VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

AUTOMATIZOVANÉ ROZVRHOVÁNÍ V PRŮMYSLOVÉ VÝROBĚ

AUTOMATED PLANNING AND SCHEDULING IN MANUFACTURING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL BUCHER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN HRUBÝ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Bucher Michal**
Program: Informační technologie
Název: **Automatizované rozvrhování v průmyslové výrobě**
Automated Planning and Scheduling in Manufacturing
Kategorie: Modelování a simulace
Zadání:

1. Prostudujte metody počítačového rozvrhování výrobních úloh.
2. Navrhněte model úlohy kompletační výroby a algoritmus jejího rozvrhování. Zahrňte potřeby rozvrhování na stroje, personál a materiál.
3. Navržený algoritmus implementujte do podoby programu, který na zadanou úlohu výroby vypočte její rozvrh.
4. Program testujte na úloze reálného výrobního podniku.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

1. První dva body zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Hrubý Martin, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 31. října 2019

Abstrakt

Bakalářská práce se zabývá tématem automatizovaného rozvrhování v průmyslové výrobě. Problém je formálně popsaný matematickým modelem Resource Constrained Project Scheduling Problem. Na základě tohoto modelu byl vytvořený systém na rozvrhování, který využívá optimalizátor založený na principu Genetických algoritmů. V části práce o testování se věnuje využití systému nad vybraným reálným výrobním podnikem. Výsledný systém umožňuje i vykreslovat rozvrh v podobě Ganttova diagramu.

Abstract

Thesis deals with a problem of automated planning and scheduling in manufacturing. Problem is formally defined by mathematical model Resource Constrained Project Scheduling Problem. Based on this model scheduling system was developed, which is using the optimizer based on Genetic algoritms. Developed system was then tested in real manufacturing. System can also visualize schedules in form of Gantt chart.

Klíčová slova

Rozvrhování, Plánování, RCPSP, Průmyslová výroba, Genetické algoritmy

Keywords

Scheduling, Planning, RCPSP, Manufacturing, Genetic algoritms

Citace

BUCHER, Michal. *Automatizované rozvrhování v průmyslové výrobě*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Hrubý, Ph.D.

Automatizované rozvrhování v průmyslové výrobě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Bucher
27. května 2020

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu práce Ing. Martinovi Hrubému, Ph.D. za umožnění vytvoření této práce pod jeho vedením. Dále bych chtěl poděkovat zaměstnancům z firmy Lenoxi Automation s.r.o., kteří mi umožnili několik setkání s lidmi z MAHLE Behr Mnichovo Hradiště s.r.o. a zároveň podporovali práci detailními informacemi o průběhu výrobních procesů v praxi.

Obsah

Kapitola 1

Úvod

V dnešní době se díky rychlému rozvoji informačních technologií rozšiřují možnosti automatizace průmyslové výroby. Snaha využívat zdroje s co nejvyšší efektivitou, může v některých případech být podstatně rozsáhlý problém, kde hledání řešení může zabrat dlouhou dobu. Z tohoto důvodu vznikají požadavky na plánovácí a rozvrhovací systémy. Tyto systémy se pak stávají nedílnou součástí řízení a optimalizace výroby.

Plánování výroby se zabývá následujícími otázkami: jaký produkt se má vyrábět, kolik těchto produktů bude třeba vyrobit, do jakého termínu musí být vyrobeny. Plánování také zahrnuje předpovědi a důležitá fakta, jako například dostupnost materiálů, kapacity, strategie a cíle. Rozvrhování výroby pracuje s vyhotoveným plánem. Soustředí se na přiřazení plánovaných operací určitému času a prostoru možným zdrojům. Za tímto účelem je nutné znát výrobní operace a postupy. Výstupem je rozvrh, který představuje řešení.

Pro řešení rozvrhovacích úloh byl navržen formální model „Resource Constrained Project Scheduling Problem“ (RCPSP). Model definuje operace, které mají určitou dobu trvání a požadavky na zdroje. Zároveň definuje vztahy mezi operacemi. Dále definuje omezení kapacit zdrojů. Hlavní myšlenkou je minimalizovat dobu dokončení rozvrhu projektu tzv. „makespan“ za splnění všech omezení. Problematika RCPSP je zkoumána už řadu let, existuje tak velké množství studií, které problém rozšiřují o prakticky využitelné části.

Tato bakalářská práce se dále zabývá využitím modelu a modifikací některých částí v praxi. První kapitola definuje základní model RCPSP a popisuje důležité části teorie, na kterou zbytek práce navazuje. V druhé kapitole práce jsou popsané návrhy a úpravy modelu včetně algoritmu pro řešení rozvrhování nad reálnou výrobou. Zabývá se zde nedostatky a požadavky na model z reálného světa. Třetí kapitola naznačuje, jak je návrh implementovaný do podoby programu. Poslední kapitola se zabývá zhodnocením implementace a testováním.

Model a algoritmus je navrhovaný pro výrobní podnik MAHLE Behr Mnichovo Hradiště s.r.o., který poskytnul reálná data. Podnik se zabývá výrobou automobilových klimatizací a chladičů.

Kapitola 2

Rozbor tématu

Následující kapitola obsahuje dvě části. První část je věnována rozboru a definici základního modelu RCPSP. V druhé části jsou popsané varianty rozšíření základního modelu.

2.1 Resource Constrained Project Scheduling Problem

Model RCPSP je definovaný následujícím způsobem. Předpokládáme projekt, který se skládá z množiny $\mathcal{J} = \{0, 1, ..., n, n+1\}$ operací. Pro každou operaci z množiny $\mathcal{J}$ platí, že musí být provedena právě jednou. Operace 0 a $n+1$ jsou fiktivní. 0 představuje počáteční operaci projektu a $n+1$ naopak představuje konečnou operaci projektu. Platí pro ně $p_0 = 0$ a $p_{n+1} = 0$. Každá operace $j \in \mathcal{J}$ má definované dvě omezení. První omezení určuje precedenci, kde operace j nesmí začít dříve, než jsou dokončené všechny precedeční operace z množiny $\mathcal{P}_j$. Jednotlivé vztahy mezi operacemi lze znázornit grafem Activity-on-node (AON). Operace v grafu jsou představovány jako uzly a precedenční vazby jako hrany (příklad grafu 2.1). Druhé omezení popisuje požadavek operace na množství využití kapacity zdroje, kterou potřebuje po dobu vykonávání. K určuje počet typů obnovitelných zdrojů , které reprezentuje množina $\mathcal{K} = \{1, ..., K\}$. Při vykonávání operace j je zapotřebí $r_{j,k}$ množství typu zdroje $k \in \mathcal{K}$ každou periodu po dobu trvání aktivity p_j. Zdroj k má omezenou kapacitu $\mathcal{R}_k$ v každém časovém bodě. Předpokládáme, že parametry p_j, $r_{j,k}$ a $\mathcal{R}_k$ jsou deterministické. Pro počáteční a koncovou operaci projektu platí, že $p_j = 0$ a $r_{j,k} = 0$ pro všechny $k \in \mathcal{K}$. Obecně platí, že všechny udávané parametry jsou celá čísla.

F_j označuje koncový čas operace j. Rozvrh S je definovaný vektorem koncových časů operací $S = (F_1, ..., F_n)$. Probíhající operace v čas t lze popsat množinou $\mathcal{A}(t) = \{j \in \mathcal{J} | F_j - p_j \leq t < F_j\}$. Poté je možné problém lze popsat následovně:

$$\text{Min } F_{n+1} \tag{2.1}$$

$$F_h \leq F_j - p_j \qquad j = 1, ..., n+1; h \in \mathcal{P}_j \tag{2.2}$$

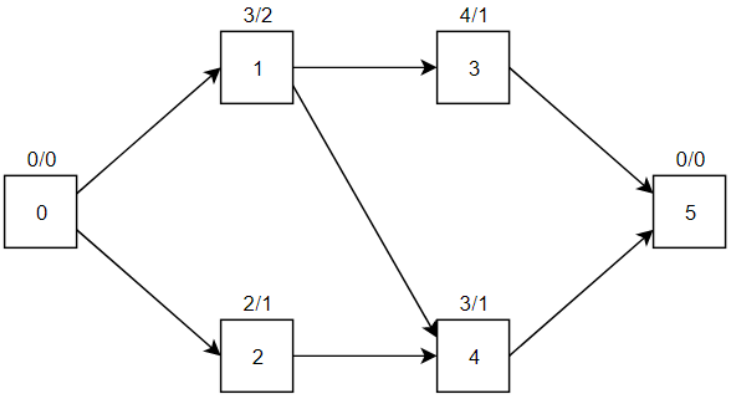
$$\sum_{j \in \mathcal{A}(t)} r_{j,k} \leq R_k \qquad k \in \mathcal{K}; t \geq 0 \tag{2.3}$$

$$F_j \geq 0 \qquad j = 1, ..., n+1 \tag{2.4}$$

RCPSP minimalizuje dobu trvání projektu 2.1. Vztah 2.2 popisuje omezení na základě precedence operací udává, že operace nesmí začínat dříve než jsou dokončeni jeho předchůdci. Vztah 2.3 kontroluje, zda nedojde k překročení maximální kapacity zdrojů v kaž-

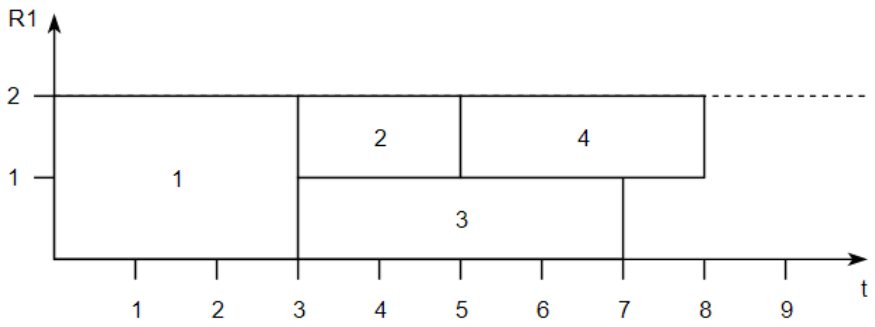
dou časovou periodu. Poslední vztah 2.4 udává, že koncové časy operací nesmí být záporné. Definice vztahů a pojmenování jednotlivých označení je převzato z [8].

Ukázkový příklad: Parametry $K = 1$ a $R_1 = 2$. Operace a jejich požadavky včetně precedencí jsou definované v grafu 2.1.



Obrázek 2.1: AON graf precedence aktivit a jejich ohodnocení ve tvaru $p_j/r_{j,1}$

Doba trvání řešení je 8. Výsledný rozvrh lze vizualizovat Ganttovým diagramem 2.2. Ganttův diagram slouží pro grafické znázornění využití zdrojů v čase. Osa x reprezentuje čas a osa y představuje využití zdroje. O historii vzniku a další informace lze dohledat zde [11].



Obrázek 2.2: Řešení úkázkového příkladu

2.2 Rozšíření RCPSP

Model RCPSP avšak nepokrývá všechny situace při využití v praxi. Vznikají tak rozšíření, která využívají základní model jako jádro. V této práci je popsaných pouze několik variant rozšíření důležitých pro aplikace modelu v reálné průmyslové výrobě. Tato sekce čerpá informace z [7], kde je možné nalézt i další rozšíření modelu.

2.2.1 Přerušitelnost operací

Model RCPSP předpokládá, že operace nemohou být po začátku přerušené. Zavedením možnosti přerušení operace je nutné zajistit, aby se operace provedla celá. Model se rozšíří o maximální počet přerušení operace. Pro nepřerušitelné operace je parametr nastavený na 0. Přerušení mohou nastat pouze v celé časové jednotky. Využití této vlastnosti v praxi lze modelovat přestávky a plánované odstávky strojů.

2.2.2 Příprava zdroje

V základu se předpokládá, že zdroj je hned připravený k využití, avšak v praxi některé zdroje vyžadují přípravu k tomu, aby dále mohly být využívány. Musí tedy být splněna operace, která uvede stroj do připraveného stavu. Je možné přípravy rozdělit do tří typů. První typ pouze kontroluje, zda je nutné před rozvrhnutou operací přiřadit její přípravu. Druhý typ zohledňuje posloupnost operací, které jsou vykonávány za sebou na zdroji, a zda kontroluje nutnost provádět přípravy opakovaně. Poslední typ mění čas přípravy podle délky vykonávané operace, včetně jejich předchůdců.

2.2.3 Minimální zpoždění operace

Operace v základním modelu RCPSP čeká na dokončení všech operací, na kterých je závislá. Rozšířením operace o vlastnost minimálního zpoždění dovolí překrývání na sobě závislých operací. Operace tedy nemusí čekat na úplné dokončení a je ji možné naplánovat o nějaký čas dříve. Hodnota d_{ij}^{FS} udává minimální dovolené zpoždění operace. Upravením vztahu o začátku následující operace pomocí hodnoty zpoždění lze dostat $C_i + d_{ij}^{FS} \leq S_j$. Pro základní model platí, že hodnota $d_{ij}^{FS} = 0$. Povolením záporných hodnot nastává překryv na sobě závislých operací.

2.2.4 Maximální zpoždění operace

Naopak od minimálního zpoždění omezuje s jakým maximálním zpožděním může nasledující operace nastat. Podobně jako u předchozího problému je zavedená hodnota zpoždění d_{ij}^{FS}. Následně upravením vztahu na $C_i + d_{ij}^{FS} \geq S_j$ je řečeno, že následující operace může začít s určitým zpožděním. V některých případech tato modifikace může vést k nevyřešitelným problémům.

2.2.5 Neobnovitelné zdroje

Model RCPSP využívá pouze obnovitelné zdroje, které po vykonání operace vrací zabranou kapacitu operací zdroji. U neobnovitelných zdrojů naopak aktivita vyčerpá požadovanou kapacitu a v modelu už se dále nevyskytuje. Dovoluje tedy modelovat situaci dostupnosti limitovaného počtu materiálu.

Případ neobnovitelných zdrojů rozšiřuje a modifikuje základní model RCPSP následujícím způsobem. Množina $\mathcal{K}^p$ představuje množinu typů obnovitelných zdrojů, $\mathcal{K}^v$ jsou typy zdrojů s omezenou kapacitou. Musíme pak dále upravit i ostatní množiny, kde R_k^p představuje kapacitu obnovitelného zdroje typu k. Množina R_k^v naopak představuje kapacitu neobnovitelného zdroje typu k. Dále je nutné upravit i parametry $r_{j,k}$ na $r_{j,k}^p$ pro využití kapacity aktivitou j obnovitelných zdrojů a $r_{j,k}^v$ využití kapacity aktivitou j neobnovitel-

ných zdrojů. p_j^p pak udává čas, za jakou obnovitelný zdroj zpracuje aktivitu j a p_j^v čas využití aktivity j neobnovitelných zdrojů.

2.2.6 Více módů

Základní model RCPSP definuje operace, kde každá operace se dá vykonat pouze jedním způsobem. Má tím pádem pevně definovanou dobu trvání a využití zdrojů. Koncept více módů (Multi-mode) rozšiřuje model RCPSP o možnost vykonávat operace více než jedním způsobem. Každá operace má definovaný určitý počet módů, kde každý mód má určenou dobu trvání a využití zdrojů.

Model lze následovně upravit. Operace j musí být vykonána v jednom z jejích módů. M_j určuje počet módů operace j. Jednotlivé módy jsou značeny $1, ..., M_j$. Po začátku operace ve vybraném módu nelze v průběhu mód měnit. Čas potřebný k vykonání operace j v módu m je dán $p_{j,m}$ a požadované množství zdroje operace j v módu m je dán $r_{j,m,k}$. Dále se zde berou v potaz i neobnovitelné zdroje (více popsáno v sekci 2.2.5). Rozvrh přiřazuje každé operaci j koncový čas F_j a mód m_j.

Kapitola 3

Metody řešení

Tato kapitola se věnuje řešením RCPSP. Seznámení se základním principem jednotlivých algoritmů a jejich rozdělení s případným použitím při řešení optimalizačních problému RCPSP. Pro další možné způsoby řešení a přehled zkoumaných přístupů lze dohledat v [10] a [3].

3.1 Heuristické metody

Základní myšlenkou těchto algoritmů je najít možné řešení, není avšak zaručené, že řešení bude optimální. Nejdříve je popsaná metoda „Schedule Generation Schemes“, která je pak dále využita v ostatních heuristických a metaheuristických metodách.

3.1.1 Schedule Generation Schemes

SGS tvoří jádro většiny heuristických a metaheuristických funkcí, které řeší RCPSP. Staví proveditelný rozvrh postupnými kroky. Při procesu sestavování rozvrhu SGS postupně rozšiřuje částečný rozvrh, dokud nerozvrhne všechny operace. Částečný rozvrh je rozvrh, který má rozvrhnutou pouze podmnožinu všech operací $n + 2$.

Existují dva typy SGS. První způsob sériový SGS (Serial SGS) postupuje po operacích, na rozdíl od paralelního SGS (Paralel SGS), který postupuje po čase [8]. Ačkoliv se využívají obě možnosti, u metaheuristických metod je nejčastěji využívaná modifikace klasického sériového SGS pro tzv. "Activity list".

Serial Schedule Generation Scheme

Sériový SGS (SSGS) začíná v čase 0 s přiřazenou startovní operací projektu do množiny rozvržených operací. Následně tvoří rozvrh postupným výběrem jedné operace v každém kroku $g = 1, ..., n$. Vybírá operaci z omezené množiny, která splňuje určitá pravidla na základě zvoleného pravidla. Vybranou aktivitu se snaží rozvrhnout na co nejdřívější čas podle omezení precedence 2.2 a stavu zdrojů 2.3. V každém kroku g jsou k dispozici dvě disjunktní množiny S_g a D_g. Množina S_g představuje všechny rozvrhnuté operace. Množina D_g obsahuje všechny operace, které je možné v kroku g rozvrhnout. Zbývající kapacitu zdroje typu k v čase t lze definovat jako $\tilde{R}_k(t) = R_k - \sum_{j \in \mathcal{A}(t)} r_{j,k}$. Množinu $\mathcal{F}_g = \{F_j \mid j \in S_g\}$ tvoří časy všech rozvrhnutých operací v kroku g. $\mathcal{D}_g = \{j \in \mathcal{J} \setminus S_g \mid \mathcal{P}_j \subseteq S_g\}$ definuje množinu všech operací, pro které platí, že jejich předchůdci jsou obsažení v množině rozvrhnutých operací v kroku g. SSGS lze popsat následovně:

Výpis 3.1: Serial Schedule Generation Scheme

$F_0 = 0,\ S_0 = \{0\}$
`for` $g = 1$ `to` n `do`
    `Calculate` $\mathcal{D}_g,\ \mathcal{F}_g,\ \tilde{R}_k(t)(k \in \mathcal{K}; t \in \mathcal{F}_g)$
    `Select one` $j \in \mathcal{D}_g$
    $EF_j = \max_{h \in \mathcal{P}_j}\{F_h\} + p_j$
    $F_j = \min\{t \in [EF_j - p_j, LF_j - p_j] \cap \mathcal{F}_g \mid r_{j,k} \leq \tilde{R}_k(\tau), k \in \mathcal{K}, \tau \in [t, t + p_j[\cap \mathcal{F}_g\} + p_j$
    $S_g = S_{g-1} \cup \{j\}$
`end`
$F_{n+1} = \max_{h \in \mathcal{P}_{n+1}}\{F_h\}$

V první části algoritmu se vloží počáteční operace $j = 0$ s časem dokončení $F_0 = 0$ do částečného rozvrhu S_g. V průchodu cyklu v každém kroku vypočítá množinu $\mathcal{D}_g$, množinu všech koncových časů rozvrhnutých operací $\mathcal{F}_g$, a aktuální stav zdrojů $\tilde{R}_k(t)$ pro časy $t \in \mathcal{F}_g$. Následně vybere jednu operaci z množiny $j \in \mathcal{D}_g$. Pro vybranou operaci j je následně vypočítaný nejdřívější čas konce EF_j na základně precedence. Poté vypočítá koncový čas F_j, pro který jsou splněné podmínky 2.2 a 2.3 z intervalu $[EF_j, LF_j]$. LF_j je nejpozdější koncový čas operace vypočítaný pomocí zpětné rekurze z horní hranice času projektu.

Tabulka 3.1: Ilustrační ukázka řešení pomocí SSGS pro příklad 2.1.

g	**1**	**2**	**3**	**4**
$\mathcal{D}_g$	{1,2}	{2,3}	{2}	{4}
j	1	3	2	4
$\mathcal{S}_g$	{0,1}	{0,1,3}	{0,1,3,2}	{0,1,3,2,4}

Při výběru operace z $\mathcal{D}_g$ dochází podle pravidla s nejdelší dobou p_j. Teda v bodě $g = 1$ je z možných operací $\{1, 2\}$ vybraná operace 1. Ta se vloží do množiny S_1. Výsledná posloupnost operací je $(1, 3, 2, 4)$.

Serial Schedule Generation Scheme for Activity list

Další možností SSGS je využít seznam operací λ tzv. „Activity list“. Tento seznam musí splňovat podmínky precedence. Aktivita j je na pozici v seznamu, kde všechny aktivity $h \in P_j$ jsou v seznamu před aktivitou j a zároveň následníci po j jsou umístění v seznamu za aktivitou j. j_g popisuje aktivitu, která byla vybrána v bodě g. Zápis seznamu potom obecně vypadá takto $\lambda = \langle j_1, ..., j_n]$

Pro dekódování „Activity list“ byl tak upravený původní SSGS, kde na místo hledání dalších možných rozvrhnutelných operací v kroku g se vybere operace ze seznamu na pozici, který odpovídá kroku g. Algoritmus lze popsat následovně:

Výpis 3.2: Serial Schedule Generation Scheme for Activity list

$F_0 = 0,\ S_0 = \{0\}$
`for` $g = 1$ `to` n `do`
    `Calculate` $\mathcal{F}_g,\ \tilde{R}_k(t)(k \in \mathcal{K}; t \in \mathcal{F}_g)$
    $j = j_g$
    $EF_j = \max_{h \in \mathcal{P}_j}\{F_h\} + p_j$
    $F_j = \min\{t \in [EF_j - p_j, LF_j - p_j] \cap \mathcal{F}_g \mid r_{j,k} \leq \tilde{R}_k(\tau), k \in \mathcal{K}, \tau \in [t, t + p_j[\cap \mathcal{F}_g\} + p_j$

$S_g = S_{g-1} \cup \{j\}$
end
$F_{n+1} = \max_{h \in \mathcal{P}_{n+1}}\{F_h\}$

Tato modifikace je často využívaná v metaheuristických metodách pro vyhodnocení fitness hodnoty jednotlivých jedinců. Vždy totiž existuje seznam λ^*, který po dekódování vygeneruje optimální řešení.

PSGS

Na rozdíl od předchozího algoritmu PSGS postupuje po čase. V každém kroku g je vypočtený čas t_g. Dále pracuje s množinami všech operací, které byly dokončené $\mathcal{C}_g\{j \in \mathcal{J} \mid F_j \leq t_g\}$ do času t_g a množinou všech aktivních operací $\mathcal{A}_g = \mathcal{A}(t_g) = \{j \in \mathcal{J} \mid F_j - p_j \leq t < F_j\}$ v kroku g. Množinu operací, kterou je možné rozvrhnout lze popsat následovně $\mathcal{D}_g = \{j \in \mathcal{J} \setminus (\mathcal{C}_g \cup \mathcal{A}_g) \mid \mathcal{P}_j \subseteq \mathcal{C}_g \wedge r_{j,k} \leq (\tilde{R}_k - \sum_{j \in A(t)} r_{j,k})\}$. Operace v množině D_g splňují podmínky precedence a kapacity zdrojů. Aktuální stav zdrojů v čas t_g lze popsat jako $\tilde{R}_k(t_g) = R_k - \sum_{j \in \mathcal{A}_g} r_{j,k}$.

Výpis 3.3: Paralel Schedule Generation Scheme

$g = 0,\ t_g = 0,\ \mathcal{A}_0 = \{0\},\ C_0 = \{0\},\ \tilde{R}_k(0) = R_k$

while $\mid \mathcal{A}_g \cup \mathcal{C}_g \mid \leq n$ **do**
    $g = g + 1$
    $t_g = \min_{j \in \mathcal{A}_g}\{F_j\}$
    Calculate $\mathcal{C}_g,\ \mathcal{A}_g,\ \tilde{R}_k(t_g),\ \mathcal{D}_g$

    while $D_g \neq \emptyset$ **do**
        Select one $j \in \mathcal{D}_g$
        $F_j = t_g + p_j$
        Calculate $\tilde{R}_k(t_g),\ \mathcal{A}_g,\ \mathcal{D}_g$
    while end

while end

$F_{n+1} = \max_{h \in \mathcal{P}_{n+1}}\{F_h\}$

Algoritmus je rozdělený na dvě části, kde první vypočítá čas t_g a hodnoty množin C_g, A_g a D_g, včetně stavu zdrojů $R_k(t_g)$. Druhá část rozvrhuje podmnožinu aktivit, které je možné začít v čase t_g.

Tabulka 3.2: Ilustrační ukázka řešení pomocí PSGS pro příklad 2.1.

g	**1**	**2**	**2**	**3**
t_g	0	3	3	5
$\mathcal{D}_g$	{1,2}	{2,3}	{2}	{4}
j	1	3	2	4

V tabulce řešení 3.2 v kroku $g = 2$ byl prvně nalezený čas t_g, který odpovídá dokončení aktivity 1. V čas 3 jsou dále vypočítány ostatní množiny. Po výběru operace z $\mathcal{D}_g$ vypočítá

její koncový čas na základně aktuálního času t_g a délky operace p_j. Následně se přepočítají stavy zdrojů, poté se zaktualizují množiny aktivních operací a množina operací, které lze rozvrhnout v čas t_g. V případě, že $\mathcal{D}_g$ je prázdná, se pokračuje dalším krokem.

Pravidla pro výběr aktivity

Z důvodu, že množina $\mathcal{D}_g$ může obsahovat více operací, je zde nutné zavést pravidla, na základě kterých je možné vybrat jednu operaci. Zvolená pravidla velmi ovlivňují, jak bude finální rozvrh vypadat, a jak kvalitní bude jeho řešení. Zvolení vhodného pravidla pro některá řešení může vést k nalezení optimálního řešení, avšak tato situace není příliš častá při použití v praxi. Zde jsou zmíněná pouze některá základní pravidla pro výběr operace.

- MINSLK - Vybere první aktivitu s nejnižší volností. Volnost „slack“ aktivity lze vypočítat $LF_j - EF_j$.
- LFT - Vybere aktivitu s nejnižší hodnotou LF_j.
- LST - Vybere aktivitu s nejnižší hodnotou $LF_j - p_j$.
- SPT - Vybere nejrychleji zpracovanou aktivitu (podle hodnoty p_j).
- LPT - Vybere nejpomaleji zpracovanou aktivitu (podle hodnoty p_j).
- RAN - Vybírá aktivitu náhodně.

Článek zabývající se srovnáním různých pravidel s vyhodnocením chování [4].

3.1.2 Více průchodové metody

Využívají jednoho nebo obou SGS s cílem vytvořit více než jeden rozvrh. Kombinací typu SGS s pravidly výběru operace z $\mathcal{D}_g$ se snaží vytvořit různá řešení a následně porovnáním vyhodnotit nejlepší rozvrh. Nejčastější jsou rozdělené do tří skupin:

Multi priority rule methods

několikrát využívá algoritmus SGS, kde v každém průchodu použije jiné pravidlo pro výběr operace. Často se pravidla řadí sestupně, podle kvality jejich řešení.

Forward-backward scheduling methods

vytváří rozvrh pomocí SGS, kde se v každé iteraci přetáčí směr průchodu mezi dopředným a zpětným. Pro dopředný postup se nic nemění, avšak při zpětném průchodu je třeba otočit vazby mezi rodiči a předky. Poslední operace $n + 1$ se tedy stává startovní a původně počáteční operace 0 se stává koncovou aktivitou.

Sampling methods

pracuje s jedním SGS a s jedním pravidlem. Místo klasického výběru pomocí pravidla je výběr doplněný o náhodnou hodnotu, která ovlivní selekci operace. Za účelem získání více rozvrhů se provádí více iterací.

Sekce o heuristických metodách čerpá z [8].

3.2 Metaheuristické metody

V dnešní době rozsáhle zkoumaná oblast pro řešení obtížných optimalizačních problémů. V sekci jsou uvedené pouze základní možnosti řešení RCPSP. Metaheuristické metody využívají postupů se snahou, co nejvíce se přiblížit optimálnímu řešení aproximací. Při správném použití a implementaci mohou vést k velmi dobrým výsledkům v rozumném čase.

3.2.1 Tabu vyhledávání

Patří mezi metody lokálního hledání. Algoritmus vyhodnocuje sousedy, porovnává je s aktuálním stavem a vybírá vždy nejvhodnějšího souseda. Tento koncept je náchylný na zacyklení, kdy by se opakovaně střídaly dva stavy. Z tohoto důvodu vzniklo omezení zamezující navrácení do bodů, ve kterých už byl v předchozích několika stavech. K tomuto využívá tabu seznam, který slouží jako seznam zakázaných změn. Každá nově provedená změna je umístěna na vrchol tabu seznamu. Seznam má stanovenou pevnou délku a při přesáhnutí velikosti je nejstarší záznam odstraněn. Malá délka tabu seznamu způsobí nebezpečí zacyklení, naopak příliš velká délka omezí prohledávání příliš. Slovní popis průběhu algoritmu pro řešení RCPSP [1]:

1. nastav $k = 1$ a vyber počáteční rozvrh S_1 za použití heuristiky. Ulož $S_{\text{best}} = S_1$.
2. Vyber nového kandidáta na rozvrh z okolních sousedů $S_c \in N(S_k)$, jestliže je změna S_k na S_c zakázaná z důvodu záznamu v tabu seznamu, opakuj krok 2., jinak pokračuj.
3. Jestliže je možná změna S_k na S_c, ulož $S_{k+1} = S_c$ a zároveň posuň dosavadní záznamy v tabu seznamu. Při přetečení kapacity tabu seznamu poslední položku vymaž a vlož nový záznam.
4. Jestliže $F(S_c) < F(S_{\text{best}})$, tak $S_{best} = S_c$.
5. Inkrementuj hodnotu $k = k + 1$, pokud platí koncová podmínka, cyklus ukonči jinak opakuj krok 2..

Podmínky ukončení mohou být různého typu: zadaného počtu iterací, omezení doby po jakou algoritmus může pracovat, nenalezení žádného lepšího řešení po určitém počtu iterací či počet porovnání ohodnocující funkce.

3.2.2 Simulované žíhání

„Simulated Annealing“ (SA) je pravděpodobnostní optimalizační metoda k prohledávání stavového prostoru. Je založena na simulaci procesu žíhání oceli, kde se zahřívá na vysokou teplotu, a postupným pomalým snižováním teploty se odstraňují vnitřní špatné vlastnosti oceli. Algoritmus začíná s náhodně vygenerovaným počátečním řešením. V každé iteraci generuje sousední řešení narušením aktuálního řešení a porovnává aktuální řešení se sousedními. Pokud je nové sousední řešení lepší, převezme se a vyhledává se nové řešení. V případě, kdy sousední řešení je horší, bude vybráno s praděpodobností závisející na rozsahu špatných vlastností řešení a aktuální teplotě. Při průběhu algoritmu se redukuje teplota, a tím se snižuje i pravděpodobnost vybrání špatných sousedních řešení. Více informací lze nalézt v článku o SA v RCPSP [2].

3.2.3 Genetické algoritmy

„Genetic Algorithm“ (GA) jsou inspirované procesy evoluce v biologii. Na rozdíl od uvedených vyhledávacích metod, které pracují pouze nad jedním aktuální stavem, berou genetické algoritmy ohled na populace. Populace jsou tvořeny několika jedinci, kteří představují jednotlivá řešení. Počáteční generace, ze které algoritmus vychází, je možné generovat náhodně nebo za pomocí pravidel. V každé generaci probíhá vzájemné křížení populace, které vytváří nové jedince. Zároveň probíhá mutace stávajících jedinců. Na konci každé generace se vypočítá fitness hodnota jedinců a následně proběhne výběr, který vytvoří novou generaci populace. Fitness hodnota jedince určuje kvalitu, v případě RCPSP jde o čas dokončení rozvrhu. Jelikož se hledá nejmenší možný čas dokončení řešení, znamená to pro GA, že čím menší hodnota fitness, tím je jedinec kvalitnější. Informace jsou čerpané z [5] a [6].

Výpis 3.4: Genetický algoritmus [6]

$G = 1$
`Create initial` POP
`Compute fitness for individuals` $I \in POP$

`while` $G \leq GENMAX$ `do`
    $G = G + 1$
    `Create children` CHI `from` POP `crossover`
    `Mutate children` $I \in CHI$
    `Compute fitness for` $I \in CHI$
    $POP = POP \cup CHI$
    `Reduce population` POP `with selection`
`while end`

Jedinec

Jedinec v GA nese informaci o řešení prohlému. Nejčastější reprezentace jedince při řešení RCPSP je seznam naplánovaných operací tzv. „Activity list“. Vysvětlení podmínek, které musí být pro tento seznam splněny, je v sekci včetně popisu dekódování řešení 3.1.1. O kvalitě jedince rozhoduje fitness hodnota, která v případě RCPSP je doba dokončení projektu tzv. „Makespan“. Pro zjištění fitness hodnoty se využívá SSGS, který rozvrhuje operace podle předloženého seznamu operací.

Počáteční populace

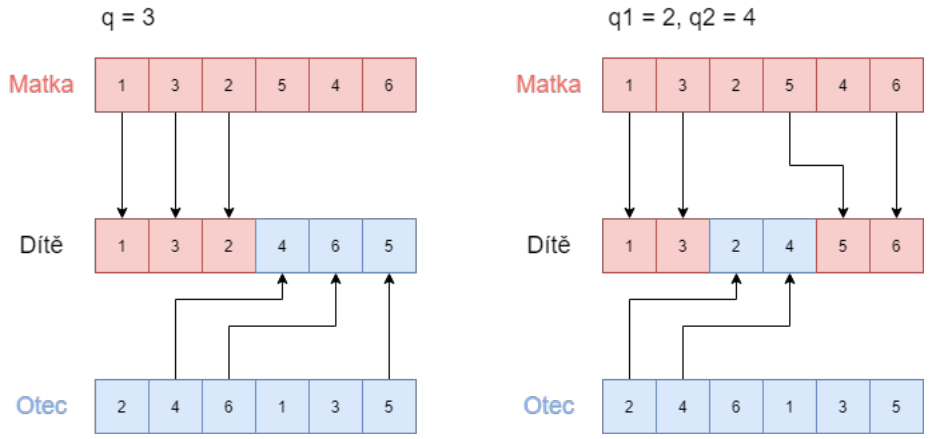
Způsobů, které lze použít pro vytvoření počáteční populace, je více. Nejčastějším způsobem je využití jedné z metod SGS. Následně se určí pravidlo, podle kterého bude SGS vybírat operaci z možných operací připravených k rozvrhnutí v jednotlivých krocích. Nejjednodušším pravidlem je vybrání náhodné operace, avšak pro dosažení lepší kvality lze využít různých pravidel. Velikost populace se v průběhu algoritmu nemění.

Křížení

V případě, že jsou vybráni dva rodiče z populace, dochází ke vzniku nových jedinců tzv. dětí. Tento jedinec je vytvořený z vlastností matky a otce, nese informace od obou rodičů.

Jednobodové křížení je jedním způsobem, jak lze přistupovat k vytváření nových jedinců. Po výběru matky λ^M a otce λ^F se vybere náhodná hodnota q, která je v rozsahu $1 \leq q \leq n$. Na základě této hodnoty se pak následně vybere část $i = 1, ..., q$ od matky λ^M nastavením $j_i^C := j_i^M$, druhá část $i = q + 1, ..., n$ je vybrána od otce λ^F. Zde je ale důležité vybrat pouze operace, které se v dítěti nenachází. Nastavíme tedy $j_i^C := j_i^F$, kde k je nejnižší index takový, že $j_k^F \notin \{j_1^C, ..., j_{i-1}^C\}$.

Pro názornou ukázku je použit příklad z [8], kde seznam operací $\lambda^M = \langle 1, 3, 2, 5, 4, 6]$ a $\lambda^F = \langle 2, 4, 6, 1, 3, 5]$. Poté je vybrána hodnota $q = 3$. Operace $1, 3, 2$ převezme nový jedinec od matky a následně je doplněná o operace v pořadí $4, 6, 5$. Vznikne tak nový jedinec $\lambda^F = \langle 1, 3, 2, 4, 6, 5]$. Platí, že nově vytvořený seznam neporušuje podmínky precedence operací.



Obrázek 3.1: Jednobodové křížení na levé straně a dvoubodové křížení na pravé straně

Další možností je dvoubodové křížení. Tento způsob rozšiřuje jednobodové křížení o další hodnotu q. Generují se dvě hodnoty q_1 a q_2 pro které platí, že $1 \leq q_1 \leq q_2 \leq n$. Následně má nový jedinec část $i = 1, ..., q_1$ od matky λ^M nastavením $j_i^C := j_i^M$. Pro úsek $i = q_1 + 1, ..., q_2$ přebírá po otci λ^F, kde jsou vybrané pouze operace, které se v dítěti nenachází. Nastavením $j_i^C := j_i^F$, kde k je nejnižší index takový, že $j_k^F \notin \{j_1^C, ..., j_{i-1}^C\}$. Pro zbylou část $i = q_2 + 1, ..., n$ se zase přebírá po matce λ^M se stejným pravidlem jako pro výběr z otce. Nastavením $j_i^C := j_k^M$, kde k je nejnižší index takový, že $j_k^M \notin \{j_1^C, ..., j_{i-1}^C\}$. Příklad dvoubodového křížení je na obrázku 3.1 na pravé straně.

Mutace

Mutace je proces, který náhodně mění vlastnosti jedince. K tomu využívá pravděpodobnost $p_{mutation}$, která udává, že se náhodně vybraný gen zmutuje. Pro případ RCPSP se jedná o proces, kdy dojde k prohození dvou operací v seznamu. To může narušit podmínky a lze operaci mutace vykonat pouze v případě, že změna nenaruší podmínku precedence operací v seznamu. Postupně se prochází seznam operací zleva doprava, kde se pro každou operaci na základě pravděpodobnosti $p_{mutation}$ určí, zda bude mutovat. Pokud je možné mutaci provést, prohodí se operace na aktuální pozici s následující operací.

$$\lambda = \langle j_1, ..., j_i, j_{i+1}, ..., j_n]$$

Po provedení mutace v pozici $i \in \{1, ..., n - 1\}$.

$$\lambda' = \langle j_1, ..., j_{i+1}, j_i, ..., j_n]$$

Kapitola 4

Návrh řešení

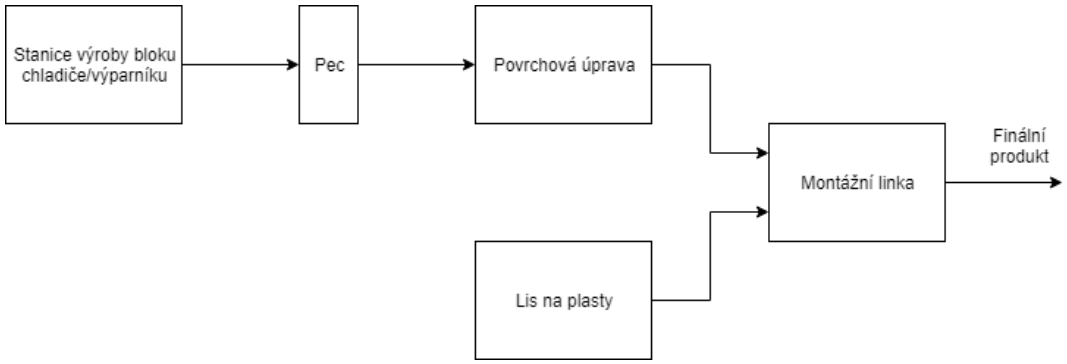
V následující kapitole budou formulované požadavky a návrhy řešení na model a algoritmus, který lze využít pro rozvrhování v průmyslové výrobě. Model bude přizpůsobený a upravený pro výrobní podnik MAHLE Behr Mladá Boleslav s.r.o., který zároveň poskytl data pro otestování na reálném případu. Kapitola je rozdělena na popis řešeného problému. Uvádí informace o průběhu výrobního procesu a reprezentaci jednotlivých operací do formy modelu. V části navrhovaných řešení se popisuje postup, jak jsou požadavky zakomponované do rozvrhu. Ilustrační obrázky rozvrhů sekce 4.1.3 jsou generované rozvrhovacím nástrojem vytvořeným jako součást práce.

4.1 Požadavky na model

Sekce popisuje výrobní postupy, které se používají v podniku MAHLE Behr Mladá Boleslav s.r.o.. V další části jsou popsané jednotlivé požadavky na model RCPSP, které vyplývají z charakteru operací.

4.1.1 Popis výrobního postupu

Výrobní podnik vyrábí různé typy klimatizací a chladičů do automobilů. Ačkoliv se zde vyrábí velké množství různých typů, výrobní proces je pro většinu z nich identický a pro různé typy se mění pouze stroj, na kterém je možné kus vyrobit. Základní proces se skládá z lisování plastových součástí, výroby bloku chladiče či výparníku, spečením součástí bloku, povrchové úpravy bloku a finální smontování.



Obrázek 4.1: Postup výroby chladiče/výparníku

Graf 4.1 reprezentuje základní průběh a vazbu mezi jednotlivými operacemi. Nejčastěji se produkty liší v části povrchové úpravy, kde je možné, že se povrchová úprava provádí vícekrát v různých materiálech. Velká část součástí potřebná k sestrojení finálního produktu je kupovaná a předpokládá se, že bude vždy pro každou operaci dostatek požadovaného materiálu a součástí.

4.1.2 Výrobní operace

V této sekci jsou blíže popsané výrobní operace z odlišných částí výrobního postupu. Navazuje se na ni pak v další sekci 4.1.3, kde jsou detailněji popsané, jak zmiňované vlastnosti ovlivňují model, tak přístup k řešení.

Pro každý produkt z katalogu se výrobní postup liší zdrojem, na kterém je možné produkt vyrobit. Zdroje spadají do určitých skupin, kde v rámci skupiny se vlastnosti příliš nemění a dochází zde pouze k rozdílné délce operací. Vzhledem k velkému počtu zdrojů a operací se v následujících sekcích práce zabývá obecně operacemi a zdroji, a uvádí jaké vlastnosti mají v modelu.

Pro časy operací se využívají informace z výrobních linek. Důležitý parametr je zde způsob, podle jakého se časy operací plánují. Práce bere v potaz tři verze typu plánování času. Všechny časy jsou v systému uváděné v minutách.

- Osobního času - Informační systém uvádí osobní čas pracovníků na výrobu sto kusů při určitém počtu pracovníků. Při přepočtu času operace v modelu je nutné vědět, kolik bude pracujících na pracovišti po dobu práce.
- Strojního čas - Informační systém uvádí strojní čas na výrobu sto kusů. V některých případech je nutné vědět přesný postup plánování, protože uvedený čas je podělený určitou konstantou a nebyl by tak při přepočtu na jeden kus přesný. U strojů, které využívají forem, se čas udává v cyklech formy.
- Čas přestavby - Popisuje délku přestavby na daný produkt v minutách. Nezáleží na počtu kusů a čas není nutné přepočítávat.

Kazetování

Kazetování je úplně počáteční operace, která staví hlavní chladící blok. Vyrábí se podle typu produktu na jednom ze dvou fraktálů AC nebo ET. AC je zaměřený na výrobu bloků do klimatizací. ET se specializuje na výrobu výparníků a chladičů. Pro oba fraktály platí, že výrobní operace mají stejné požadavky, mění se pouze typ zdroje. Stroj prvně nařeže kusy z namotaného plochého železa na určitou délku. Následně pak každý kus projede strojem, který nařezané kusy naohýbá do zvlněného tvaru. Poté se navlněné kusy skládají ručně, nebo strojem do formy pro chladící blok. Proces končí při naplnění formy a následným svázáním žeber.

V modelu a v informačním systému je proces reprezentovaný jedním pracovním strojem. Stroj pak v modelu má kapacitu 1. Není zde nutno řešit více operací zároveň na jednom stroji, protože stroj před výrobou musí být prvně přestavěný a seřízený na určitý typ produktu, který je pak vyráběný. U zdroje je tedy zároveň nutné zavést informaci o přestavbách, více v sekci 4.1.3. Dále je zde uvedené, po jaké pracovní době je nutné udělat přestávku.

Operace mají konečný čas výroby jednoho kompletního chladícího bloku, který se uvádí v informačním systému ke každé formě zvlášť. Operace pro každý zdroj využívají celou jeho

kapacitu, jak je zmíněné v předchozím odstavci. Operaci nepředchází žádné jiné operace a je možné přerušení s nutností zaručení, že se po obnovení provede celá.

Spékání jádra chladícího bloku

Spékání chladícího bloku probíhá pomocí jedné ze dvou pecí na fraktálech AC nebo ET. Po zhotovení chladícího bloku se nechá blok spéct. Při průchodu pecí jsou seskládané části v bloku letované k sobě při vysoké teplotě.

Model s pecí pracuje velmi jednoduše. Zdrojová kapacita každé pece je 1. Vždy se předpokládá, že pec bude připravená předem, tudíž není třeba zavádět žádné přípravy pro operace. Délka operace je dána rychlostí zpracování jednoho bloku. Zahrnuje dobu přeskládání na nosič a dobu průchodu pecí. Využití zdroje pro každou operaci je 1. Před průchodem pecí musí být prvně vyhotovený základní chladící blok, je tedy závislá na operaci kazetování. Operace je zároveň možné přerušit za běhu s nutností zaručení, že se po obnovení provede celá.

Při přepočtu času na jeden kus je nutné brát ohled na kapacitu nosiče. Časy výrobní operace jsou podělené velikostí nosiče. Zde se jedná o konstantu 12.

Zavírání chladícího bloku

Zavírání kompletuje chladící blok přivařením bočnic a vytváří tak hlavní jádro chladiče nebo výparníku.

Pracovní skupina je rozdělena na několik pracovních stanic, kde v modelu je pro každou pracovní stanici zavedený zdroj s kapacitou 1. Před operací musí být vyhotovené jádro chladícího bloku a operace může využívat podle typu různé pracovní stanice. Není zde nutná žádná příprava pro operace a je možné vyvolat přerušení operace, kdy se po obnovení operace dokončí. Délka operace je závislá pouze na typu vyráběného produktu.

Povrchová úprava

U některých případů je nutné provádět povrchové úpravy pro zlepšení vlastností chladiče. Proces je velmi podobný peci, kde místo procházení pecí se chladič ponoří do jednoho z požadovaných materiálů. Operaci je možné provádět vícekrát nad různými typy materiálů. V případě, kdy se pro produkt povrchová úprava opakuje s různými materiály, musí být uvedené přesné pořadí, v jakém se budou povrchové úpravy vykonávat.

Zdroj pro povrchové úpravy v modelu je velmi podobný případu pece, kde kapacita je rovna 1 a není zde nutné brát v potaz přípravu materiálu pro aplikování. Předpokládá se, že bude vždy dostatek požadovaného materiálu pro povrchovou úpravu a vždy bude připravený. Pro operace pak platí, že vždy zaberou celou kapacitu zdroje a doba trvání zahrnuje dobu, po jakou je nutné kus nechat ponořený, přeskládání do nosiče a z nosiče. Operace povoluje přerušení v průběhu s nutností zaručení, že se po obnovení provede celá.

Vodní zkouška

Jedna z možných zkoušek kompletního chladícího bloku. Blok je ponořený do vodní nádrže a je kontrolovaný, zda není kus chybně svařený. Nedostatky při úniku vzduchu se opravují opakovaným svářením.

Na model operace vodní zkoušky ani zdroje nemají speciální požadavky. V systému se uvádí průměrná doba testování a nebere se ohled na opakované opravy sváření. Každá linka

v systému se chová jako zdroj s kapacitou 1, kterou obsazuje pouze jedna operace. Operace může být přerušena a po obnovení se musí dokončit.

Héliová zkouška

Další možnou zkouškou kvality výrobku je héliová zkouška. Narozdíl od vodní zkoušky je podstatně lepší a přesnější. Při héliové zkoušce je nutné stroj připravit na daný typ produktu. Následně jsou produkty naskládané do speciálního nosiče, který projde strojem a vyhodnotí testy.

Stroj je rozdělený na několik komor, kde každá komora v modelu představuje jeden samostatný zdroj, kterému je možné přiřadit operace. Každá komora má kapacitu 1. Přestavby také probíhají na každé komoře zvlášť, je možné provádět dvě různé operace na dvou odlišných komorách zároveň. Operace zabírají celou kapacitu jedné komory. Vyžadují, aby komora byla připravena pro daný typ produktu. Operace může probíhat pouze pokud má vyhotovený blok se všemi požadovanými úpravami. Operaci zkoušky je možné přerušit, musí se ale po obnovení provést celá.

Alternativní možný přístup by byl považovat komory testovacího stroje jako kapacitu. Zde by se podstatně komplikovalo řešení doby příprav pro každou komoru zvlášť. Proto je zvolený první zmíněný přístup.

Lisování

Lisování plastových součástí chladiče nebo výparníku se provádí odděleně od výroby jádra chladiče. Využívá se zde 17 lisů a několik forem pro různé typy plastových součástí. Formy je možné instalovat na jeden nebo více lisů a na základně specifikace. Formy udávají speciální čas pro výpočet doby trvání operace. Určuje dobu trvání jednoho cyklu formy.

Lis je v modelu modelovaný jako zdroj s kapacitou 1. Zároveň využívá definované typy forem jako typ přípravy, u každé formy je uvedený čas instalace, a které operace je možné provádět. Operace zabírá vždy celou kapacitu zdroje a je pro každou operaci uvedeno, na kterých všech zdrojích, s jakou formou lze operaci vykonávat. Operace lisování nemá žádné předchůdce. Lisování v průběhu je možné přerušit, pokud je možné po obnovení operaci dokončit.

Některé formy umožňují lisování více kusů produktu najednou. Nastává tak speciální případ, kdy se čas lisování jednoho produktu nemění a podle počtu vylisovaných produktů jednou operací se upraví konečná délka operace.

Sestavení finálního produktu

Posledním výrobním procesem je sestavení výparníku nebo chladiče. Součástí tohoto procesu je i celkový test finálního produktu. Proces probíhá pásově, postupným sestavováním, na jednotlivých stanicích výrobní linky. Požaduje všechny komponenty, které jsou tvořené v předchozích krocích.

Model zahrnuje celou linku jako jeden zdroj, který má kapacitu 1. Čas zpracování operace závisí na počtu pracovníků na lince a vždy zabírá celou kapacitu přiřazeného zdroje. Operaci předchází všechny předešle zmíňené operace na základně typu produktu. Operaci přerušit lze pouze v případě, že je možné po obnovení zdroje dokončit operaci.

Odstávky

Speciální případ operace, který uvede stroj do stavu, ve kterém není možné vyrábět. V praxi může být důvod odstávky, například oprava stroje, kontrola nebo seřízení. Jsou předem plánované do provozu. Práce nebere v potaz odstávky v podobě náhlé poruchy strojů a jejich opravy během samotné výroby. Pouze poruchy, které jsou známé před začátkem plánování a jsou uvedené v informačním systému.

Operace vždy zabere na stroji celkovou kapacitu a není možné v čase odstávky další operace provádět. Precedence operace se zde neřeší. Operace musí přesně dodržet dobu, ve které začínají a kdy končí, proto není možné u odstávek vyvolávat přerušení, či jakkoliv modifikovat čas nebo průběh. Všechny zdroje v případě této práce umožňují rozvržení předem naplánovaných odstávek.

Přípravy

Příprava je další speciální případ operace, která není přímo definovaná ve výrobních operacích. Slouží k připravení zdroje do stavu, ve kterém je možné provádět operaci. U různých zdrojů se příprava může v praxi lišit podle vykonávané operace. Pro model je pouze důležitá doba přípravy, a jaké možné operace je po přípravě možné vykonávat. U některých zdrojů přípravy mohou výrazně ovlivnit konečný rozvrh, kde délka samotné instalace může být podstatně delší, než na ní závislá operace.

V modelu má každá operace uvedenou přípravu, kterou po zdroji požaduje a při rozvrhnutí se bere ohled na obě dvě. Precedence není u přípravy řešená.

4.1.3 Požadavky

Ačkoliv je model RCPSP velmi kvalitní, je nutné některé části modifikovat, aby byly splněné požadavky z praxe. V následující části se práce věnuje návrhům, jak lze části modelu transformovat pro využití nad popsaným výrobním postupem v sekci 4.1.1.

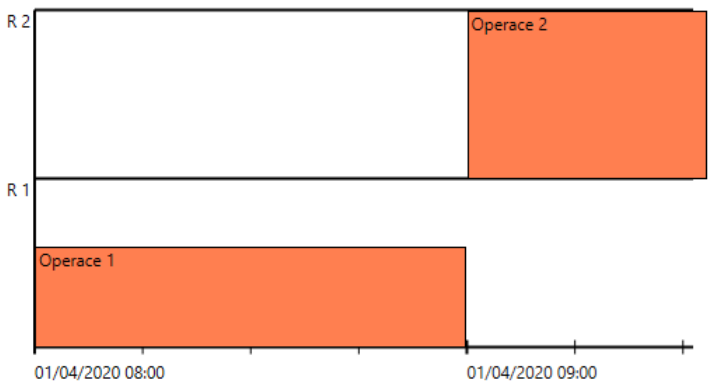
Reálný čas

V základním modelu RCPSP se nebere ohled na čas. Délky operací nemají stanovenou jednotku a zároveň není jasné, od kdy se začíná plánovat. Jako řešení tohoto problému lze přidat datum začátku projektu, od kterého se má začít plánovat. Dále je nutné určit, kterou časovou jednotku bude délka operace p_j reprezentovat v modelu. Z důvodu délky jednotlivých operací jsou nejideálnější sekundy, aby nedošlo k zaokrouhlování z důvodu práce s celými čísly. Ukázka Ganttova diagramu 4.2 s reálným časem. Pro výpis je použitý formát „DD/MM/YYYY hh:mm:ss“.

Zároveň je zde předpoklad, že se nebude plánovat s překročením 24 dnů dopředu, aby nedošlo k přetečení datového typu.

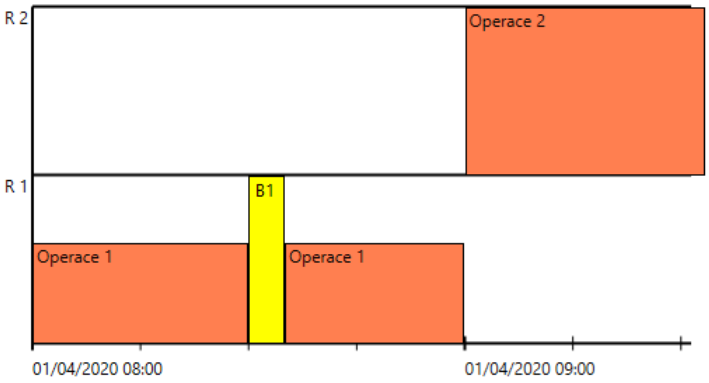
Odstávky

V praxi existují různá omezení s prací se zdrojem. Zdroje v podobě strojů se mohou při dlouhodobém neustálém provozu přehřívat, a tak je možné v kuse pracovat jen po určitou dobu. Další případ může být porouchaný stroj, na kterém probíhá oprava. V případě lidských zdrojů musí plán dodržovat určitá nařízení ohledně pracovní doby. To vede na požadavek vytvoření přestávek. Přestávka je operace, která se vygeneruje po určité době



Obrázek 4.2: Ganttův diagram s ukázkou plánu s reálným časem

provozu na zdroji. Obrázek 4.3 znázorňuje přestávku na zdroji R1, která proběhne po půl hodině práce.



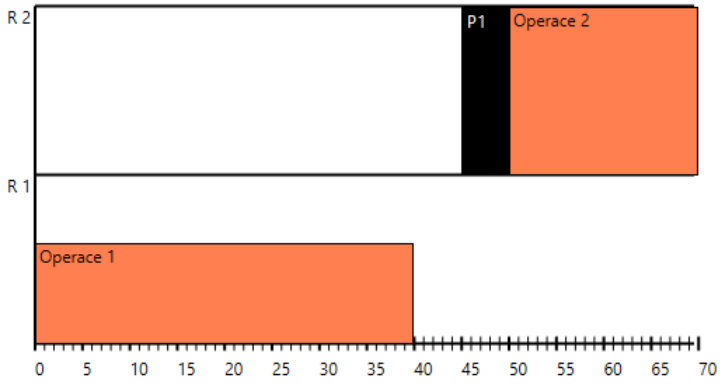
Obrázek 4.3: Ganttův diagram s ukázkou přestávky

Dalším typem je plánovaná odstávka, která definuje, kdy bude zdroj nepřístupný a po jakou dobu jej nelze využívat. Případ lze využít i pro omezení pracovní doby, jako jsou například dovolené, svátky.

Přerušení operací

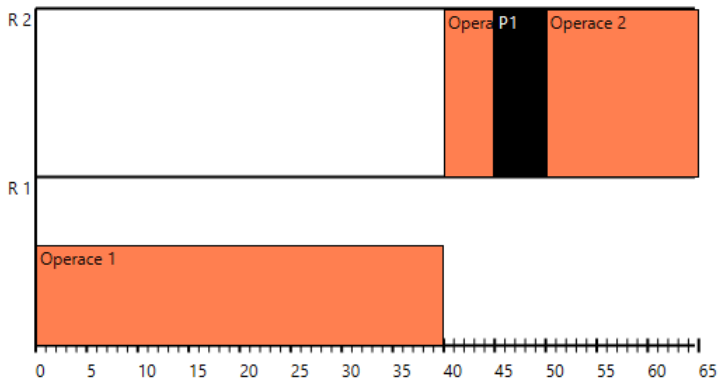
Při rozvrhování je důležité zohlednit doby, kdy je přístroj mimo provoz. Takový případ může nastat z důvodů plánované odstávky nebo pracovního volna. Je nutné, aby operace neprobíhaly v čas, kdy není dostupný požadovaný zdroj. To může vést k nevyplněným místům v rozvrhu při nemožném přerušení operace. Ilustrační obrázek 4.4 znázorňuje příklad, kde v čase 45 je naplánovaná odstávka zdroje R2 na dobu 5. Operace 2 ačkoliv by mohla být rozvrhnuta v čas 40, musí z důvodu odstávky, která by zasáhla do jejího průběhu počkat na její dokončení, teprve pak je možné operaci rozvrhnout.

Povolením přerušení některých operací, pro které je to možné, lze zabránit případům, kde by se z důvodu nedostatku času na dokončení, díky odstávkám zbytečně protahovala



Obrázek 4.4: Ganttův diagram rozvrhu s nemožností přerušení operace 2

doba, za kterou je možné rozvrh vykonat. Povolení přerušení u operace 2 znázorňuje obrázek 4.5. Na rozdíl od předchozí situace se operace 2 rozdělí na dvě části. První část začne před odstávkou v čase 40 a probíhá až do doby odstávky, která operaci pozastaví na dobu 5. Po ukončení odstávky se operace 2 dokončí.

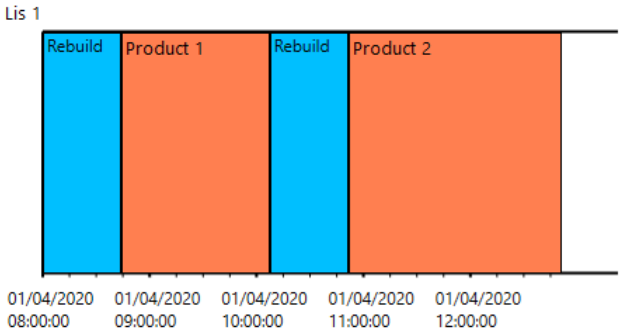


Obrázek 4.5: Ganttův diagram rozvrhu s umožněním přerušení operace 2

Vztahy mezi operacemi

Některé stroje požadují přípravu na vykonání určité operace. V praxi se to týká například lisů, které využívají forem a je možné v rámci jednoho lisu využívat různých typů. Při rozvrhování operací je třeba zohlednit, zda operace využívá formu, která byla použita při předchozí operaci. V modelu je nutné vést informaci o tom, jaká operace vyžaduje typ přípravy. Řešením je uvádět u stroje, v jaký čas byla provedená předchozí příprava, a v závislosti na této informaci rozvrhovat následující operace. Zároveň u každého stroje je nutné uvádět seznam možných příprav s její délkou.

Obrázek 4.6 naznačuje průběh s přestavbami na jednom z lisů výrobní linky. Pro operaci „Product 1“ je nutné prvně vyměnit lisovací formu a poté je možné začít vyrábět plánovaný



Obrázek 4.6: Ganttův diagram rozvrhu s přestavbami

výrobek. Stejně tak po dokončení operace „Product 1“ je nutné znovu vyměnit formu na zcela jiný výrobek.

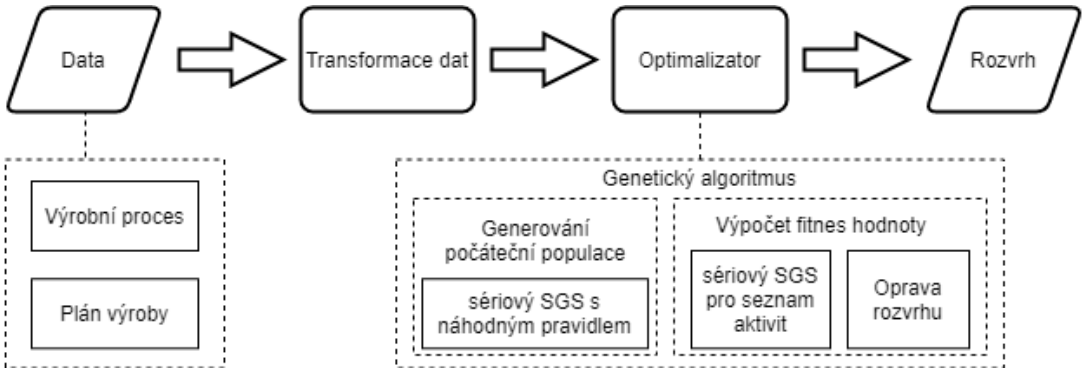
Formy se dají v praxi aplikovat na více strojů a tudíž je i v návrhu toto nutné zahrnout. Pro každou operaci je nutné uvádět informaci, jaké formy jsou pro ní dostupné a zároveň tak upravovat i dobu zpracování operace. Použití jedné formy na dvou různých strojích může vést na různé doby trvání operace a rozdílné využití kapacit zdroje.

4.2 Návrh algoritmu

Následující část práce popisuje, jak je algoritmus navržen a z jakých částí se skládá. Popis přístupu k jednotlivým požadavkům popsaným v předchozí sekci a možnostech řešení. V rámci práce se předpokládá, že všechny informace budou předem dostupné a v průběhu nebude docházet ke změnám. Pracuje se s předem definovaným balíkem dat.

4.2.1 Obecný návrh

Při návrhu algoritmu bylo důležité upřesnit si požadavky a možnosti popsaných metod v části 3. Poté vybrat nejlepší možnou metodu a následně jí poupravit. Další část algoritmu se stará o přetransformování reálných dat z informačního systému do podoby, ve které je možné s nimi pracovat v rámci modelu. Jednotlivé struktury, se kterými algoritmus pracuje, jsou inspirované knihovnou testovacích úloh PSPLIB. Některé části pak lze pomocí dat z knihovny jednoduše otestovat.



Obrázek 4.7: Průběh algoritmu

Průběh algoritmu je naznačený na obrázku 4.7. Data se skládají ze dvou částí. Data výrobního procesu obsahují důležité infomace k jednotlivým operacím a zdrojům, definují délky operací, jejich návaznosti a další požadavky, které se při rozvrhovaní využívají. Data plánu výroby zahrnují i infomace o tom, který produkt se má vyrobit v jakém množství. Uvádí se zde doba expedice, která udává konečný čas, do kterého musí být plán splněný, a datum, od kterého se má plán začít provádět.

Data se následně transformují do podoby, se kterou může dál optimalizátor pracovat. Inicializují se zde parametry a vytvoří se model výroby podle nahraných dat. Optimalizátor následně za pomocí genetického algoritmu hledá řešení pro zadaný problém.

4.2.2 SGS

Při rozhodovaní, kterou z modifikací je vhodné zvolit, je brán ohled na požadovanou celkovou funkcionalitu systému. Ačkoliv by paralelní SGS bylo možné využít, a u některých požadavků z reálné situace by jeho přístup ulehčil. Hlavně v případě, kdy je nutné plánovat, postupně po čase, například při vyhodnocování odstávek, nebo při potřebných přípravách zdroje. Hlavní nevýhodou avšak je, že při využití optimalizátoru se podstatně komplikuje dekódování jednotlivých jedinců. Proto byl pro práci zvolený sériový SGS. Velmi snadno ho lze zakomponovat do optimalizátoru, díky jeho rozšíření s dekódováním seznamu operací.

SGS se využívá ve dvou modifikacích. První modifikace s náhodným výběrem operace z rozhodovací množiny $\mathcal{D}_g$. Náhodně se taky vybírá mód operace, ve kterém bude operace probíhat. Tato modifikace slouží pro generování počáteční populace v optimalizátoru. Tento přístup vede na dostatečně různorodou počáteční populaci. Průběh metody není v tomto případě nutné příliš měnit 3.1.1. Jediná změna pro účel práce je přidání výběru náhodného módu operace.

Druhá modifikace se využívá při výpočtu fitness hodnoty jednotlivých jedinců v optimalizátoru. Zde se návrh podstatně liší, pouze první část je stejná pro vytvoření počátečního rozvrhu. Využívá se sériového SGS pro seznam aktivit, kde se vytvoří základní rozvrh. Algoritmus zůstává stejný jako v popsané části 3.1.1. Základní rozvrh se poté transformuje opravou.

4.2.3 Oprava rozvrhu

Jak už bylo zmíněné v předešlém textu - z důvodu, že sériový SGS plánuje po aktivitách, neuvádí se zde tak čas, ke kterému by bylo možné si zapamatovat stav zdrojů. Následující sekce popisuje jednotlivé problémy a jak je navržené řešení.

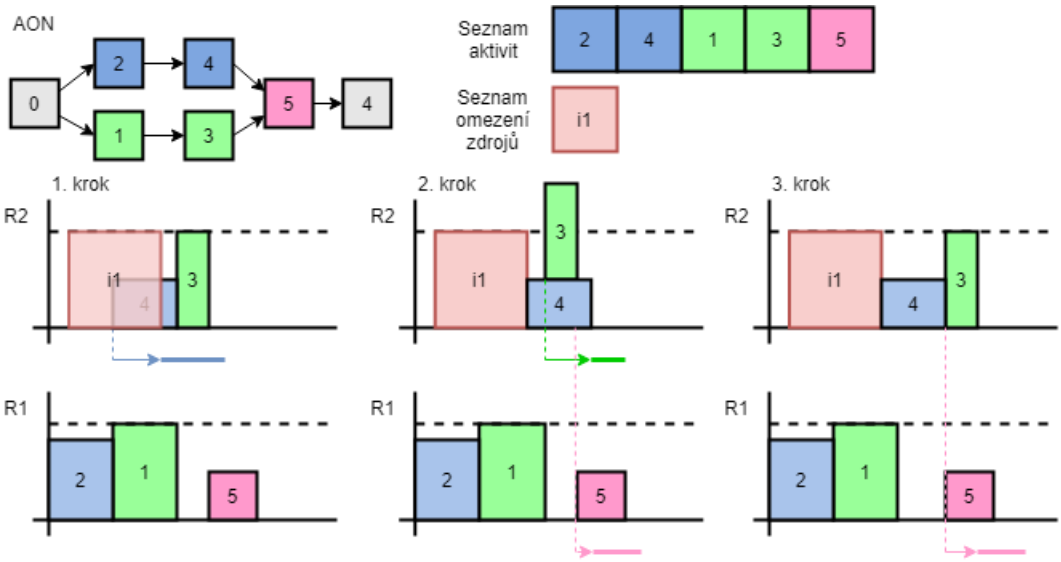
Při opravě se opakovaně prochází rozvrhem a postupně se doplňují chybějící informace. Podle potřeb se rozvrh dále transformuje. Důležitou podmínkou je zachování pořadí operací. To umožňuje pracovat s rozvrženými operacemi na jednotlivých zdrojích, kdy je doplněná chybějící informace o předchozích operacích z metody sériového SGS. Úpravy je vhodné provádět v určitém pořadí, aby nedocházelo ke zbytečným zpětným kontrolám.

Oprava operací

Následující úpravy rozvrhu mají podstatný vliv na průběh operací. Při změně rozvrhu je nutné transformovat operace, které jsou popsané v následující části. Tato část se věnuje přístupu k jednotlivým změnám a jaké mohou nastat případy. Z důvodu, že zvolené metody vždy rozvrhují operace na co nejdřívější možný čas se stává, že při přidání nových operací

se musí staré posunout a vytvořit tak dostatečné místo. Při posunu operace nesmí dojít k porušení žádného z pravidel. Rozvrh tak zůstane po změnách stále validní.

Při posunu operací byl zvolený rekurzivní přístup. V době, kdy nastane posun u některé z operací, se kontrolují i všechny závislé operace. Závislé operace jsou všechny operace, u kterých by posun porušil některé z pravidel.

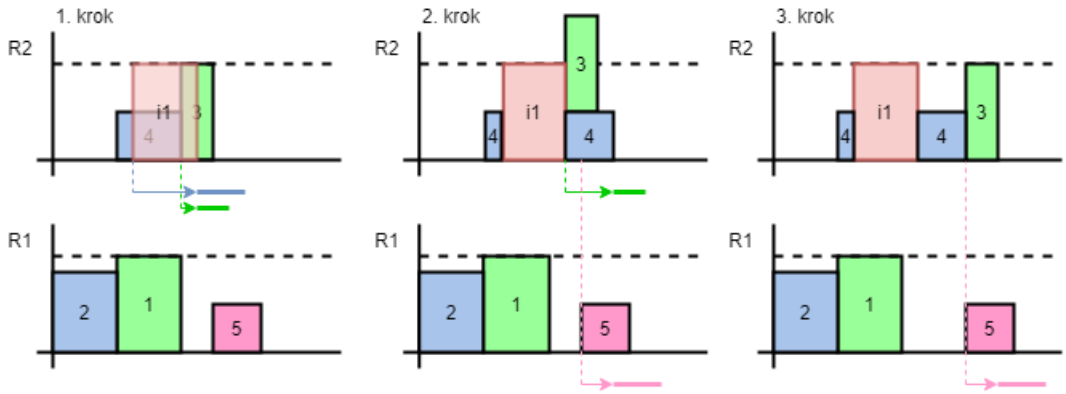


Obrázek 4.8: Ganttův diagram rozvrhu a precedenční síť s posunem operací

Jak rekurzivně probíhá posun operací v rozvrhu ilustruje 4.8. Do původního rozvrhu se přidáním operace i1 poruší podmínka využití kapacity zdroje. Z tohoto důvodu je nutné operaci 4 posunout směrem doprava. V prvním kroku se určí, jak se operace 4 v rozvrhu posune a zda konec posunuté operace nepřekračuje maximalní povolený čas dokončení projektu. V případě, že by byla tato podmínka porušena, bude rozvrh označen za neplatný. Naopak se operace posune a probíhá opakovaná kontrola, zda se posunem opětovně neporušilo některé z pravidel. V kroku dva jsou porušená obě pravidla, kde operace 4 a 3 překročí maximální kapacitu, zároveň zde dojde i k porušení precedence, kdy operace 4 není dokončená před začátkem operace 5. Posun se provede prvně u operace 3 a dále se provede posun pro operaci 5. Rekurzivní kontrola pak zaručí, že se pro následující posunuté operace opět překontrolují podmínky pro každou zvlášť. V kroku tři se zanoření posunu v operaci 3 ukončí, protože jsou splněné všechny podmínky. Naopak pro operaci 5 je porušena podmínka precedence a musí se opakovaně posunout doprava.

Přerušení operací popisuje obrázek 4.9. Platí zde stejné precedenční podmínky, jako u předchozího obrázku. Změněný je čas začátku a doba omezení zdroje i1. Ukazuje tak případ, kdy je možné provést část operace 4 před začátkem odstávky. Pokud by operace 4 neměla povolené přerušení, musela by se posunout až za konec odstávky. V ukázkovém případu je možné operaci 4 přerušit. Vypočte se čas, který se před přerušením operace provede. V dalším kroku se operace 4 rozdělí na dvě nové operace. Obě operace zachovávají stejné precedenční požadavky, pouze se upraví využití zdroje a délka operací. Zároveň se pro pokračující operaci změní i čas zahájení podle konce překážející operace. V kroku jedna je operace rozdělí na dvě části. První části se pouze upraví čas zpracování. Druhou je nutné posunout na konec odstávky. Po posunutí se dále kontroluje porušení podmínek a pokračuje se stejně, jako je popsané u přesouvání operací.

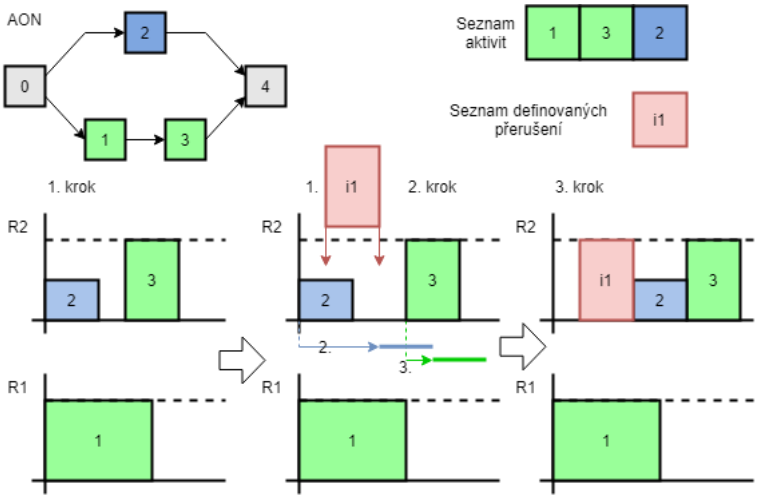
Obecně se vždy nazačátku kontroluje, zda je možné operaci přerušit. Přerušení nenastává v každém případě, kdy se operace z části překrývají. Vyvolání přerušení u výrobní operace musí splňovat podmínku, že začala před blokující operací a zároveň konec operace je za začátkem blokující operace. Zabrání se nesmyslnému prohazování a rozdělování operací na malé části.



Obrázek 4.9: Ganttův diagram rozvrhu a precedenční síť s přerušením operací

Přidání odstávek

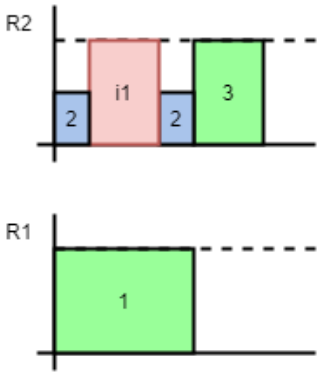
První úprava, která se nad rozvrhem provádí, je přidání odstávek. Odstávky totiž ovlivní další navazující operace a je důležité vědět při dalších opravách, zda v čase zrovna odstávka neprobíhá. Není třeba dále provádět zpětnou kontrolu při každé přidané změně, zda nebyla přidaná nová odstávka. V práci jsou stanovené dva typy odstávky. První typ má přesně stanovený začátek operace a dobu, po jakou bude stroj blokovaný. Tento typ se do rozvrhu přidává jako první.



Obrázek 4.10: Ganttův diagram rozvrhu a precedenční síť s kroky opravy rozvrhu při přidání odstávky

Obrázek 4.10 ukazuje přístup při dosazování odstávky zdroje do rozvrhu. V kroku jedna je vygenerovaný základní rozvrh pomocí sériového SGS. V kroku dva prochází seznam

přerušení zdrojů a postupně se doplňují do rozvrhu. Při doplnění dochází k několika krokům. Na začátku vloží rozvrhovací systém odstávku do rozvrhu na jeho počáteční čas a zabere zdroj po jeho dobu trvání. Následně se dohledají všechny operace na zdroji, které probíhají v čas odstávky. Pro každou operaci následuje úprava podle podmínky, zda je možné operaci přerušit. Obrázek ukazuje případ, kdy operace 2 není možné přerušit a musí být provedena najednou celá. V tomto případě se vypočítá, o kolik je nutné operaci posunout doprava a následně se upraví její čas začátku. Všechny navazující operace na zdroji a z precedenční vazby se rekurzivně posouvají taktéž doprava podle vzniklého překryvu.



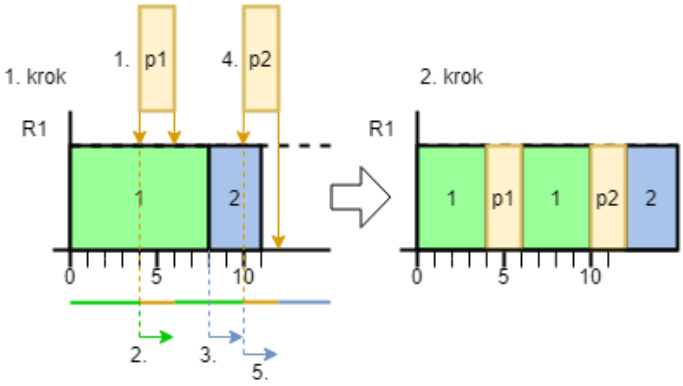
Obrázek 4.11: Ganttův diagram rozvrhu s povolením přerušení

Alternativní přístup s povolením přerušení operace 2 naznačuje výsledný rozvrh obrázek 4.11. Operace 2 se rozdělí na dvě části, první se provádí v čas, kdy byla operace původně rozvrhnuta. Druhá část provede zbytek operace v čase po ukončení plánované odstávky.

Další typ speciální odstávky je přestávka. Vlastností přestávky je, že musí být rozvrhnuta po nějaké době práce zdroje. Přestávky mají smysl pouze na zdroji, které operace mají povolené přerušení. Charakter výrobního podniku, kterým se práce zabývá, jsou všechny výrobní operace přerušitelné a nevzniká zde problém, kdy by operace požadovala přestávku, ale nebylo by povolené ji přerušit.

Doplňení přestávek do rozvrhu probíhá v dalším průchodu po přidání všech pevně daných odstávek. Proces přidání přestávky do rozvrhu je velmi podobný. Rozdílná část je v tom, kdy se přestávka má na příslušný zdroj přidat. Každý zdroj má definovaný čas, po kterém je nutné provést přestávku. Musí se procházet operace rozvržené na zdroji a kontrolovat, zda nedošlo k případu, že je nutné přidat přestávku.

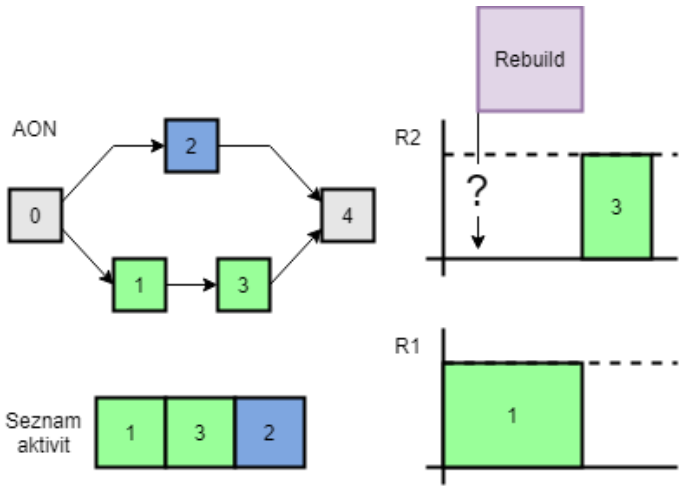
Průběh přidávání přestávek na zdroj je zobrazen na 4.12. Po vytvoření základního rozvrhu se prochází po využívaných zdrojích a kontroluje se potřeba vytvořit přestávku. Pro zdroj R1 v obrázku platí, že po každé čtvrté periodě se musí pozastavit po dobu dvě periody. Postupně se prochází po rozvrhnutých operacích a kontroluje se podmínka. Pokud dojde ke splnění podmínky, vytvoří se nová operace a rozvrhne se. Následuje kontrola, při které se hledá, zda je nutné přerušit operace, které zasahují do nově rozvrhnuté operace. V případě splněné podmínky se přeruší operace, která do přestávky zasahuje a obnoví se po dokončení přestávky. Musí se také posunout všechny následující operace směrem doprava, včetně precedenčně závislých operací. Následně se pak pokračuje od konce nově vytvořené operace a opakuje se kontrola, zda proběhla dostatečná doba. Minimální doba, po které může být přestávka vytvořená, je omezená na dobu jedné periody a délka přestávky může být minimálně jedna perioda.



Obrázek 4.12: Ganttův diagram rozvrhu s přestávkami

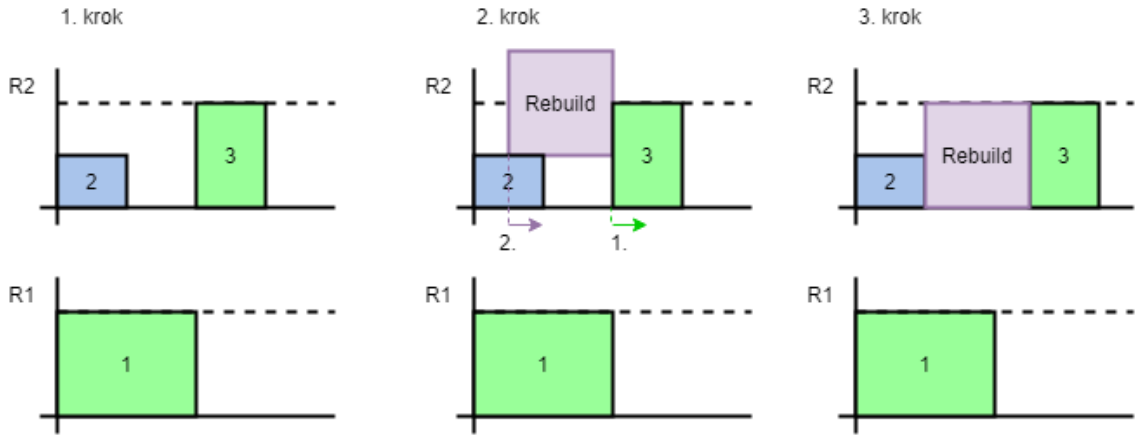
Přidání přípravy zdroje

Obrázek 4.13 ukazuje postup řešení v případě přidávání operace přípravy v sériovém SGS. Operace 3 potřebuje k vykonání určitý typ přípravy na zdroji. Při rozvrhování operace 3 dojde k problému, že algoritmus neví, zda je nutné provést přípravu nebo ne. Na otázku v základním sériovém SGS nelze odpovědět, protože algoritmus pracuje pouze s právě rozvrhovanou aktivitou a neví, v jakém stavu se zdroj nachází v předchozích časech. Možností by bylo přidat zpětnou kontrolu do algortimu, ale to by mohlo vést na podstatně velkou změnu průběhu algoritmu. Snahou bylo zanechat původní postup a pouze ho rozšířit.



Obrázek 4.13: Ganttův diagram rozvrhu a precedenční síť s přestavbou

Obrázek 4.14 znázorňuje, jak lze přistupovat k tomuto problému. Prvně se vytvoří v kroku jedna základní rozvrh pomocí sériového SGS pro seznam aktivit a následně v kroku dva se pomocí opakovaného procházení přidávají speciální operace do rozvrhu. Vždy se prvně vyzkouší možnost přidat přípravu hned před alokovanou operaci. V případě, že to možné není, se původní operace posune v rozvrhu doprava tak, aby se vytvořilo dostatečné místo pro přípravu. Při opakovaném průchodu operace 3 už ví, kdo je jejím předchůdcem na stroji. Umožní se vyhodnocení potřeby aplikovat přípravu na zdroji.



Obrázek 4.14: Ganttův diagram původního rozvrhu a rozvrhu s opravou

Druhou možností řešení by bylo přípravu vždy rozvrhnout na počáteční čas příslušné navazující operace. Zde nastává případ, že se v rozvrhu nevyužíjí prázdná místa. Tento přístup je nutné využít v případě, že příprava zdroje by byla závislá na předchozí operaci. Pro případ této práce není nutné brát přípravy předchozí operace v potaz. Vždy se jedná pouze o přenastavení zdroje na výrobu jiného typu produktu.

4.2.4 Optimalizátor

Optimalizátor pracuje nad generovanými rozvrhy pomocí SGS. Rozvrhy ohodnocuje a na konci procesu vrací nejlepší nalezené řešení. Kvalita výsledného rozvrhu záleží na stanovených parametrech a na rozsahu právě řešeného problému.

Navrhovaný algoritmus zahrnuje optimalizaci pomocí genetického algoritmu s využitím předem zmíněných metod SGS. Při rozhodování reprezentace řešení v genetickém algoritmu je zvolený seznam operací, který zahrnuje i vybraný mód každé operace. Díky sériovému SGS pro seznam aktivit, se dá velmi jednoduše získat fitness hodnota jednotlivých jedinců. Ten generuje pouze základní fitness hodnotu bez přidávaných rozšíření rozvrhu. Finální hodnota jedince je reprezentovaná hodnotou rozvrhu po opravě. Pro hodnoty v modelu platí, že čím je fitness hodnota menší, tím je jedinec silnější (kvalitnější). Pro vytvoření počáteční populace se využívá sériový SGS s náhodným výběrem operace z rozhodovací množiny. Zároveň se pro každou operaci náhodně vybere jeden z příslušných módů.

Princip původního genetického algoritmu nebylo třeba měnit. Při implementaci jednotlivých rozšíření jde pouze o úpravu rozvrhu generovaného metodou SGS. Tento problém je řešený zastřešením algoritmu SGS s opravou do jednotného navazujícího algoritmu, který vrací správný rozvrh s reálnými požadavky.

Kapitola 5

Implementace a testování

Kapitola popisuje způsob implementace rozvrhovacího systému, na kterou navazuje způsob testování a vyhodnocení výsledků. Kapitola na začátku uvádí využité technologie, které byly využity při implementaci. Na tuto část navazuje testování navrženého rozvrhovacího algoritmu a vyhodnocení. Kapitola je zakončena shrnutím testů nad reálnými situacemi. Při testování byl využit procesor I7-6700K @ 4.00 GHz a 16 Gb ram. Testování probíhalo na operačním systému Windows 10.

5.1 Implementace rozvrhovacího systému

Vlastní rozvrhovací systém je implementovaný pomocí jazyka C# na .NET frameworku 4.7.2.. Důvodem pro zvolení jazyka C# bylo využití systému jako komponenty do informačního systému. Využívá se zde SQL Server databázového systému pro získávání dat o výrobním procesu. Práce nevyužívá žádné nestandardní knihovny. Pro spuštění stačí připojení k výrobní databázi se striktně danou strukturou a standardní .NET framework 4.7.2..

Celý systém je objektově orientovaný. Využívá se při implementaci různých OOP praktik. Systém je rozdělený na několik částí, kde každá se chová jako samostatná knihovna. První z nich se zabývá nahráním dat z databázového systému a naplněním datových struktur pro další práci. Druhá část implementuje jednotlivé logické prvky a algoritmy, které se pak dále využívají v optimalizátoru. Je možné je využít i samostatně. Implementuje modifikované SGS metody a následnou opravu rozvrhu. Třetí částí je optimalizátor, který se stará o vyhodnocování a hledání řešení. Poslední část slouží pro základní vizualizaci rozvrhu. Vizualizace je implementovaná pomocí jednoduché WPF aplikace.

Při spuštění vizualizace je možné přímo vypočítat rozvrh pro jednu z objednávek ze systému, nebo načíst data z knihovny PSPLIB. Optimalizátor vypočítá rozvrh, který se pak zobrazí ve formě Ganttového diagramu.

5.1.1 Implementace SGS

Při implementaci metod SGS je využité rozhraní `IScheduleGenerationSchemes`, které definuje metodu `Generate()`. Tato metoda přijímá dva parametry a návratová hodnota je typu `Schedule`. Prvním parametrem je datový objekt typu `ProjectInformation`, který přenáší informace o částech projektu jako definice operací, zdrojů a spousty dalších důležitých dat pro rozvrhovaní. Druhý parametr je seznam operací. Parametr není při každém volání potřebný a je možné ho vynechat. Z rozhraní `IScheduleGenerationSchemes` dědí jeho metody abstraktní třída `ScheduleGenerationSchemes`. Obsahuje pomocný objekt pro

kontroly zdrojů a precedencí operací rozvrhu `ScheduleGenerationSchemesHelper`. Jednotlivé realizace metod SGS pak pouze požadují definování těla funkce `Generate()`, kde záleží na typu zvolené metody.

5.1.2 Implementace oprav rozvrhu

Všechny požadavky a rozšíření nad rámec základního modelu RCPSP obstarává objekt `PostProcessor`. Ten využívá dva pomocné objekty `JobManager` a `FormManager`. Objekt `JobManager` definuje funkce pro úpravu operací v rozvrhu a `FormManager` definuje funkce, které kontrolují a opravují průběh operací na jednotlivých linkách a doplňují nebo opravují operace rozvrhu, které jsou typu `JobType.Rebuild`. Při doplňování omezení zdrojů se využívá statického objektu `JobFactory`, který má za úkol doplnit do rozvrhu operace, které nějakým způsobem omezují průběh.

Nejdůležitější funkce při provádění oprav je `JobManager.FixScheduledJobs()`. Ta má za úkol kontrolu operací v rozvrhu a případně vzniklý problém řeší posunem, nebo rozdělením operace. O to se stará funkce `FixJob()`.

5.1.3 Implementace Ganttova diagramu

V průběhu programování práce bylo zapotřebí průběžně kontrolovat výsledky a testovat různé případy. Data bylo vhodné převést na čitelnou formu pro lepší kontrolu. Optimalizátor pouze vrací seznam operací postupně seřazených. Tento přístup není příliš čitelný, proto bylo důležité hledat další alternativy pro vyhodnocování rozvrhů. Nejpopulárnější přístup je Ganttův diagram, který graficky znázorňuje průběh operací na zdrojích. Ačkoliv je poměrně velké množství programů pro vytváření Ganttova diagramu, rozhodl jsem se pro účely jednoduché modifikace implementovat vlastní vizualizace. Výhodou je možnost následných úprav podle požadavků a netřeba omezovat se už specifikovanou implementací.

Jádro implementace tvoří WPF prvek `Canvas`, na který pomocný objekt `GanttDraw` vykresluje položky z vypočteného rozvrhu. Při vytváření Ganttova diagramu pro případy dat z knihovny PSPLIB se v některých případech operace zobrazují mimo limit zdroje. Je to pouze vizuální chyba. Součástí diagramu je i výpis důležitých informací v horní části.

5.1.4 Implementace optimalizátoru

Jádro optimalizátoru tvoří objekt `GeneticAlgorithm`. Využívá pět rozhraní, kde každé rozhraní definuje metodu z návrhu optimalizátoru. První rozhraní `IPopulationCreator` definuje metodu `Create()`. Metoda požaduje parametr typu `int` a představuje velikost tvořené populace. Druhé rozhraní `IFitnessCalculator` definuje metodu `Calculate()` pro výpočet fitness hodnot. Jako vstupní parametr požaduje objekt typu `Population`, který představuje populaci v jednom kroku algoritmu. Vytváření nových jedinců populace má na starost rozhraní `ICrossover`, které definuje metodu `Crossover()` s parametrem populace. Obdobně tak rozhraní `IMutation` zprostředkovává metodu `Mutate()`, která má na starost mutaci nových jedinců populace. Poslední rozhraní je `ISelector`, které slouží pro výběr jedinců z nového stavu populace do další generace. K tomu využívá metodu `Select()`, která jako jediný parametr požaduje populaci.

Tento přístup výrazně zjednodušil testování různých přístupů a jednotlivých částí. Objekty, které implementují chování různými způsoby, tak lze jednoduše zaměnit. Konstruktor optimalizátoru požaduje implementované objekty rozhraní. V implementaci výpočtu fitness

Tabulka 5.1: Testované parametry optimalizátoru

Parametr	Hodnoty
Velikost populace	30; 50; 100
Počet generovaných rozvrhů	1000
Pravděpodobnost křížení	0,1; 0,5; 0,7; 0,9
Pravděpodobnost mutace	0,05; 0,1; 0,25
Typ křížení	jednobodové, dvojbodové
Metoda	Sériový SGS

Tabulka 5.2: Tabulka výsledků testu nad J301_1

Velikost populace	**Typ křížení**	**Prav. křížení**	**Prav. mutace**	**Průměrná odchylka**	**% odchylka**
100	Jednobodové	0,9	0,25	0,14	0,33%
100	Dvoubodové	0,9	0,25	0,18	0,42%
50	Dvoubodové	0,9	0,25	0,4	0,93%
50	Jednobodové	0,7	0,25	0,45	1,05%
30	Dvoubodové	0,9	0,25	0,96	2,23%
30	Dvoubodové	0,7	0,25	1,23	2,86%

hodnoty se využívá rozhraní `IScheduleGenerationWrapper`, které zabaluje implementaci metody SGS a `PostProcessoru`.

5.2 Testování implementace optimalizátoru

Z počátku implementace byla využita data z knihovny testovacích úloh PSPLIB „Project scheduling problem library“. Více informací lze dohledat zde [9]. Jedná se o sady úloh, které jsou rozdělené podle počtu operací. Při práci byly využity tři sady problémů J30 s 30 operacemi a J60 s 60 operacemi, které obsahují 480 různých problémů a j120 se 120 operacemi a 600 různých problémů. Každá úloha pracuje se čtyřmi zdroji. Pomocí těchto dat lze otestovat chování optimalizátoru na základě zveřejněných řešení.

Pro důvody porovnání implementace se zveřejněnými řešeními na internetu je pravidlo ukončení genetického algoritmu změněné na počet vygenerovaných rozvrhů. Testování probíhalo s omezením 1000 vygenerovaných rozvrhů. V jednotlivých sadách byl vždy vybraný první problém. V J30 probíhalo testování nad `J301_1`, stejně tak u J60 nad `J601_1` a u J120 nad `J1201_1`. Testované velikosti populace byly 30, 50 a 100. Obdobně byly testované i pravděpodobnosti křížení a mutace. Pravděpodobnosti křížení $0,1$; $0,5$; $0,7$; a $0,9$. pravděpodobnosti mutace $0,05$; $0,1$; a $0,25$. Každá sada parametrů byla testovaná stokrát. Zvolené parametry shrnuje tabulka 5.1.

Tabulka 5.2 vyhodnocení zobrazuje výsledky nad problémem `J301_1` a jsou vybrané pouze nejlepší kombinace parametrů pro velikosti populace 30, 50 a 100. Z tabulky je možné vidět, že se zvětšujícím se počtem počáteční populace se zvyšuje šance nalezení optimalního řešení. Výsledné řešení je závislé na kvalitě a velikosti počáteční populace. Při přístupu náhodného generování počáteční populace může dojít k problému, kdy vygenerovaní jedinci nejsou dostatečně kvalitní a optimalizátor konverguje k optimálnímu řešení velmi obtížně. Na základě testů byla vybrána počáteční velikost populace o velikosti 100 jedinců. Dvojbodové křížení vykazovalo při vyšších hodnotách pravděpodobnosti křížení podstatně lepší

výsledky při opakovaném testování než jednobodové křížení. Za typ na základě testování bylo zvolené dvojbodové křížení s pravděpodobností $0,9$. Pravděpodobnost mutace byla zvolená hodnota $0,25$. Další nejbližší výsledky podávala kombinace dvojbodobvého křížení s pravděpodobností $0,7$ a pravděpodobností mutace 0.25. Při příliš nízké hodnotě křížení optimalizátor konvergoval k řešení podstatně pomaleji.

Vybrané hodnoty byly dále otestované nad problémem `J601_1`, kde docházelo téměř k nulové procentuální odchylce od dolní meze řešení. Během testování nad problémem `J1201_1` docházelo naopak k podstatně horším výsledkům. Procentuální odchylka od dolní meze řešení se pohybovala kolem 16% . Optimalizátor při velkém počtu operací není příliš spolehlivý a je nutné provádět výpočet opakovaně. Přesto, ale ve většině případů našel dostatečně kvalitní řešení.

5.3 Testování nad reálnými daty

Testování optimalizátoru nad reálnými daty bylo podstatně komplikovanější. U testovaných dat není dostupné nejlepší možné řešení. Testování probíhalo na základě nejlepšího nalezeného výsledku v průběhu testování. Omezující kritérium testů bylo stále 1000 generovaných rozvrhů. Optimalizátor s parametry nastavenými podle výsledků testů nad problémy z knihovny PSPLIB byl testovaný na čtyřech případech.

První případ obsahoval objednávku na dva finální produkty o velikosti 350 kusů. Operacím sestrojení finálního produktu předchází zhruba 17 operací. Operace se na některých zdrojích překrývají. Zároveň zde nebyla přidána žádná další omezení v podobě odstávek. Optimalizátor v tomto případě téměř vždy našel řešení se stejným časem. Začátek rozvrhování byl v čase 06:00:00 hodin ráno a poslední výrobní operace končila druhý den 01:51:00. V průběhu 100 testů se od této hodnoty lišilo 12 rozvrhů, kde konec poslední operace byl zhruba o pět minut opožděn. Mezi generovanými rozvrhy se avšak velmi lišila část operací, kterou příliš výsledný čas rozvrhu neovlivňoval. Podobnost jednotlivých rozvrhů zabrání zdrojů je poměrně nízká. U testovaného případu některé operace obsahují až 16 dostupných módů. Tento problém je způsobený náhodným generováním módů při vytváření počáteční populace. Jedním z řešení je přidání hodnoty, která by určitým způsobem ohodnotila počet využitých zdrojů.

Obrázek 5.1 ukazuje jeden z možných rozvrhů, který byl pro tuto sadu dat nalezený. Z obrázku je také možné vidět, že doba za jakou byl rozvrh vygenerovaný, je do vteřiny. Optimalizátor pro menší počty operací pracuje velmi rychle. Čas generování se může lišit podle použitého procesoru. Další generovaný rozvrh zobrazuje obrázek 5.2. Zde je vidět, že v některých případech optimalizátor vrací nepříliš vhodný rozvrh použití pro praktické využití. Od prvního obrázku tak využívá navíc tři zdroje, které by mohly zůstat volné. V případě využití takto vygenerovaného rozvrhu v praxi, by bylo nutné manuálně některé operace přesunout na jiné zdroje.

Stejné nastavení bylo využito i v druhém případě testů. Narozdíl od prvního případu zde byla přidaná omezení zdrojů. Výsledky tohoto testu byly podstatně horší než v předešlém testu. Při 100 generování byl pouze v 22 případech nalezen dosud nejlepší čas. Zhruba 42% ostatních rozvrhů se od této doby lišily o průměrně 21 minut. Zbylých 58% rozvrhů se od nejlepšího nalezeného času lišily zhruba v jedné hodině. Nejlepší nalezený koncový čas je 5. 5. 2020 v 02:10:50. Nejhorší čas během testování byl 5. 5. 2020 03:25:30.

Obrázek 5.3 představuje jeden z možných rozvrhů. Pro tento případ byla výrobní linka „Montáž L1“ v odstávce po dobu 4. 5. 2020 14:00:00 až do 5. 5. 2020 06:00:00. V případě, kdy byla snížená velikost populace na 30 jedinců docházelo k problému, kdy optimalizátor

konvergoval k optimálnímu řešení velmi špatně. Obrázek 5.3 zobrazuje část rozvrhu, kde se využila odstavená linka. Při příliš malém počtu jedinců populace dochází k problému, kdy počáteční generování příliš ovlivní výsledný rozvrh.

Třetí test probíhal nad deseti odlišnými produkty, kde docházelo k překryvu na dvou a více zdrojích. Zde byl zaznamenaný výrazný pokles rozdílů kvality mezi generovanými rozvrhy. Při 40 generovaných rozvrzích se konečný čas téměr nelišil a pouze čtyři rozvrhy měly čas rozdílný zhruba o jednu hodinu. Vzhledem k tomu, že některé operace byly podstatně delší než většina ostatních, měly největší vliv na finální čas rozvrhu. Problém nastal u operací, které neovlivňovaly finální dobu rozvrhu a rozmístění operací na zdroje bylo náhodné. Naopak nejdelší operace, na které v rozvrhu zabíraly většinu času, se rozvrhovaly velmi podobným způsobem a koncový čas vycházel téměř vždy stejně. Obrázek 5.5 ukazuje jeden z vygenerovaných rozvrhů pro tento případ. Na rozdíl od předchozích dvou testů byla doba generování výrazně delší, kde s počáteční populací 100 a omezujícím kritériem 1000 generovaných rozvrhů doba z jedné sekundy stoupla na zhruba 2 minuty a 15 sekund. Test obsahoval v základu přes 250 výrobních operací. Po zavedení požadavků se finální počet operací v rozvrhu pohyboval kolem hodnoty 1000.

Čtvrtý test pracuje s deseti objednávkami stejného kusu finálního produktu. Na rozdíl od předchozího testu byly výsledky významně horší. Rozdíl koncových časů při generování více rozvrhů byly v rámci tří hodin. Podobnost generovaných rozvrhů byla nízká. U zdrojů, které se přetěžovaly, vzniká fronta. Koncový čas rozvrhu záleží na pořadí operací na zdroji, narozdíl od zdrojů jako například héliová zkouška, kde možností pro jednu operaci je 16. V případě, že se navýší velikost populace a počet generací, je možné částečně tento problém omezit za cenu podstatně delší doby generování rozvrhů. Část generovaného rozvrhu je zobrazena na obrázku 5.6, kde červený rámeček označuje popisovaný problém. V tomto rozvrhu bylo by možné manuálně přeskládat operace na 3 zdroje.

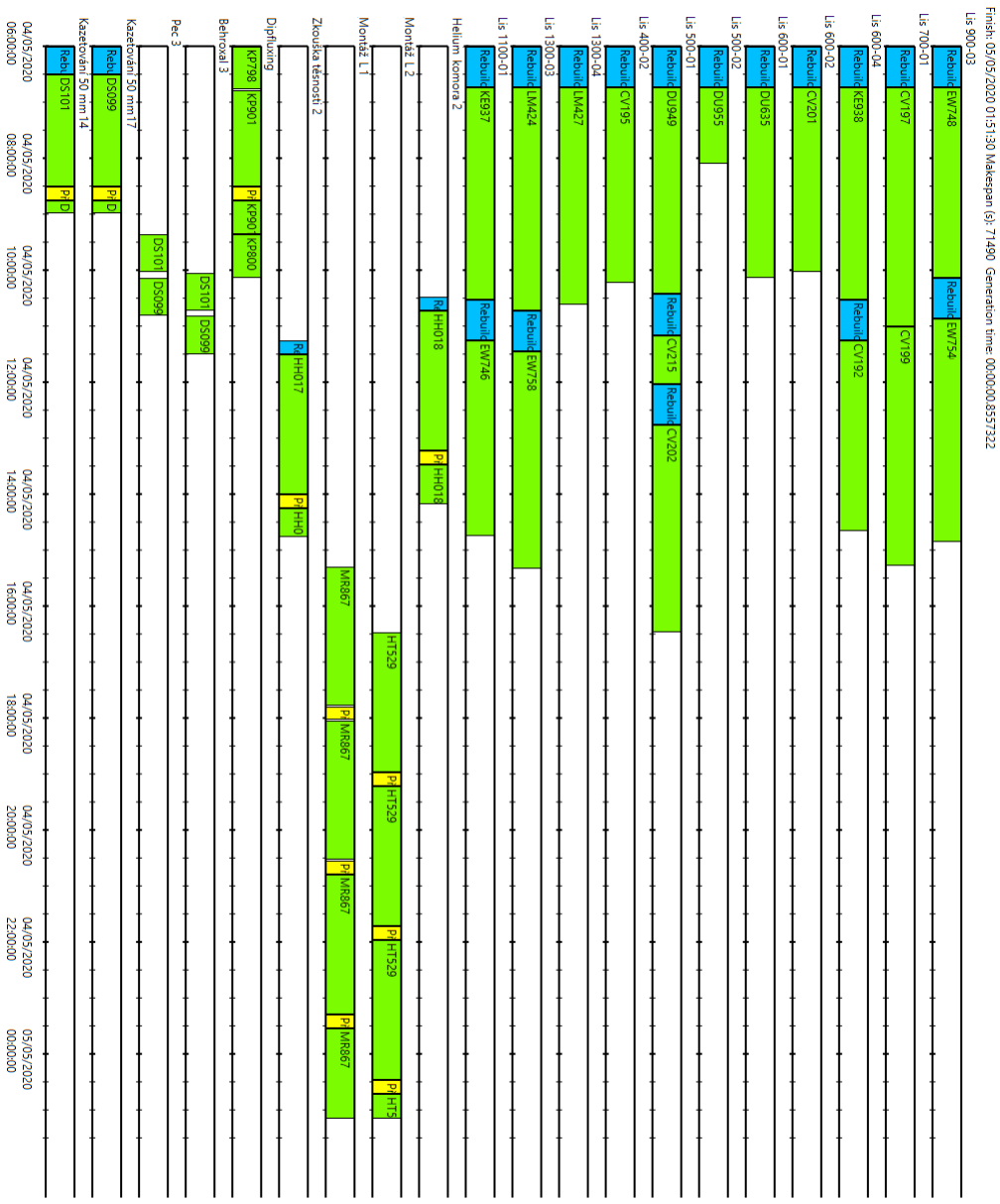
5.3.1 Shrnutí výsledků testování

Optimalizátor byl srovnatelný s některými publikovanými řešeními nad sadami řešení z knihovny PSPLIB. U případů nad reálnými daty se při menším počtu operací, se základně nastavenými parametry podávaly velmi dobré výsledky. Při velkém počtu operací optimalizátor začal být výrazně náhodnější. Tento problém lze částečně vyřešit zvětšením počáteční populace, to avšak neřeší každý problém. Pro případy rozvrhování operací, kdy většina operací využívá jeden nebo dva stejné zdroje, nebyly výsledné rozvrhy příliš kvalitní a pro využití je třeba je manuálně modifikovat a upravovat. Rychlost generování rozvrhů závisí na umístění odstávek a jejich počtu. Pro případy, kdy se odstávka některého zdroje zablokovala hned na začátku rozvrhu, se při generování rozvrhů podstatně prodloužila doba.

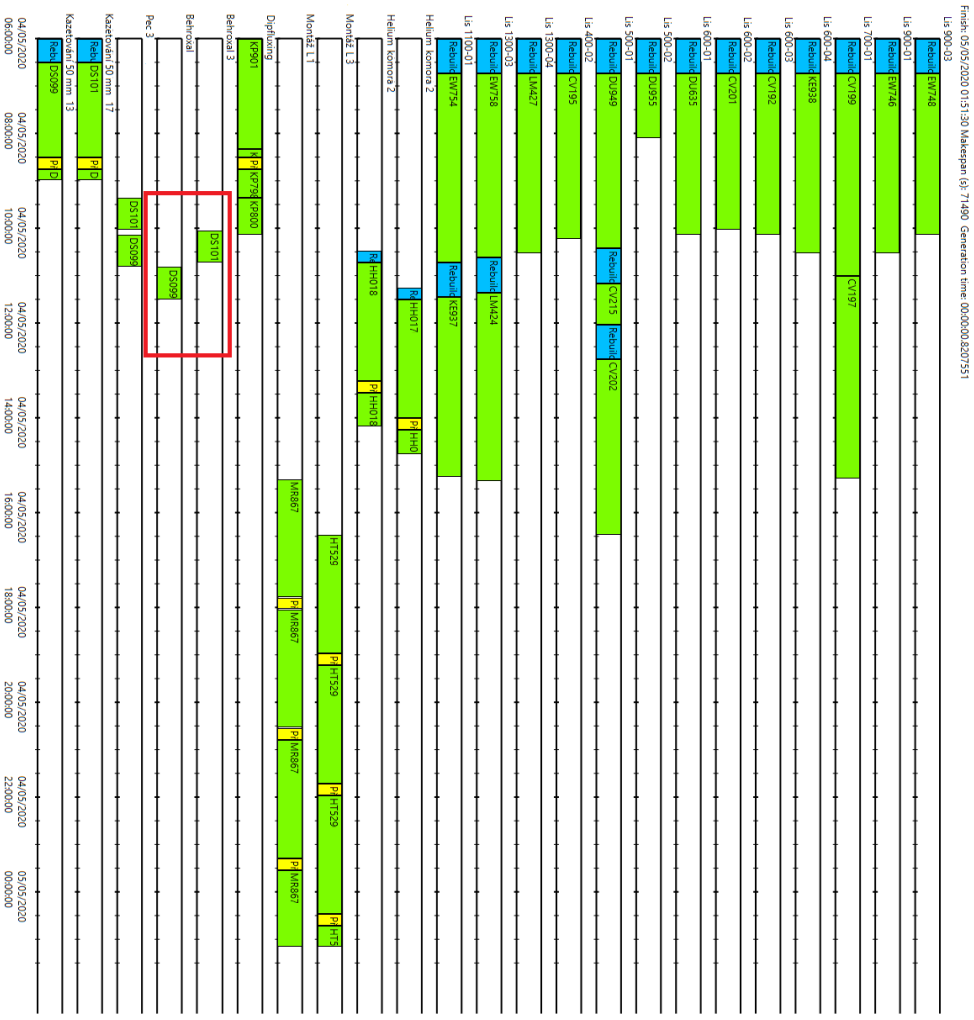
Vzhledem k typu plánování vybraného výrobního podniku, kde plánování probíhá na krátkou dobu dopředu, rozvrhovací systém podával dostatečně kvalitní výsledky. V běžné situaci se jedná o jeden nebo dva dny dopředu, s malým počtem odlišných produktů v rámci jednoho výrobního procesu. U některých vygenerovaných rozvrhů je vhodné pro využití manuálně opravit zmíněné chyby. To se hlavně týká operací, které bezdůvodně zabírají zdroj a je možné je manuálně přerozvrhnout na jiný zdroj bez postihu porušení omezení ostatních operací. Doporučené je využít systém v základním nastavení pro rozvrhování maximálně pěti finálních produktů.

Je-li třeba rozvrhovat velké množství operací, je vhodné zvětšit velikost počáteční populace a počet generací optimalizátoru za cenu podstatně pomalejšího výpočtu. Pro případy, kdy bylo nutné rozvrhovat více jak 200 operací, se při testování jevilo velmi výhodné pra-

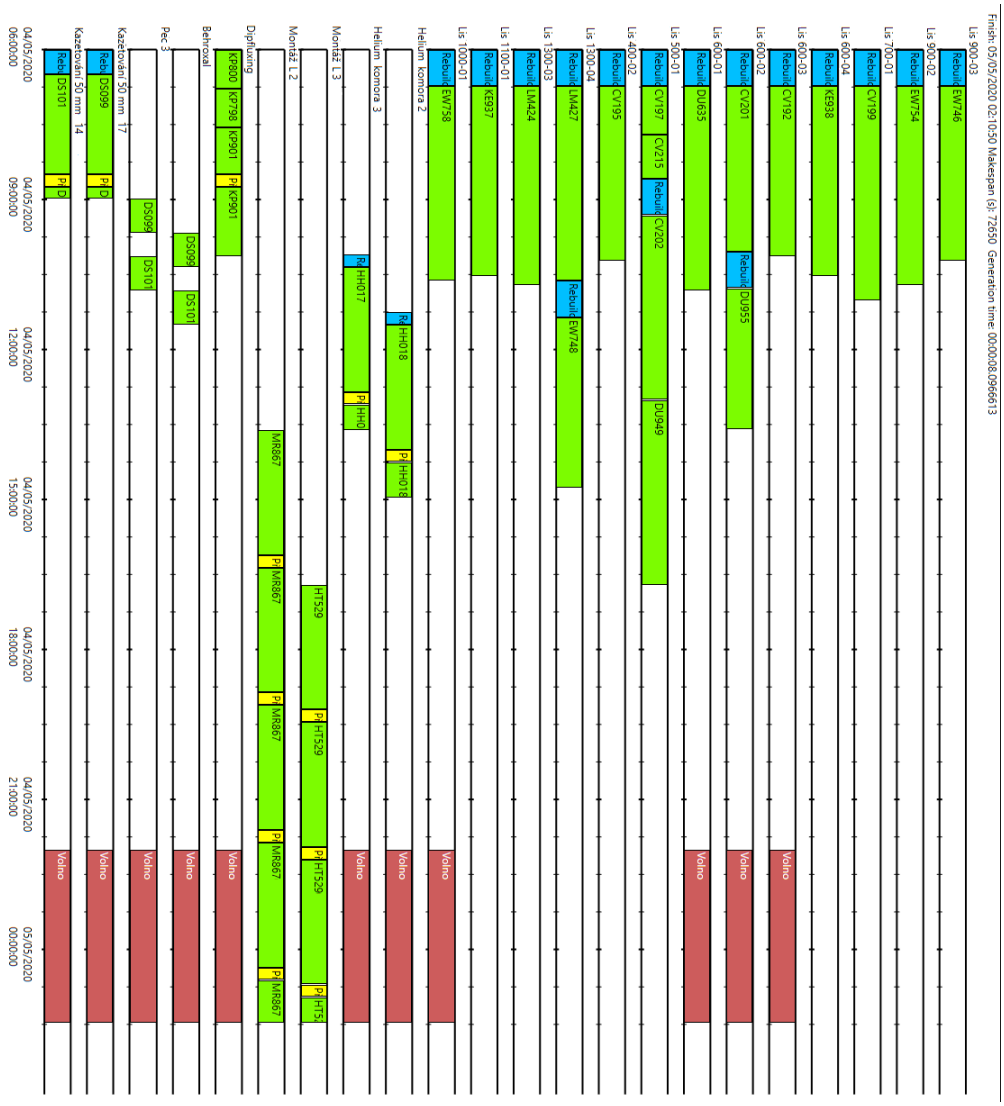
covat s velikostí populace 200 a počet generací navýšit na 5000 až 50000. Na zmíněném hardwaru byl výpočet stále poměrně efektivní, dokázal v rozumném čase nalézt dostatečně kvalitní rozvrh.



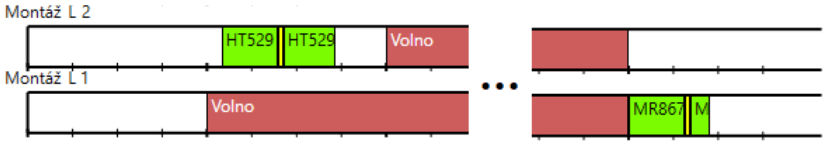
Obrázek 5.1: Ganttův diagram rozvrhu pro produkty MR867 a HT529



Obrázek 5.2: Ganttův diagram rozvrhu pro produkty MR867 a HT529

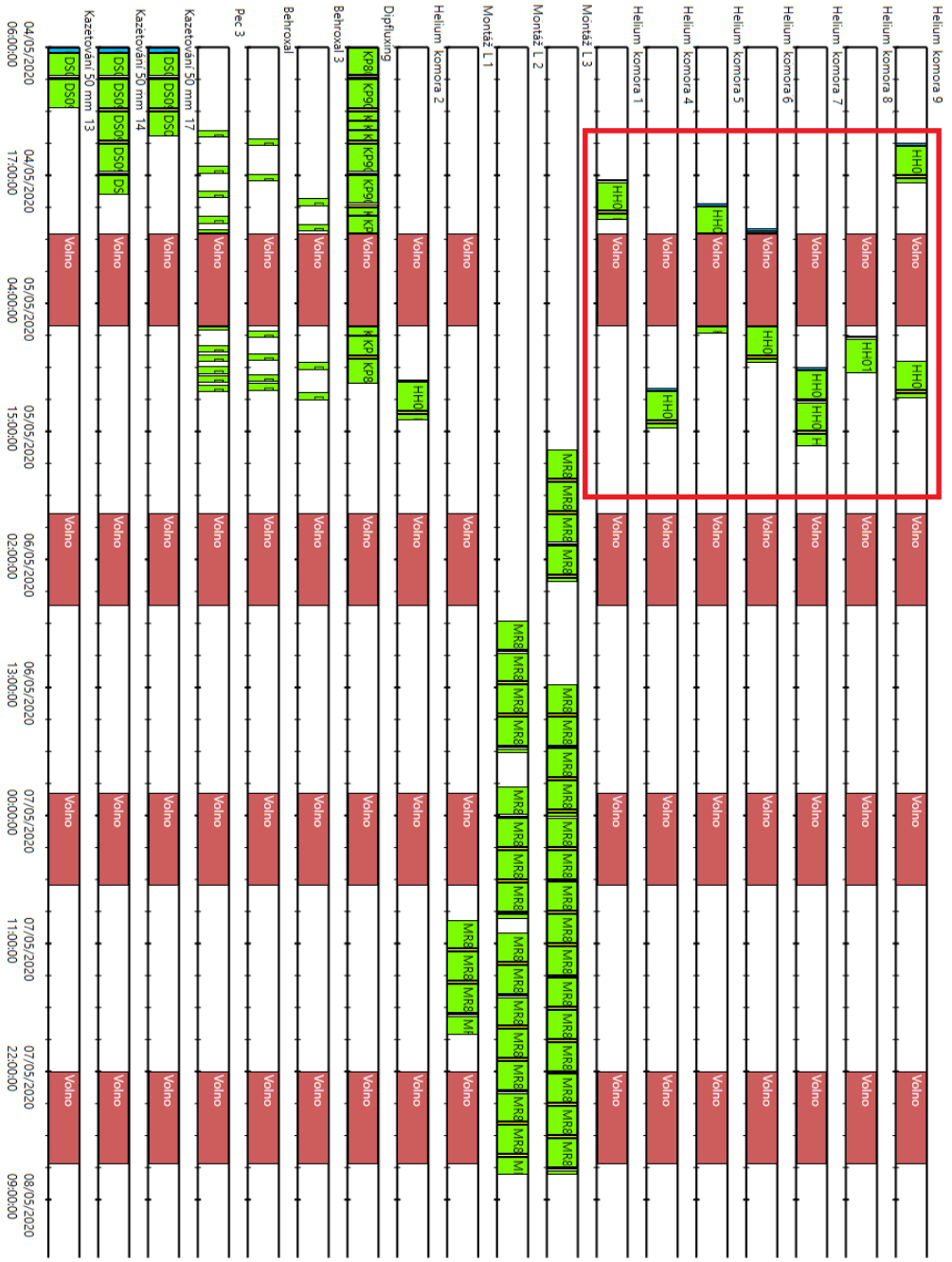


Obrázek 5.3: Ganttův diagram rozvrhu pro produkty MR867 a HT529 s odstávkami



Obrázek 5.4: Část Ganttova diagramu rozvrhu pro produkty MR867 a HT529

Obrázek 5.5: Část Ganttova diagramu rozvrhu s deseti různými finálními produkty



Obrázek 5.6: Část Ganttova diagramu rozvrhu s deseti stejnými finálními produkty

Kapitola 6

Závěr

V rámci bakalářské práce jsem vytvořil systém pro automatické plánování výroby v průmyslu. Při vytváření systému jsem pracoval hlavně s modelem RCPSP, který jsem částečně modifikoval pro požadované specifikace vybraného podniku. Navrhnul jsem vlastní řešení pro vzniklé problémy a převedl je do podoby programu. Program je možné zakomponovat do už nasazené služby MAHLE MES, která slouží jako systém pro správu výrobního podniku.

Při návrhu jsem konzultoval možnosti systému s firmou Lenoxi Automation s.r.o, která má na starost automatizaci průmyslové výroby firmy MAHLE Behr Mnichovo Hradiště s.r.o., která je jedním z největších výrobních společností automobilových součástek. Během konzultací s pracovníky jsem měl možnost se podívat na reálný průběh výrobního postupu. Seznámil jsem se s výrobními postupy a s problematikou plánování.

Dalším vývojem této práce by bylo možné rozšířit systém o možnost přeplánování stávajícího rozvrhu, podle měnícího se stavu zdrojů a operací. Další možností rozšíření by bylo optimalizovat genetický algoritmus pro případy výrob, nebo zahrnutí požadavku plánování lidských zdrojů na základě kvalifikační matice. Pro tento případ by se musel stávající model částečně předělat a rozšířit o další požadavky.

Literatura

[1] Barták, R. *PA167 Rozvrhování* [online]. 2019. Dostupné z: https://www.fi.muni.cz/~hanka/rozvrhovani/prusvitky/all_bw.pdf.

[2] Bouleimen, K. a Lecocq, H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*. 2003, roč. 149, č. 2, s. 268 – 281. Sequencing and Scheduling. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0377221702007610. ISSN 0377-2217.

[3] Christofides, N., Alvarez Valdes, R. a Tamarit, J. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*. 1987, roč. 29, č. 3, s. 262 – 273. Dostupné z: http://www.sciencedirect.com/science/article/pii/0377221787902402. ISSN 0377-2217.

[4] Davis, E. W. a Patterson, J. H. Resource-based project scheduling: which rules perform best? *Project Management Quarterly*. 1975, 6(4), s. 25 – 31.

[5] Hartmann, S. Project Scheduling with Multiple Modes: A Genetic Algorithm. *Annals of Operations Research*. 2001, roč. 102, č. 5, s. 111–135. Dostupné z: https://doi.org/10.1023/A:1010902015091.

[6] Hartmann, S. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*. 2002, roč. 49, č. 5, s. 433–448. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.10029.

[7] Hartmann, S. a Briskorn, D. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*. 2010, roč. 207, č. 1, s. 1 – 14. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0377221709008558. ISSN 0377-2217.

[8] Kolisch, R. a Hartmann, S. *Heuristic algorithms for solving the resource-constrained project scheduling problem - Classification and computational analysis* [online]. 1999. Dostupné z: https://www.om-db.wi.tum.de/psplib/files/handbook1.pdf.

[9] Kolisch, R. a Sprecher, A. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*. 1997, roč. 96, č. 1, s. 205 – 216. Dostupné z:

http://www.sciencedirect.com/science/article/pii/S0377221796001701. ISSN 0377-2217.

[10] Kyriakidis, T. S., Kopanos, G. M. a Georgiadis, M. C. MILP formulations for single- and multi-mode resource-constrained project scheduling problems. *Computers and Chemical Engineering*. 2012, roč. 36, s. 369 – 385. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0098135411001955. ISSN 0098-1354.

[11] Wilson, J. M. Gantt charts: A centenary appreciation. *European Journal of Operational Research*. 2003, roč. 149, č. 2, s. 430 – 437. Sequencing and Scheduling. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0377221702007695. ISSN 0377-2217.