu.



	Kombinace	Celkový přeseřizovací čas [min]	Zpoždění 1. zakázky [h]	Zpoždění 2. zakázky [h]	Zpoždění 3. zakázky [h]	Zpoždění 4. zakázky [h]	Zpoždění 5. zakázky [h]	Zpoždění 6. zakázky [h]	Celková dynamická mezera [h]	Celkové zpoždění [h]	Hodnota kriteriální fuknce [-]
1.	1-3-6-4-5-2	126	-70	-99,4	-19,31667	-18,3	-18,21667	-87,9	-313,1333	0	0,21
2.	3-6-1-4-5-2	99	-120	-39,91667	-29,06667	-18,75	-18,66666	-88,35	-314,75	0	0,165
3.	3-6-1-4-5-2	99	-120	-39,91667	-29,06667	-18,75	-18,66666	-88,35	-314,75	0	0,165
4.	3-6-1-4-5-2	99	-120	-39,91667	-29,06667	-18,75	-18,66666	-88,35	-314,75	0	0,165

	Kombinace	Celkový přeseřizovací čas [min]	Zpoždění 1. zakázky [h]	Zpoždění 2. zakázky [h]	Zpoždění 3. zakázky [h]	Zpoždění 4. zakázky [h]	Zpoždění 5. zakázky [h]	Zpoždění 6. zakázky [h]	Celková dynamická mezera [h]	Celkové zpoždění [h]	Hodnota kriteriální fuknce [-]
1.	3-6-1-4-5-2	99	-120	-39,91667	-29,06667	-18,75	-18,66666	-88,35	-314,75	0	0,165
2.	3-6-1-4-5-2	99	-120	-39,91667	-29,06667	-18,75	-18,66666	-88,35	-314,75	0	0,165
3.	6-2-1-4-5-3	116	-60	-169,15	-29,11667	-18,8	-18,71667	-18,06667	-313,85	0	0,1933333
4.	6-2-1-4-5-3	116	-60	-169,15	-29,11667	-18,8	-18,71667	-18,06667	-313,85	0	0,1933333
5.	1-3-6-4-5-2	126	-70	-99,4	-19,31667	-18,3	-18,21667	-87,9	-313,1333	0	0,21
6.	1-3-6-4-5-2	126	-70	-99,4	-19,31667	-18,3	-18,21667	-87,9	-313,1333	0	0,21
7.	6-3-1-4-5-2	133	-60	-99,38333	-28,5	-18,18333	-18,1	-87,78333	-311,95	0	0,2216667
8.	6-1-4-2-5-3	136	-60	-49,15	-38,83334	-128,45	-18,38334	-17,73334	-312,55	0	0,2266667
9.	6-1-4-2-5-3	136	-60	-49,15	-38,83334	-128,45	-18,38334	-17,73334	-312,55	0	0,2266667
10.	2-4-6-1-5-3	191	-190	-59,26667	-18,65	-7,800003	-17,46667	-16,81667	-310	0	0,3183334
11.	5-6-4-1-3-2	222	-100	-39,7	-38,68333	-7,716667	-37,11667	-86,3	-309,5167	0	0,37
12.	6-5-4-1-3-2	239	-60	-79,06667	-38,4	-7,433334	-36,83334	-86,01667	-307,75	0	0,3983333
13.	4-5-1-6-3-2	169	-80	-79,91667	-29,41667	1,383331	-38	-87,18333	-313,1334	1,383331	1,526665
14.	5-1-4-6-2-3	187	-100	-49,5	-39,18333	1,433334	-107,7167	-16,88333	-311,85	1,433334	1,601668
15.	6-5-1-2-4-3	197	-60	-79,06667	-28,56667	-128	2,73333	-16,71667	-309,6167	2,73333	2,78833
16.	2-6-1-4-3-5	183	-190	-39,56667	-28,71667	-18,4	-37,85	3,050003	-311,4833	3,050003	3,050003
17.	6-3-4-1-2-5	192	-60	-99,38333	-38,4	-7,433334	-106,8667	3,199997	-308,8833	3,199997	3,199997
18.	2-6-4-1-3-5	235	-190	-39,56667	-38,55	-7,583336	-36,98334	3,916664	-308,7667	3,916664	3,916664
19.	6-3-1-2-4-5	173	-60	-99,38333	-28,5	-127,9333	2,799995	2,883331	-310,1333	5,683327	5,403327
20.	2-4-1-6-3-5	241	-190	-59,26667	-28,3	2,5	-36,88333	4,01667	-307,9333	6,51667	6,26667
21.	5-4-6-2-1-3	166	-100	-59,33333	-18,71667	-127,8667	12,16666	-17,23334	-310,9833	12,16666	11,22666
22.	4-2-5-6-1-3	132	-80	-169,6167	-59,55	0,75	11,6	-17,8	-314,6166	12,35	11,335

Obrázek 7.7: Řešení ukázkového příkladu získané genetickým algoritmem

7.10.7 Zavedení priority a tolerance zpoždění

V předešlých částech byly představeny parametry, jejichž pomocí se hodnotí kvalita navrhované posloupnosti zakázek. Za důležitější parametr se pokládá zpoždění, což je pochopitelné, protože cílem podniku je dosáhnout maximalizace zisků a pro to je potřeba vyhotovovat zakázky a snažit se, aby byly dokončeny všechny včas a předejít tak zpožděním. Snaha dosáhnout co nejvyššího zisku vede k tomu, že do výrobního procesu vstupuje největší možné množství zakázek. Dochází tak tomu, že výrobní proces je u hranice výrobních kapacit. Proto není ve výrobním procesu mnoho rezerv. Ve výrobním procesu jsou vždy nevyhnutelné nepředvídatelné události, které ve výsledku budou znamenat, že nebude zbývat dostatek času vše včas dokončit. Primárně je tedy snaha dokončit všechny zakázky bez zpoždění, ale nastane stav, kdy bude potřeba volit, které zakázky budou dokončeny včas a které ne. Z tohoto důvodu vstupuje do výpočtů další parametr. Tím je priorita zakázek. Priorita určuje důležitost zakázek. Větší důležitost některé zakázky například může být dána na základě smlouvy se zákazníkem. Ve smlouvě může být zaneseno, že pokud nebude žádané zboží dodané včas, cena kterou zákazník zaplatí se sníží v závislosti na velikosti zpoždění. Při rozhodování o tom, v jakém pořadí se mají zakázky vyrábět, tak bude záležet v první řadě na prioritě. Vyšší priorita neznamená, že taková zakázka bude vždy přednostní. Zakázka s vyšší prioritou se bude vyrábět dříve než méně prioritní zakázky pouze tehdy, pokud by hrozilo, že pokud přednost nedostane, bude zpožděna. To je patrné z následující rovnice (7.7), kde

priorita zakázky násobí zpoždění zakázky. Pokud je zakázka dokončena s předstihem nebo v přesně stanovený čas, zpoždění se rovná nule a priorita tak nehraje roli.

Termín zakázky udává, kdy je požadováno, aby zakázky bylo hotová. Termín se většinou určuje s přesností na hodiny. Stav, kdy dojde ke zpoždění zakázky, je nežádoucí, ale když je například zakázka dokončena se zpožděním 15 minut, neznamená to, že je potřeba tento stav nějakým způsobem penalizovat. Proto se zavádí parametr tolerance (může být také označen jako dovolená odchylka). Tolerance zpoždění udává, že může dojít ke zpoždění zakázek o nějakou dobu a přesto to nebude mít vliv. Pokud je zpoždění zakázky menší než povolená odchylka, vliv na hodnotu kriteriální funkce je nulový, přestože nějaké zpoždění vzniklo a přestože váhový koeficient zpoždění a_z má větší váhu, než koeficient použitý pro přeseřízení.

$$a_p \cdot \sum_{i \in N} \sum_{j \in N} s_{ij} \cdot x_{ij} + a_z \cdot \sum_{i \in N} p_i \cdot [z_i(b_i) - u] \rightarrow min \tag{7.7}$$

$$[z_i(b_i) - u] \geq 0 \tag{7.8}$$

$$\begin{gathered} \sum_{j \in N} x_{ij} = 1 \quad i \neq j \quad j = 1, \ldots, n \\ \sum_{i \in N} x_{ij} = 1 \quad i \neq j \quad i = 1, \ldots, n \\ d_i = b_i + v_i - t_i \\ z_i(b_i) = d_i \quad pro\, d_i \geq 0 \quad i = 1, \ldots, n \\ z_i(b_i) = 0 \quad pro\, d_i < 0 \quad i = 1, \ldots, n \\ b_i \geq 0 \quad i = 1, \ldots, n \\ x_{ij} \in \{0,1\} \quad i \in N \quad j \in N \end{gathered} \tag{7.9}$$

kde

u ..... Povolená odchylka zpoždění [h]

p_i ..... Priorita i-té zakázky ($i = 1, \ldots, n$) [-]

n ..... Počet zakázek [-]

t_i ..... Termín i-té zakázky ($i = 1, \ldots, n$) [h]

v_i ..... Doba výroby i-té zakázky ($i = 1, \ldots, n$) [h]

b_i ..... Doba startu práce na i-té zakázce [h]

s_{ij} ..... Přeseřizovací čas z i-té na j-tou zakázku ($i \neq j \quad i = 1, \ldots, n \quad j = 1, \ldots, n$) [h]

d_i ..... Dynamická mezera (i = 1,…, n) [h]

z_i ..... Zpoždění zakázky i-té zakázky (i = 1,…, n) [h]

N ..... Množina všech uzlů [-]

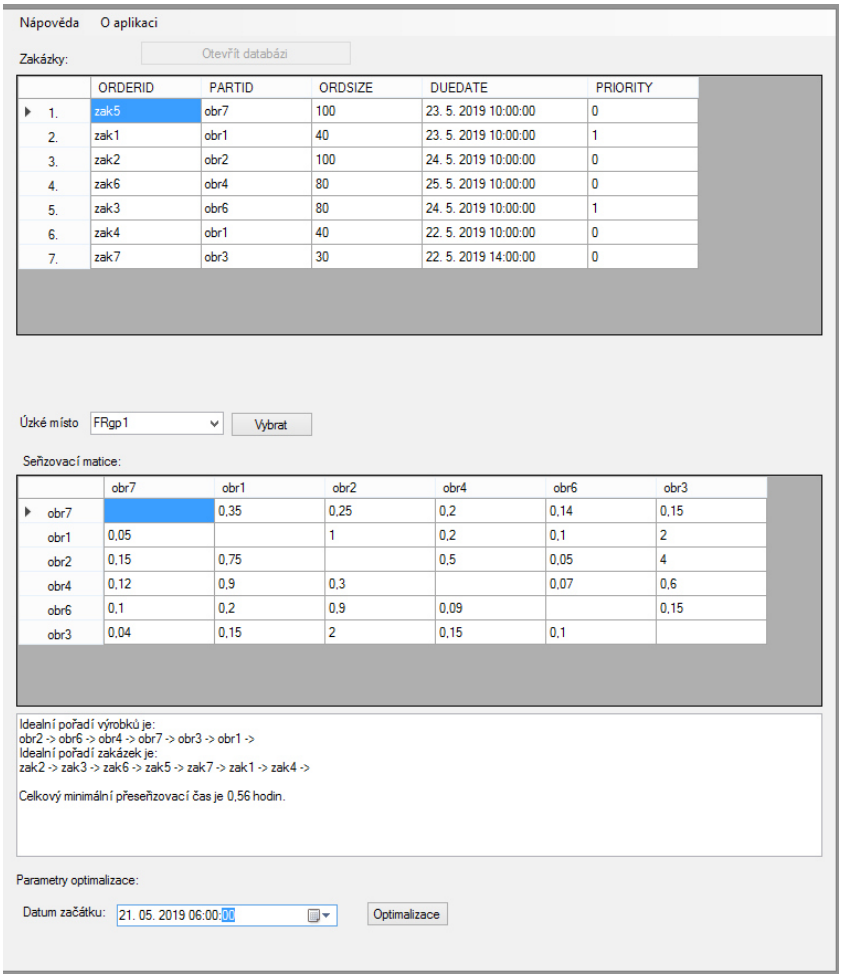
a_p ..... Váha přeseřízení [-]

a_z ..... Váha zpoždění [-]

7.10.8 Upravení vstupu programu

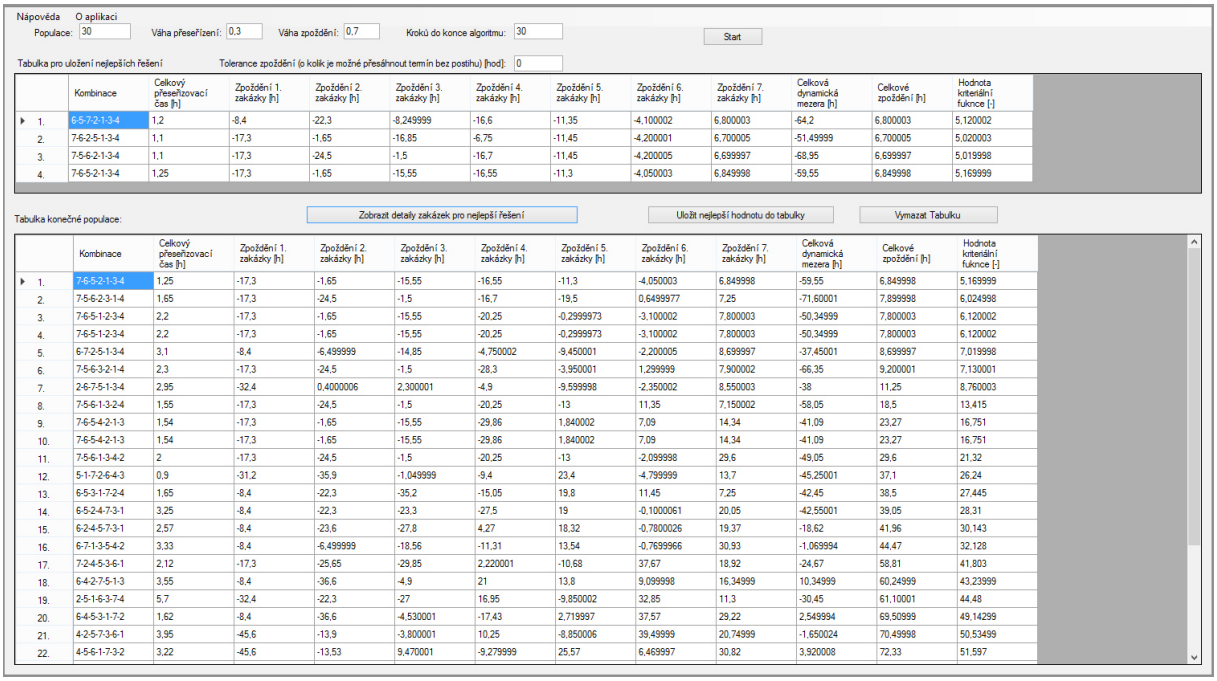
Pro předešlé příklady, kdy bylo testováno fungování genetického algoritmu, probíhalo zadávání zakázek a přeseřizovacích časů ručně. Ve výrobním procesu je samozřejmě nemyslitelné, že by bylo pro rozvrhování posloupností zakázek nutné zadávat všechno zvlášť ručně. Ve firmách se využívají informační systémy, které obsahují velké množství informací z různých oblastí řízení podniku. Pokud přijde do firmy zakázka, následně je pouze jednou zadána do systému a informace o ní jsou sdílené se všemi potřebnými stanovišti. Všechna data se musí nějakým způsobem ukládat. Jeden ze způsobů je ukládání dat do databází. Pokud tedy, jak už bylo řečeno, přijde podniku nějaká zakázka, údaje o zakázce se buď automaticky vygenerují nebo je zadá do informačního systému nějaký člověk. Informační systém poté vygeneruje nový záznam do databáze. Aby byl program skutečně využitelný, je nutná jeho úprava tak, aby byl schopen načítat všechna data o zakázkách, výrobcích, postupech apod. z databáze.

Na obrázku 7.8 je skupina zakázek, které byly načteny ze zkušební databáze. Záznam pro každou zakázku obsahuje označení zakázky (sloupec ORDERID), označení výrobku (sloupec PARTID), množství výrobků (sloupec ORDSIZE), termín zakázky (sloupec DUEDATE) a prioritu (sloupec PRIORITY). Zakázky zak1 a zak3 mají prioritu 1 a mají tedy vyšší důležitost než zbylé zakázky. Úzkým místem, tedy strojem pro který se bude provádět optimalizace, je FRgp1. Program z databáze vygeneroval přeseřizovací tabulku všech výrobků, které jsou obsaženy v zakázkách. Údaje o přeseřízení jsou v hodinách. Pomocí výpočtu všech možností je zjištěna posloupnost zakázek, kdy je celkový přeseřizovací čas nejmenší. Minimální přeseřizovací čas je 0,56 minut. Jako poslední je zadáno datum a hodina, kdy se začne pracovat na této skupině zakázek.



Obrázek 7.8: Zobrazení zakázek a přeseřizovací tabulky z databáze

Následující částí je samotná optimalizace, tedy aplikace genetického algoritmu. Na obrázku 7.9 je nastavení jednotlivých parametrů genetického algoritmu. Populace algoritmu byla nastavena na 30, váha přeseřízení na 0,3, váha zpoždění na 0,7 a počet kroků, po nichž se ukončí běh algoritmu v případě, že kroky nezlepší hodnotu kriteriální funkce, je roven 30. Každé uspořádání zakázek je reprezentováno posloupností čísel, v níž jednotlivá čísla odpovídají řádkům v tabulce zakázek. Do horní tabulky byly uloženy čtyři nejlepší řešení. Tato řešení byla získána opakovaným během algoritmu. Spodní tabulka obsahuje konečnou populaci.



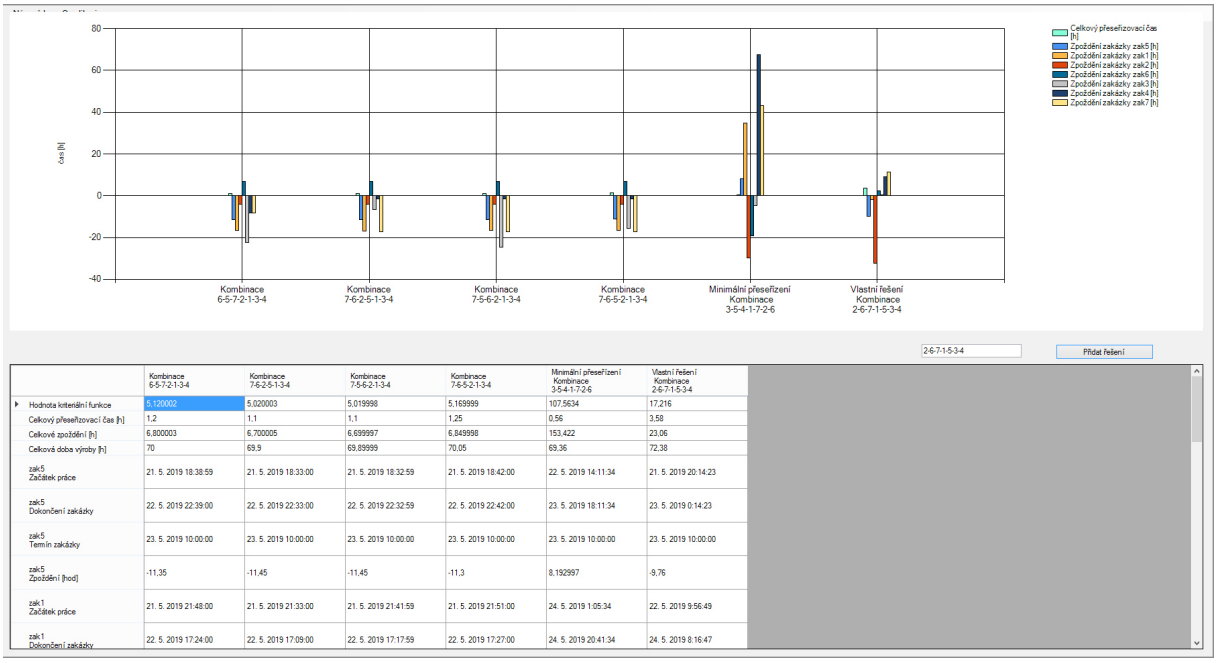
	Kombinace	Celkový přeseřizovací čas [h]	Zpoždění 1. zakázky [h]	Zpoždění 2. zakázky [h]	Zpoždění 3. zakázky [h]	Zpoždění 4. zakázky [h]	Zpoždění 5. zakázky [h]	Zpoždění 6. zakázky [h]	Zpoždění 7. zakázky [h]	Celková dynamická mezera [h]	Celkové zpoždění [h]	Hodnota kriteriální fuknce [-]
1.	6-5-7-2-1-3-4	1,2	-8,4	-22,3	-8,249999	-16,6	-11,35	-4,100002	6,800003	-64,2	6,800003	5,120002
2.	7-6-2-5-1-3-4	1,1	-17,3	-1,65	-16,85	-6,75	-11,45	-4,200001	6,700005	-51,49999	6,700005	5,020003
3.	7-5-6-2-1-3-4	1,1	-17,3	-24,5	-1,5	-16,7	-11,45	-4,200005	6,699997	-68,95	6,699997	5,019998
4.	7-6-5-2-1-3-4	1,25	-17,3	-1,65	-15,55	-16,55	-11,3	-4,050003	6,849998	-59,55	6,849998	5,169999

	Kombinace	Celkový přeseřizovací čas [h]	Zpoždění 1. zakázky [h]	Zpoždění 2. zakázky [h]	Zpoždění 3. zakázky [h]	Zpoždění 4. zakázky [h]	Zpoždění 5. zakázky [h]	Zpoždění 6. zakázky [h]	Zpoždění 7. zakázky [h]	Celková dynamická mezera [h]	Celkové zpoždění [h]	Hodnota kriteriální fuknce [-]
1.	7-6-5-2-1-3-4	1,25	-17,3	-1,65	-15,55	-16,55	-11,3	-4,050003	6,849998	-59,55	6,849998	5,169999
2.	7-5-6-2-3-1-4	1,65	-17,3	-24,5	-1,5	-16,7	-19,5	0,6499977	7,25	-71,60001	7,899998	6,024998
3.	7-6-5-1-2-3-4	2,2	-17,3	-1,65	-15,55	-20,25	-0,2999973	-3,100002	7,800003	-50,34999	7,800003	6,120002
4.	7-6-5-1-2-3-4	2,2	-17,3	-1,65	-15,55	-20,25	-0,2999973	-3,100002	7,800003	-50,34999	7,800003	6,120002
5.	6-7-2-5-1-3-4	3,1	-8,4	-6,499999	-14,85	-4,750002	-9,450001	-2,200005	8,699997	-37,45001	8,699997	7,019998
6.	7-5-6-3-2-1-4	2,3	-17,3	-24,5	-1,5	-28,3	-3,950001	1,299999	7,900002	-66,35	9,200001	7,130001
7.	2-6-7-5-1-3-4	2,95	-32,4	0,4000006	2,300001	-4,9	-9,599998	-2,350002	8,550003	-38	11,25	8,760003
8.	7-5-6-1-3-2-4	1,55	-17,3	-24,5	-1,5	-20,25	-13	11,35	7,150002	-58,05	18,5	13,415
9.	7-6-5-4-2-1-3	1,54	-17,3	-1,65	-15,55	-29,86	1,840002	7,09	14,34	-41,09	23,27	16,751
10.	7-6-5-4-2-1-3	1,54	-17,3	-1,65	-15,55	-29,86	1,840002	7,09	14,34	-41,09	23,27	16,751
11.	7-5-6-1-3-4-2	2	-17,3	-24,5	-1,5	-20,25	-13	-2,099998	29,6	-49,05	29,6	21,32
12.	5-1-7-2-6-4-3	0,9	-31,2	-35,9	-1,049999	-9,4	23,4	-4,799999	13,7	-45,25001	37,1	26,24
13.	6-5-3-1-7-2-4	1,65	-8,4	-22,3	-35,2	-15,05	19,8	11,45	7,25	-42,45	38,5	27,445
14.	6-5-2-4-7-3-1	3,25	-8,4	-22,3	-23,3	-27,5	19	-0,1000061	20,05	-42,55001	39,05	28,31
15.	6-2-4-5-7-3-1	2,57	-8,4	-23,6	-27,8	4,27	18,32	-0,7800026	19,37	-18,62	41,96	30,143
16.	6-7-1-3-5-4-2	3,33	-8,4	-6,499999	-18,56	-11,31	13,54	-0,7699966	30,93	-1,069994	44,47	32,128
17.	7-2-4-5-3-6-1	2,12	-17,3	-25,65	-29,85	2,220001	-10,68	37,67	18,92	-24,67	58,81	41,803
18.	6-4-2-7-5-1-3	3,55	-8,4	-36,6	-4,9	21	13,8	9,099998	16,34999	10,34999	60,24999	43,23999
19.	2-5-1-6-3-7-4	5,7	-32,4	-22,3	-27	16,95	-9,850002	32,85	11,3	-30,45	61,10001	44,48
20.	6-4-5-3-1-7-2	1,62	-8,4	-36,6	-4,530001	-17,43	2,719997	37,57	29,22	2,549994	69,50999	49,14299
21.	4-2-5-7-3-6-1	3,95	-45,6	-13,9	-3,800001	10,25	-8,850006	39,49999	20,74999	-1,650024	70,49998	50,53499
22.	4-5-6-1-7-3-2	3,22	-45,6	-13,53	9,470001	-9,279999	25,57	6,469997	30,82	3,920008	72,33	51,597

Obrázek 7.9: Genetický algoritmus pro zakázky vybrané z databáze

V následujícím kroku je možné zobrazit podrobné informace o všech zakázkách (viz obr. 7.10). Nejnižší hodnotu kriteriální funkce má kombinace 7-5-6-2-1-3-4 a to 5,02. Kombinace 7-6-2-5-1-3-4 má téměř totožnou hodnotu kriteriální funkce. Rozdíly mezi těmito dvěma kombinacemi jsou téměř zanedbatelné. Celková doba přeseřizování je 1,1 hodin a celkové zpoždění je 6,7 hodin. Zakázky zak1 a zak3 mají vyšší prioritu a obě jsou v tomto uspořádání zakázek dokončeny včas. Zajímavé je uspořádání zakázek, kde se dosahuje minimální přeseřizovacího času. Hodnota kriteriální funkce je totiž 107. Takto vysoká hodnota je způsobena tím, že jedna z prioritních zakázek (konkrétně zak1) má veliké zpoždění a nastává penalizace za nedodržení termínu u prioritní zakázky. Při této kombinaci zakázek se dosáhne většího využití zdroje, ale to je vykoupeno vysokými zpožděními.

Ještě je přidána jedna kombinace. Ta je výsledkem jednoduchého algoritmu zaměřeného pouze na termíny, velikosti a priority zakázek. Posloupnost zakázek se tvoří tak, že se postupně vybírají zakázky podle toho, kdy musí být dokončeny. Pokud je více zakázek s podobnými termíny, rozhoduje se podle velikosti zakázek a priorit. I přesto kombinace získaná tímto postupem má celkové zpoždění 23 hodin, což je hodně. V potaz vůbec nebyl brán přeseřizovací čas. Proto je celkový přeseřizovací čas dlouhý, má hodnotu 3,58 hodin. Genetickým algoritmem bylo získáno řešení, kde je nejmenší zpoždění a zároveň je minimalizován i přeseřizovací čas.



	Kombinace 6-5-7-2-1-3-4	Kombinace 7-6-2-5-1-3-4	Kombinace 7-5-6-2-1-3-4	Kombinace 7-6-5-2-1-3-4	Minimální přeseřízení Kombinace 3-5-4-1-7-2-6	Vlastní řešení Kombinace 2-6-7-1-5-3-4
Hodnota kriteriální funkce	5,120002	5,020003	5,019998	5,169999	107,5634	17,216
Celkový přeseřizovací čas [h]	1,2	1,1	1,1	1,25	0,56	3,58
Celkové zpoždění [h]	6,800003	6,700005	6,699997	6,849998	153,422	23,06
Celková doba výroby [h]	70	69,9	69,89999	70,05	69,36	72,38
zak5 Začátek práce	21. 5. 2019 18:38:59	21. 5. 2019 18:33:00	21. 5. 2019 18:32:59	21. 5. 2019 18:42:00	22. 5. 2019 14:11:34	21. 5. 2019 20:14:23
zak5 Dokončení zakázky	22. 5. 2019 22:39:00	22. 5. 2019 22:33:00	22. 5. 2019 22:32:59	22. 5. 2019 22:42:00	23. 5. 2019 18:11:34	23. 5. 2019 0:14:23
zak5 Termín zakázky	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00
zak5 Zpoždění [hod]	-11,35	-11,45	-11,45	-11,3	8,192997	-9,76
zak1 Začátek práce	21. 5. 2019 21:48:00	21. 5. 2019 21:33:00	21. 5. 2019 21:41:59	21. 5. 2019 21:51:00	24. 5. 2019 1:05:34	22. 5. 2019 9:56:49
zak1 Dokončení zakázky	22. 5. 2019 17:24:00	22. 5. 2019 17:09:00	22. 5. 2019 17:17:59	22. 5. 2019 17:27:00	24. 5. 2019 20:41:34	24. 5. 2019 8:16:47

Obrázek 7.10: Podrobné informace o řešeních získaných genetickým algoritmem

7.11 Skupina zdrojů jako úzké místo

Jak již bylo popsáno v dřívějších kapitolách, za úzké místo se považuje stroj. Úzkým místem ovšem může být i skupina strojů. Tyto stroje jsou identické a mají stejné parametry, vyrábějí se na nich stejné výrobky a provádějí se na nich stejné výrobní operace. Pokud je úzkým místem skupina zdrojů, vyvstává problém nejen v jakém pořadí zakázky vyrábět, ale také jak zakázky jednotlivým zdrojům rozdělovat. Na jiném místě v této práci bylo popsáno, že optimalizaci úzkého místa lze chápat jako speciální variantu problému obchodního cestujícího, kdy nedochází k návratu do výchozího bodu a kdy není přesně určen uzel, ze kterého se startuje. Stejně lze k tomuto problému přistupovat i pokud je úzkým místem ne jeden, ale skupina zdrojů. V případě více zdrojů se jedná o variantu problému obchodního cestujícího, kde vystupuje několik obchodních cestujících. V angličtině se taková varianta označuje jako multiple travelling salesman problem (zkratka mTSP). Řešení tohoto problému je složitější než obvyklá varianta jednoho cestujícího. Pokud je nalezen nějaký způsob, jak řešit problém, kde vystupuje několik obchodních cestujících, je možné tímto způsobem řešit i obvyklou variantu s jedním cestujícím, protože se jedná o speciální případ mTSP, kde je jenom jeden cestující. Uvedené problémy s jedním a více cestujícími jsou pochopitelně navzájem příbuzné a k jejich řešení je možné využívat stejných postupů. Hovoří o hledání nejlepší cesty mezi několika městy. Města jsou objekty, které mohou reprezentovat uzly v grafu, nebo v našem konkrétním příkladu jednotlivé zakázky.

Snahou je dosáhnout převedení problému několika obchodních cestujících na problém pouze s jedním obchodním cestujícím. Existuje několik možností, jak k mTSP přistupovat. V [21] se tento problém řeší rozdělením na dvě fáze.

V první fázi se množina všech měst (nebo v našem případě uspořádání zakázek) rozdělí na několik podmnožin. Každá podmnožina má vlastního obchodního cestujícího. Rozdělením na podmnožiny se mTSP rozdělí na několik obvyklých variant problému obchodního cestujícího. Druhou fází je řešení klasické varianty TSP (Travelling Salesman Problem). Problém mTSP je tedy převeden na problém vhodného rozdělení prohledávané množiny. Pokud se řeší mTSP dvoufázově, je snahou aby každý obchodní cestující nalezl nejlepší cestu, ale zároveň dosáhnout toho, že všichni cestující se dostanou do cíle po stejné cestě. Pokud se hledí na mTSP z časového hlediska, tzn. hledá se nejkratší doba za kterou cestující procestuje všechna města, snahou je, aby se všichni cestující dostali do cíle v podobně dlouhém čase. Protože pokud by všichni cestující navštívili všechny body své podmnožiny, ale jeden cestující měl před sebou ještě několik míst, která musí navštívit, těžko lze hovořit o nalezení optimálního řešení. Zbylá místa by mohl navštívit někdo, kdo už procestoval celou svou podmnožinu.

Problém rozdělení množiny na několik podmnožin, které si jsou na základě nějakého parametru dostatečně podobné, má mnoho řešení. Nejjednodušším řešením je náhodně rozdělit množinu tak, aby každá podmnožina obsahovala stejný počet prvků. Je předem zřejmé, že tento postup v drtivé většině případů neobstojí. Mnohem lepší je prvky množiny rozdělit na základě nějaké blízkosti. Pokud je cílem procestovat několik měst, znamená to rozdělit města na skupiny, kde většinu měst, která jsou libovolnému městu nejblíže, patří do stejné skupiny. Rozdělení prvků do shluků na základě euklidovské vzdálenosti se věnuje například metoda K-means (z anlického means – průměry). V [21] je metoda K-means kombinována s omezením nejvyššího možného množství prvků ve shluku. Toto omezení má zabránit příliš velké nevyrovnanosti ve velikosti jednotlivých shluků. Pro použití metody K- means se využívá euklidovská vzdálenost. U problematiky úzkých míst ovšem použití euklidovské vzdálenosti není možné.

Mnou navrhované řešení rozdělení zakázek do n skupin, kde n je počet zdrojů, vychází z následující úvahy. Rozdělit zakázky náhodně tak, aby každý zdroj měl stejné množství zakázek, není dobré řešení. Zakázky totiž neobsahují stejné množství výrobků a pokud by bylo jednomu zdroji přiřazeno větší množství zakázek, které obsahují výrazně více výrobků než ostatní, zdroj nemůže tyto zakázky dokončit včas. U zakázek je potřeba vypočítat dobu potřebnou na jejich dokončení a podle této doby zakázky mezi zdroje rozdělit tak, aby na každý zdroj připadaly zakázky, které budou mít v součtu stejný výrobní čas. Pro každou skupinu zakázek se poté využije naprogramovaný genetický algoritmus.

8 Výsledky

Proto, aby bylo možné nějakým způsobem demonstrovat, zda se podařila optimalizace úzkého místa, je potřeba použít nějaké další algoritmy nebo postupy, jejichž výsledky poté mohou být porovnány s výsledky z genetického algoritmu. Jeden způsob, jak lze získat řešení, je jít cestou maximalizace využití zdroje. To znamená najít takové uspořádání zakázek, kdy je přeseřizovací čas minimální. Toto řešení bude mít nejmenší možnou hodnotu v první části kriteriální funkce, kde vystupuje celkový seřizovací čas. Nalezení takovéhoto uspořádání zakázek bude mít další výhodu a to tu, že bude známý minimální seřizovací čas a jeho hodnota bude užitečná pro zhodnocení výsledku genetického algoritmu, protože porovnáním se seřizovacím časem dalších řešení lze zjistit, jak se hodnota seřízení blíží ideální hodnotě. Další uspořádání se zajistí na základě jednoduchého postupu, který zkoumá pouze termíny, priority a velikosti zakázek. Při tomto způsobu se postupně vybírají zakázky, které mají nejbližší termín, zohledňuje se velikost zakázek a priorita. I když nějaká zakázka má blízko termín dokončení, může být vybrána přednostně jiná, protože druhý nejbližší termín má prioritní zakázka, která je větší. Tímto postupem se získá řešení, které má zpravidla menší hodnotu zpoždění, ale rozhodně není nejlepším možným. K něčemu takovému by bylo potřeba provést optimalizaci. Tento postup simuluje způsob, jakým by problém řešil člověk metodou nevyžadující provádět větší množství náročných výpočtů.

Dalším důležitým aspektem je určení hodnot vah. Váha seřízení a váha zpoždění by se měly volit tak, aby byly z intervalu od nuly do jedné. Dalším doporučením je, aby váhy byly normalizované a tedy součet obou vah se rovnal jedné. Pro běh algoritmu nejsou nutné normalizované váhy, ale já jsem s nimi ve všech příkladech a výpočtech pracoval. Poměr obou vah, jsou-li normalizované, je totiž velmi názorný. Pokud je některá z vah nastavená na nulu, vyřazuje se tím zcela vliv daného parametru. Důležitější pro správnou optimalizaci úzkého místa je jednoznačně vliv zpoždění a váha zpoždění musí být tedy větší, než váha seřízení. Zároveň nesmí být vliv seřízení příliš malý, protože se snažíme dosáhnout co nejvyššího využití zdroje. Pro většinu výpočtů jsem používal nastavení vah, kde váha přeseřízení je 0,3 a váha zpoždění je 0,7. Při zkušebních výpočtech jsem zjistil, že výsledná řešení nejsou příliš citlivá na menší změny vah. Může se zdát, že při mírně pozměněných vahách dochází k tomu, že jsou nacházena zcela odlišná řešení, ale to na malých vzorcích může být způsobeno pouhou náhodou. V tomto problému bylo požadavkem minimalizovat přednostně zpoždění a algoritmus to skutečně plní. Výsledkem jsou víceméně stejné hodnoty, když je váha zpoždění 0,6 nebo 0,9. To je také ovlivněno velikostí přseřizovacích časů. Pokud jsou přeseřizovací časy vzhledem k potenciálním zpožděním zanedbatelné, pak i při navzájem podobně velkých váhách bude upřednostněna minimalizace zpoždění. Pokud by se v problému vyskytovaly skutečně velké přeseřizovací časy, volba vah už by měla mnohem větší dopad.

Soubor, na kterém bude testována funkčnost a použitelnost genetického algoritmu, je tvořen několika skupinami zakázek. V předešlé kapitole bylo popsáno, že programové vybavení bylo upraveno tak, aby data vstupovala do programu ve formě databází. Pro každou skupinu zakázek byla vytvořena vlastní databáze, která obsahuje všechny potřebné údaje pro

optimalizaci úzkého místa. Konkrétně vstupem je pět databází. Pro každou databázi se postupuje následovně.

Nejdříve je zjištěno pořadí zakázek, kde je využití stroje maximální, a tím se tedy zjistí nejmenší přeseřizovací čas. V druhém kroku se použije genetický algoritmus, váha přeseřízení se nastaví na hodnotu 0, váha zpoždění na 1 a algoritmu se nastaví velká populace. Genetický algoritmus se opakovaně spouští. Cílem je najít nejmenší možné zpoždění, které je možné dosáhnout. Nelze vyloučit, že takto nalezená hodnota celkového zpoždění nebude optimální, na rozdíl od hodnoty minimálního celkového přeseřizovacího času, který byl získán prohledáním všech možností. Tyto dvě hodnoty, i když jedna z nich nemusí být optimální, se použijí jako referenční, podle nichž lze zhodnotit konečné řešení. Cílem optimalizace úzkého místa je nalézt vhodný kompromis mezi přeseřizovacím časem (potažmo využití zdroje) a celkovým zpožděním. Kvalitu řešení tak lze posoudit podle toho, jak se blíží těmto dvěma hodnotám parametrů. Poté, co jsou získána tato řešení, následuje spuštění genetického algoritmu pro databázi, kde už jsou nenulové hodnoty obou vah. Všechna získaná řešení jsou následně porovnána.

V tabulce 8.1 se nachází hodnoty, které byly získány při následujícím nastavení. Váha přeseřízení je 0,3 a váha zpoždění je 0,7. Populace genetického algoritmu je 40 prvků a běh se bude opakovat třicetkrát. V tabulce je uvedena nejlepší hodnota, která byly získána. V tabulce jsou výsledky pro čtyři algoritmy. První algoritmus, který je označený jako minimální přeseřízení, je už výše popsaný algoritmus, který zkoumá všechny možnosti a vybere tu, kde je nejmenší celkové přeseřízení. Algoritmus minimální zpoždění je použitím genetického algoritmu, kde váha přeseřízení je nula. Optimalizace je výsledek samotného genetického algoritmu. Empirické řešení označuje již popsaný algoritmus, který simuluje jednoduchý rozhodovací proces člověka. Pro porovnání byly dosazeny hodnoty celkového zpoždění a přeseřízení do kriteriální funkce použité u optimalizace pomocí genetického algoritmu.

Tabulka 8.1: Podrobné hodnoty jednotlivých řešení získaných různými algoritmy

	Parametry	**Minimální přeseřízení**	**Minimální zpoždění**	**Optimalizace**	**Empirické řešení**
1. Databáze	Kriteriální funkce [h]	47,76	11,36	10,16	29,62
	Celkové zpoždění [h]	67,80	12,8	12,8	40,60
	Celkové přeseřízení [h]	1	8	4	4

	Parametry	Minimální přeseřízení	Minimální zpoždění	Optimalizace	Empirické řešení
2. Databáze	Kriteriální funkce [h]	90,28	1,923	0,36	6,67
	Celkové zpoždění [h]	128,67	0	0	7,18
	Celkové přeseřízení [h]	0,69	6,41	1,2	5,47
3. Databáze	Kriteriální funkce [h]	90,45	1,44	0,708	11,52
	Celkové zpoždění [h]	129,00	0	0	13,67
	Celkové přeseřízení [h]	0,52	4,79	2,36	6,5
4. Databáze	Kriteriální funkce [h]	72,43	0,54	0,213	1,07
	Celkové zpoždění [h]	103,229	0	0	0
	Celkové přeseřízení [h]	0,56	1,8	0,71	3,58
5. Databáze	Kriteriální funkce [h]	111,94	1,39	1,16	2,168
	Celkové zpoždění [h]	159,69	0,80	0,80	0,80
	Celkové přeseřízení [h]	0,52	2,78	2,01	5,36

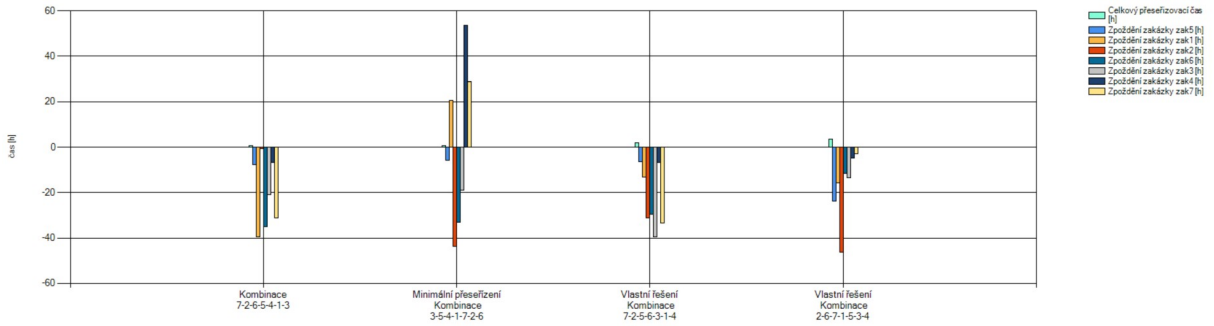
Pro správnou interpretaci hodnot z tabulky 8.1 je dobré uvést několik základních informací ohledně použitých databází. Stručný výpis obsahu databází je v tabulce 8.2. První sloupec obsahuje označení databáze, druhý sloupec počet zakázek obsažených v databázi. Třetí sloupec obsahuje údaj o tom, jak dlouho se budou všechny zakázky v databázi vyrábět. Jedná se o čistý čas a přeseřízení se nebere v úvahu. Doba výroby je závislá na počtech výrobků a

délce jejich výroby. V posledním sloupci je obsaženo datum, které udává, kdy je započata práce na zakázkách v dané databázi. Podle tohoto data se odvíjí všechny výpočty zpoždění apod.

Tabulka 8.2: Základní informace o použitých databázích

Databáze	Počet zakázek	Celková doba výroby [h]	Datum začátku práce
1. Databáze	6	74,8	16. 2. 2002 6:00:00
2. Databáze	10	108,9	14. 5. 2019 10:00:00
3. Databáze	15	138,9	18. 5. 2019 9:00:00
4. Databáze	7	68,8	20. 5. 2019 16:00:00
5. Databáze	10	96,3	19. 5. 2019 14:00:00

Zhodnocení výsledků z tabulky 8.1 je následující. Pro první databázi bylo získáno genetickým algoritmem uspořádání, kdy je celkové zpoždění minimální. Celkový přeseřizovací čas je mnohem vyšší než minimální hodnota, ale jedná se o skupinu pouze 6 zakázek a počet druhů výrobků je ještě nižší. Pokud je snahou opravdu primárně dosáhnout co nejnižšího zpoždění, je velice málo pravděpodobné, že se podaří najít řešení, v němž je podobné zpoždění, ale nižší celkový přeseřizovací čas. U druhé databáze bylo nalezeno řešení s nejnižším možným zpožděním a zároveň se velice dobře podařilo minimalizovat přeseřízení, a to tak, že se liší od minimální hodnoty pouze o půl hodinu. Zajímavé také je, že empirický algoritmus nalezl řešení s poměrně malým zpožděním, ale objevuje se zde jeho slabina, kdy nebere v potaz druhý parametr, kterým je přeseřízení. U třetí databáze bylo nalezeno opět uspořádání, kdy je zpoždění nejmenší možné, a to nula. Rozdíl ve velikosti přeseřizovacího času oproti minimální hodnotě je už větší. Výroba zakázek ve třetí databázi ovšem trvá skoro 139 hodin. Proto je potřeba zohlednit také to, jak velké je přeseřízení vůči celkové době. V tomto případě se proto takto velký rozdíl nejeví jako až tak významný problém. U čtvrté databáze bylo nalezeno velice kvalitní uspořádání, protože zpoždění je opět nula a přeseřízení se od ideální hodnoty liší pouze o 9 minut. U páté databáze bylo opět nalezeno nejmenší možné zpoždění, ale doba přeseřízení se liší už poměrně značně a nejedná se ani o velkou skupinu zakázek. Za zmínku také stojí, že u posledních dvou databází našel empirický algoritmus nejmenší zpoždění.



Obrázek 8.1: Graf nalezených řešení pro 4. databázi

Diskutabilní ovšem je to, zda je dobré použít stejný algoritmus pro hledání minimálního zpoždění a poté k optimalizaci. Je možné, že jím nalezené řešení jsou skutečně nejmenší. Na cvičných příkladech, které jsou popsány v 7. kapitole, genetický algoritmus skutečně vždy našel nejmenší možné zpoždění. Ale je také možné, že genetický algoritmus optimální nejmenší zpoždění nenalezne. V případě, kdy je minimální zpoždění rovno nule, lze říci s jistotou, že bylo nalezeno ideální řešení. Proto považovat za úspěch fakt, že při optimalizaci se dosáhne stejného zpoždění jako v předešlém kroku, je velice diskutabilní. Ovšem pomůže to ilustrovat skutečnost, že algoritmus s danými váhami bude vždy tíhnout především k minimalizaci zpoždění a stav, kdy se vybere řešení, které má o něco vyšší zpoždění, ale nižší čas přeseřízení, příliš často nenastane.

Způsob, jakým program prezentuje výsledky je demonstrován na 4. databázi a graf zpoždění je na obrázku 8.1. První kombinace uspořádání je získána genetickým algoritmem, druhá kombinace je uspořádání zakázek, kde je minimální celkový přeseřizovací čas. Třetí kombinace je vložené řešení, kde bylo minimalizováno zpoždění a poslední kombinace je vložené řešení, kde byl použit empirický postup k získání řešení. Hodnoty v tomto grafu korespondují s tabulkou v obrázku 8.2. V tabulce jsou uvedeny celkové doby zpoždění a přeseřízení a také celková doba výroby. Program také vypíše parametry jednotlivých zakázek. Tzn. datum a čas začátku práce na zakázce, datum a čas, kdy bude práce na zakázce dokončena. Dále také obsahuje termín zakázky a zpoždění, které vznikne.

	Kombinace 7-2-6-5-4-1-3	Minimální přeseřízení Kombinace 3-5-4-1-7-2-6	Vlastní řešení Kombinace 7-2-5-6-3-1-4	Vlastní řešení Kombinace 2-6-7-1-5-3-4
Hodnota kriteriální funkce	0,213	72,42829	0,54	1,074
Celkový přeseřizovací čas [h]	0,71	0,56	1,8	3,58
Celkové zpoždění [h]	0	103,229	0	0
Celková doba výroby [h]	69,51	69,36	70,6	72,38
zak5 Začátek práce	21. 5. 2019 22:15:36	22. 5. 2019 0:11:34	21. 5. 2019 7:51:00	21. 5. 2019 6:14:23
zak5 Dokončení zakázky	23. 5. 2019 2:15:36	23. 5. 2019 4:11:34	22. 5. 2019 3:27:00	22. 5. 2019 10:14:23
zak5 Termín zakázky	23. 5. 2019 10:00:00	23. 5. 2019 10:00:00	22. 5. 2019 10:00:00	23. 5. 2019 10:00:00
zak5 Zpoždění [hod]	-7,739998	-5,807003	-6,549999	-23,76
zak1 Začátek práce	20. 5. 2019 22:45:00	23. 5. 2019 11:05:34	21. 5. 2019 16:47:59	21. 5. 2019 19:56:49
zak1 Dokončení zakázky	21. 5. 2019 18:21:00	24. 5. 2019 6:41:34	22. 5. 2019 20:47:59	23. 5. 2019 18:16:47

Obrázek 8.2: Snímek tabulky obsahující parametry jednotlivých zakázek pro 4. databázi

9 Závěr

Diplomová práce má dvě hlavní části: teoretickou a praktickou. Smyslem teoretické části je připravit základ, na němž pak staví část praktická. Teoretická se zabývá pokročilými metodami plánování výroby, rozebírá základní typy používané v praxi, krátce přihlíží k jejich historickému vývoji a pak se soustřeďuje na metody používané v současnosti. Věnuje se podrobně optimalizaci plánování a analyzuje ho až na úroveň typických úloh, jež musí být při plánování řešeny. Mezi jinými je to hledání cest v síti uzlů s využitím teorie grafů, problém obchodního cestujícího, určení vhodných metrik pro vyhodnocení kvality nalezených řešení, která mají být blízká optimálnímu plánování, zabránění kolize více úloh, naplánovaných k provedení na stejném zařízení ve stejný čas apod. Jádro této části práce je v přípravě vhodných existujících algoritmů nebo v jejich úpravách, kombinacích či návrhu nových tak, aby byly použitelné pro konkrétní výpočty úloh.

Na databázích bylo ukázáno, jak jsou data výrobních procesů ukládána a jakým způsobem je lze reprezentovat. Byly vysvětleny pojmy vyskytující se ve výrobním procesu a byly vysvětleny jejich parametry a také to, jak jsou vzájemně ve výrobním procesu provázány.

Cílem praktické části práce bylo vytvořit programové vybavení implementující algoritmy, které pomohou získat kvalitní řešení. Kvalitním řešením se chápe takový návrh nastavení výrobního systému, kde se dosáhne zlepšení v oblasti úzkého místa a tím pádem dojde ke zlepšení celkové kvality systému. Úpravou nastavení systému se rozumí především úprava pořadí zakázek. Používají se dva hlavní parametry pro zhodnocení kvality nabízeného řešení. Prvním z nich je celkový přeseřizovací čas, který ovlivňuje využití úzkého místa. Cílem je přseřizovací čas minimalizovat a tím dosáhnout co nejvyššího využití zdroje. Druhým stěžejním parametrem je zpoždění zakázek. Snahou je najít řešení, kdy bude zpoždění minimální a tím předejít penalizacím za nedodržení termínu zakázek. Pro získání kvalitního řešení optimalizačního problému je potřeba najít vhodný kompromis, který minimalizuje jak hodnotu přeseřízení, tak hodnotu zpoždění. Je potřeba, aby řešení především minimalizovalo zpoždění, protože negativní efekt plynoucí z pozdního dokončení zakázek je větší než negativní efekt způsobený větším celkovým přeseřizovacím časem (menším využití zdroje).

Pro řešení bylo postupně použito několik algoritmů. Na začátku byl použit algoritmus, který prohledá všechna dostupná řešení. Tento algoritmus vždy nalezne optimální řešení, ale vzhledem k náročnosti problému ho nelze použít ani pro mírně složitá řešení. Algoritmus byl použit pouze k tomu, aby se na jednodušších příkladech ověřila funkčnost algoritmů jiných. Dále byl zkoušen jednoduchý algoritmus založený na metodě nejbližšího souseda. Řešení navržená tímto algoritmem byla téměř nepoužitelná. Proto jsem se nakonec rozhodl pro použití genetického algoritmu. Genetický algoritmus (zkratka GA) sice nedosáhne optimálních řešení, ale dosáhne dostatečně kvalitních řešení v rozumném čase.

Před realizací genetického algoritmu byla vytvořena kriteriální funkce, podle které se posuzovala kvalita řešení. Nejdříve bylo otestováno, že navržená účelová funkce má skutečně

nejnižší hodnotu pro řešení, které se považuje jako optimální. To bylo ověřeno použitím algoritmu, který navrací všechna řešení. Po sestavení kriteriální funkce jsem postupně vytvářel a testoval genetický algoritmus. U příkladů, kde byla známa všechna možné řešení, genetický algoritmus vždy nalezl optimální řešení. U složitějších případů, kdy už byl znám pouze minimální přeseřizovací čas, již nelze hovořit o kvalitě řešení s takovou jistotou. Oproti jednodušším algoritmům, GA nalezl řešení s výrazně nižším celkovým zpožděním. Při tomto testování GA jsem ověřil, že jedna z hlavních slabin GA je, že kvalita výstupu je značně závislá na počáteční populaci. V některých běžích algoritmus vracel velice špatná řešení a uvízl tak v lokálním extrému, který byl značně vzdálen od globálního. Tento problém má jednoduché řešení a tím je opakované spouštění algoritmu. S tím souvisí další zjištění, že než volit větší populaci, je lepší využít výpočetní výkon na opakovaný běh algoritmu.

Pro zkoušené příklady GA často nalezl řešení, kde celková hodnota zpoždění je nula. Což je optimální hodnota, protože hodnota zpoždění je nezáporná. Nelze ovšem s jistotou říci, že to svědčí o kvalitě algoritmu, protože to záleží na tom, kolik řešení ze všech možných řešení má nulovou hodnotu zpoždění. Každopádně GA je schopen velice rychle nalézt řešení, kde je hodnota zpoždění značně menší než u jiných algoritmů. GA je také schopen bez větších problému pracovat s prioritami a dovolenými odchylkami zpoždění. Co se týče hodnot přeseřízení, GA v některých případech nalezl, řešení kde je přeseřízení blízké optimální hodnotě. V některých případech se lišil už více, ale to může být způsobeno tím, že přeseřízení bylo až druhotným parametrem.

Výsledkem této práce je kromě uceleného souboru informací o problematice úzkých míst také program v jazyce Visual Basic .NET. Tento program je schopen načíst data z databáze a následně po rychlém výpočtu nabídnout přehledný návrh řešení. Program může být použit při plánování výroby nebo když je potřeba rychle získat kvalitní řešení problému, když se ve výrobním procesu vyskytne úzké místo.

Vytvořený program nabízí mnoho prostoru pro případné další využití a doplnění nových funkcí. Jednou z oblastí, kde by bylo možné pokračovat ve vývoji, je situace, kdy je úzkým místem více zdrojů. Určitě je také možné navrhnout algoritmus, který by byl schopen provádět rozdělení zakázek mezi jednotlivé zdroje. Rozdělení zakázek mezi zdroje jako výchozí krok pro využití metody mTSP je však velmi náročné na algoritmizaci a zpracování. Svým rozsahem by řešení takového problému bylo podobné rozsahu celé této práce.

V souhrnu mohu konstatovat, že vytyčené cíle se podle mého názoru podařilo splnit. V teoretické části jsem provedl podrobný rozbor, který poskytl dostatek podkladů pro praktickou část. Vytvořený program je plně funkční, dává dostatečně přehledné výsledky a umožňuje porovnat řešení nalezená více způsoby. Kromě plánování výroby jej lze využít i pro praktickou analýzu chování jednotlivých algoritmů a zjišťování vlivu vstupních parametrů na výsledek.

10 Seznam použitých zdrojů

[1] UPADHYA, Sujata. What is Forward and Backward Scheduling?. *Better Business Blog* [online]. 2016 [cit. 2019-02-19]. Dostupné z: https://betterbusiness.deskera.com/what-is-forward-and-backward-scheduling/

[2] Forward Scheduling. *Aras' Integrated Online Knowledge Center* [online]. Aras Corporation, 2005 [cit. 2019-02-19]. Dostupné z: http://aras10.mitekusa.com/Client/WebHelp/en/mergedProjects/ProgramManagement/Forward_Scheduling.htm

[3] Backward Scheduling. *Aras' Integrated Online Knowledge Center* [online]. Aras Corporation, 2005 [cit. 2019-02-19]. Dostupné z: http://aras10.mitekusa.com/Client/WebHelp/en/mergedProjects/ProgramManagement/Backward_Scheduling.htm

[4] SAGBANSUA, L. Information Technologies and Material Requirement Planning (MRP) in Supply Chain Management (SCM) as a Basis for a New Model. *Bulgarian Journal of Science and Education Policy* [online]. University of Sofia, 2010, **4**(2), 236-247 [cit. 2019-02-19]. ISSN 1313-1958. Dostupné z: https://doaj.org/article/9f7b550eba7f4871955554f64731c737

[5] MABERT, Vincent. The early road to material requirements planning. *Journal of Operations Management*. 2007, **2007**(25), 346-356. ISSN 0272-6963.

[6] SNAPP, Shaun. *How to Understand The History of MRP and DRP* [online]. 2012 [cit. 2019-02-19]. Dostupné z: https://www.brightworkresearch.com/scmhistory/2012/08/the-history-of-mrp-and-drp/

[7] WILSON, James. *The origin of materials requirements planning in Frederick W. Taylor's planning office*. University of Glasgow, 2015 [cit. 2019-02-19]. Dostupné z: http://eprints.gla.ac.uk/112204/1/112204.pdf

[8] MRP: Material Requirements Planning - Plánování materiálových požadavků. *Svět produktivity* [online]. 2012 [cit. 2019-02-19]. Dostupné z: http://www.svetproduktivity.cz/slovnik/MRP.htm

[9] VÁGNER, L. Rozvrhování v diskrétní výrobě. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 88 s. Vedoucí diplomové práce doc. Ing. Simeon Simeonov CSc..

[10] SURKA, Vladimir, Gabriela KRIZANOVA, Miriam IRINGOVA, Pavel VAZAN a Jaroslav ZNAMENAK. Implementation of manufacturing resource planning issues in practice. In: *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)* [online]. IEEE, 2016, s. 151-156 [cit. 2019-02-25]. DOI: 10.1109/INES.2016.7555110.

[11] BAHSSAS, Dania Mahmoud, Adnan Mustafa ALBAR a Rakibul HOQUE. Enterprise Resource Planning (ERP) Systems: Design, Trends and Deployment. *The International Technology Management Review* [online]. 2015, **5**(2), 72-81 [cit. 2019-03-07]. ISSN 1835-5269. Dostupné z: https://www.researchgate.net/publication/279515140_Enterprise_Resource_Planning_ERP_Systems_Design_Trends_and_Deployment

[12] HARRISON, David K a David PETTY. *Systems for Planning and Control in Manufacturing*. Butterworth Heinemann, 2002. ISBN 0750649771.

[13] WANG, Yong-Cai, Qian-Chuan ZHAO a Da-Zhong ZHENG. *Bottlenecks in production networks: An overview* [online]. [cit. 2019-03-07]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.2113&rep=rep1&type=pdf

[14] DANNHOFEROVÁ, Jana a Tomáš FOLTÝNEK. *Teorie grafů* [online]. [cit. 2019-04-28]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/index.pl?opora=1627

[15] JIROVSKÝ, Lukáš. *Teorie grafů* [online]. [cit. 2019-04-28]. Dostupné z:https://teorie-grafu.cz/

[16] BURDOVÁ, Jana. *Heuristické a metaheuristické metody řešení úlohy obchodního cestujícího* [online]. Praha, 2011. [2019-04-28]. Diplomová práce. Fakulta informatiky a statistiky VŠE v Praze, Katedra ekonometrie. Vedoucí práce Mgr. Jana Kalčevová, Ph.D. Dostupné z: https://vskp.vse.cz/id/1237286

[17] MALÉŘ, Petr. *Algoritmy pro problém Euklidovského obchodního cestujícího* [online]. Olomouc, 2014. [cit. 2019-04-28]. Bakalářská práce. Přírodovědecká fakulta Univerzity Palackého, Katedra informatiky. Vedoucí práce Petr Osička. Dostupné z: https://theses.cz/id/eonx8j/TSP.pdf

[18] CHIANG, S.-Y., C.-T. KUO a M. MEERKOV. Bottlenecks in Markovian Production Lines: A Systems Approach. *IEEE Transactions on Robotic and Automation* [online]. 1998, **14**(2). [cit. 2019-05-03]. Dostupné z https://pdfs.semanticscholar.org/11b5/cc7a10f41df93aa1d6d2859e12081a32efba.pdf

[19] WEIGNER, Martin. *Problém obchodního cestujícího – paralelní řešení pomocí vláken (SMP)* [online]. Brno, 2009. [2019-05-03]. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav informačních systémů. Vedoucí práce Tomáš Kašpárek. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=116425

[20] TAŞ, Duygu, Michel GENDREAU, Ola JABALI a Gilbert LAPORTE. The traveling salesman problem with time-dependent service times. *European Journal of Operational Research* [online]. Elsevier B.V, 2016, **248**(2), 372-383 [cit. 2019-05-19]. DOI: 10.1016/j.ejor.2015.07.048. ISSN 0377-2217.

[21] XU, Xiaolong, Hao YUAN, Mark LIPTROTT a Marcello TROVATI. Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing* [online]. Berlin/Heidelberg: Springer Berlin Heidelberg, 2018, **22**(19), 6567-6581 [cit. 2019-05-19]. DOI: 10.1007/s00500-017-2705-5. ISSN 1432-7643.

[22] HUSSAIN, Abid, Yousaf shad MUHAMMAD, M. NAUMAN SAJID, Ijaz HUSSAIN, Alaa MOHAMD SHOUKRY a Showkat GANI. Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. *Computational Intelligence and Neuroscience* [online]. Hindawi, 2017, **2017** [cit. 2019-05-19]. DOI: 10.1155/2017/7430125. ISSN 1687-5265.

11 Seznam použitých zkratek

ASP	angl. Active Server Pages, skriptovací platforma Microsoft
BOM	angl. Bill of Materials, kusovník
CRP	angl. Capacity Requirements Planning, kapacitní plánování
E	angl. Edges, hrany (grafu)
ERP	angl. Enterprise Resource Planning, plánování podnikových zdrojů
GA	genetický algoritmus
HTML	angl. Hypertext Markup Language, značkovací jazyk webových stránek
MPS	angl. Master Production Schedule, hlavní plán výroby
MRP	angl. Material Requirement Planning, plánování potřeby materiálu
mTSP	angl. multiple Travelling Salesman Problem, vícenásobný problém obchodního cestujícího
NP	nedeterministicky polynomiální
NPC	NP-Complete, nedeterministicky polynomiální úplný (problém)
PIP	angl. Performance In Processing, výkonnost zpracovávání
POP	angl. Purchase Order Processing, zpracování požadavků pro nákup
PR	angl. Production Rate, rychlost výroby, rychlost produkce
ROP	angl. Reorder Point, stanovení okamžiku začátku práce na zakázce
SOA	angl. Services Oriented Architecture, architektura orientovaná na služby
SOP	angl. Sales Order Processing, zpracování požadavků pro prodej
TSP	angl. Travelling Salesman Problem, problém obchodního cestujícího
V	angl. Vertices, vrcholy (grafu)

12 Seznamy obrázků a tabulek

12.1 Seznam obrázků

12.2 Seznam tabulek

13 Seznam příloh

Příloha A: Datový nosič

Příloha A

Datový nosič

- Program pro optimalizaci úzkého místa, v jazyku Visual Basic .NET, složka
  - Zdrojový kód
  - Spustitelný soubor
  - Pomocné soubory
- Databáze s daty
- Elektronická verze diplomové práce