České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Simulace a rozvrhování výrobních procesů

Praha, 2010 **Autor: Marek Elbl**

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Marek Elbl**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Simulace a rozvrhování výrobních procesů**

Pokyny pro vypracování:

1. Seznamte se s algoritmy pro rozvrhování výrobních procesů.
2. Seznamte se se systémy umožňující plánování výroby s více uživateli (např. MS Share Point).
3. Pro vybranou případovou studii navrhněte a implementujte systém pro rozvrhování a simulaci výroby. Snažte se navrhnout takové řešení, aby data algoritmu mohlo zadávat více uživatelů.
4. Odzkoušejte a zdokumentujte Vámi navržené řešení.

Seznam odborné literatury:

[1] M. Pinedo. Planning and Scheduling in Manufacturing and Services. Springer, USA, 2005.
[2] Birger Franck, Klaus Neumann and Christoph Schwindt, Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling, OR Spectrum, Springer Berlin / Heidelberg, 2001.
[3] Lixin Tanga, Jiyin Liu, Aiying Rong and Zihou Yang, A mathematical programming model for scheduling steelmaking-continuous casting production, European Journal of Operational Research, 2000.

Vedoucí: Ing. Přemysl Šůcha, Ph.D.

Platnost zadání: do konce letního semestru 2009/10

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Zde bych v první řadě rád poděkoval Ing. Přemyslu Šůchovi, Ph.D. za odborné rady a pomoc při vedení diplomové práce. Dále pak chci poděkovat své rodině a všem přátelům, kteří mě při studiích a na cestě k vytvoření této práce podporovali.

Abstrakt

Cílem této diplomové práce je vytvořit návrh systému pro rozvrhování výroby, který by mohlo sdílet více uživatelů najednou. Za tímto účelem jsou prozkoumány systémy pro hromadnou správu dokumentů využívající webové rozhraní jako Microsoft Sharepoint a Google Docs. Pro účely rozvrhování je popsáno využití Petriho sítí. Systém pro rozvrhování výroby je aplikován na problém rozvrhování dávkových procesů, pro který je navržen optimalizační algoritmus. Tento algoritmus využívá smíšeného lineárního programování (MIP) a metaheuristiky tabu search.

Výsledkem této práce je aplikace pro rozvrhování výroby dávkových procesů, využívající technologii Microsoft Sharepoint pro vytvoření webového rozhraní s podporou správy více uživatelů. Jako grafické uživatelské rozhraní sloužící uživateli k zadávání parametrů výroby je využit Microsoft Excel. Optimalizační algoritmus je implementován v MATLABu.

Abstract

The aim of this diploma theses is to design a system for production scheduling, which should by controllable by multiple users simultaneously. Therefore multiuser systems for documents sharing with web interface like Microsoft Sharepoint and Google Docs are explored. For scheduling purpose utilization of Petri nets is described and next is described the batch scheduling problem. For this problem new optimalization algorithm is designed. This algorithm uses mixed integer programming (MIP) and meta-heuristic tabu search.

The work result is an application for production scheduling of batch processes. This application uses Microsoft Sharepoint technology to generate the multiuser web interface where Microsoft Excel is used as a graphical user interface for entering of productions parameters. The scheduling algorithm is implemented in MATLAB.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

1.1 Motivace

Využití výpočetních prostředků se v současnosti stalo již samozřejmostí na všech úrovních lidské činnosti. Jedním ze způsobů využití těchto prostředků je počítačová simulace a rozvrhování. S rozvrhováním se můžeme setkat v rozličných oborech od rozvrhování pracovních směn na místech, kde je nutno zabezpečit jejich správné střídání, přes hledání optimálního pořadí výrobků ve výrobních linkách až k plánování fází složitých projektů. Počítačová simulace může sloužit k ověřování spolehlivosti nových systémů, případně změn v nich, aniž bychom museli provádět testy v reálném prostředí, kde by mohl hrozit vznik škod. Všechny tyto aplikace mají za cíl snížení nákladů a zvýšení bezpečnosti a pružnosti při provozu a návrhu rozličných systémů.

Velké rozšíření výpočetní techniky ovšem automaticky neznamená, že jsou tyto prostředky jednoduše aplikovatelné a že se s nimi každý rychle seznámí. Přílišná složitost uživatelského rozhraní, i když jinak třeba kvalitního systému, může způsobit, že uživatelé nebudou využívat tyto systémy správně, nebo je začnou různými způsoby obcházet, což způsobuje snížení celkové efektivity. V této práci je navržen systém pro rozvrhování výroby využívající k zadávání dat obecně známých výpočetních prostředků jakým je například Microsoft Excel. Systém je navržen pro rozvrhování dávkové výroby na výrobních linkách s paralelními pracovními stanicemi (FFL), které jsou odděleny buffery s omezenou kapacitou.

1.2 Přehled souvisejících prací

Problematikou rozvrhování dávkových procesů se podrobně zabývá kapitola 2. Ve své práci jsem vycházel z matematického popisu modelu dávkové výroby s paralelními pracovními stanicemi tzv. *flexible flow lines* (FFL) pomocí smíšeného celočíselného programování (MIP) viz (Sawik, T., 2002). Optimalizačním kritériem v této práci byla minimalizace délky rozvrhu. Tento přístup přináší výhodu nalezení optimálního řešení, nicméně není vhodný pro velké instance výroby.

Rozvrhováním dávkových procesů ve výrobní lince FFL se také zabývá práce (Quadt, M., Kuhn, H., 2007). Pojednává o rozvrhování výrobků s identickou *dobou trvání* na všech strojích pomocí genetického algoritmu. Jednalo se o multikriteriální optimalizaci zohledňující minimalizaci ceny za nastavení (*setup cost*) a střední dobu průchodu (*mean flow time*). Stejně jako v případě mnou navrženého algoritmu je časová náročnost závislá převážně na počtu druhů dávek, které se ve výrobě vyskytují. V tomto případě nárůst počtu dávek způsobuje zvětšování prostoru, ve kterém prohledává genetický algoritmus řešení s rychlostí faktoriálu.

Rozvrhování dávkových procesů pro dávkový stroj (*Batch machine*) je popsáno v (Muthu Mathirajan, Bhargav, V., Ramachandran, V., 2009). Řešení bylo navrženo pomocí celočíselného lineárního programování (ILP). Optimalizačním kritérium bylo v tomto případě „*weighted tardiness*“.

V (Yeong-Dae Kim, Hyeong-Gyu Lim, Moon-Won Park, 1996) jsou porovnány výkonnosti a vhodnosti různých metaheuristik pro rozvrhování výroby plošných spojů. Tato výroba byla charakterizována jako *Flow shop* (bez paralelních pracovních stanic) a stejné druhy výrobků byly seskupeny do dávek. Výroba potom byla zjednodušena tak, že se na každou dávku pohlíželo jako na jeden *job* s určitou dobou trvání (*processing time*) na každém stroji. V těchto experimentech se jako jedna z nejvýkonnějších metaheuristik ukázala metoda *Tabu search*.

V (Nowicki, E., Smutnicki, C., 1998) je popsáno rozvrhování výroby typu FFL pomocí metaheuristiky *Tabu search*. Kritériem byla délka rozvrhu. V (Quadt, D., Kuhn, H., 2007) jsou souhrnně prezentovány metody použitelné pro rozvrhování výroby typu FFL.

1.3 Vlastní přínos práce

Jedním z cílů této diplomové práce bylo navrhnout systém pro plánování výroby, který by v co největší míře využíval pro uživatele obecně známé výpočetní prostředky, jakým je například balík kancelářských aplikací Microsoft Office. Aplikace Microsoft Excel je využita k tvorbě uživatelského rozhraní, sloužícího k zadávání parametrů výroby. Tento systém umožňuje ovládání více uživateli najednou. Za tímto účelem jsou v práci prozkoumány možnosti systémů pro hromadnou zprávu dokumentů poskytujících webové rozhraní jakým je například „Google Docs“ a „Microsoft Windows SharePoint Services 3.0“, který je v navrženém systému pro rozvrhování výroby využit. Tento rozvrhovací systém se konkrétně zabývá rozvrhováním dávkových procesů ve výrobní lince typu FFL. Pro rozvrhování dávkových procesů bylo prozkoumáno několik možných přístupů. Mezi ně patří smíšené celočíselné programování, využití metaheuristik nebo Petriho sítí. Výsledné řešení je navrženo s pomocí metaheuristiky „tabu search“. Hlavní přínosy práce jsou shrnuty v následujících bodech:

- Návrh systému pro rozvrhování výroby s využitím aplikace MS Excel jako uživatelského rozhraní a technologie MS SharePoint pro vytvoření internetového portálu správy dokumentů.
- Heuristické řešení MIP modelu výroby viz (Sawik, T., 2002) umožňující řešit větší instance problémů s využitím metaheuristiky „tabu search“.
- Rozšíření heuristického řešení problému rozvrhování výroby (Sawik, T., 2002) o kritérium optimalizace „weighted tardiness“.

V kapitole 2 je popsána problematika rozvrhování dávkových procesů. Dále se tato kapitola zabývá rozvrhováním těchto procesů na výrobních linkách s buffery a nakonec je zaveden matematický MIP model výrobní linky dávkových procesů na m paralelních pracovních stanicích za sebou tzv. flexible flow line (FFL), oddělených omezenými buffery. Tento model je dále rozšířen o kritérium „weighted tardiness“. Kapitola 3 popisuje systémy pro zprávu dokumentů a podnikovou spolupráci a aplikaci GLPK určenou k řešení úloh lineárního programování. Dále jsou v této kapitole popsány způsoby využití Petriho sítí pro modelování a rozvrhování výroby. Kapitola 4 vysvětluje heuristiky použité pro urychlení doby řešení MIP modelu výroby z kapitoly 2, dále heuristickou tvorbu matematického modelu výroby bez využití MIP a nakonec aplikaci metaheuristiky „tabu search“. V kapitole 5 je popsána struktura výsledného systému pro rozvrhování výroby,

jeho použití a nezbytné kroky k jeho instalaci. V kapitole 6 jsou experimentální výsledky výkonů heuristik navržených v kapitole 4.

Kapitola 2

Popis problému

V této kapitole je vysvětleno základní rozdělení dávkových procesů společně s přehledem algoritmů pro řešení těchto úloh. Dále bude zaveden matematický MIP model dávkové výroby pro montážní linku s paralelními pracovními stanicemi tzv. *flexible flow line* (FFL) a omezenými buffery.

2.1 Dávkové procesy

Problém dávkové výroby znamená, že množina jednotlivých úloh, která je vykonávána na stejném stroji, musí být rozdělena do dávek. Úloha (job) může být například jeden konkrétní výrobek dané výroby. Dávkou (batch) se potom rozumí podmnožina původní množiny všech úloh, mající typicky nějaké společné vlastnosti popřípadě odpovídající stejnému typu výrobku. Tato dávka je zpracovávána na stejném stroji a úlohy v ní obsažené by měly být vykonány společně.

U rozvrhování dávkových procesů se tedy zabýváme problémem, jak rozdělit úlohy do dávek a jak tyto dávky rozvrhnout. Čas dokončení dávky je stejný jako čas dokončení poslední úlohy v dávce. Každý stroj může mít definovanou nastavovací dobu (*setup time*) a dobu vyjmutí (*removal time*) pro každou dávku. Mezi jednotlivými úlohami v dávce je nastavovací doba a doba vyjmutí nulová. V závislosti na typu výroby jsou dva způsoby, jak zjišťujeme délku dávky. Rozlišujeme mezi dávkami typu „p-batch“ (délka dávky je maximum z doby trvání všech úloh v dávce) a „s-batch“ (délka dávky je suma dob trvání všech úloh) (Brucker, P., 2007). Dávky „p-batch“ se můžou vyskytnout například u strojů, zpracovávajících jednotlivé výrobky paralelně. Naopak o dávky typu „s-batch“ se

jedná u strojů zpracovávajících jednotlivé výrobky sériově.

2.1.1 Dávkové procesy pro jednoprocesor

Pro dávky typu „s-batch“ se nejprve řeší problém rozdělení množiny n úloh do dávek a dále hledáme sekvenci těchto dávek. To lze řešit pomocí jednoduchých polynomiálních algoritmů například pro úlohy typu: $1\,|s-batch|\sum C_i$, $1\,|p_i = p; s-batch|\sum w_i C_i$ nebo $1\,|prec; p_i = p; s-batch|\sum C_i$. Pomocí metod dynamického programování lze řešit problém $1\,|s-batch|\sum w_i U_i$ (Brucker, P., 2007).

Dávky typu „p-batch“ lze využít u stroje, který dokáže jednotlivé úlohy v dávce, vykonávat současně tzv. dávkový stroj (*batch machine*). U těchto typů dávek potom rozlišujeme, zda je stroj neomezený a tedy dokáže zpracovávat více úloh než jich je v dávkách $b > n$, nebo zda je omezený $b < n$. Problémy typu: $1\,|p-batch|\sum f_i$, $1\,|p-batch|\sum U_i$, $1\,|p-batch|\sum f_max$, $1\,|p-batch, b<n|\sum f_i$ jsou řešitelné pomocí polynomiálních algoritmů. Naproti tomu problémy jako $1\,|p-batch|\sum w_i, U_i$, $1\,|p-batch|\sum w_i, T_i$ nebo $1\,|p-batch, b=2|\,L_{max}$ jsou NP obtížné (Brucker, P., 2007). Tyto NP obtížné úlohy lze řešit pomocí různých heuristik, dynamického programování, genetických algoritmů atd. viz (Cheng-Shuo Wang, Reha Uzsoy, 2000).

2.1.2 Složitější systémy

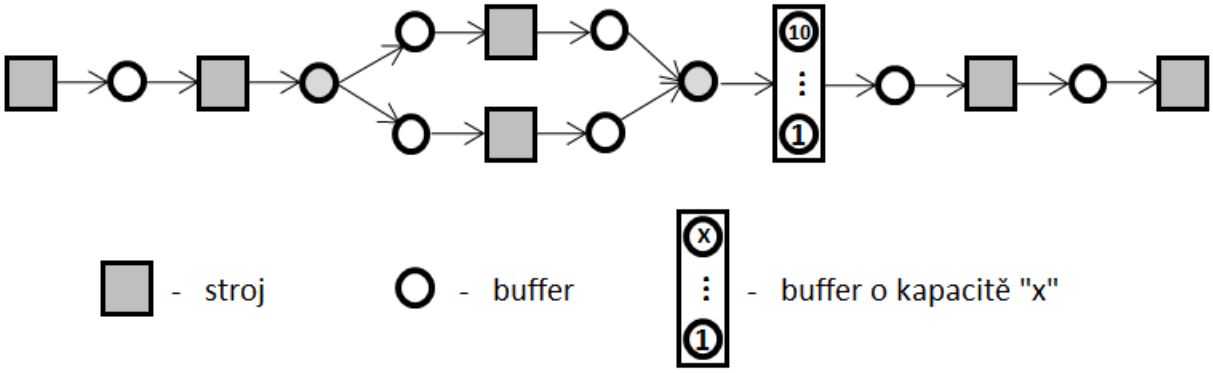
U složitějších struktur výroby, nebo pokud je např. zapotřebí rozvrhnutí na více než dva dávkové stroje, lze využít metody lineárního programování viz níže, nebo „branch and bound“ algoritmu viz (Schwindt, Ch., Trautmann, N., 2001).

2.1.3 Dávkové procesy s buffery

Většina literatury se věnuje problematice dvou strojů s omezeným resp. neomezeným bufferem, mezi těmito stroji, pro flowshop $F2\,||\,C_{\max}$. Úlohy jsou sestavovány do dávek tak, že mezi úlohami v dávkách není potřeba žádná nastavovací doba, případně doba vyjmutí. Cílem rozvrhování je seřadit dávky tak, aby se minimalizovalo kritérium, typicky C_{max}. Pro dva stroje se jedná o polynomiální úlohu. O NP obtížnou úlohu se jedná pro případ flowshop $Fm\,||\,C_{\max}$, kde $m \geq 3$, nebo pokud lze dávky dále dělit. (Jianbin, D., 2003)

2.2 Matematický model výrobní linky

Tato část popisuje matematický model, pro problém rozvrhování dávkových procesů, na m paralelních pracovních stanicích za sebou tzv. *flexible flow line* (FFL), oddělených omezenými buffery (viz obr. 2.1). Pro řešení je použito smíšené celočíselné programování (MIP). Hledání optimálního rozvrhu pomocí MIP je vhodné pro malé počty dávek malých velikostí (do 50 výrobků rozdělených až do 6-ti dávek) pro různé konfigurace linky FFL. (Sawik, T., 2002)



Obrázek 2.1: Příklad výrobní linky FFL s paralelní pracovní stanicí a bufferem o kapacitě 10

2.2.1 Definice proměnných

Definice 2.1 (Použité proměnné):

g - dávka (typ výrobku) $g \in G = \{1, ..., v\}$

i - pracovní stanice $i \in I = \{1, ..., m\}$

j - stroj ve stanici i, $j \in J_i = \{1, ..., m_i\}$

k - výrobek $k \in K = \{1, ..., n\}$

Definice 2.2 (Vstupní proměnné):

b_g - velikost dávky g (počet výrobků typu g)

v - počet dávek (druhů výrobků)

m - počet pracovních stanic

m_i - počet strojů v pracovní stanici i, $m_i \geq 1$

n - absolutní počet všech výrobků ve všech dávkách

r_{ig} - doba trvání (processing time) v pracovní stanici i dávky g

K_g - podmnožina výrobků typu g

Q_{ifg} - velké číslo, větší než délka rozvrhu.

Definice 2.3 (Rozhodovací proměnné):

C_{max} - délka rozvrhu

c_{ik} - koncový čas (completion time) výrobku k v pracovní stanici i

d_{ik} - čas odchodu (departure time) výrobku k z pracovní stanice i

x_{ijk} - 1, jestliže výrobek k je na stroji $j \in J_i$ v pracovní stanici $i \in I$; jinak $x_{ijk} = 0$

y_{fg} - 1, jestliže dávka f je před dávkou g; jinak $y_{fg} = 0$

2.2.2 Definice MIP modelu

Výrobní linka je sestavena z několika pracovních stanic v sérii, které jsou vzájemně odděleny buffery s omezenou kapacitou. Každá pracovní stanice má jeden, nebo více paralelních identických strojů. Výrobní linka produkuje několik typů rozdílných výrobků. Každý výrobek musí být zpracováván jen na jednom stroji v každé pracovní stanici bez preempce. Výrobky se ve výrobní lince nemohou „předbíhat“ a proto se FFL linka často označuje jako *flow shop* s paralelními stanicemi. Dokončený výrobek na pracovní stanici je přesunut na volný stroj v následující pracovní stanici, nebo do bufferu za pracovní stanicí. Každý výrobek musí projít postupně v pořadí všemi pracovními stanicemi respektive buffery od stanice 1 až po stanici m. Kvůli omezené kapacitě bufferů se jedná o rozvrhování s blokováním. Dokončený výrobek totiž může ve stroji zůstat a daný stroj blokovat, pokud je následující pracovní stanice respektive buffer obsazen.

Modelování FFL linky je založeno na tom, že se na buffery pohlíží jako na stroje s nulovou dobou trvání (processing time). Všechny výrobky jsou rozděleny do dávek. Každá dávka obsahuje výrobky jednoho typu a výrobky v dávkách jsou zpracovávány postupně jeden po druhém. Pořadí zpracovávání jednotlivých výrobků, v rámci jedné dávky, v každé pracovní stanici je identické. Předpokládá se, že pro různé dávky není potřeba nastavovací doba, případně doba vyjmutí.

Startovací čas lze vyjádřit jako $c_{ik} - r_{ig}$, jelikož u strojů není povolená preempce. Výrobek dokončený v pracovní stanici i v čase c_{ik} opouští stanici v čase $d_{ig} \geq c_{ik}$ a pokračuje na volný stroj ve stanici $i + 1$. Jestliže jsou v čase c_{ik} všechny stroje m_{i+1} na následující stanici $i + 1$ obsazeny, potom je stroj ve stanici i blokován výrobkem k do času $d_{ik} = c_{i+1k} - r_{i+1g}$, ve kterém je přiřazen na volný stroj ve stanici $i + 1$.

Nechť J_i je kruhová množina reprezentující jednotlivé paralelní stroje v pracovní stanici i. Na této množině je definována funkce $next(j, J_i)$. Tato funkce vrací identifikátor následujícího stroje v pracovní stanici $j + 1 = next(j, J_i)$. Pokud $j = m_i$, neboli ukazatel j odkazuje na poslední stroj v množině, vrátí funkce ukazatel na první stroj: $1 = next(m_i, J_i)$.

Nechť $K_g = \left\{ \sum_{f \in G: f \leq g-1} b_g + 1, \ \ldots \ , \sum_{f \in G: f \leq g-1} b_f + b_g \right\}$ je vzestupně seřazená množina identifikátorů jednotlivých výrobků k, které odpovídají příslušné dávce g. Tato množina je důležitá proto, aby měla každá dávka g jednoznačně definovaný seznam výrobků k, které do ní patří.

Příklad 2.1 (Příklad množiny K_g):

V tomto příkladu je demonstrována tvorba množiny K_g pro výrobu pěti druhů dávek, mající tento výrobní plán:

Tabulka 2.1: Výrobní plán

Typ dávky	Velikost dávky	Typ dávky	Velikost dávky	Typ dávky	Velikost dávky	Typ dávky	Velikost dávky
1	2	3	3	4	1	5	2

Řešení: Ze zadaných hodnot je zřejmé, že hodnoty proměnné b_g kde $g \in \{1, 2, 3, 4, 5\}$ jsou následující: $b_1 = 2$, $b_2 = 0$, $b_3 = 3$, $b_4 = 1$, $b_5 = 2$.

Po dosazení těchto hodnot do vztahu pro výpočet K_g, získáme tyto hodnoty množin K_g: $K_1 = [1, 2]$, $K_2 = []$, $K_3 = [3, 4, 5]$, $K_4 = [6]$, $K_5 = [7, 8]$. ✓

Úkolem optimalizace výroby je nalézt minimální hodnotu kritéria $C_{\max} = \max_{k \in K} (c_{mk})$, kde c_{mk} znamená koncový čas výrobku k v poslední pracovní stanici m.

$$C_{max} \tag{2.1}$$

Omezení (2.2) zajišťuje, že každý výrobek může být přiřazen jen na jeden stroj resp. buffer v určité pracovní stanici.

$$\sum_{j \in J} x_{ijk} = 1; \quad i \in I, \quad k \in K, \tag{2.2}$$

Omezení (2.3) zajišťuje, že následujícímu výrobku $k+1$ z dávky g, bude přiřazen následující stroj $next(j, J_i)$ v pracovní stanici i.

$$x_{i,next(j,J_i),k+1} = x_{ijk}; \quad i \in I, \quad j \in J_i, \quad g \in G, \quad k \in K_g : k < last(K_g), \quad m_i > 1; \tag{2.3}$$

Omezení (2.4) a (2.5) zajišťuje, že startovací čas na aktuální pracovní stanici i nemůže být menší, než čas odchodu z předchozí stanice $i-1$ respektive nemůže být menší než 0 na první pracovní stanici.

$$c_{1k} \geq r_{1g}; \quad g \in G, \quad k \notin K_g, \tag{2.4}$$

$$c_{ik} - c_{i-1k} \geq r_{ig}; \quad i \in I, \quad g \in G, \quad k \in K_g : i > 1; \tag{2.5}$$

Omezení (2.6) zajišťuje, že pro všechny pracovní stanice (s výjimkou poslední), musí být koncový čas menší než čas odchodu.

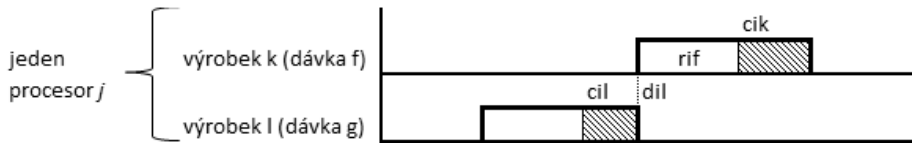
$$c_{ik} \leq d_{ik}; \quad i \in I, \quad k \in K : i < m \tag{2.6}$$

Podle (2.7) je na poslední pracovní stanici čas dokončení roven času odchodu.

$$c_{mk} = d_{mk;} \quad k \in K; \tag{2.7}$$

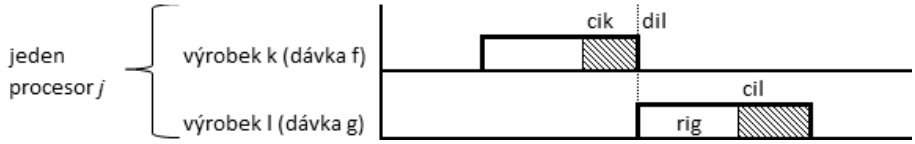
Omezení (2.8) resp. (2.9) se kvůli velkému číslu Q_{ifg} resp. Q_{igf} neuplatní, pokud jsou na jeden stroj j, v jedné pracovní stanici i, přiřazeny dva výrobky z různých dávek - k z dávky f a l z dávky g. A dále, pokud dávka f resp. g předchází v rozvrhu dávku g resp. f. V tomto případě se začne uplatňovat omezení (2.9) resp. (2.8). Pokud dávka g resp. f předchází v rozvrhu dávku f resp. g, tak se omezení (2.8) resp. (2.9) uplatní a zajistí, že výrobky l resp. k z předchozí dávky g resp. f budou zařazeny na daný stroj před výrobky k resp. l z následující dávky f resp. g. Jinými slovy je zajištěno, že pokud je v rozvrhu dávka f před dávkou g, tak na každý stroj musí být zařazeny nejprve výrobky z dávky f a až poté výrobky z dávky g. Grafické zobrazení viz obr. 2.2 a obr. 2.3.

$$\begin{gathered} c_{ik} + Q_{ifg}\,(2 + y_{fg} - x_{ijk} - x_{ijl}) \geq d_{il} + r_{if} \\ i \in I \quad J \in J_i \quad f, g \in G \quad k \in K_f \quad l \in K_g : i > 1 \end{gathered} \tag{2.8}$$



Obrázek 2.2: Grafické zobrazení omezení (2.8)

$$\begin{aligned} &c_{il} + Q_{igf}\,(3 - y_{fg} - x_{ijk} - x_{ijl}) \geq d_{ik} + r_{ig} \\ &i \in I \quad J \in J_i \quad f, g \in G \quad k \in K_f \quad l \in K_g : i > 1 \ ; \end{aligned} \tag{2.9}$$



Obrázek 2.3: Grafické zobrazení omezení (2.9)

Rovnice (2.10) zajišťuje, že výrobek začne být zpracováván na pracovní stanici i okamžitě po opuštění pracovní stanice $i - 1$.

$$c_{ik} = d_{i-1k} + r_{ig}; \quad i \in I, \quad g \in G, \quad k \in K_g : f < g \tag{2.10}$$

(2.11) určuje dolní omezení hodnoty C_{max}

$$c_{mk} \leq C_{\max}; \quad k \in K, \tag{2.11}$$

Omezení (2.12) slouží k zredukování celkového stavového prostoru (urychluje dobu hledání optimálního řešení) zavedením horního omezení hodnoty času odchodu d_{ik} výrobku z pracovní stanice i.

$$d_{ik} + \sum_{h \in I : h > i} r_{hg} \leq C_{\max} \quad i \in I \quad g \in G \quad k \in K_g : i < m \tag{2.12}$$

Část a v omezení (2.13) odpovídá sumě dob trvání všech výrobků, které už byly zařazeny do výrobní linky před výrobkem l a část b odpovídá sumě dob trvání všech výrobků, které ještě zařazeny do výrobní linky nebyly. Toto omezení zavádí horní omezení

pro dobu, kterou bude výrobek ve výrobní lince $c_{ml} - (c_{1l} - r_{1g})$.

$$c_{ml} - (c_{1l} - r_{1g}) \leq C_{\max} - \underbrace{\sum_{f \in G: f<g} b_f r_{1f} y_{fg} - \sum_{f \in G: f>g} b_f r_{1f}(1 - y_{gf}) - \left(l - \sum_{f=1}^{g-1} b_f\right) r_{1g} -}_{a}$$
$$\underbrace{- \left(\sum_{f=1}^{g} b_f - l\right) r_{mg} - \sum_{f \in G: f<g} b_f r_{mf}(1 - y_{fg}) - \sum_{f \in G: f>g} b_f r_{mf} y_{gf};}_{b}$$
$$g \in G, \quad l \in K_g : m_1 = 1, \quad m_m = 1; \tag{2.13}$$

Omezení (2.14) určuje, že v pracovní stanici s více paralelními stroji, může být zpracováváno současně jen tolik výrobků, kolik je její kapacita.

$$c_{ik} + m_i \geq d_{ik} + r_{ig}; \quad i \in I, \quad g \in G, \quad k \in K_G : k + m_i \leq last(K_g), \quad m_i > 1, \tag{2.14}$$

Omezení (2.15) zajišťuje pro pracovní stanici s více paralelními stroji, že výrobky jsou v rámci jedné dávky zařazovány v pořadí vzestupném, dle hodnoty identifikátoru výrobku $k \in K_g$ a identickém pro všechny pracovní stanice.

$$c_{ik} + 1 \geq c_{ik}; \quad i \in I, \quad g \in G, \quad k \in K_G : k < last(K_g), \quad m_i > 1, \tag{2.15}$$

Omezení (2.16) zajišťuje pro pracovní stanici s jedním strojem, že výrobky v rámci jedné dávky, jsou zařazovány v pořadí vzestupném, dle hodnoty identifikátoru výrobku $k \in K_g$ a identickém pro všechny pracovní stanice. Dále zajišťuje, že v takovéto pracovní stanici se může zpracovávat pouze jeden výrobek současně.

$$c_{ik+1} \geq d_{ik} + r_{ig}; \quad i \in I, \quad g \in G, \quad k \in K_G : k < last(K_g), \quad m_i = 1; \tag{2.16}$$

Omezení (2.17) redukuje výsledný stavový prostor problému, odstraněním redundantní informace o pořadí dávek z y_{fg}

$$y_{fg} = 0; \quad f, g \in G : f \geq g; \tag{2.17}$$

Rovnice a omezení (2.18) až (2.21) určují vlastnosti jednotlivých proměnných.

$$c_{ik} \geq 0; \quad i \in I, \quad k \in K, \tag{2.18}$$

$$d_{ik} \geq 0; \quad i \in I, \quad k \in K, \tag{2.19}$$

$$x_{ijk} \in \{0,1\}; \quad i \in I, \quad j \in J_i, \quad k \in K, \tag{2.20}$$

$$y_{fg} \in \{0,1\}; \quad f,g \in G. \tag{2.21}$$

V rovnicích (2.8) a (2.9) je využíván parametr Q_{igf}. Jedná se o „velké číslo“ ne menší než délka rozvrhu a lze získat podle následující rovnice:

$$Q_{ifg} = UB - \sum_{h \in I:h<i} r_{hf} - \sum_{h \in I:h>i} r_{hg}; \quad i \in I, \quad f,g \in G \tag{2.22}$$

kde

$$UB = \frac{\sum_{i \in I} \sum_{g \in G} b_g r_{ig}}{m_i} \tag{2.23}$$

Pomocí rovnice (2.24) lze získat hodnotu dolního omezení hodnoty C_{max} rozvrhu.

$$LB = \max_{i \in I} \left\{ \frac{\sum_{g \in G} b_g r_{ig}}{m_i} + \min_{g \in G} \left(\sum_{h \in I:h<i} r_{hg} \right) + \min_{g \in G} \left(\sum_{h \in I:h>i} r_{hg} \right) \right\} \tag{2.24}$$

2.2.3 Úprava modelu pro kritérium „weighted tardiness“

Mimo kritéria C_{max}, které zaručuje rozvržení výroby tak, aby byla výroba dokončena co nejrychleji, existují i jiná užitečná kritéria jako například hodnota váženého nezáporného zpoždění *weighted tardiness*. Pro toto kritérium je důležité, že každý výrobek má ještě navíc určený termín (*due date*), kdy má být dokončen a prioritu dokončení tohoto výrobku. Toto kritérium je například užitečné, pokud je třeba dodat výrobek do určitého data a při překročení tohoto data hrozí penále. Dosažení určitého termínu dokončení *due date* pro každý výrobek nemusí být vždy dodrženo a proto ani hodnota celkového kritéria nemusí být vždy nulová. Hlavní vlastností je tedy minimalizace nákladů za nedodržení termínu dodání *due date* pro všechny výrobky. Jinými slovy výrobek, u kterého se například platí malé penále za zpoždění, se vyplatí zpozdit za cenu, že se stihne dokončení výrobku, u kterého by hrozilo velké penále za nedodržení termínu dodání. Tato kapitola tedy popisuje rozšíření stávajícího MIP modelu výroby o toto kritérium.

Problémem modelování výroby a jejího rozvrhování pomocí ILP s minimalizací kritéria *weighted tardiness* se zabývala práce (Muthu Mathirajan, Bhargav, V., Ramachandran, V., 2009). Jednalo se o výrobu obsahující jeden dávkový stroj tzv. (*batch*

machine). Tato výroba odpovídá definici dávek typu „p-batch“. Definici kritéria v této práci jsem využil v modelu dávkové výroby na výrobní lince FFL z předchozí kapitoly, která naopak odpovídá definici dávek typu „s-batch“.

2.2.4 Definice proměnných

Definice 2.4 (Použité proměnné):

d_k - termín dokončení výrobku k

w_k - priorita termínu dokončení výrobku k

t_k - hodnota nezáporného zpoždění výrobku k vůči hodnotě d_k

2.2.5 Úprava MIP modelu

Vlastnosti výrobní linky zůstávají nezměněny, je pouze potřeba dodefinovat výpočet nového kritéria. Pro získání jeho hodnoty je nutné vypočítat tzv. nezáporné zpoždění (*tardiness*) každého výrobku k. To lze získat ze vztahu $t_k = \max(c_{mk} - d_k, 0)$, kde c_{mk} znamená koncový čas výrobku k v poslední pracovní stanici m a samotná hodnota kritéria odpovídá vážené sumě těchto hodnot t_k a lze získat ze vztahu viz (2.25). V upraveném modelu tedy bude naším úkolem minimalizovat hodnotu kritéria z rovnice (2.25) namísto původní rovnice (2.1).

$$WT = \sum_{k=1}^{n} t_k w_k \tag{2.25}$$

Z tohoto důvodu můžeme z našeho modelu vypustit rovnice (2.11), (2.12) a (2.13). Dále musíme zavést rovnice pro získání hodnot nezáporného zpoždění t_k pro každý výrobek k viz rovnice (2.26) a (2.27).

$$t_k \geq c_{mk} - d_k; \quad k \in K \tag{2.26}$$

$$t_k \geq 0; \quad k \in K \tag{2.27}$$

Kapitola 3

SW platformy pro rozvrhování výroby

V této kapitole jsou popsány různé softwarové platformy a matematické přístupy, kterými jsem se v průběhu řešení diplomové práce zabýval a které jsem dále použil při vytváření systému pro rozvrhování a simulaci výroby. Nejprve se budu věnovat využití platformy Windows SharePoint Services 3.0 a Google DOCs při rozvrhování za účelem vytvoření uživatelsky příjemného rozhraní s možností obsluhy více uživateli najednou. Dále jsou popsány možnosti využití Petriho sítí při rozvrhování a na závěr stručný popis možností profesionálních rozvrhovacích softwarových balíků jako například ASPROVA.

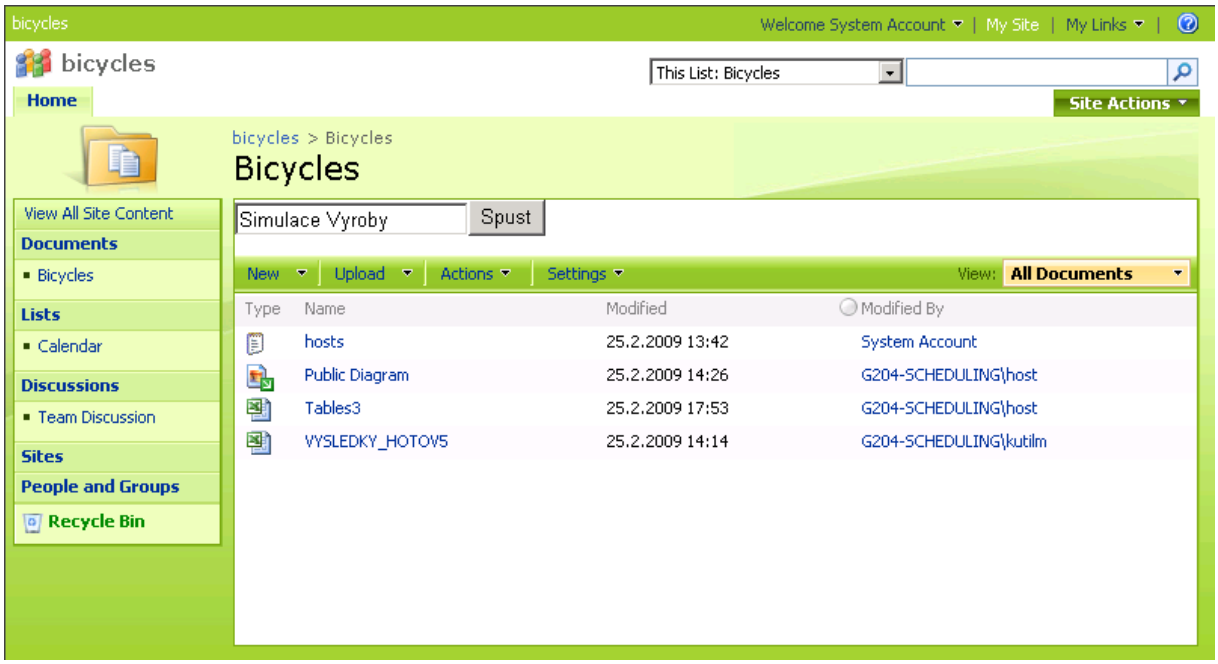
3.1 Produkty hromadné správy elektronických dokumentů

V této podkapitole jsou popsány hlavní funkce a vlastnosti systémů pro elektronickou správu dokumentů s využitím internetu. Jedná se o produkt *Microsoft SharePoint* a *Google Docs*

3.1.1 MS - Sharepoint

Sada produktů a technologií Microsoft SharePoint Products and Technologies, která se primárně skládá ze služby Microsoft Windows SharePoint Services 3.0 (WSS 3.0, je zdarma s licencí na Microsoft Windows Server) a serveru Microsoft Office SharePoint

Server 2007 (MOSS, nadstavba pro WSS 3.0 s placenou lincencí), byla vyvinuta jako platforma určená pro použití v oblasti portálů a spolupráce. (O'Connor, E., 2008) Služby SharePointu jsou využitelné hlavně pro vytváření specializovaných webů viz obr. 3.1, které lze pomocí dostupných šablon, knihoven a webových částí dále rozšiřovat. Jelikož jsou WSS 3.0 pro naše účely naprosto dostačující, zaměřím se v dalším textu na tento produkt. MOSS poskytuje rozšíření o další šablony, knihovny a webové části a dále umožňuje hlubší integraci s produkty MS Office.



Obrázek 3.1: Ukázka webové stránky vygenerované pomocí WSS 3.0

Struktura WSS 3.0 se skládá z pěti částí:

- Microsoft SQL server
- Databázové služby, kontrola přístupu, správa dokumentů
- Mechanismus zajištění webu skládající se z jednoho nebo několika webových serverů ISS
- Vzorové šablony webů
- Webové části (Web parts) - softwarové komponenty

WSS 3.0 obsahuje řadu předpřipravených šablon webů, které lze okamžitě použít. Jsou mezi nimi například:

- Týmový web. Obsahuje knihovnu dokumentů, posílání zpráv a oznámení, kalendáře, úkolovník, diskusi
- Wikiweb. Obsahuje snadno upravitelné stránky, které lze vzájemně propojovat
- Blog. Možnost ukládání článků, postřehů.
- Prázdný web. Umožňuje pomocí komponent vkládat potřebné části webu.

Jelikož druhů portálů a knihoven, které lze v WSS vytvořit a používat, je poměrně mnoho, zaměřím se pouze na popis těch částí, které jsem použil při tvorbě diplomové práce. Nově vytvořený portál má v horní části pruh obsahující odkazy na možnosti správy vlastního uživatelského účtu. V levé části je umístěn sloupec s odkazy pro navigaci na webu a v pravé části je prostor pro zobrazení požadovaných knihoven, diskusí atd.

Dle přidělených práv může uživatel přistupovat ke sdíleným knihovnám dokumentů, případně pracovat se svými dokumenty v soukromé složce.

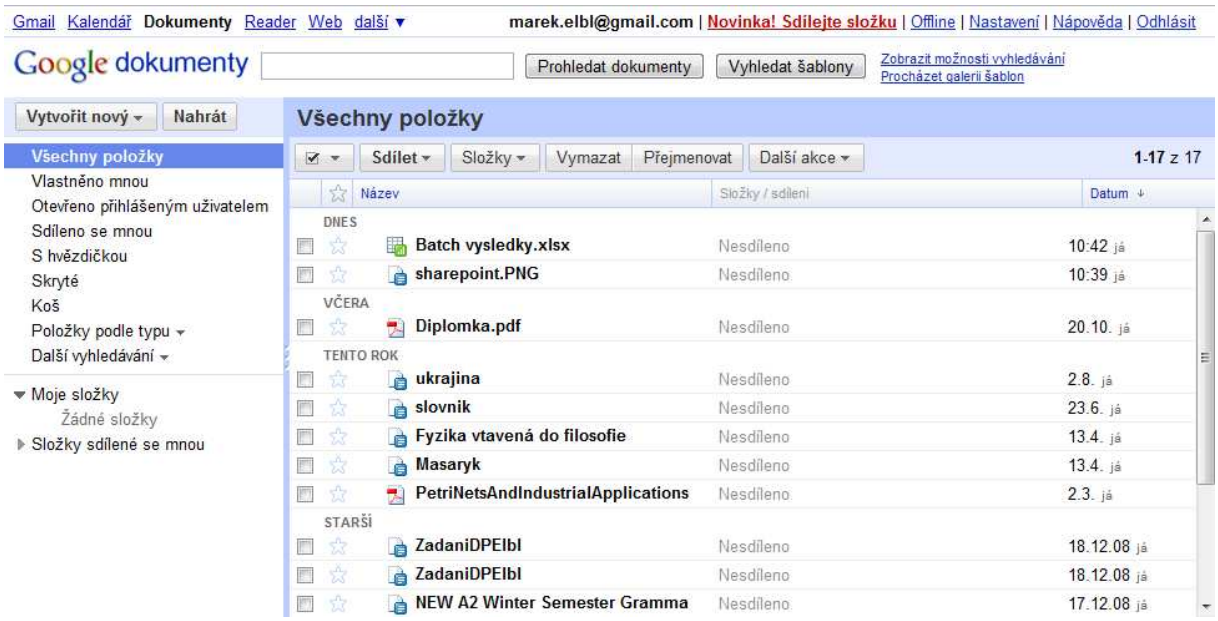
Knihovny dokumentů mohou obsahovat soubory jakéhokoliv typu. WSS se stará o správu verzí dokumentů a ke každé verzi je možno se vrátit. Editace probíhá tak, že se soubor otevře v příslušném editoru přímo ze serveru. Až do ukončení editace je soubor chráněn proti zápisu a jiní uživatelé nemohou soubor editovat. Uživatel má možnost navíc provést rezervaci dokumentu (check-out). Při rezervaci se vytvoří (není to však povinnost) lokální kopie souboru na klientském počítači. Až do uvolnění dokumentu (check-in) nemohou ostatní soubor editovat a ani nevidí jeho změny. Rezervace a uvolnění souboru se provádí rovnou na webovém portálu. U dokumentů typu MS Office je možno provádět rezervaci a uvolnění přímo z aplikace.

Jelikož uživateli většinou nevyhovuje vzhled stránek, případně nestačí funkce v předem připravených šablonách, existuje několik možností jak si daný web přizpůsobit svým potřebám. Pro úpravy vzhledu stránek a rozložení jednotlivých komponent na nich lze použít nástroj MS Sharepoint Designer. Jedná se o program, který natáhne aktuální vzhled webu a umožní ho editovat pouze pomocí myši bez nutnosti programování. Přidání dalších komponent na stránku, v terminologii WSS 3.0 takzvaných knihoven, jako je například správa dokumentů, obrázků, formulářů, atd., může provést uživatel s dostatečnými právy přímo na dotyčné webové stránce. I v tomto případě je uživatel vázán pouze knihovnami, které jsou ve WSS 3.0 k dispozici (případně může využít rozšíření v MOSS). Třetí

možností je využití webových částí (web parts). Ty je možné vkládat jako jakési bloky na webové stránky. Zde je opět možné využít řady předpřipravených částí, jako jsou například seznamy, diskuse, kalendář, zobrazování úkolů, načítání dat z XML souborů atd. a tyto části vložit přímo na dané stránce přes webové rozhraní. Webové části si ale může člověk vytvořit sám pomocí MS Visual Studia.

Jelikož celý systém pracuje na platformě ASP.NET, lze s takovouto webovou částí pracovat jako s klasickou .NET aplikací. Díky tomu lze využívat například přístupu k souborům fyzicky na serveru i v SharePointu, využívat „ActiveX“ pro manipulaci s daty například v dokumentech Office, použít „Excel services“ pro přístup k datům v tabulkách MS Excel uložených v SharePoint serveru atd. Detailnější popis využití MS SharePoint Services 3.0 pro systém rozvrhování výroby je uveden v kapitole 5.

3.1.2 Google docs



Obrázek 3.2: Ukázka úvodní stránky Google Docs

Google docs je soubor webových aplikací zdarma poskytovaných firmou Google. Zaměřuje se na kancelářské aplikace. Pro vytvoření uživatelského účtu je nutná registrace. Do Google docs je možno ukládat běžně užívané typy souborů jako například dokumenty MS Office, OpenOffice, PDF, HTML, prostý text atd. viz obr. 3.2. Import neznámých typů

souborů není povolen. Rozložení webu je podobné jako u „MS - SharePoint“, tedy v horním pruhu odkazy týkající se uživatelského účtu a nalevo sloupec s navigací na webu.

Každý soubor nebo složka v uživatelském účtu je možné sdílet s ostatními uživateli v rámci registrovaných uživatelů Google, případně povolit přístup neregistrovaným uživatelům. Je možno vracet se k předchozím verzím dokumentů.

Editace dokumentů je umožněna v rámci webových aplikací a obsahuje:

- textový editor
- tabulkový procesor
- tvůrce prezentací
- tvorbu formulářů

Soubory může editovat více uživatelů najednou. Změna se po uložení dokumentu okamžitě zobrazí i na druhé straně. Nejlépe tuto funkci podporuje Tabulkový editor, kde se zobrazuje aktuální buňka a činnost druhého uživatele.

K tabulkám na serveru Google docs lze přistupovat pomocí programového API. Google nabízí potřebné knihovny pro čtyři programovací jazyky (Java, .NET, PHP, Python) na `http://code.google.com/p/google-gdata/downloads/list`. V prostředí .NET je po instalaci třeba přidat na tyto knihovny reference. Toto API umožňuje načítat mimo jiné i data o obrázcích ve fotogalerii, videu na serveru youtube, kalendářů atd. a některá data i editovat. U tabulek je možno načíst seznam uložených souborů, a dále načítat a editovat data v jednotlivých buňkách tabulek. Soubory s tabulkami na serveru není možné přidávat ani mazat.

3.2 GLPK

GLPK, neboli „GNU Linear Programming Kit“ je softwarový balík, který je (jak již z názvu vyplývá) šířen pod GNU licencí a je určen k řešení úloh lineárního programování (LP), celočíselného lineárního programování (ILP) a smíšeného lineárního programování (MIP). GLPK podporuje „MathProg modeling language“ což je jazyk pro popis lineárních matematických modelů a je podmnožinou jazyka AMPL (AMPL, 2004).

Úlohy klasického lineárního programování jsou řešitelné v polynomiálním čase. Při řešení úloh celočíselného programování nebo smíšeného celočíselného programování nelze

použít řešení LP problému a pouze ho zaokrouhlit. Není totiž zajištěno, že by bylo řešení optimální a dokonce ani, bylo-li by přípustné. V tomto případě se již jedná o úlohy NP-obtížné (NP-hard).

GLPK využívá pro hledání řešení úloh ILP respektive MIP metody větví a mezí (Branch and Bound). Tato metoda se skládá ze dvou fází. V první fázi je zanedbán požadavek na celočíselnost a je klasickými metodami LP nalezeno pouze neceločíselné řešení. V případě že by byli všechny požadované proměnné celočíselné, tak výpočet končí. V opačném případě se vybere libovolná proměnná x_i, u které je požadavek na celočíselnost $x_i = y_i$ taková, že $y_i \neq \boldsymbol{Z}$. Vyšetřovaná oblast dané proměnné se rozdělí na dvě množiny $x_i \leq |y_i|$ a $x_i \geq 1 + |y_i|$. Pro tyto dvě množiny je výpočet LP rekurzivně opakován, dokud není nalezeno přípustné řešení, kde jsou všechna x_i celočíselná (Šůcha, P., 2004). Tyto skutečnosti jsou využity při návrhu heuristik v kapitole 4.

Program je volně dostupný na internetu a po stažení je nutno ho zkompilovat pro vlastní počítač. Výsledkem je spustitelný soubor, kterému se jako vstup vkládají soubory, s nadefinovaným MIP modelem a se vstupními daty. Příklad použití viz kapitola 5.

3.3 Petriho sítě

V moderních složitých systémech je zapotřebí prostředku pro modelování a analýzu. Může se jednat o výrobu, procesní řízení, komunikační systémy, úlohu rozvrhování atd. Jedním z vhodných prostředků poskytujících jednoduchou grafickou reprezentaci a zároveň matematický formalismus pomocí stavových rovnic, jsou Petriho sítě (Zurawski , R., MengChu Zhou, 1994) (dále PN).

3.3.1 Druhy Petriho sítí

Mimo obyčejných Petriho sítí existují i různá rozšíření. Je to například rozšíření o čas, dále hierarchické Petriho sítě, kde je základní myšlenkou rozdělení sítě do více modulů (Jensen, K., 1998). V obyčejných Petriho sítích tokeny nenesou žádnou informaci o své identitě, což je požadováno například pro mapování trasy zdrojů ve výrobě nebo zpráv. Tento problém lze řešit buď rozložením původní sítě na více sub-sítí pro každý druh tokenu, nebo použít *High-level* Petriho sítí, jako jsou například:

- Predicate-transition nets

- Barevné Petriho sítě - CPN (komunikační systémy, produkční systémy ...)
- Sítě s „jednotlivými tokeny“

V těchto sítích může token nést informaci (int, string, seznam ...). Dále byly vyvinuty nástroje jako objektové Petriho sítě, kde token je instancí třídy, která je definována v určitém seznamu. To bylo použito například u analýzy flexibilních výrobních systémů (FMS) nebo sestavovacích úloh. Pro průmyslové řízení byly použity Fuzzy Petriho sítě. Petriho sítě jsou většinou konstruovány Ad-hoc. Stále častěji se však objevují práce, snažící se tvorbu Petriho sítí systematizovat viz (Zurawski , R., MengChu Zhou, 1994).

3.3.2 Použití Petriho sítí

Petriho sítě jsou vhodné pro modelování složitých systémů, ve „workflow managementu“ nebo v dávkových procesech. Modely výroby jsou chápány jako systémy diskrétních událostí a využívají se pro její verifikaci (Delgadillo, G. M., Llano, S. P., 2007). Dále se Petriho sítě využívají pro modelování a analýzu komunikačních protokolů (jednoduchý příklad pomocí CPN viz (Jensen, K., 1998, strana 3)) nebo komunikačních sítí (Expressnet, Fastnet, D-net, U-net, Token Ring...). Ve výrobních procesech byly Petriho sítě aplikovány u výrobních linek s buffery, job-shop, machine-job, procesy výroby automobilů, FMS, systémů se sdílenými zdroji, systému Kanban. Petriho sítě s rozšířením o čas a heuristické prohledávání lze uplatnit pro úlohu rozvrhování ve výrobě (Zurawski , R., MengChu Zhou, 1994) nebo lze pomocí časových Petriho sítí provádět pouze sledování (mapování) úlohy rozvrhování (jednotlivých úloh)(Van der Aalst, Llano, S. P., 1996). Dále se s Petriho sítěmi můžeme setkat u robotických systémů, při plánování trajektorie, nebo při modelování sekvenčního řídicího systému. V neposlední řadě můžeme Petriho sítě vidět při návrhu velkých softwarových balíků pro jejich specifikaci, simulaci, atd., kde se používají barevné hierarchické Petriho sítě (Zurawski , R., MengChu Zhou, 1994). Praktické ukázky na příkladech a další možnosti využití PN v produkčních systémech jsou ukázány např. v článku (Silva, M. et al., 1998). Další oblastí použití PN byla implementace řídící logiky. Zde se úspěšně používá modifikace Petriho sítě - Grafcet (Delgadillo, G. M., Llano, S. P., 2007).

3.3.3 Simulace a Petriho sítě

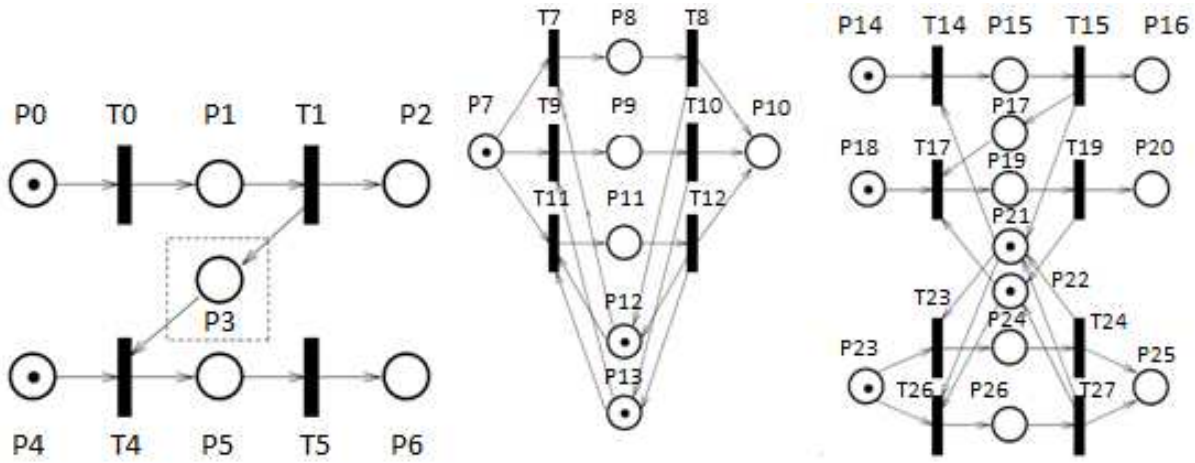
V článku (Jensen, K., 1998) je věnována pozornost simulacím barevných Petriho sítí, nicméně závěry lze použít obecně. Simulace PN modelů lze použít pro ladění a prozkoumání vlastností složitých modelů. Různé PN simulátory lze použít buď tak, že uživatel prochází simulací krok po kroku a ověřuje všechny detaily různých značení. Tento postup je značně pomalý. Simulátory lze také využívat v automatickém režimu, kdy potřebná „náhodná“ značení jsou volena např. generátorem náhodných čísel, čímž se značně urychlí simulace. Výsledky simulace lze ukládat různě, například jako seznam všech značení v jednotlivých krocích pro každé místo nebo přidáním míst do PN modelu tzv. „report places“ ve kterých se nám ukládá informace o množství prošlých tokenů. (Jensen, K., 1998)

3.3.4 Analýza PN

Pro ověření správnosti modelu není dostatečná pouze simulace s „nějakým“ počátečním značením (Jensen, K., 1998). PN modely jsou vhodné pro analýzu problému, ze které lze například zjistit pravděpodobné hodnoty rychlosti výroby, dobu výroby, kapacitu pro komunikační a mikroprocesorové modely, multiprocesorové systémy, spolehlivosti modelu atd. (Zurawski , R., MengChu Zhou, 1994). Zde se používá analýza stavového prostoru pro všechna dosažitelná značení. Tato analýza se provádí na základě grafu dosažitelných značení, ve kterém každý vrchol značí dosažitelné značení a každá hrana značí přechod od jednoho dosažitelného značení ke druhému (odpálení přechodu). Tvorba grafu dosažitelných značení z CPN viz (Jensen, K., 1998). Z těchto grafů lze generovat reporty, ze kterých je patrné například to, jak veliký je stavový prostor pro daný model, lze nalézt silně souvislou komponentu stavového grafu, maximální resp. minimální počet daných tokenů v místech. Dále lze prohledávat nejkratší cesty v grafu (nejmenší počet odpálených přechodů, případně tak řešit úlohu optimalizace) atd. Hlavní nevýhodou analýzy stavového prostoru je tzv. „state explosion problem“. To znamená, že se stavový prostor zvětší tak, že ho nelze již analyzovat. Další nevýhodou je, že analýza je stále jen pro konkrétní počáteční značení a může postihovat jen jednu z mnoha konfigurací systému. Existují i jiné metody analýzy poskytující matematický důkaz, že systém pracuje správně, např. analýza P/T invariant. Nicméně použití těchto metod je většinou složitější a její výsledky lze interpretovat pro konkrétní problém různě (Jensen, K., 1998). Proto se o těchto metodách zmíním až u konkrétních příkladů níže.

3.3.5 Rozvrhování a Petriho sítě

Jedna z možností využití PN při rozvrhováním je popsána v (Van der Aalst, Llano, S. P., 1996). Jedná se o způsob mapování a analýzy rozvrhu (Job-shop, Machine-scheduling) s využitím časovaných PN. Analýzou PN lze potom zjistit konfliktní, redundantní relace následností nebo horní/dolní omezení v místech. Pro mapování nějakého rozvrhu pomocí PN je zapotřebí provést překlad z oblasti rozvrhování, do oblasti časovaných PN (formální zápis viz (Van der Aalst, Llano, S. P., 1996, strana 11)). Jinými slovy převést „úlohy“ a „zdroje“ na „místa“ a „přechody“ v časovaných PN. Každá „úloha“ je složena ze třech míst, které odpovídají stavu, kdy „úloha t“ čeká na zpracování, je zpracovávána a je dokončena. Přechody s použitým časem značí release time a complition time úlohy. Dalším místem lze modelovat sdílený zdroj atd. viz obr. 3.3 a obr. 3.4.



Obrázek 3.3: a) Model dvou Úloh s relací následností b) Úloha může být vykonán třemi různými zdroji c) Příklad modelu Job-shopu s dvěma joby 1 a 2 a třemi úlohami A, B a C.

Na takto namodelovaný rozvrh lze použít výše popsané analýzy a popsat tak jeho:

a) Strukturální vlastnosti: Na základě P-invariant lze zjistit konflikt v relacích následností, případně takto odstranit nadbytečné relace následností. Na základě T-invariant jsme schopni určit, jestli je síť propojená/připojená (connected) a na základě toho určit, jestli by nešlo rozdělit rozvrhovací problém na dva oddělené problémy.

b) Vlastnosti dynamického chování sítě: Na základě grafu dosažitelných značení lze získat všechny dosažitelné odpalovací sekvence, které vždy odpovídají dosažitelnému rozvrhu. To ovšem neznamená všechny dosažitelné rozvrhy, ale pouze ty, které lze

generovat algoritmem *eager* (často označovaným jako *list schedule*). Algoritmus *eager* generuje rozvrhy, kde jsou úlohy vykonávány hned, jak je to možné. Graf může narůst do velkých rozměrů, což lze částečně omezit pomocí metod a heuristik viz (Van der Aalst, Llano, S. P., 1996, strana 16). Z tohoto grafu lze také určit horní a dolní omezení doby rozvrhu.

Další oblastí použití, je hledání optimálního rozvrhu, přímo pomocí produkčního modelu v PN. Pro hledání optimálního výrobního rozvrhu lze implementovat „Beam Search algoritmus“. Dále se používá metoda „branch and bound“, která byla následně rozšířena o „dispatching rules“ pro výběr odpalovaného přechodu. V praxi nejpoužívanější heuristiky jsou právě „dispatching rules“ (LPT, SPT ...) s Produkčním modelem např. právě v PN. Také byly navrženy Heuristky pro generování částí grafu dosažitelných značení (prohledávání do hloubky, do šířky, smíšeně a A* algoritmus) (Delgadillo, G. M., Llano, S. P., 2007). V (Mejía, G., Odrey, N., 2005) byl navržen přístup, kombinující algoritmus A* se strategií výběru nodů. Tento postup měl výsledky bližší optimu s menší výpočetní náročností.

Výzkum je většinou zaměřen na hledání algoritmů pro rozvrhování, nebo na vývoj nástrojů pro modelování složitých systémů. V (Delgadillo, G. M., Llano, S. P., 2007) byl představen přístup, kde je rozvrhování založené na PN a v němž je modelovací platforma oddělená od plánovače. Tento postup byl v praxi testován v Kolumbijské tiskové společnosti INTERGRÁFICAS S.A. (flexibilní výroba, Job-shop, mnoho paralelismů atd ...). Nejprve je produkční plán namodelován pomocí PN (Marked Timed - Place Petri Nets (TPPNs)). Potom se vykonává algoritmus simulace PN, ve které může dojít ke konfliktní situaci (operační konflikt - lze odpálit např. jeden i druhý přechod). V případě možnosti odpálení více přechodů $u_i \in \vec{u}$, je podle některého z kritérií určeno, který přechod je pro odpálení nejvhodnější (zde dochází ke spolupráci s rozvrhovačem - popsáno níže). Operační konflikt v PN modelu může znamenat, že pro jeden job vybíráme jeden nebo druhý stroj (alternativní výrobní postupy), případně že 2 joby soupeří o jeden stroj. Popsaná aplikace fungovala tak, že načetla informace z PN modelu, vztahující se k příslušnému výrobnímu plánu a vytvořila Ganttův diagram. Aplikaci lze popsat následujícím algoritmem:

1. Vytvoří instanci třídy Petriho sítě

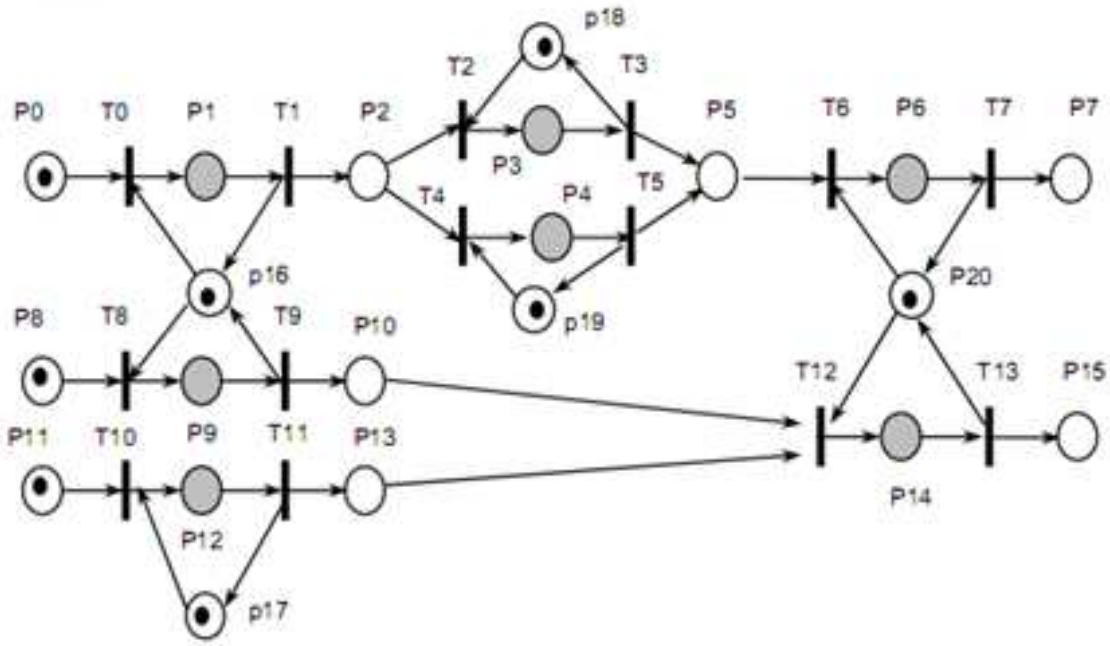
2. Vytvoří instanci třídy plánovače (Scheduler) a vybere pravidla pro odpalování přechodů (Dispatching rules - Shortest Processing Time (SPT), Longest processing

time (LPT), weighted shortest process time (WSPT), Earliest Due Date (EDD), Critical Ratio (CR), Minimum Slack (MS), Apparent Tardiness Cost (ATC))

3. Spustí algoritmus simulace PN až do konečného stavu (v průběhu simulace jsou na základě vybraných dispatching rules vybírány úlohy/zdroje, které se provedou/použijí)

4. Vygeneruje Ganttův diagram a další systémové ukazatele.

Pro zhodnocení výkonu systému byly měřeny a spočítány tyto systémové ukazatele: seznam Jobů, které byly vykonány pozdě; nejdelší doba vykonání jobu (makespan); vytížení strojů %; celkový čas; „nedochvilnost“ jobů (Total Weighted tardiness).



Obrázek 3.4: Systém se sdíleným zdrojem a alternativní cestou při výrobě

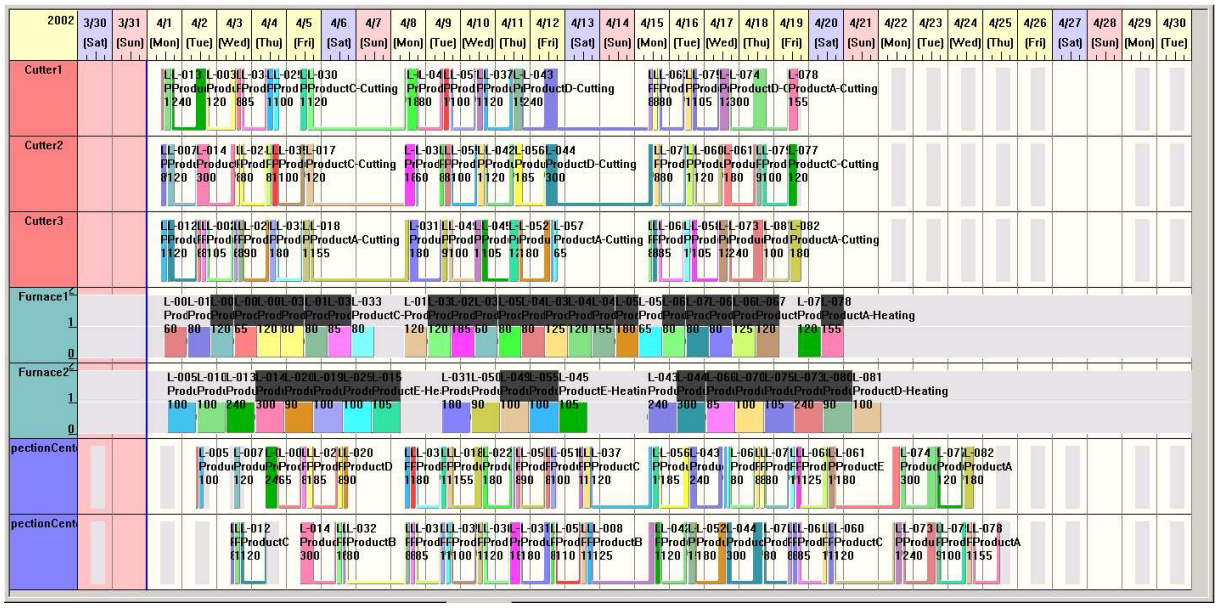
3.3.6 Automatická tvorba PN

Při tvorbě modelu v PN se často postupuje tak, že se vytvářejí bloky PN modelů. Z těchto bloků se potom celý systém sestavuje. Tento postup není příliš vhodný pro analýzu celého systému. Poslední dobou se objevují snahy o automatizaci tvorby Petriho sítí. Například

v (Gradisar, D., Music, G., 2007) je využit popis flexibilních systémů pomocí modelovacího jazyka FMS, nebo maticového modelu diskrétních událostí FMS. Tento popis je přeložen do Petriho sítí. Dále je zde představen přístup, kde je na základě produkčních dat automaticky vytvořena PN. Jako zdrojová data jsou použita data z výrobního systému plánování zdrojů (MPR11). Místo modelu matice diskrétních událostí, používá účet za materiál (bill of materials BOM) a trasy materiálu (routing). Výstupem je model systému pomocí časové PN, kde čas je zaveden pomocí principu „holding-duration“. Takto vytvořená PN je potom vložena do simulátoru, který vygeneruje Ganttův diagram. V tomto případě simulátor opět pro rozhodnutí o přiřazení určitého zdroje k úloze využívá „dispatchig rules“.

3.4 Asprova

Jedním z komerčních produktů pro rozvrhování výroby je program Asprova. Jedná se o takzvaný *Advanced Planning & Scheduling (APS) system*. APS lze obecně definovat jako proces řízení výroby, ve kterém dochází k optimalizované alokaci zdrojů a materiálu, nutných k zajištění poptávky. Výsledkem tohoto procesu je tedy plán výroby viz (Petr Plaček, 2008).



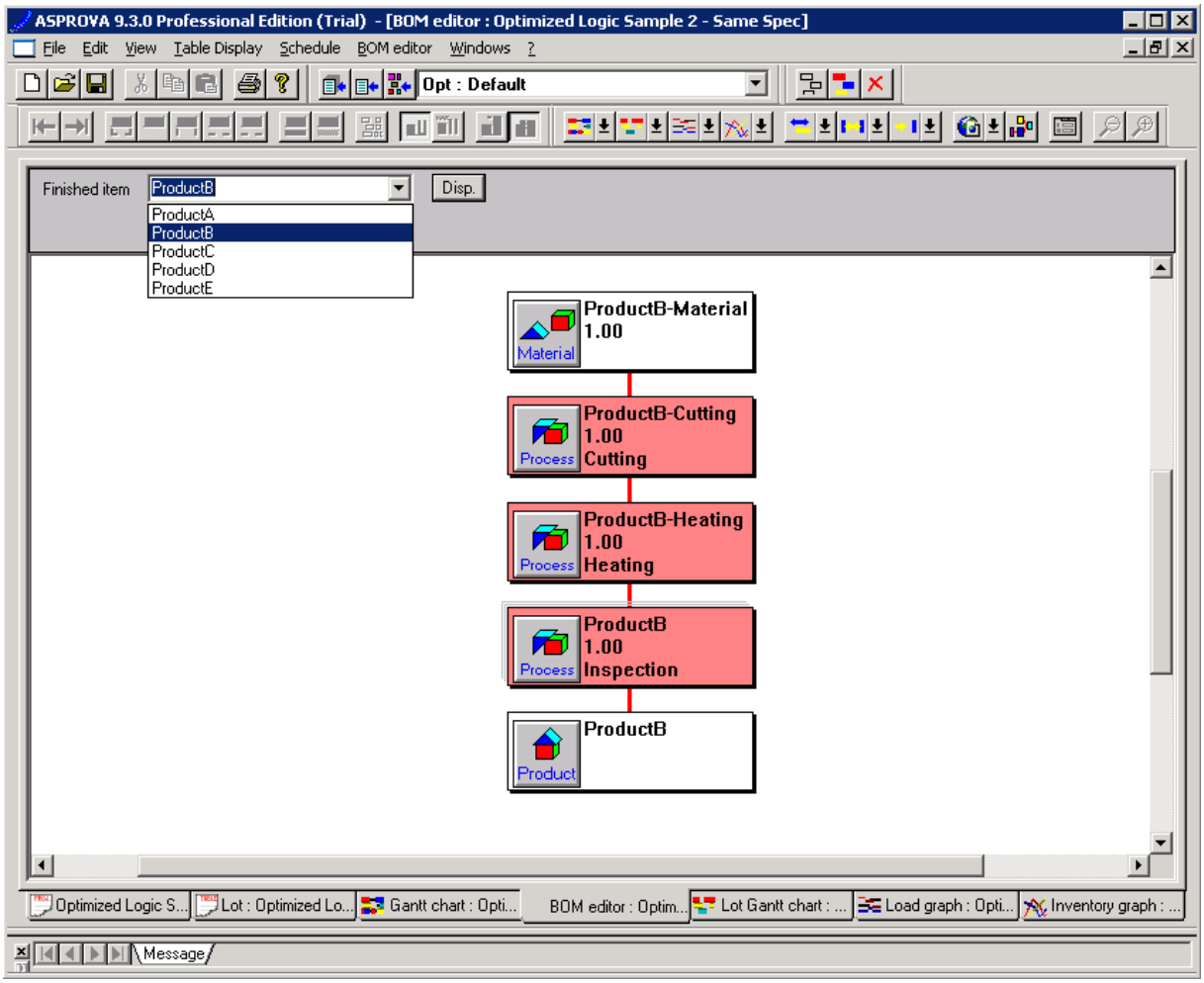
Obrázek 3.5: Ganttův diagram výrobního plánu pomocí ASPROVA v9

Jednou s pozitivních vlastností systému ASPROVA je grafický interface, pomocí kterého je zadávána struktura výroby (uživatel definuje tzv. *Bill of materials (BOM)*). Zde definuje jednotlivé výrobky a strukturu jejich výroby. To znamená potřebné zdroje materiálů, stroje a posloupnosti těchto operací viz obr. 3.6.

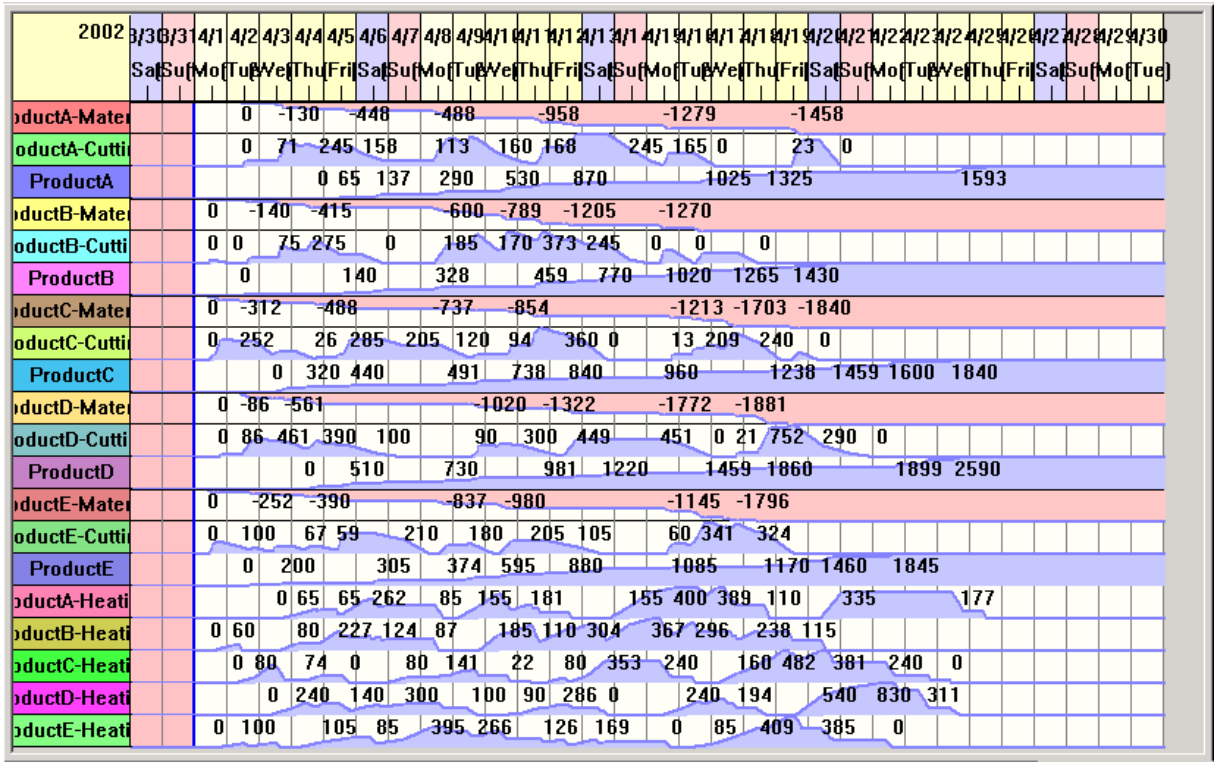
Pro takto definovanou výrobu lze spustit optimalizaci výrobního plánu. Aplikace umožňuje zobrazit několik možných výstupů. Jedním z nich je klasický Ganttův diagram (viz obr. 3.5), množství vytížení skladů v průběhu výroby (viz obr. 3.7), průměrné vytížení strojů za den atd.

Tento systém poskytuje řadu možností nastavení jako například pracovní doby a svátky, umožňuje export dat do tabulek, dále je poskytnut interface prostřednictvím aplikace Microsoft Excel, pomocí kterého může uživatel systém ovládat.

Dalším produktem s podobnými vlastnostmi je například aplikace Orchestrate.



Obrázek 3.6: Způsob definování BOM v ASPROVA v9



Obrázek 3.7: Zobrazení průběžného vytížení skladů

Kapitola 4

Heuristické řešení MIP modelu

V kapitole 2 bylo na základě článku (Sawik, T., 2002) popsáno MIP řešení problému rozvrhování dávkových procesů, na m paralelních pracovních stanicích za sebou (FFL), oddělených omezenými buffery. Řešení tohoto problému pomocí MIP sice garantuje nalezení optimálního řešení, nicméně je vhodné pouze pro malé počty dávek a výrobků. V této kapitole jsou popsány mnou navržené algoritmy, které urychlují dobu hledání řešení problému. Nejprve se jedná o heuristiky urychlující řešení MIP a na závěr je popsána heuristika využívající svůj vlastní matematický model odvozený od MIP modelu a pracující na základě metody Tabu Search.

Hlavní myšlenkou následujících algoritmů je neřešit problém jako celek, ale řešit vždy jen část celkového problému a postupnými kroky se blížit k optimálnímu řešení. Algoritmy byly programovány v prostředí Matlab a pro řešení MIP byl použit program GLPK viz podkapitola 3.2.

4.1 Fixace y_{fg}

Tento algoritmus urychluje výpočet zafixování proměnných v matici $\boldsymbol{y}$. Matice $\boldsymbol{y}$ udává, zda je dávka f před dávkou g. V takovém případě je hodnota $y_{fg} = 1$, v opačném případě je $y_{fg} = 0$. Algoritmus má tolik kroků, kolik je jednotlivých dávek. V každém kroku se řeší postavení jedné dávky, kterou si označíme l, vůči ostatním. To je zajištěno tak, že se zafixují hodnoty y_{fg}, které nijak s dávkou l nesouvisí a takto definovaný problém se nechá vyřešit. Jsou to takové prvky y_{fg}, kde $f \neq l$ a zároveň $g \neq l$ viz obr. 4.1. Samotná fixace je realizována tím způsobem, že se hodnoty určené k zafixování zapíší jako další podmínky

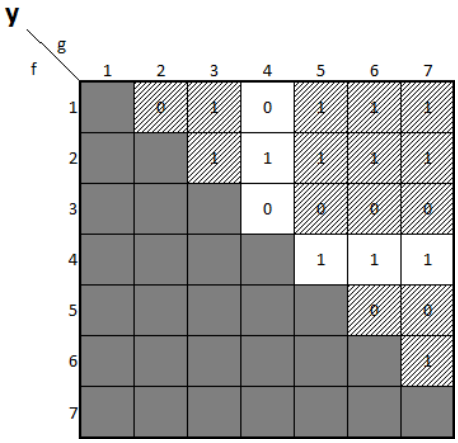
k matematickému modelu výroby. Urychlení výpočtu pomocí tohoto algoritmu spočívá v tom, že se pomocí zafixování hodnot matice $\boldsymbol{y}$ zredukuje celkový stavový prostor, který je v rámci řešení MIP prohledáván. Pseudokód algoritmu:

```
begin
    y = vytvoř inicializační matici;  // například y_fg=0 pro všechny f a g
    batches = seřazený vektor s identifikátory všech dávek;
    for i=1:length(batches)
        sol = batches(i);
        Zapiš do souboru s modelem jako omezení všechny hodnoty yfg, kde f!=sol a g!=sol;
        y = Spusť řešení MIP; // GLPK solver
    end
end
```

Tento algoritmus má následující vlastnoti:

+ Zafixování části proměnných y_{fg} velmi zrychluje část výpočtu, hledající celočíselné řešení

- Zásadně neurychluje LP část hledání řešení (počet zafixovaných proměnných y_{fg} je vzhledem k celkovému počtu proměnných zanedbatelný)



Obrázek 4.1: Matice $\boldsymbol{y}$ pro $l = 4$, šrafovaná část označuje fixované hodnoty

4.2 Fixace prvních dávek, postupná optimalizace

Tento algoritmus se oproti minulému zaměřuje na větší zredukování celkového stavového prostoru. V každém iteračním kroku řeší jen omezený počet dávek. Tento algoritmus opět

začíná vytvořením inicializační matice $\boldsymbol{y}$, kde mohou být například všechny $y_{fg} = 0$ a z ní vytvoření inicializačního rozvrhu pořadí dávek. Z inicializačního rozvrhu vezme algoritmus tolik prvních dávek, kolik odpovídá definované velikosti „optimalizačního okna“. Pro tyto dávky vypočteme pomocí MIP optimální rozvrh. Dávku, která je ve výsledném částečném rozvrhu jako první (a které ještě není zafixovaná) zafixujeme (příslušné hodnoty x_{ijk}, c_{ik}, d_{ik}). K aktuálně řešené úloze přidáme další dávku z inicializačního rozvrhu a opět vyřešíme pomocí MIP. Takto se algoritmus opakuje, dokud není odebrána z inicializačního rozvrhu poslední dávka.

```
1       begin
2           y = "vytvoř inicializační matici";  // například yfg=0 pro všechny f a g
3           batches = "seřazený vektor s identifikátory všech dávek dle" y;
4           temp_schedul = []; // částečný rozvrh určený pro řešení pomocí GLPK
5           window = 3;
6           while batches !=0
7               if temp_schedul==0
8                   temp_schedul = batches(1 "až" window);
9                   batches = batches(window+1 "až" end);
10              else
11                  temp_schedul = [temp_schedul, batches(1)];
12                  batches = batches(2 "až" end);
13              end
14              temp_schedul = Spusť_řešení_MIP(temp_schedul); // GLPK solver
15              "Zafixuj všechny hodnoty xijk, cik, dik pro výrobky odpovídající dávce" temp_schedul(1);
16          end
17      end
```

Tento algoritmus má následující vlastnosti:

+ Lze s ní řešit velké instance problému

- Pro malé optimalizační okno má výsledný rozvrh velkou odchylku od optimálního řešení

4.3 Redukce modelu výroby

Tento algoritmus se zaměřuje na získání přesnějšího inicializačního rozvrhu. Toho je dosaženo redukcí modelu výroby (snížení počtu výrobních stanic v modelu výroby) a následným vyřešením. To znamená, že není vyřešen celý původní model výroby, ale je vytvořen „podobný“ model s menším počtem pracovních stanic. Snahou je zanechat ve výrobním modelu jen ty pracovní stanice, které ovlivňují výsledný rozvrh nejvíce

(s největší dobou trvání). Celou výrobní linku zredukujeme na určitý definovaný počet strojů i s největším mediánem doby trvání r_{ig}, $\underset{g \in G}{med}(r_{ig}) : \forall i$.

MIP řešení zredukovaného modelu za účelem získání inicializační hodnoty matice $\boldsymbol{y}$ má tyto vlastnosti:

+ Velké urychlení části výpočtu hledající neceločíselné řešení

- Pomalé hledání celočíselného řešení

Pokud tuto získanou matici $\boldsymbol{y}$ využijeme při řešení kompletního modelu, docílíme urychlení celočíselného řešení, nicméně urychlení prvotní fáze hledání neceločíselného řešení je zanedbatelné.

4.4 Kombinace jednotlivých algoritmů pro řešení MIP modelu

Výsledný algoritmus vychází z podkapitoly 4.3 a k potlačení nevýhod tohoto postupu využívá algoritmy z kapitol 4.1 a 4.2. V prvním kroku je získán inicializační rozvrh pomocí redukce modelu výroby viz 4.3. Zde byl nevýhodou fakt, že měl tento postup časově náročnou část hledání celočíselného řešení. Ke zlepšení je zde uplatněn algoritmus z kapitoly 4.1 využívající fixace $\boldsymbol{y_{fg}}$. Výsledkem je získaná inicializační matice $\boldsymbol{y}$. Tato matice řeší postavení všech dávek vůči sobě, a proto je použitelná i pro kompletní model výroby. V následujícím kroku se tedy řeší již kompletní model. Pro urychlení hledání řešení je použit algoritmus z kapitoly 4.2, který používá výše získanou inicializační matici $\boldsymbol{y}$.

```
1   int x // počet strojů v redukované výrobní lince
2   vyber "x" strojů s největším mediánem rig
3   vytvoř z těchto strojů nový model výroby
4   vyřeš model pomocí heuristiky: fixace yfg
5   použij výslednou hodnotu yfg jako inicializační
6   Vyřeš kompletní model pomocí heuristiky: Fixace prvních dávek, postupná optimalizace
```

4.5 Heuristické řešení MIP modelu a algoritmus Tabu search

4.5.1 Minimalizace kritéria C_{max}

Všechny dosud popsané algoritmy využívaly k urychlení výpočtu převážně metody vedoucí k zredukování stavového prostoru MIP modelu. Určité proměnné byly zafixovány a pro ostatní byl MIP model vyřešen. Tento postup byl aplikován několikrát za sebou se snahou minimalizovat hodnotu C_{max}. V heuristice popsané v podkapitole 4.4 je pomocí MIP hledán optimální částečný rozvrh a výsledný rozvrh je z těchto dílčích rozvrhů složen. „optimalizační okno“ postupuje v rozvrhu dávek „z leva do prava“ a ačkoliv jsme díky využití MIP schopni garantovat, že částečný rozvrh bude v „optimalizačním okně“ vždy optimální, neumožňuje nám tento postup v každém kroku zjištění, jaká bude celková hodnota C_{max}. Pro zjištění této hodnoty (i se znalostí posloupnosti všech dávek) by bylo nutno dopočítat MIP model jako celek, což by byla časově velmi náročná operace. Z těchto výše popsaných důvodů nebylo možno využít obecně známé metaheuristiky jako například *metoda lokálního prohledávání*, *tabu search* a jiné.

Za tímto účelem jsem vytvořil algoritmus generující matematický model výroby na základě vložené posloupnosti dávek bez nutnosti jeho řešení pomocí MIP. Vstupem algoritmu je tedy pořadí dávek v rozvrhu v podobě vektoru *schedule* a konfigurace výrobní linky (hodnoty v, n, b_g, m_i, r_{ig}). Výstupem jsou hodnoty matematického modelu c_{ik}, d_{ik} a x_{ijk}. Tento algoritmus vychází z pravidel matematického modelu popsaných v kapitole 2.2.3. Základní myšlenkou je dopočítávání hodnot c_{ik}, d_{ik} a x_{ijk} pro každý výrobek k zvlášť. Tyto hodnoty jsou získávány postupně podle vložené posloupnosti dávek a jim odpovídajícím výrobkům. Každý výrobek postupně prochází, v rámci algoritmu, výrobním modelem od stanice $i = 1$ až k stanici $i = m$, kde jsou dopočítávány výše zmíněné hodnoty c_{ik}, d_{ik} a x_{ijk}. Algoritmus je popsána následujícím pseudo kódem:

```
begin
schedule = [posloupnost indexů dávek odpovídající rozvrhu];
načti data "v", "n", "bg", "mi", "rig"
Kg = getKg(v,Bg); // vytvoř matici K_g
mi_now(1:length(mi))=1 // ukazatel na stroj "j "ve stanici "i", určený k přiřazení výrobku
    for iter = 1 "až" length(schedule)
        g = schedule(iter);
        for k = "první až poslední výrobek v dávce"
            for i = 1 "až" m // přes všechny pracovní stanice
                j = mi_now(i); // stroj určený k přiřazení
                ksolved = find(xijk(i,j,:)==1); // seznam již rozvržených výrobků "k" na stroji "j"
```

```
12                  if isempty(ksolved)
13                      dik_min = 0;
14                  else
15                      dik_min = max(dik(i,ksolved));
16                  end
17                  xijk(i,j,k)=1; //výrobek "k" je přiřazen na stroj "j" stanice "i"
18                  cik1=dik_min+rig(i,g); // čas kdy bude výrobek dokončen na procesoru "i"
19                  if i==1
20                      cik2=0;
21                  else
22                      cik2=dik(i-1,k)+rig(i,g); // čas kdy byl výrobek dokončen na predchozím procesoru
23                  end
24                  if cik1 >= cik2
25                      cik(i,k)= cik1;
26                  else
27                      cik(i,k)= cik2;
28                  end

29                  if i == m // určím dik_min_next, neboli "departure time" na nísledující stanici
30                      dik_min_next=0;
31                  else // pokud nejsem na poslední stanici, tak zjistím kdy se uvolní přislušný procesor
                         // na následující stanici
32                      jnext = mi_now(i+1); // volný procesor na následující stanici
33                      ksolved = find(xijk(i+1,jnext,:)==1); // seznam výrobků "k" které jsou na daný
                                                              //procesor "j" již rozvrženy
34                  if isempty(ktemp)
35                      dik_min_next=0;
36                  else
37                      dik_min_next=max(dik(i+1,ktemp));
38                  end
39                  end
                    // zjistim kdy muze vyrobek opustit stage a nasatvim "departure time"
40                  if cik(i,k) <= dik_min_next
41                      dik(i,k)=dik_min_next;
42                  elseif cik(i,k) > dik_min_next
43                      dik(i,k) = cik(i,k);
44                  end
                    // provedu zvyseni hodnoty aktualniho procesoru na prirazovani
                    // ve viceprocesorove stanici
45                  if j < mi(i)
46                     mi_now(i)=mi_now(i)+1;
47                  elseif j==mi(i)
48                      mi_now(i)=1;
49                  end

50              end
51          end
52      end
53  end
```

Na základě výše popsaného způsobu získání hodnot matematického modelu jsem vytvořil optimalizační algoritmus s využitím metaheuristiky *tabu search* (Pelikán, J.,

2001). Metaheuristiky jsou zobecněné metody řešení optimalizačních úloh. Jejich úlohou není najít optimální řešení, ale řešení, které se k tomuto co možná nejvíce blíží v přijatelném čase. Množina přípustných řešení optimalizační úlohy se značí X a naším úkolem je nalézt optimální řešení $x^* \in X$, které minimalizuje účelovou funkci $f(x)$. Dále je definována množina $N(x)$, což je takzvané okolí bodu x, kde $x \in N(x)$. Existuje celá řada těchto metaheuristik. Patří mezi ně například metoda lokálního bodu, „tabu search“, prahové akceptace, SIAM nebo genetické algoritmy.

Metoda „tabu search“ využívá při řešení optimalizační úlohy seznamu již zpracovaných řešení. Tento seznam se označuje jako takzvaný „tabu list“ (TL). Při hledání následujícího řešení v okolí předcházejícího řešení se již nemusí zpracovávat řešení uložená v TL. Díky tomuto postupu se přesouváme v rámci okolí aktuálního řešení do takových řešení, která ještě nebyla navštívena. Tato metoda lze popsat následovně:

Krok 1 Zvolme výchozí přípustné řešení $x \in S$ a položme $x^* := x$.
Krok 2 Nalezneme $x' \in \bar{N}(x) \subset N(x)$, které nabývá minima funkce $f(x)$ na množině $\bar{N}(x)$. Množina $\bar{N}(x)$ obsahuje ta řešení z $N(x)$, která nejsou v TL
Krok 3 Zapišme řešení x do seznamu TL a dosaďme $x := x'$. Pokud je $f(x) < f(x*)$, pak dosaďme $x* := x$.
Krok 4 Pokud je splněno ukončovací kritérium, pak výpočet končí (a řešení $x*$ je výsledkem), jinak krok 2.

Ukončovacím kritériem může být například dosažení počtu určitého počtu kroků, případně pokud v určitém počtu kroků nedošlo ke zlepšení rozvrhu.

Mnou navržený algoritmus vychází právě z metaheuristiky *tabu search*. Algoritmus provádí částečnou optimalizaci rozvrhu v rámci „optimalizačního okna“ pro určitý počet sousedících dávek. Rozvrhem se myslí pořadí dávek tak, jak jsou zařazovány do výroby a je reprezentován vektorem, ve kterém má každý prvek hodnotu indexu určité dávky.

V prvním kroku hledá algoritmus takové pořadí *první* až *n-té* dávky (záleží na velikosti „optimalizačního okna“) v rozvrhu, které snižuje hodnotu celkového C_{max} a zároveň snižuje maximální hodnotu *času odchodu* d_{ik} na poslední stanici pro poslední výrobek v „optimalizačním oknu“ oproti nejlepšímu nalezenému řešení. Podmínkou, že musí být místo celkového C_{max} snížena i hodnota „C_{max} optimalizačního okna“, bylo dosaženo zrychlení algoritmu při stejné kvalitě nalezeného řešení. Při hledání takového rozvrhu, vytvoří algoritmus všechny možnosti pořadí dávek v rámci optimalizačního okna a pro takto

upravené rozvrhy dopočítá hodnoty matematického modelu s využitím výše popsaného algoritmu. Pokud rozvrh splňuje popsané podmínky, je uložen jako nejlepší nalezené řešení. Při hledání řešení pomocí tohoto algoritmu je snaha mít co možná největší „optimalizační okno“. Na druhou stranu počet všech možných pořadí dávek v „optimalizačním okně“ odpovídá permutaci a je určen vztahem $P(n) = n!$, čímž roste počet výpočtů modelu v rámci jednoho kroku algoritmu. Z těchto důvodů jsem jako nejvhodnější velikost optimalizačního okna určil $n = 3$.

Při každém testování, jestli daná „trojice“ dávek nezlepšuje dosud nejlepší nalezené řešení, ukládá algoritmus tuto „trojici“ do tabulky tzv. „tabu listu“. K této trojici ještě uloží informaci, na jaké pozici se tato trojice v rozvrhu nacházela. V každém dalším kroku algoritmu se před samotným výpočtem hodnot modelu provede s pomocí „tabu listu“ test, jestli již nebyla tato „trojice“ na dané pozici zkoušena. Algoritmus tak postupuje se svým „optimalizačním oknem“ v rozvrhu od první pozice až k nejvyšší možné. Pokud hodnota C_{max} nedosáhne hodnoty dolního omezení velikosti rozvrhu viz (2.24) a algoritmus se dostane na konec rozvrhu, začíná opět od první pozice. Pokud nedojde ke zlepšení rozvrhu v určitém počtu kroků (počet dávek v rozvrhu - délka optimalizačního okna + 1), znamená to, že jsme našli nejlepší rozvrh, který je tento algoritmus schopen najít, a proto je tato část algoritmu ukončena. Algoritmus tedy nenašel v okolí nejlepšího nalezeného řešení žádné lepší řešení a dosáhl tedy jakéhosi lokálního minima. V následující části bude algoritmus „zkoušet“, jestli nenajde v prostoru všech dosažitelných řešení jakékoli lepší řešení, než doposud nejlepší nalezené.

Princip hledání lepšího řešení v prostoru dosažitelných řešení je založen na prostém přemístění konkrétní dávky na všechny možné pozice v rozvrhu. Toto je provedeno pro každou dávku. Vždy, když je nalezen lepší rozvrh, tak je uložen jako nejlepší nalezené řešení. Pokud bylo nalezeno lepší řešení, je opět spuštěn algoritmus prohledávání okolí tohoto nejlepšího řešení popsaný v předchozím odstavci.

Vstupní hodnoty algoritmu jsou data o konfiguraci výrobní linky (v, n, b_g, m_i a r_{ig}) a výstupem je nejlepší nalezený rozvrh v podobě vektoru *schedule*. Výše popsaný optimalizační algoritmus je popsán následujícím pseudokódem:

```
LoBopt = Vypočti hodnotu "LB" dolního omezení "C_max";
načti data z modelu // "v", "n", "bg", "mi", "rig", "Kg"
schedule= [inicializační rozvrh]; //prosté seřazení dávek
[xijk,cik,dik] = VypoctiDataVyroby(schedul);
window=3; // velikost optimalizačního okna
terminator=0; // určuje, zda je optimalizační okno na konci rozvrhu
tabulist=[];
```

```
8       solve=0; // pokud není "trojice" dávek z optimalizačního okna v Tabu Listu tak "solve=0"
9       stopp = 0; // pokud už heuristika tabu search nenalezne lepší řešení "stopp = 1"
10       stop = 0; // počet kroků bez zlepšení rozvrhu v heuristice "tabu search"
11       end_tabu = 0; // pokud tabu search nenalezne po druhem spuštění žádné zlepšení, tak je konec celého
                 // algoritmu
12       interrup = 0; // přerušení celého algoritmu
13       next_tabusearch = 1; // pokud se podaří v 2. fázi prohledávání celého prostoru lepší řešení, je
                        // opět spuštěn tabu search a next_tabusearch = 1
14       next_k=[];

15       while next_tabusearch == 1
16           while 1
17              // postupuj s optimalizačním oknem od pozice jedna v rozvrhu výše
18               for start = 1:length(schedul)-window+1
19                   [xijk,cik,dik] = VypoctiDataVyroby(v,n,bg,mi,rig,schedul);
20                   if start == length(schedul)-window+1
21                       terminator=1; //pokud je optimalizační okno na konci rozvrhu
22                   end
23                   next_k = max(Kg(schedul(start+window-terminator),:)); // index prvního výrobku
                                                                  //následujícího za posledním
                                                                  // výrobkem v optimalizačním okně
24                   lastcikmin = cik(m,next_k); // hodnota "c_ik" výrobku "next_k" na posledním stroji
25                   lastcmax = max(max(dik)); // hodnota "C_max" testovaného rozvrhu
26                   schedul_win_list = "permutace dávek v oknu" //(všechny možnosti)
27                   schedul_temp=schedul;
28                   // vyzkouší všechny kombinace dávek v optimalizačním oknu
29                   for i = 1:size(schedul_win_list,1)
30                       schedul_temp(start:start+window-1)=schedul_win_list(i,:); // testovaný rozvrh
31                       id = najdi pozici, kde je daná "trojice" dávek uložená v "tabulistu"
32                       if id > 0
33                           if find(tabulist(id,window+1:end)==i) > 0 // pokud se "trojice" v tabulistu
                                                                  // nachází, zjistí jestli již byla
                                                                  // na této pozici v rozvrhu
34                               solve=0; // již bylo zkoušeno, model se nebude řešit
35                           else
36                               tabulist(id,end+1)=i; // ulož trojici do tabulistu
37                               solve=1; // vyřeš model
38                           end
39                       else
40                           // pokud trojice v tabulistu není, přidej ji na konec a vyřeš model
41                           tabulist(end+1,1:window)=schedul_win_list(i,:);
42                           tabulist(end,end+1)=i;
43                           solve=1; // vyřeš model
44                       end
45                       if solve ==1
46                           [xijk,cik,dik]= VypoctiDataVyroby(v,n,bg,mi,rig,schedul_temp);
47                           if i == 1 // zjisti index následujícího výrobku za dávkami v optimalizačním
                                 // okně (pouze jednou v každém kroku alg.)
48                               next_k = max(Kg(schedul_temp(start+window-terminator),:));
49                           end
50                           cmax = max(max(dik));
51                           nextcikmin = cik(m,next_k);
```

```
52                          if nextcikmin < lastcikmin && cmax < lastcmax
53                              schedul(start:start+window-1)=schedul_win_list(i,:); // použij toto řešení
                                                                                    // jako nejlepší nalezené
54                              stop = 0; // vynulování počtu kroků bez zlepšení
55                              // pokud je dosaženo dolního omezení "LB", algoritmus končí
56                              if(max(max(dik))/LoBopt*100-100 == 0)
57                                  stopp = 1; // konec tabusearch
58                                  interrup = 1; // konec celého algoritmu
59                                  break;
60                              end
61                              end_tabu=0;
62                              lastcikmin = nextcikmin;
63                              lastcmax = cmax;
64                          end
65                      end
66                  end
67                  stop = stop + 1;
68                  if stop == length(schedul)-window+2 || stopp == 1
69                      stop=0;
70                      stopp = 1;
71                      break;
72                  end
73              end
74              if stopp ==1
75                stopp = 0;
76                next_tabusearch = 0;
77                break;
78              end
79          end

80          // vyhledá jestli nejde jednotlivé dávky přemístit na jinou pozici
81          if end_tabu ~= 1
82              tabulist=[]; // vymazat Tabulist
83              schedul_best = schedul;
84          for i=1:length(schedul)
85              if interrup == 1
86                 break;
87              end
88              batch_temp = schedul(i); // vyberu dávku
89              batch_temp_pos = find(schedul_best == batch_temp);
          // vyjmu ji z rozvrhu
90              schedul_temp=[schedul_best(1:batch_temp_pos-1) schedul_best(batch_temp_pos+1:end)];
91              for j=1:length(schedul)
92                  // umísťuji dávku všechny možné pozice v rozvrhu
93                  schedul_temp2 = [schedul_temp(1:j-1) batch_temp schedul_temp(j:end)];
94                  [xijk,cik,dik]=VypoctiDataVyroby(v,n,bg,mi,rig,schedul_temp2);
95                  cmax_temp=max(max(dik));
96                  if cmax_temp < lastcmax
97                     // při zlepšení rozvrhu ho uložím jako nejlepší známý
98                     next_tabusearch = 1;
99                     schedul_best=schedul_temp2;
100                     lastcmax = cmax_temp;
101                  end
```

```
                    if(lastcmax/LoBopt*100-100) == 0
                        interrup = 1;
                        break;
                    end
                end
            end
            schedul=schedul_best;
            end_tabu =1;
            end
            if interrup == 1
               break;
            end
        end
```

4.5.2 Minimalizace kritéria „weighted tardiness“

Při minimalizaci kritéria „weighted tardiness“ je algoritmus tvorby MIP modelu a hledání řešení pomocí metaheuristiky „tabu search“ prakticky totožný jako v předchozí podkapitole. Déle musí být navíc načteny hodnoty w_k a d_k pro jednotlivé výrobky, kritérium je vypočteno podle rovnice (2.25) a podmínka, za které je daný rozvrh považován za lepší než nejlepší nalezený, je taková, že je akceptován jakýkoliv rozvrh s menší hodnotou WT než nejlepší doposud nalezený. Algoritmus by bylo možné ještě dále vylepšit o získání lepšího inicializačního rozvrhu, například využitím některého z řídících pravidel jako třeba „apparent tardiness cost (ATC)“ viz (Rudová, H., 2009). Aktuálně je využito pouze prosté seřazení dávek dle indexu identifikátoru. Pro zrychlení výpočtu v jazyce programu MATLAB jsem upravil výpočet kritéria v rovnici (2.25) tak, že jsem místo sumy uvažoval skalární součin vektorů a toto kritérium jsem vypočítával následovně:

```
weighTardiness = dot(wk(1:n),max(cik(m,1:n)-dk(1:n),0));
```

Kapitola 5

Systém pro rozvrhování a simulaci výroby

V této kapitole je navržen a popsán systém pro rozvrhování a simulaci výroby. Základním požadavkem byla možnost ovládání systému více uživateli najednou, dále aby bylo zadávání dat pro uživatele příjemné a aby nebyl uživatel nucen být školen na nový pro něj neznámý systém. K tomuto účelu byl vybrán program MS Office Excel 2007 z důvodu velké rozšířenosti v oblasti kancelářského softwaru. Tento tabulkový editor je využit jako jednoduché GUI pro zadávání vstupních dat simulace resp. rozvrhování. Při tvorbě systému jsem se zaměřil na rozvrhování dávkové výroby pro montážní linky s paralelními pracovními stanicemi (FFL) a omezenými buffery viz kapitola 2 a na simulaci případové studie výroby kol viz (Costel Alin Nicola, 2008).

5.1 Popis produktového řešení

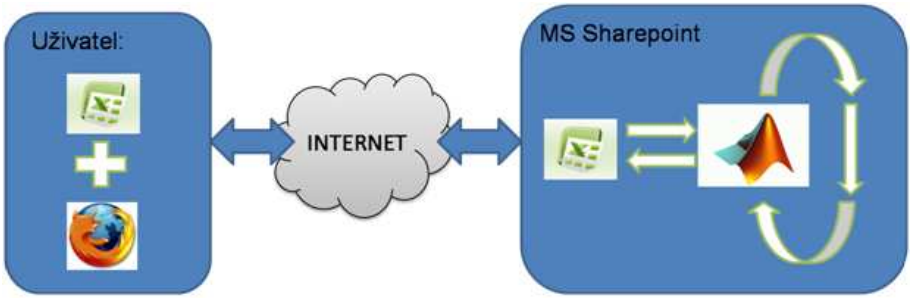
Výsledné řešení spočívá v tom, že se uživatel přihlásí ze svého klientského počítače přes internet na portálu, který je zajištěn pomocí MS SharePoint Services 3.0. Zde v knihovně dokumentů otevře tabulku programu MS Office Excel 2007, ve které vyplní informace o plánované výrobě. Dále pomocí jednoho tlačítka spustí rozvrhovač a po dokončení výpočtu je do jeho složky automaticky nahrán dokument s výsledky rozvrhování respektive simulace.

5.2 Struktura systému

Problém rozvrhování FFL linky s omezenými buffery byl řešen smíšeným celočíselným programováním v programu GLPK. Heuristické řešení (viz kapitola 4) bylo implementováno v prostředí Matlab. Vstupní data o modelu výroby a množství vyrobených výrobků jsou zadávána v aplikaci MS Office Excel 2007.

Tyto soubory programu MS Office Excel 2007 obsahující grafické rozhraní by mohl mít uživatel s potřebnými programy na svém lokálním počítači. Jelikož ale pořizovací náklady dílčích aplikací nemusí být zanedbatelné a zároveň jejich správné nastavení může být poměrně komplikované, je zřejmé, že nasazení tohoto systému na server přinese nejen časovou a finanční úsporu, ale i zvýší komfort ovládání systému a umožní pracovat se systémem více uživatelům najednou.

Jak již bylo zmíněno výše, pro zajištění webu, správy dokumentů, jejich verzí a uživatelských účtů byl využit systém MS SharePoint Services 3.0 běžící na platformě MS Windows Server 2003. V tomto systému je pro každého uživatele založen uživatelský účet, který se do něj přihlásí pomocí internetového prohlížeče. Aplikace MS Office Excel 2007 má v sobě integrovanou podporu přístupu a práce s dokumenty na serveru MS SharePoint. Po otevření dokumentu z webu se na klientský počítač nahraje lokální kopie souboru, kterou může uživatel editovat a změny uložit zpět na server. Pak už jen pomocí tlačítka na webových stránkách portálu spustí simulaci a vyčká na vytvoření nového dokumentu s výsledky. Ilustrace struktury systému viz obr. 5.1

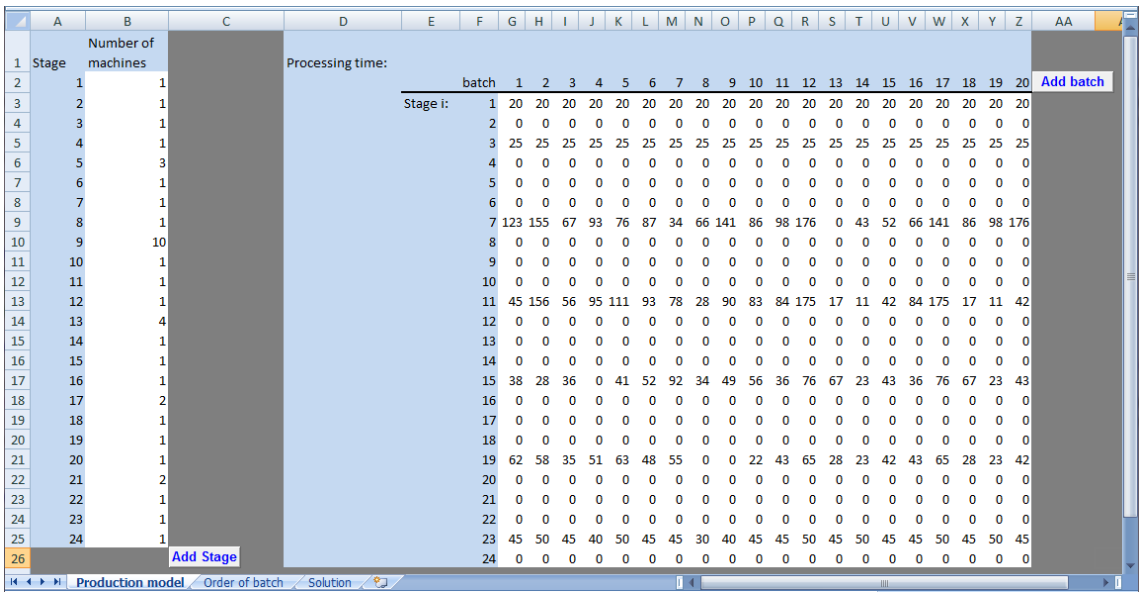


Obrázek 5.1: Struktura systému pro rozvrhování a simulaci

5.3 Použití systému

V této kapitole je popsána práce se systémem pro rozvrhování výroby. Po přihlášení uživatel poklepe na knihovnu dokumentů v pravém sloupci s názvem „Batch Scheduling“ pro rozvrhování dávkových procesů, nebo na „Bicycles“ pro simulaci výroby kol. Jelikož ovládání rozhraní v MS Excel pro výrobu kol je popsáno v (Costel Alin Nicola, 2008), budu se dále věnovat pouze rozhraní pro rozvrhování dávkových procesů. V knihovně „Batch Scheduling“ se nachází dokument aplikace MS Excel s názvem „batch“. Přihlášený uživatel si dokument zarezervuje (zabrání přístupu k dokumentu jiným uživatelům v průběhu simulace) a otevře pro editaci v aplikaci MS Excel.

Dokument obsahuje tři listy. V listu s názvem „Production model“ může uživatel měnit parametry výrobní linky viz obr. 5.2. V levé tabulce je udán počet paralelních strojů na dané pracovní stanici. V pravé tabulce je zadána „doba trvání“ (processing time) pro určitou dávku na určité pracovní stanici.



Stage	Number of machines
1	1
2	1
3	1
4	1
5	3
6	1
7	1
8	1
9	10
10	1
11	1
12	1
13	4
14	1
15	1
16	1
17	2
18	1
19	1
20	1
21	2
22	1
23	1
24	1

Add Stage

Processing time:

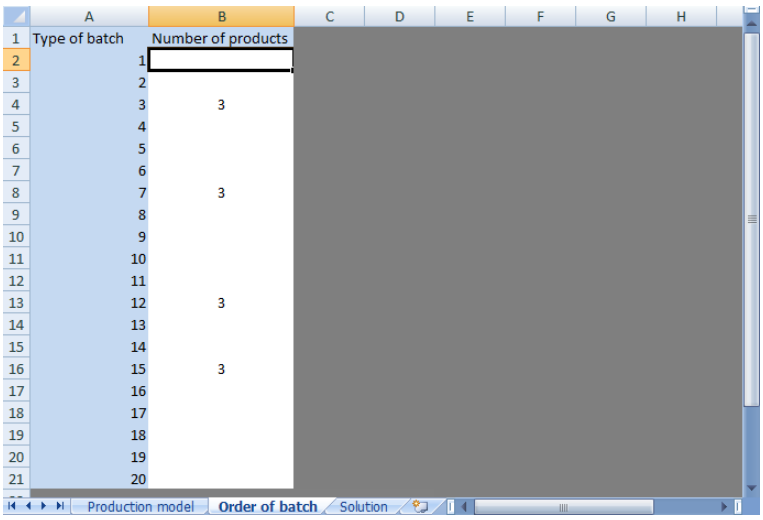
Stage i:	batch 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	123	155	67	93	76	87	34	66	141	86	98	176	0	43	52	66	141	86	98	176
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	45	156	56	95	111	93	78	28	90	83	84	175	17	11	42	84	175	17	11	42
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	38	28	36	0	41	52	92	34	49	56	36	76	67	23	43	36	76	67	23	43
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	62	58	35	51	63	48	55	0	0	22	43	65	28	23	42	43	65	28	23	42
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	45	50	45	40	50	45	45	30	40	45	45	50	45	50	45	45	50	45	50	45
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Add batch

Production model | Order of batch | Solution

Obrázek 5.2: Editace modelu výrobní linky

V listu „Order of batch“ lze editovat množství výrobků dané simulace viz obr. 5.3.

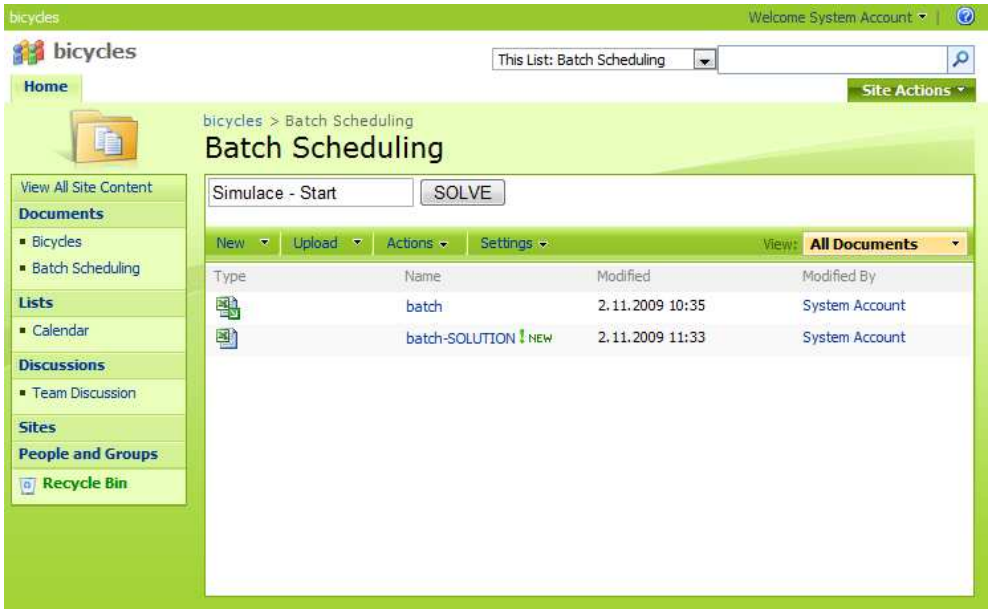


Type of batch	Number of products
1	
2	
3	3
4	
5	
6	
7	3
8	
9	
10	
11	
12	3
13	
14	
15	3
16	
17	
18	
19	
20	

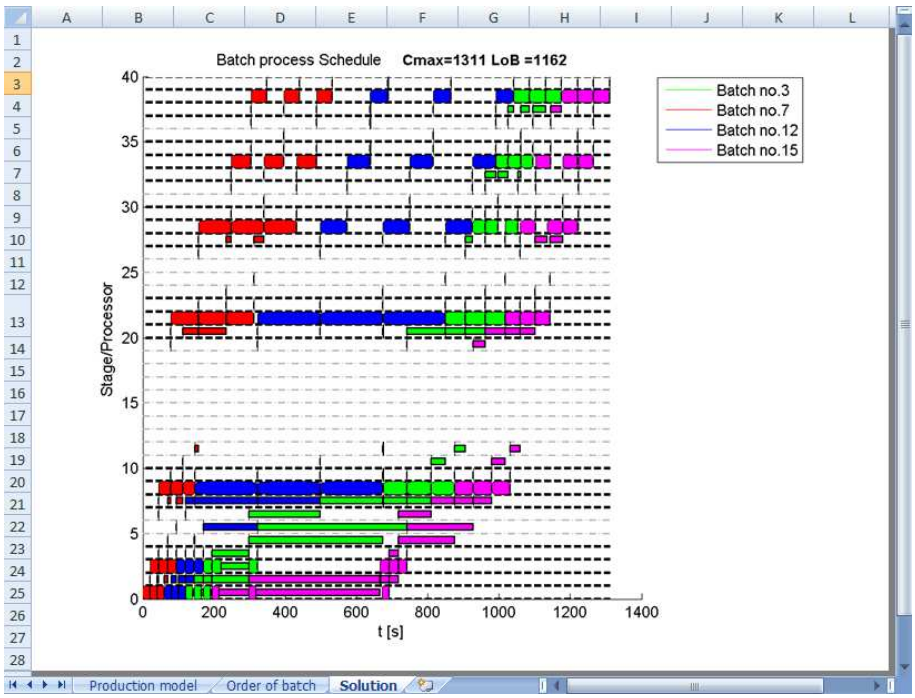
Production model | Order of batch | Solution

Obrázek 5.3: Editace počtu výrobků

Takto upravený dokument nahraje uživatel zpátky na server a v knihovně dokumentů spustí rozvrhovač pomocí tlačítka „SOLVE“. Po dokončení výpočtu je do knihovny automaticky přidán dokument viz obr. 5.4 s Ganttovým diagramem odpovídajícím zadaným požadavkům viz obr. 5.5.



Obrázek 5.4: Knihovna dokumentů s výsledky simulace



Obrázek 5.5: Výsledný dokument s Ganttovým diagramem

5.4 Popis instalace

V této kapitole jsou popsány nezbytné postupy, které jsem použil při tvorbě systému. Jedná se o:

- Vložení webové části (Web Part) respektive tlačítka, na webovou stránku v SharePointu
- Práce s MS Excel jako ActiveX objektem pomocí události tlačítka
- Předávání dat mezi MS Excel a Matlabem a spouštění výpočtů
- Zpětné uložení aktualizovaných dat na server SharePoint

5.4.1 Vložení webové části na webovou stránku v SharePointu

Uživatelské webové části je možno s určitými úpravami programovat pomocí MS Visual Studio 2008:

1. Do MS Visual Studia 2008 je nutno doinstalovat Rozšíření: „Visual Studio 2008 extensions for Windows SharePoint Services vision 1.2“, která přidává možnost tvorby, ladění a nasazení balíčků (předpřipravených projektů) do SharePointu. V našem případě budeme využívat přidání „Web Part“ na vytvořené stránky.

2. Po spuštění Visual Studia 2008, zvolíme: File → New Project, vybereme v levém sloupci SharePoint a zvolíme WebPart - Zadáme její jméno.
   Poznámka: V této fázi nastává problém, protože při založení nového projektu uloží Visual Studio jeho jméno interně na „WebPart1“. Při pokusu o vytvoření dalších webových částí SharePoint hlásí chybu, že WebPart s tímto jménem již existuje. Nejjednodušší způsob odstranění tohoto problému je:

   a) V Solution Exploreru VisualStudia smazat složku s názvem „WebPart1“

   b) Kliknout pravým tlačítkem na projekt a zvolit: Add → New Item

   c) V novém okně vybrat v levém sloupci „SharePoint“ a vytvořit novou webovou část se jménem odpovídajícím jménu celého projektu.

3. Příklad kódu v C# pro přidání tlačítka, textového popisu a obsluhu jejich událostí, je ukázán na následujícím kódu:

```
using System;
using System.Runtime.InteropServices;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Serialization;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using Microsoft.SharePoint.WebPartPages;
using System.Web.UI.HtmlControls;

namespace Tlacitko1
{
    [Guid("cfe99c20-614e-4656-ae68-176b25c7a318")]
    public class Tlacitko1 : System.Web.UI.WebControls.WebParts.WebPart
    {

        HtmlButton oButton;
        HtmlInputText oTextBox;
```

```
        public void oButton_click(object sender, EventArgs e)
        {
                        // Obsluha udalosti Tlacitka
        }

        protected override void CreateChildControls()
        {
            oButton = new HtmlButton();
            oButton.InnerText = "Text Tlacitka";
            oButton.ServerClick += new EventHandler(oButton_click);
            Controls.Add(oButton);

            oTextBox = new HtmlInputText();
            oTextBox.Value = "Informacni text";
            Controls.Add(oTextBox);
        }

        protected override void Render(System.Web.UI.HtmlTextWriter output)
        {
            oTextBox.RenderControl(output);
            oButton.RenderControl(output);
            output.WriteLine("<BR>");
        }
    }
}
```

4. Dále je nutné nastavit adresu serveru, na který se bude nahrávat WebPart. V Solution Exploreru tedy klepnout pravým tlačítkem na název projektu, zvolit „Properties“ a v levém sloupci zvolit „Debug“. Do pole „Start browser with URL:“ vyplnit adresu:
   `http://localhost:<PORT SHAREPOINT PORTALU>/`

5. Pro umístění vytvořené webové části na server je nutno v hlavní liště z nabídky „Build“ zvolit „Deploy Solution“.

6. Pokud vše proběhne v pořádku a projekt je umístěn na server, zobrazíme si v internetovém prohlížeči portál, který jsme si vytvořili. Po přihlášení jako uživatel s administrátorskými právy se přepneme na stránku, na kterou chceme vložit webovou část a v pravém horním rohu zvolíme „Site Actions“ → „Edit Page“

7. Na stránce se nám zobrazila místa s textem: „Add a Web Part“. Poklepeme na tento text a ve vyskakovacím okně klepneme na „Advanced Web Part gallery and options“

8. Nejjednodušším způsobem jak najít naši Web Part je v pravém sloupci klepnout na text „Browse“, vybrat „Search“ a vyhledat název našeho projektu s webovou částí a pomocí tlačítka Add ho vložit.

5.4.2 Práce s MS Excel v SharePoint z webové části

Tato kapitola popisuje práci s MS Excel jako ActiveX objektem z webové části a zpětné uložení aktualizovaných dat na server SharePoint. Budu vycházet ze založeného projektu webové části a webového portálu pro zprávu dokumentů viz výše. Jinou možností jak programově přistupovat k datům v souborech typu MS Excel je využití služby Excel Services, která mimo jiné funguje jako jakési API. S takovým souborem lze manipulovat v .NET jako s objektem. V rámci tohoto objektu je dostupná většina matematických funkcí stejně jako v MS Excel. Rovněž lze přistupovat k jednotlivým listům a buňkám. Nicméně služba Excel Services byla navržena pouze pro distribuci dat souborů MS Excel ze SharePoint serveru, proto nelze do těchto souborů zapisovat a ukládat je.

1. K dokumentům na serveru MS SharePoint lze přistupovat pomocí adresy `http://<ADRESA_SERVERU>:<PORT>/<CESTA_V_MS_SP>/<jmeno_souboru>`. Nicméně upload souborů na server není žádnou zvláštní službou defaultně podporován. Tato služba je ale volně dostupná a lze ji do SharePoint serveru doinstalovat. Na adrese `www.codeplex.com` ji lze najít pod názvem: „WSUploadService - Web Service for uploading documents into SharePoint“. Tato služba musí být do projektu v MS Visual Studiu přidána jako webová reference. Blíže k přidávání referencí viz bod 2.

2. Do projektu v MS Visual Studiu založeného v podkapitole 5.4 je potřeba následovně zadat příslušné reference kvůli ukládání dat na server a manipulaci s MS Excelem:

   - V Solution Exploreru klepnout pravým tlačítkem na název projektu, zvolit „Add Reference“ a z karty COM přidat referenci „Microsoft Excel 12.0 Object Library“
   - Přidat webovou referenci:
     - Klepnout pravým tlačítkem v Solution Exploreru na název projektu, zvolit „Add Service Reference“ (pro MS VS2005 jen „Add Web Reference“). Poklepat na tlačítko „Advanced...“ a v dalším okně poklepat na„Add Web Reference“

  - Jako URL webové reference zvolit:
    `http://localhost:<PORT_PORTALU>/_vti_bin/Files.asmx?wsdl`
  - Přidat referenci pod názvem „localhost“

3. Následující příklad funkce obsluhy události zobrazuje jak:

   - Stáhnout dokument z SharePoint serveru z umístění `http://g204-scheduling:18123/Documents/Tables4.xls` na místní disk serveru jako `D:\SP_bikes\VYSLEDKY.xls`
   - Spustit Makro, které je součástí dokumentu MS Excel a jmenuje se „project“
   - Uploadovat aktualizovaný dokument na SharePoint server do: `http://g204-scheduling:18321/Documents`

```
public void oButton_click(object sender, EventArgs e)
        {
            // download souboru - prepisuje
            oTextBox.Value = "Simulace - Start";
            WebClient client = new WebClient();
            client.Credentials = CredentialCache.DefaultCredentials;
            string uri = "http://g204-scheduling:18100/Bicycles/Tables3.xls";
            string fileName = "D:\\SP_bikes\\VYSLEDKY5.xls";
            try
            {
                client.DownloadFile(uri, fileName);
            }
            catch (WebException ex)
            {
                Console.WriteLine("Pri stahovani souboru doslo k vyjimce : {0}",
                    ex.ToString());
            }
```

```
    // Provedení Makra v Excelu
     ApplicationClass app = new ApplicationClass();
     Workbook workBook = app.Workbooks.Open(fileName, 0, false, 5, "", "",
         true, XlPlatform.xlWindows, "\t", false, false, 0, true, 1, 0);
     Worksheet workSheet = (Worksheet)workBook.Sheets[1];
     app.Application.Run("project", Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value);
     workBook.SaveAs("D:\\SP_bikes\\VYSLEDKY_HOTOV5.xls", Missing.Value,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlNoChange,
          Missing.Value, Missing.Value, Missing.Value, Missing.Value,
          Missing.Value);
     workBook.Close(false, fileName, null);
     app.Quit();
     // upload souboru na server - neprepisuje
     string strPath = @"D:\SP_bikes\VYSLEDKY_HOTOV5.xls";
     try
     {
         localhost.Files oUploader = new localhost.Files();
         oUploader.PreAuthenticate = true;
         oUploader.Credentials = CredentialCache.DefaultCredentials;
         string strFile = strPath.Substring(strPath.LastIndexOf("\\") + 1);
         string strDestination = "http://g204-scheduling:18100/Bicycles";
         FileStream fStream = new FileStream(strPath, System.IO.FileMode.Open);
         byte[] binFile = new byte[(int)fStream.Length];
         fStream.Read(binFile, 0, (int)fStream.Length);
         fStream.Close();
         //string str =
         oUploader.UploadDocument(strFile, binFile, strDestination);
         //System.Console.Write(str);
         oTextBox.Value = "Simulace - OK";
     }
     catch (Exception ex)
     {
         oTextBox.Value = "Soubor nelze nahrat";
         System.Console.Write(ex.Source + " - " + ex.Message +
             " - " + ex.InnerException + " - " + ex.StackTrace);
     }
     //Mazani Zaloznich souboru:
     File.Delete(strPath);
     File.Delete(fileName);
}
```

5.5 Výhody, nevýhody, problémy

Navržený systém poskytuje řadu výhod. Hlavní z nich je možnost ovládání aplikace více uživateli najednou přes webové rozhraní. Tento systém je zajištěn službou Windows SharePoint Services 3.0., která je poskytována zdarma s licencí na Microsoft Windows Server 2003. Celé produktové řešení je umístěno na jednom serveru společně s potřebnými výpočetními aplikacemi. Odpadá tedy nutnost instalace produktu u každého uživatele zvlášť. To přináší finanční úsporu za pouze jednu licenci např. pro MATLAB na server. Jako nástroj realizace grafického rozhraní pro zadávání parametrů výroby byl zvolen MS Excel. Zde je výhodou všeobecná znalost tohoto programu mezi uživateli. Data získaná ze souborů MS Excel jsou předávána programu MATLAB, který je využit pro implementaci algoritmů rozvrhování a simulace výroby a poskytuje celou řadu funkcí a toolboxů. Další výhodou použití MATLABu je možnost spolupráce s rozličnými externími výpočetním prostředky, v našem případě programem GLPK pro řešení úloh lineárního programování a simulátorem Petriho sítí.

Při porovnání Windows SharePoint Services 3.0. a Google Docs je možno spatřovat několik rozdílů. WSS 3.0 jsou jako služba MS Windows Server jeho součástí a potenciálně citlivá data o výrobě se nedostávají „ven“ mimo okruh pověřených uživatelů a případná firma si může svůj server spravovat sama. Při využití Google Docs a ukládání dat na server společnosti Google musí být bráno v potaz, že se tyto informace dostávají do rukou třetí strany. WSS 3.0 oproti Google Docs poskytují mnoho funkcí a možností editace a úprav, které ale nejsou ve velké míře využity a zmíněná jednoduchost Google Docs může být výhodou.

Systém postavený na vzájemné spolupráci více nezávislých aplikací na druhou stranu zvyšuje jeho složitost. Dále lze mezi problémy a nedostatky jmenovat to, že podpora Windows SharePoint Services 3.0. není standardně v MS Visual Studiu 2008 nainstalována a vytváření webových částí nefunguje správně viz výše. Nicméně tento nedostatek by měl být odstraněn s novou verzí MS Visual Studia 2010, kde je již podpora Windows SharePoint standardně integrována.

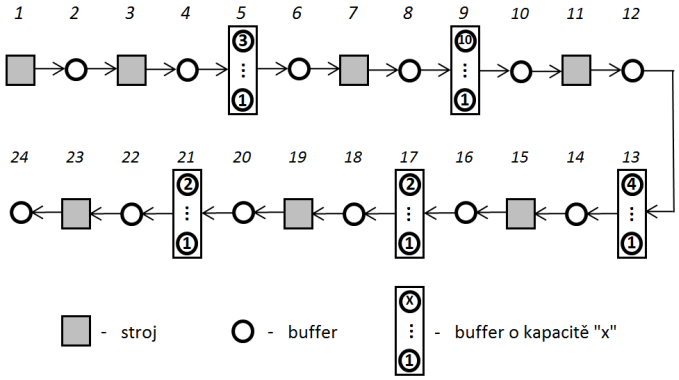
Kapitola 6

Experimentální výsledky

V této kapitole jsou prezentovány experimentální výsledky algoritmů popsaných v kapitole 4. Měření bylo prováděno na modelu z reálné výroby viz (Sawik, T. et al., 2002). Jednalo se o výrobní linku SMT (čili surface mount technology). Výpočty byly prováděny na notebooku HP Compaq 6510b s 64-bitovým procesorem Intel Core2 duo T9300 2x2.50 GHz s využitím matematického modelovacího jazyka AMPL (AMPL, 2004) a výpočetním programem GLPK v.4.38 (viz podkapitola 3.2) pro řešení MIP modelu. Pro implementaci algoritmů byl využit MATLAB R2009a 64-bit.

6.1 Popis instance výroby pro experimenty

Konfigurace výrobní linky SMT, která byla použita pro experimenty, je zobrazena na obr. 6.1.



Obrázek 6.1: Výrobní linka se stroji oddělenými buffery o určité kapacitě

Linka se skládá z nakladače, potisku, čtyř umisťovacích strojů a vizuální kontroly v sérii za sebou, oddělených buffery s omezenou kapacitou.

Tabulka 6.1: Doba trvání v sekundách

Druh	Pracovní stanice						
výrobku	1	3	7	11	15	19	23
1	20	25	123	45	38	62	45
2	20	25	155	156	28	58	50
3	20	25	67	56	36	35	45
4	20	25	93	95	0	51	40
5	20	25	76	111	41	63	50
6	20	25	87	93	52	48	45
7	20	25	34	78	92	55	45
8	20	25	66	28	34	0	30
9	20	25	141	90	49	0	40
10	20	25	86	83	56	22	45
11	20	25	98	84	36	43	45
12	20	25	176	175	76	65	50
13	20	25	0	17	67	28	45
14	20	25	43	11	23	23	50
15	20	25	52	42	43	42	45
16	20	25	66	84	36	43	45
17	20	25	141	175	76	65	50
18	20	25	86	17	67	28	45
19	20	25	98	11	23	23	50
20	20	25	176	42	43	42	45

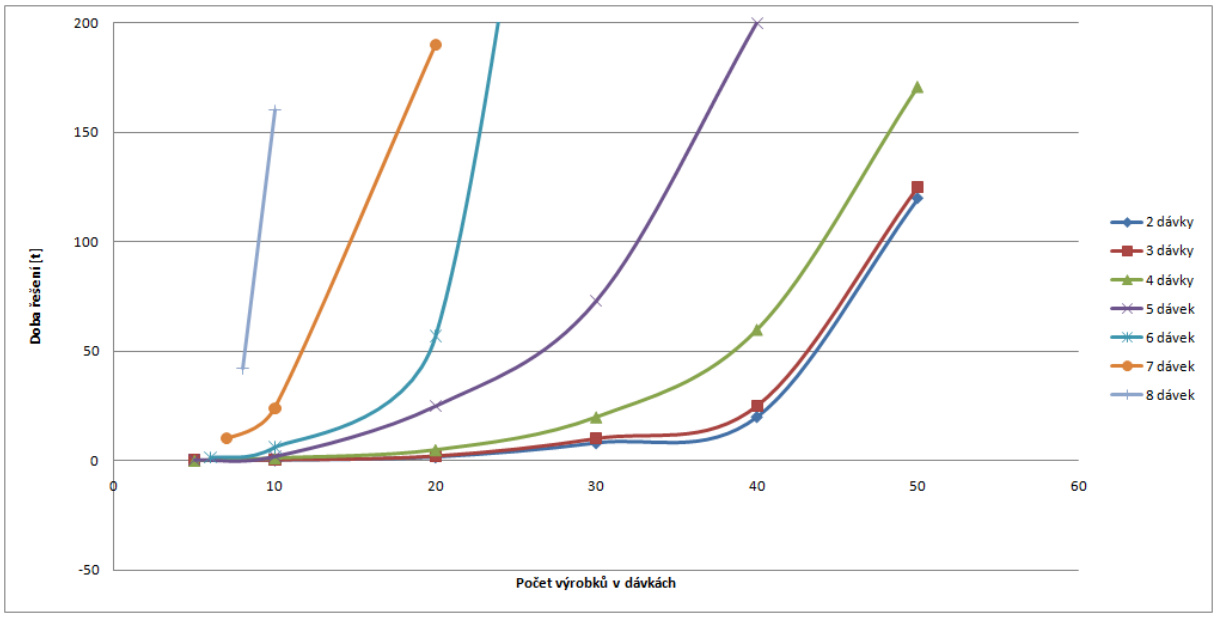
Doby trvání pro všechny druhy výrobků na všech pracovních stanicích jsou zobrazeny viz tabulka 6.1 a spolu s konfigurací linky vycházejí z (Sawik, T. et al., 2002). Buffery jsou sice ve výrobním modelu brány jako pracovní stanice, ale s nulovou dobou trvání a proto nejsou v tabulce zapsány.

6.2 Experimenty s minimalizací kritéria C_{max}

V této podkapitole je jako hodnota kritéria uvažována hodnota C_{max}. Při hodnocení o kolik je heuristicky nalezený rozvrh horší, než rozvrh optimální, jsem využil rychlého způsobu výpočtu dolní hranice hodnoty C_{max} viz rovnice (2.24). Nejedná se sice přímo o hodnotu C_{max} optimálního rozvrhu, nicméně ve většině případů jsou tyto hodnoty totožné, případně velmi blízké.

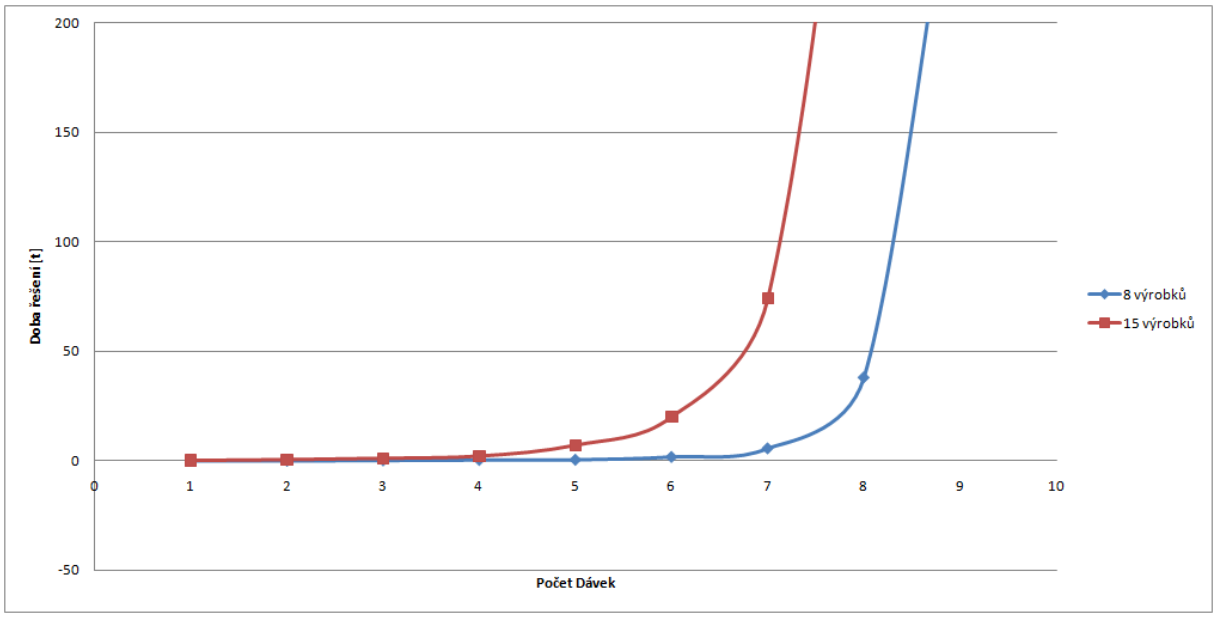
6.2.1 Experiment 1

Tento experiment popisuje rychlost hledání optimálního rozvrhu pouze pomocí MIP a s využitím matematického modelu z podkapitoly 2.2.3. Na obrázku obr. 6.2 je zobrazena závislost doby řešení na celkovém počtu výrobků ve všech dávkách. Toto je vyobrazeno pro různé počty dávek. Z obrázku je zřejmé, že dobu výpočtu neovlivňuje pouze celkový počet výrobků v produkčním plánu, ale také do kolika dávek se tyto výrobky dělí. Tato metoda je použitelná, pokud je počet dávek do pěti a celkový počet výrobků do čtyřiceti.

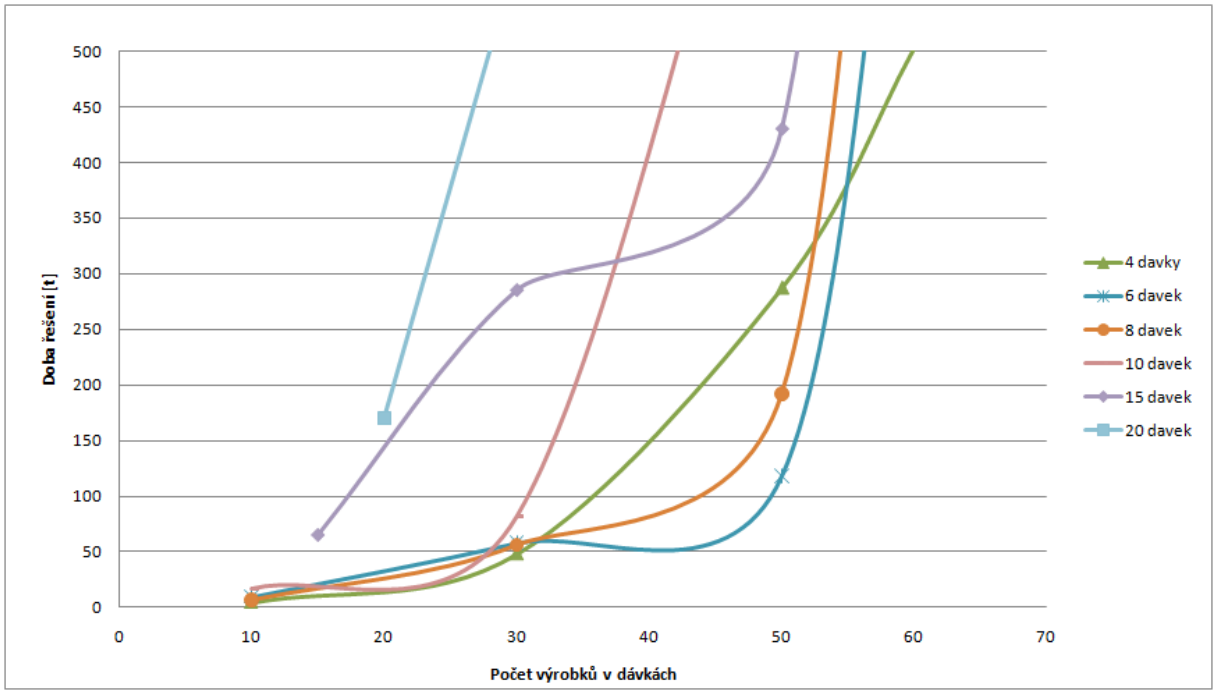


Obrázek 6.2: Doba řešení MIP

Na obr. 6.3 je znázorněna závislost doby výpočtu na počtu dávek pro různé počty výrobků.



Obrázek 6.3: Doba řešení MIP

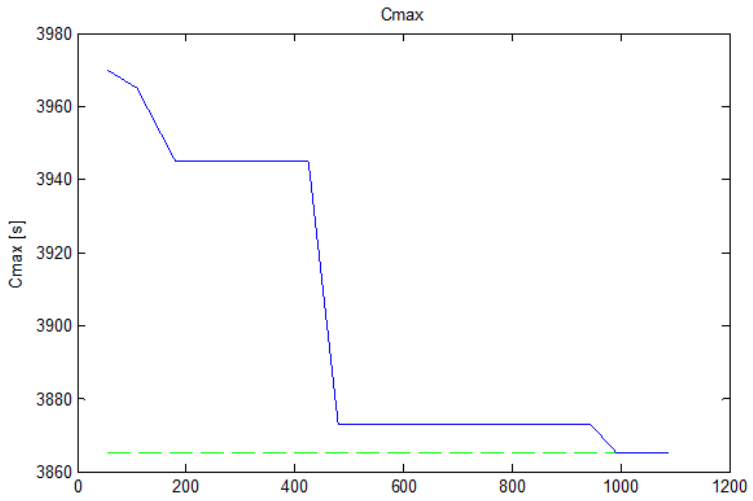


Obrázek 6.4: Časová náročnost výpočtů při použití fixace y_{fg}

6.2.2 Experiment 2

Zde je popsána rychlost řešení problému při použití heuristiky „fixace y_{fg}“ viz podkapitola 4.1, jejíž výsledky jsou zobrazené na obr. 6.4 Heuristika zlepšuje dobu výpočtu hlavně s ohledem na počet dávek, do kterých jsou výrobky rozděleny. Tato metoda došla ve většině případů k řešení mající hodnotu C_{max} rovnou hodnotě dolního omezení délky rozvrhu LB viz rovnice (2.24). Tato metoda je použitelná pro úlohy do padesáti výrobků rozdělených až do patnácti dávek.

Postupné snižování hodnoty C_{max} v každém iteračním kroku heuristiky je zobrazeno v obr. 6.5. Jednalo se o výrobu obsahující 40 výrobků rozdělených do 20-ti dávek.

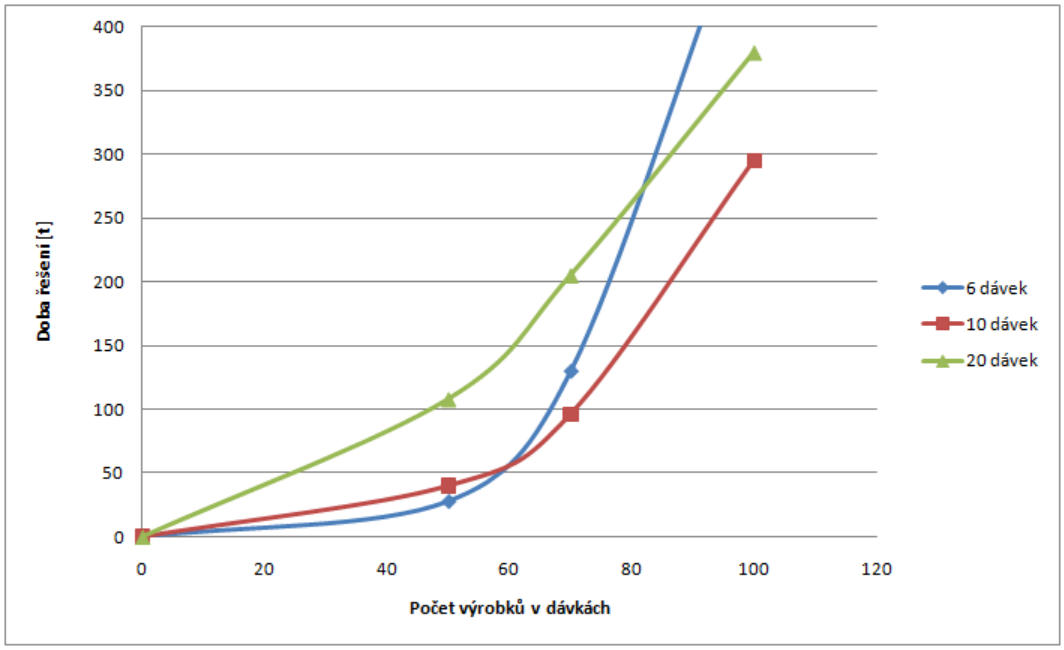


Obrázek 6.5: Změna hodnoty C_{max} v iteračních krocích algoritmu y_{fg} (40 výrobků ve 20-ti dávkách)

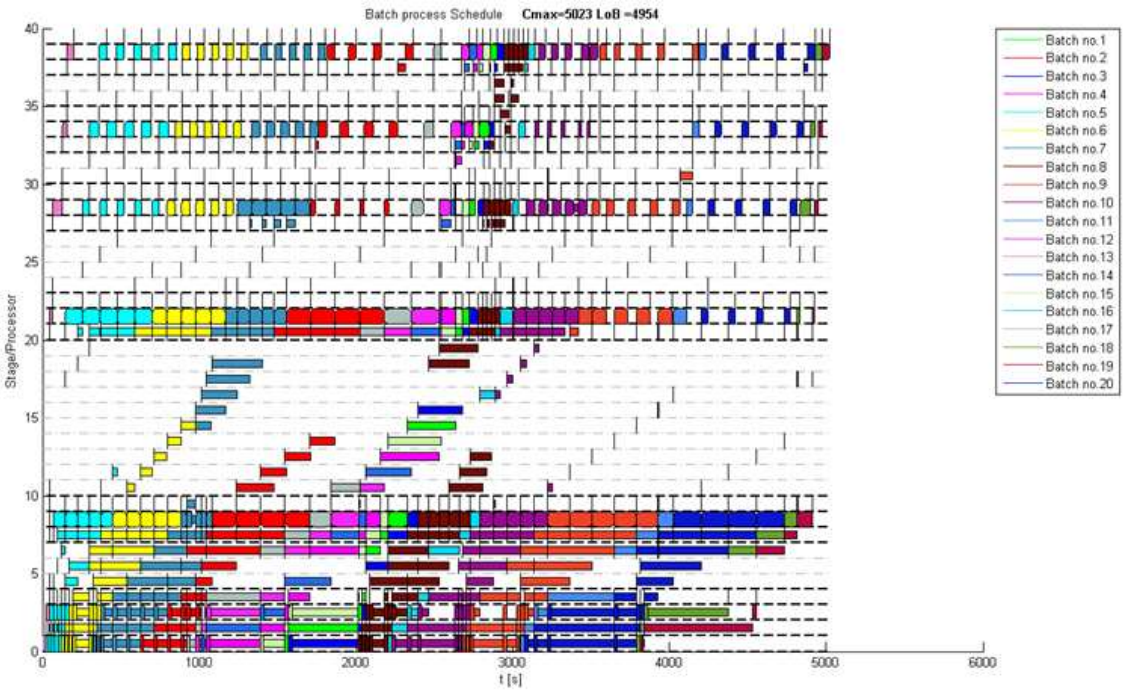
6.2.3 Experiment 3

Heuristika v tomto experimentu kombinuje algoritmy použité v experimentech 1 a 2 a je popsaná v podkapitole 4.4. Tato metoda ještě více snižuje závislost doby výpočtu na počtu dávek. Dále nárůst této doby v závislosti na celkovém počtu výrobků není tak strmý jako v předchozích případech (viz obr. 6.6). Tato metoda ovšem nedosahuje čistě optimální řešení. To, jestli bude řešení optimální, nebo zda bude odchylka od C^*_{max} malá (do 5-ti procent) je závislé na poměru velikosti optimalizačního okna (počtu dávek, které se v daném iteračním kroku řeší) ku celkovému počtu dávek. Z pohledu časové náročnosti a přesnosti jsem určil jako nejvhodnější velikost optimalizačního okna rovnou

třem. Heuristika je vhodná pro úlohy obsahující do sta výrobků rozdělených až do dvaceti dávek. Od dvaceti dávek se již odchylka od C^*_{max} pohybuje kolem 15%.



Obrázek 6.6: Časová náročnost výpočtů při použití výsledné heuristiky

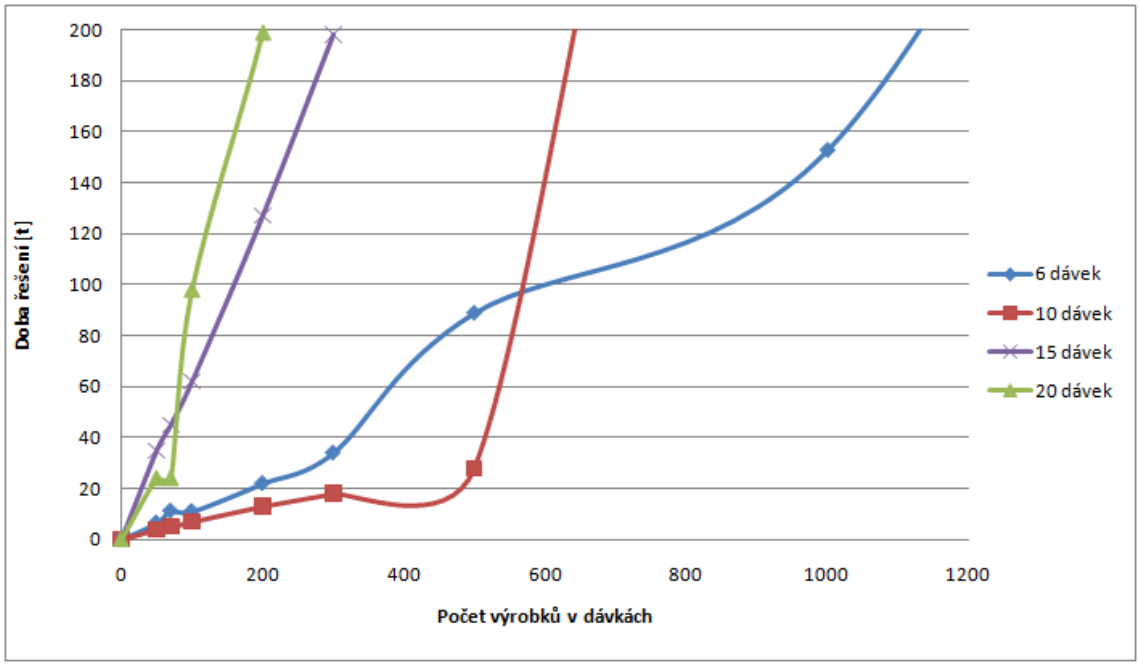


Obrázek 6.7: Ganttův diagram pro 51 výrobků rozdělených do 20-ti dávek

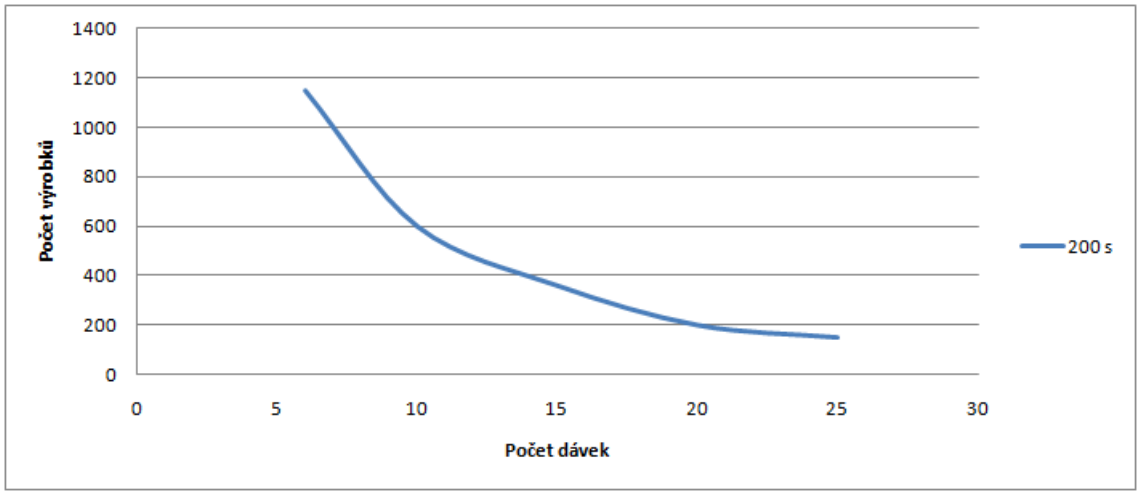
Na obr. 6.7 je příklad výsledného Ganttova diagramu zobrazujícího úlohu s 51 výrobky rozdělenými do 20-ti dávek. Výsledná hodnota $C_{max} = 5023$ a dolní hranice délky rozvrhu $LB = 4954$.

6.2.4 Experiment 4

Heuristika popsaná v kapitole 4.5, využívající metaheuristiky „tabu search“ a dále vlastního matematického modelu výroby získaného bez použití MIP, vykazuje řádově vyšší výkony než heuristiky navržené s využitím MIP. Tato heuristika nezaručuje nalezení optimálního řešení. Odchylky od dolního omezení délky rozvrhu LB, pro všechna měření zobrazená v grafu obr. 6.8, byly maximálně do 1%. Z grafu je patrné, jak je délka hledání řešení závislá na počtu dávek. Tato závislost je zobrazená v grafu obr. 6.9.



Obrázek 6.8: Časová náročnost výpočtů při použití ”tabu search”



Obrázek 6.9: Počet výrobků rozvrhnutelných do 200s v závislosti na počtu dávek

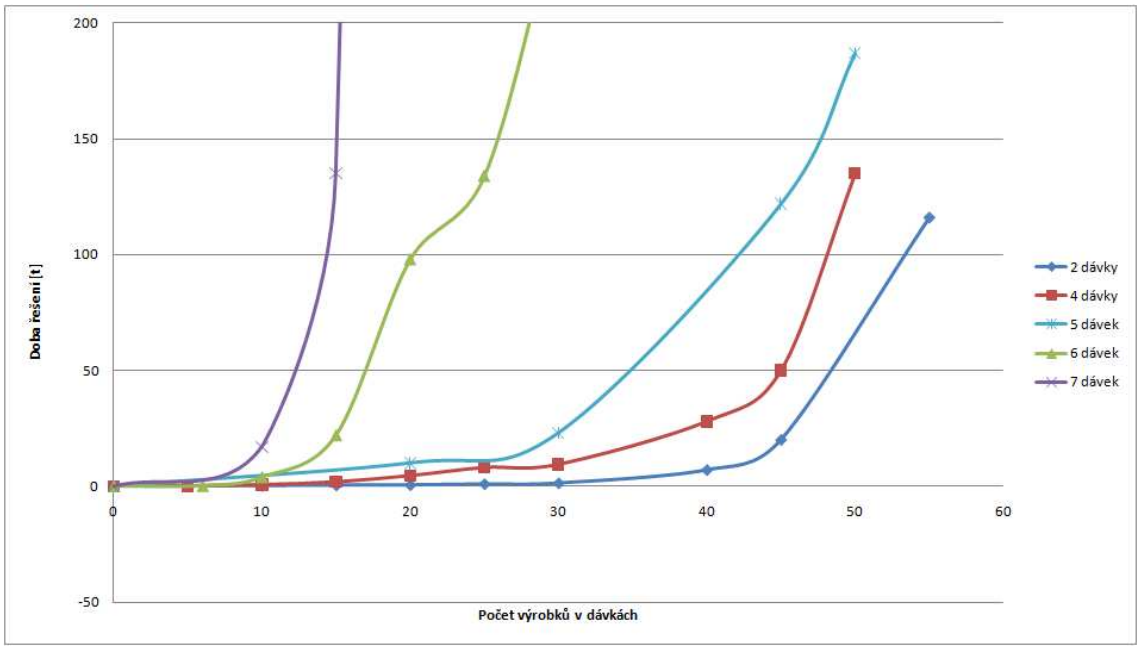
6.3 Experimenty s minimalizací kritéria *weighted tardiness*

V této kapitole jsou prezentovány výsledky algoritmů minimalizující kritérium *weighted tardiness*. Nejprve je prezentována výkonnost řešení problému pomocí MIP a dále pomocí metaheuristiky „tabu search“. Ve srovnání s kritériem C_{max}, kde bylo možné vypočítat dolní omezení této hodnoty, nebyl v tomto případě žádný takový vzorec k dispozici. Proto také nebylo možné vyčíslit procentuální odchylku výsledku od takového dolního omezení. Nicméně při porovnání výsledků pro instance, které byly řešitelné pomocí MIP, dávala heuristika stejné (optimální) výsledky. Pro větší instance jsem jako test použil postup, kdy jsem inicializační rozvrh seřadil v jednom pořadí a hodnoty *due date* pro jednotlivé výrobky seřadil v pořadí opačném. Poslední výrobek v inicializačním rozvrhu tedy měl hodnotu *due date* rovnou nule a první měl tuto hodnotu rovnou hodnotě dolního omezení C_{max}. Hodnoty *due date* pro ostatní výrobky jsem z těchto hodnot interpoloval. Z jednoduché úvahy vyplývá, že dávka zařazená v inicializačním rozvrhu na posledním místě by měla po provedení algoritmu být na prvním místě a naopak. Tímto způsobem se heuristika také chovala. To samozřejmě nemusí platit vždy a rozvrh může být ovlivňován velikostí dávek a přiřazenými prioritami dávek. Při kontrole rozvrhů velkých instancí to

byl však jediný způsob jak zjistit kvalitu výsledného řešení.

6.3.1 Experiment 5

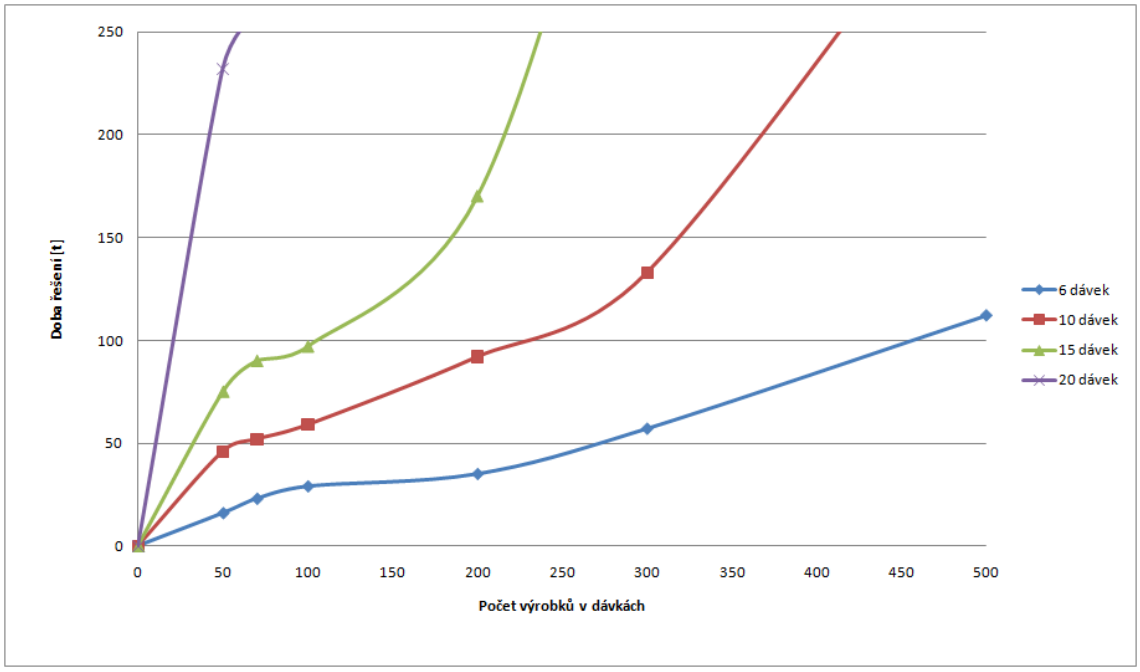
Na obrázku obr. 6.10 je zobrazena výkonnost řešení problému čistě pomocí MIP. Rychlost nalezení řešení je podobná jako při použití kritéria C_{max}. Je vhodná pro instance obsahujících do 50-ti výrobků v 5-ti dávkách.



Obrázek 6.10: Doba řešení MIP modelu při použití kritéria „weighted tardiness“

6.3.2 Experiment 6

Na obrázku obr. 6.11 je zobrazena výkonnost metaheuristiky „tabu search“. Heuristika zvládá řešit poměrně velké instance i když ve srovnání s výkonností heuristiky minimalizující kritérium C_{max} je doba hledání delší. To je způsobeno tím, že výpočet hodnoty kritéria *weighted tardiness* v každém kroku algoritmu je časově náročnější než pouhé určením maximální hodnoty C_{max}.



Obrázek 6.11: Časová náročnost výpočtů při použití heuristiky „tabu search“ a kritéria „weighted tardiness“

Kapitola 7

Závěr

Tato diplomová práce se zabývá vytvořením systému pro rozvrhování výroby. Cílem bylo vytvořit systém využívající obecně známé výpočetní prostředky, a tím snížit náročnost přechodu na nový systém pro uživatele. Systém byl umístěn na server, takže uživatel potřeboval k ovládání systému pouze internetový prohlížeč a tabulkový editor. Všechny ostatní potřebné aplikace a s tím spojené komplikace, jako například správné nastavení a propojení všech výpočetních programů, jsou již řešeny pouze na straně serveru, s čímž je spojené i snížení nákladů za licence k těmto produktům.

Pro vytvoření uživatelského rozhraní systému byl použit MS Excel. Hromadná správa dokumentů více uživateli najednou a vytvoření internetového portálu bylo realizováno pomocí technologie MS SharePoint. Nutná rozšíření funkcionality produktu MS SharePoint byla doprogramována pomocí technologie .NET a jazyka C#. Pro implementaci algoritmů výroby a pro generování Ganttova diagramu byl využit MATLAB. Problém celočíselného lineárního programování byl řešen s pomocí programu GLPK. Systém byl navržen pro řešení problému rozvrhování dávkové výroby na výrobních linkách s paralelními pracovními stanicemi (flexible flow lines) obsahující buffery s omezenou kapacitou.

V první fázi práce byly popsány možnosti využití systému MS SharePoint a bylo provedeno porovnání s možnostmi webové aplikace Google Docs. Dále jsou popsány vlastnosti komerčního produktu pro rozvrhování výroby *Asprova*.

V dalším kroku byly popsány a zdokumentovány možnosti využití Petriho sítí (PN) pro účely rozvrhování. PN jsou vhodným nástrojem pro modelování a verifikaci modelů výroby. V úloze rozvrhování mohou poskytnout užitečnou podporu. Například pomocí analýzy *P/T - invariant* lze určit chyby v návrhu modelu výroby, co se týče redundantích relací následností. Dále lze určit jak rozdělit zadaný rozvrhovací problém na více podproblémů bez vlivu na kvalitu výsledného rozvrhu. Pro samotné úlohy rozvrhování

lze různými metodami získávat rozvrhy, které ovšem odpovídají rozvrhům získaným za použití jednodušších řídicích pravidel (LPT, SPT, EDD atd. viz kapitola 3.3.5). PN v tomto případě nejsou příliš vhodné k implementaci složitějších heuristických metod.

Nakonec byl implementován MIP model dávkové výroby na výrobní lince typu FFL s buffery o omezené kapacitě, který byl oproti stávajícímu kritériu C_{max} rozšířen o kritérium *weighted tardiness*. Pro takto definovaný model byly navrženy heuristiky urychlující hledání optimálního rozvrhu. Takto navržené řešení ovšem stále trpělo velkou časovou náročností, a proto byl vytvořen algoritmus pro heuristickou tvorbu matematického modelu výroby bez využití MIP. Na základě tohoto algoritmu byla implementována metaheuristika *Tabu search* pro hledání optimálního rozvrhu. Tato heuristika již dosahovala dobrých výsledků, odpovídajícím reálným instancím výroby.

Výsledný systém používá jako rozhraní aplikaci MS Excel. Výpočty a algoritmy jsou však implementovány v programu Matlab, který nabízí lepší možnosti implementace algoritmů, využití vlastních toolboxů a spolupráce s ostatními výpočetními programy. Výsledný systém je tedy snadno rozšiřitelný o další optimalizační úlohy, které mohou být založené na rozličných přístupech od heuristických metod, přes celočíselné programování, až k modelování a verifikaci systémů založených na Petriho sítích.

Literatura

Brucker, P.; (2007), *Scheduling Algorithms*: Springer

Cheng-Shuo Wang, Reha Uzsoy; (2000), *A genetic algorithm to minimize maximum lateness on a batch processing machine*: Computers and Operations Research archive, Volume 29, Issue 12, Pages 1621-1640

Schwindt, Ch., Trautmann, N.; (2001), *Batch scheduling in process industries: an application of resource-constrained project scheduling*: OR Spectrum, Volume 22, Issue 4, Pages 501-524

Jianbin, D.; (2003), *Batch scheduling of Two-machine Limited/buffer Flowshop with Setup and Removal Times*: Georgia Institute of Technology.

Sawik, T.; (2002), *An Exact Approach for Batch Scheduling in Flexible Flow Lines with Limited Intermediate Buffers* : Mathematical and Computer Modelling, Volume 36, Issues 4-5, September 2002, Pages 461-471

Sawik, T., Schaller, A., Tirpak, T.M.; (2002), *SCHEDULING OF PRINTED WIRING BOARD ASSEMBLY IN SURFACE MOUNT TECHNOLOGY LINES*: Journal of Electronics Manufacturing, Volume: 11, Issue 1, pages 1-17

McCormick, S.T., Pinedo, M.L.; (1989), *Sequencing in an assembly line with blocking to minimize cycle time*: Operations Research archive, Volume 37 , Issue 6, Pages: 925 - 935

O'Connor, E.; (2008), *Mistrovství ve Windows Sharepoint Services 3.0*: Computer Press, a.s.

Zurawski , R., MengChu Zhou; (1994), *Petri Nets and Industrial Applications: A Tutorial*: IEEE Transactions on industrial electronics, volume 41, NO. 6

Jensen, K.; (1998), *An Introduction to the Practical Use of Coloured Petri Nets*: Lectures on Petri Nets II: Applications, Pages 237-292

Delgadillo, G. M., Llano, S. P.; (2007), *SCHEDULING APPLICATION USING PETRI NETS : A CASE STUDY: INTERGRÁFICAS S.A.*: 19th International Conference on Production Research

Van der Aalst; (1996), *Petri net based scheduling*: OR Spectrum, Volume 18, Issue 4, Pages 219-229

Silva, M., Teruel, E., Valette, R., Pingaud, H.; (1998), *Petri Nets and Production Systems*?: Lectures on Petri Nets II: Applications, Pages 85-124

Mejía, G., Odrey, N.; (2005), *An approach using Petri Nets and improved heuristic search for manufacturing system scheduling*: Journal of Manufacturing Systems, Volume 24, Issue 2, 2005, Pages 79-92

Gradisar, D., Music, G.; (2007), *Production-process modelling based on production-management data: a Petri-net approach*: International Journal of Computer Integrated Manufacturing archive, Volume 20, Issue 8, Pages 794-810

AMPL FAQ; (2004), *http://www.ampl.com/FAQ/index.html#WhatisAMPL*

Rudová, H.; (2009), *http://www.fi.muni.cz/h̃anka/rozvrhovani/prusvitky/sedma_bw.pdf*: Masarykova univerzita, Fakulta informatiky

Šůcha, P.; (2004), *Celočíselné lineární programování*: ČVUT, FEL, Katedra řídicí techniky

Palpant, M., Artigues, CH., Mireille, P.; (2004), *LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighbourhood Search*: Annals of Operations Research, Volume 131, Numbers 1-4, October 2004 , pp. 237-257(21)

Costel Alin Nicola; (2008), *Production Scheduling; report*: ČVUT, FEL, Katedra řídicí techniky

Petr Plaček; (2004), *Pokročilé plánování výroby, http://www.systemonline.cz/aps-scm/pokrocile-planovani-vyroby-pohled-uzivatele.htm*

Quadt, M., Kuhn, H.; (2007), *Batch scheduling of jobs with identical process times on flexible flow lines*: International Journal of Production Economics, Volume 105, Issue 2, February 2007, Pages 385-401

Muthu Mathirajan, Bhargav, V., Ramachandran, V.; (2009), *Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates*: The International Journal of Advanced Manufacturing Technology

Yeong-Dae Kim, Hyeong-Gyu Lim, Moon-Won Park; (1996), *Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process*: European Journal of Operational Research, Volume 91, Issue 1, 24 May 1996, Pages 124-143

Nowicki, E., Smutnicki, C.; (1998), *The flow shop with parallel machines: A tabu search approach*: European Journal of Operational Research, Volume 106, Issues 2-3, 16 April 1998, Pages 226-253

Quadt, D., Kuhn, H.; (2007), *A taxonomy of flexible flow line scheduling procedures*: European Journal of Operational Research, Volume 178, Issue 3, 1 May 2007, Pages 686-698

Pelikán, J.; (2001), *Diskrétní modely v operačním výzkumu*: Professional Publishing, Praha