# ŠKODA AUTO VYSOKÁ ŠKOLA o.p.s.

Studijní program: B6208 Ekonomika a management

Studijní obor/specializace: 6208R088 Podniková ekonomika a management provozu


# Algoritmy rozvrhování produkce na paralelní procesory v prostředí VBA for Excel

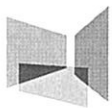
## Diplomová práce


**Bc. Roman MATERN**

Vedoucí práce: doc. Ing. Jan Fábry, Ph.D

ŠKODA AUTO VYSOKÁ ŠKOLA o.p.s.
Katedra logistiky, kvality a automobilové techniky

Akademický rok: 2019/2020



ŠKODA AUTO Vysoká škola

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Zpracovatel: **Bc. Roman Matern**

Studijní program: Ekonomika a management

Obor: Podniková ekonomika a management provozu

Název tématu: **Algoritmy rozvrhování produkce na paralelní procesory v prostředí VBA for Excel**

Cíl: Práce bude zaměřena na rozvrhování produkce na paralelní procesory. Cílem je zpracovat v prostředí VBA for Excel optimalizační algoritmy i heuristické metody pro řešení základních typů rozvrhovacích úloh v produkčních systémech. Výstupem práce bude aplikace, která uživateli nabídne výběr typu úlohy, zadání vstupních dat a jejich zpracování příslušným makrem VBA for Excel.

Rámcový obsah:

1. Základní pojmy z oblasti produkčních systémů a rozvrhování produkce na procesory.
2. Popis jednotlivých typů úloh rozvrhování produkce na paralelní procesory a metod pro jejich řešení.
3. Algoritmizace vybraných metod v prostředí VBA for Excel.
4. Vytrvoření finální aplikace pro rozhodování uživatele.

Rozsah práce: 55 – 65 stran

Seznam odborné literatury:

1. SIXTA, J. – MAČÁT, V. *Logistika.: Teorie a praxe.* 1. vyd. Brno: CP Books, 2005. 315 s. ISBN 80-251-0573-3.
2. FIALA, P. *Modely produkčních systémů.* 2. vyd. V Praze: Oeconomica, 2013. 231 s. ISBN 978-80-245-1966-1.
3. PINEDO, M. *Scheduling: theory, algorithms, and systems.* Springer, 2016. 670 s. ISBN 978-3-319-26578-0.
4. *Principles of financial modelling: model design and best practices using Excel and VBA.* Wiley, 2018. 512 s. ISBN 9781118904015.

Datum zadání diplomové práce: únor 2019

Termín odevzdání diplomové práce: leden 2020

L. S.

**doc. Ing. Jan Fábry, Ph.D.**
Vedoucí práce

**prof. Ing. Radim Lenort, Ph.D.**
Vedoucí katedry

**Mgr. Petr Šulc**
Prorektor ŠAVŠ

**Bc. Roman Matern**
Autor práce

Prohlašuji, že jsem závěrečnou práci vypracoval(a) samostatně a použité zdroje uvádím v seznamu literatury. Prohlašuji, že jsem se při vypracování řídil(a) vnitřním předpisem ŠKODA AUTO VYSOKÉ ŠKOLY o.p.s. (dále jen ŠAVŠ) směrnicí OS.17.10 Vypracování závěrečné práce.

Jsem si vědom(a), že se na tuto závěrečnou práci vztahuje zákon č. 121/2000 Sb., autorský zákon, že se jedná ve smyslu § 60 o školní dílo a že podle § 35 odst. 3 je ŠAVŠ oprávněna mou práci využít k výuce nebo k vlastní vnitřní potřebě. Souhlasím, aby moje práce byla zveřejněna podle § 47b zákona č. 111/1998 Sb., o vysokých školách.

Beru na vědomí, že ŠAVŠ má právo na uzavření licenční smlouvy k této práci za obvyklých podmínek. Užiji-li tuto práci, nebo poskytnu-li licenci k jejímu využití, mám povinnost o této skutečnosti informovat ŠAVŠ. V takovém případě má ŠAVŠ právo ode mne požadovat příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to až do jejich skutečné výše.

V Mladé Boleslavi dne .....

Děkuji doc. Ing. Janu Fábrymu, Ph.D za odborné vedení závěrečné práce, poskytování rad, podnětů a informačních podkladů.

# Obsah

**Seznam použitých zkratek a symbolů**

| | |
|---|---|
| batch | Batch processing (dávkové zpracování) |
| brkdwn | Breakdown (selhání) |
| FIFO | First-In, First-Out |
| prec | Precedence constraints (precedenční relace) |
| prmp | Preemptions (přerušení) |
| prmu | Permutation (omezení) |
| rcrc | Recirculation (recirkulace) |
| SW | Software |
| VBA | Visual Basic for Application |

# Úvod

V dnešním světě se každý výrobní podnik snaží o to, aby se zlepšilo jeho postavení na trhu. Jeden z aspektů, které k tomu napomáhají, je optimální využití dostupných zdrojů, ze kterých se v konečném důsledku vytváří přidaná hodnota. Proto je hodně důležité pro každou společnost správně plánovat své procesy. Součástí těchto plánovacích aktivit je rozvrhování. S urychleným technologickým pokrokem je společnost obecně řízena a organizována pomocí různých algoritmů rozvrhování a pomocí automatizace rozvrhovacích procesů dokáže prudce reagovat na jakékoli změny.

Cílem této práce je zpracovat v prostředí VBA for Excel optimalizační algoritmy i heuristické metody pro řešení základních typů rozvrhovacích úloh v produkčních systémech. Výstupem práce bude aplikace, která uživateli nabídne výběr typu úlohy, zadání vstupních dat a jejich zpracování příslušným makrem VBA for Excel.

Důvodem sloužícím k výběru tématu je vytvoření aplikace, která by mohla zautomatizovat proces řešení rozvrhovacích úloh. Kromě toho, by mohla tato aplikace sloužit jako podporující prvek pro uživatele, kteří se chtějí seznámit s problematikou rozvrhování a pochopit princip fungování jednotlivých metod a algoritmů.

Tato práce je rozdělena do čtyř kapitol. První kapitola pojednává o problematice teorie rozvrhování, popisuje a vysvětluje základní pojmy, které jsou spojeny s touto oblastí. Druhá kapitola je zaměřena na zkoumání konkrétních základních modelů s paralelně seřazenými procesory. V této části práce jsou rovněž představeny metody a algoritmy pro řešení daných typů úloh. V třetí kapitole se rozebírají jednotlivé kroky programování těchto metod a algoritmů a vysvětluje se obecný princip práce s VBA. Poslední kapitola obsahuje představení finální aplikace pro rozhodování uživatele.

# 1 Produkční systémy a rozvrhování produkce na procesory

Problémy teorie rozvrhování a kalendářního plánování jsou spojeny s přidělováním omezených zdrojů k provádění sady úloh a nalezení nejlepšího rozvrhu s předem definovanými vstupy. Teorie rozvrhování najde uplatnění v takových oblastech, jako je řízení výrobního podniku, organizace dopravních toků, plánování projektů, řízení zdrojů v počítačových systémech atd.

Tato kapitola se zaměřuje na teoretická východiska a základní pojmy z oblasti produkčních systémů a rozvrhování produkce na procesory. Zejména se zde rozebírají složky, na nichž je založen proces rozvrhování. Probírají se zde také různé modely s jedním či více procesory. Tato kapitola slouží jako úvod do problematiky teorie rozvrhování produkce na paralelně seřazené procesory.

## 1.1 Základní pojmy a definice

Rozvrhování je proces rozhodování, který se často používá v mnoha průmyslových odvětvích a službách. Zabývá se přidělováním zdrojů na úkoly v daném časovém úseku a jeho účelem je optimalizovat jeden nebo více parametrů. Pojem rozvrhování je neoddělitelně spjat s dalšími důležitými pojmy jako procesor a dávka (Fiala, 2013).

Pojem procesor znamená zařízení, které se používá k provádění jednotlivých činností či operací (Brucker, 2007). Jeden z hlavních předpokladů je, že počet procesorů je konečný a je označen písmenem *m*. Skupina procesorů v systému se označuje množinou *P*: $P = \{P_1, P_2, \ldots, P_m\}$.

Další důležitý pojem z problematiky rozvrhování je dávka, která představuje činnost provádějící se na jednotlivých procesorech jako samostatná operace (Brucker, 2007). Stejně jako u procesorů se počet dávek, neboli také úloh, považuje za konečný a označuje se písmenem *n*. Samotná dávka je označována písmenem *D* a množina dávek konkrétní úlohy může vypadat takto: $D = \{D_1, D_2, \ldots, D_n\}$.

Aby bylo možné ukázat, jak budou jednotlivé dávky zpracovány na procesorech, je nutné vytvořit rozvrh. Jedním z nejjednodušších příkladů rozvrhu je tzv. Ganttův diagram, který vytvořil Henry Gantt – zakladatel teorie rozvrhování (Pinedo, 2016). Gantt byl první, kdo použil analytické metody ke studiu posloupnosti výrobních

operací a vyvinul metody pro jejich plánování. Ganttův diagram je demonstrován níže na jednoduchém příkladu.

Předpokládá se následující scénář, kde musí kuchař připravit tříchodovou večeři – zeleninová polévka, vařené brambory a grilované kuře. Jednotlivé pokrmy jsou označeny jako dávky. Proces vaření se u každého z chodů skládá ze dvou činností (množin) – příprava a samotné vaření. Celkový počet dávek v tomto případě je šest, kde každá z množin má tři dávky. Aby bylo možné zahájit zpracování dávek z množiny vaření, musí toto jídlo projít nejdříve operací z množiny přípravy. Předpokládá se, že dávky z množiny přípravy před vařením mohou být zpracovány pouze na jednom procesoru, neboť je k dispozici pouze jeden kuchař (procesor). Ostatní dávky z množiny vaření mohou být zpracovány na separátních nezávislých procesorech, neboť stálá přítomnost kuchaře u těchto operaci není vyžadována a mohou se provádět samostatně a paralelně.

Doba přípravy jednotlivých surovin je označená parametrem $t_j$ a doba samotného vaření proměnnou $b_j$. Brambory se loupají po dobu 15 minut a vaří se po dobu 20 minut. Kuře se omyje a nadívá se po dobu 10 minut, pak se peče po dobu 40 minut. Suroviny na polévku se připravují 30 minut, poté se polévka vaří 60 minut. Tyto údaje lze shrnout do tabulky 1. Cílem dané úlohy je připravit večeři za co nejkratší čas. Proto se za optimální považuje rozvrh s minimální celkovou dobou trvání celé úlohy.

***Tab. 1 Příklad 1 - část 1***

| | **Příprava, $t_j$** | **Vaření, $b_j$** |
|---|---|---|
| **Brambory** | 15 | 20 |
| **Kuře** | 10 | 40 |
| **Polévka** | 30 | 60 |

Jako první je třeba uspořádat jednotlivé dávky z množiny vaření podle doby jejich zpracování $b_j$ od nejdelší po nejkratší - nejprve polévku, pak kuře, potom brambory. Jelikož operace vaření polévky trvá nejdelší dobu ze všech ostatních operací, musí být zpracování dávky z množiny přípravy surovin na polévku zahájeno v čase $t = 0$. Zbylé dávky budou uspořádány na procesory stejným způsobem podle hodnoty $b_j$.

Pro sestavení Ganttova diagramu, na kterém budou zobrazeny všechny dávky, rozvržené na jednotlivé procesory v čase, je nutné spočítat celkovou dobu přípravy surovin $T_j$ a celkový čas přípravy pokrmů $T_j + b_j$ dle vzorců:

$$T_1 = t_1, \tag{1}$$

$$T_2 = t_1 + t_2, \tag{2}$$

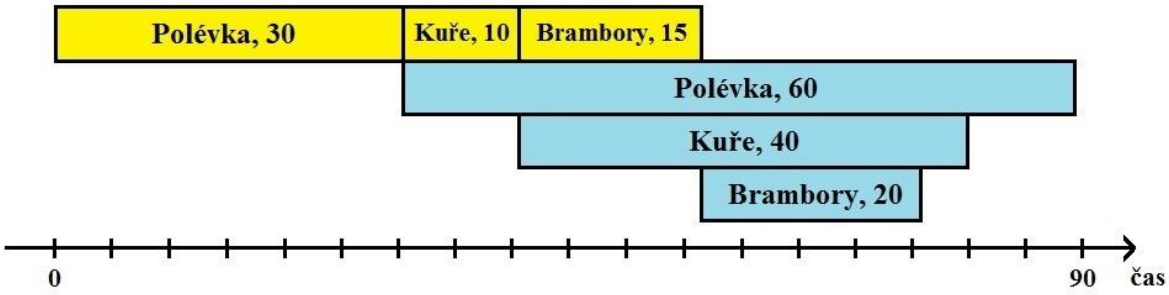
$$T_3 = t_1 + t_2 + t_3. \tag{3}$$

Výsledky výpočtu jsou uvedeny v následující tabulce:

***Tab. 2 Příklad 1 - část 2***

| | $t_j$ | $T_j$ | $b_j$ | $T_j + b_j$ |
|---|---|---|---|---|
| **Polévka** | 30 | 30 | 60 | 90 |
| **Kuře** | 10 | 40 | 40 | 80 |
| **Brambory** | 15 | 55 | 20 | 75 |
| **Celkem** | - | - | - | 90 |

Ganttův diagram dané úlohy je zobrazen na obrázku 1, kde žlutě jsou označeny dávky z množiny přípravy ingrediencí a modře dávky z množiny vaření pokrmů.




***Obr. 1 Příklad 1 - Ganttův diagram***

Z tohoto grafu je patrné, že minimální doba přípravy celé večeře je 90 minut a odpovídá celkové době vaření polévky. Jedná se o optimální rozvrh dané úlohy, který nelze dál zkrátit.

## Charakteristiky procesorů

V různých rozvrhovacích úlohách mohou mít procesory určité specifické vlastnosti, které do jisté míry ovlivňují způsob sestavení rozvrhu a konečný výsledek. Tyto vlastnosti jsou přesně definovány konkrétními modely. Podle Leunga (2004) lze za základní charakteristiku modelu považovat počet procesorů, na kterých mohou být zpracovány dávky. Jednoprocesorové modely jsou takové modely, v nichž je k dispozici pouze jeden procesor, na kterém jsou zpracovávány dávky. Pokud je počet procesorů větší než jeden, je takový model považován za víceprocesorový. Kromě toho, víceprocesorové modely mohou obsahovat dvě různá uspořádání procesorů, a to paralelně anebo sériově řazené procesory (Fábry, 2019). Předchozí úloha je příkladem jak paralelně, tak i sériově řazených procesorů – procesory, na kterých se zpracovávají dávky z množiny vaření, jsou seřazeny paralelně, protože dávky na těchto procesorech mohou být zpracovány nezávisle na sobě. Avšak tyto procesory jsou sériově řazené ve vztahu k prvnímu procesoru, na kterém jsou zpracovány dávky z množiny přípravy, protože dávky k těmto procesorům mohou být přiřazeny pouze tehdy, když bude dokončena předchozí dávka na prvním procesoru.

## Charakteristiky dávek

Stejně jako procesory mají i dávky různé charakteristiky. Mezi základní patří:

- Doba trvání $t_j$ - představuje dobu zpracování dávky na procesoru (Fábry, 2019). Doba trvání je často vyjádřena v minutách, může být také vyjádřena v hodinách. Důležitou podmínkou ve většině rozvrhovacím úloh je, že jednotky času musí být konzistentní. Pokud dávka $D_j$ je přiřazena k procesoru $P_i$, potom doba realizace $j$-té dávky je označena jako $t_{ij}$ (Leung, 2004).
- Nejdříve možný termín zahájení $r_j$ – jedná se o čas, ve kterém může dávka vstoupit do systému, tj. nejdřívější čas, ve kterém může být zahájeno zpracování $j$-té dávky (Leung, 2004).
- Požadovaný termín dokončení $d_j$ – představuje závazný termín dokončení $j$-té dávky (Leung, 2004).
- Váha dávky $w_j$ – vyjadřuje důležitost dávky. Pokud mají dávky v systému různé váhy, prioritně by měly být zpracovány nejdříve dávky s největší váhou.

Tato váha může představovat například skutečné náklady na udržení dávky v systému nebo množství přidané hodnoty k dávce (Fábry, 2019).

## Typy modelů rozvrhování a jejich definice

V závislosti na vstupních údajích, které jsou k dispozici pro řešení rozvrhovací úlohy, lze modely dělit na deterministické a stochastické. Deterministické modely jsou takové modely, kde jsou všechny vstupy známé. Na rozdíl od deterministických, jsou některé parametry ve stochastických modelech představeny hodnotami náhodnými (Pinedo, 2016).

V případě, že jsou všechny dávky známé a jejich množství je konstantní, model je definován jako statický z hlediska znalosti dávek. To znamená, že jsou na začátku rozvrhování k dispozici úplné informace o úkolech, které je třeba rozvrhnout, naplánovat, a zároveň jsou k dispozici veškeré informace o dostupných zdrojích. Rozvrh je tedy vytvořen najednou v jednom časovém okamžiku. Změna počtu dávek v průběhu jejich zpracování vede k vytváření tzv. dynamického modelu.

Každá rozvrhovací úloha může být popsána určitým popisem notace $\alpha \mid \beta \mid \gamma$. Pole $\alpha$ popisuje prostředí procesorů (jejich počet a uspořádání) a obsahuje pouze jeden vstup. Pole $\beta$ poskytuje podrobnosti o charakteristikách zpracování a omezeních a nemusí obsahovat vůbec žádný vstup anebo jeden či více vstupů. Pozice $\gamma$ označuje vybrané kritérium a často obsahuje pouze jeden vstup (Leung, 2004).

Parametr $\alpha$, který popisuje prostředí procesorů, má následující možné scénáře:

- 1 Systém s jedním procesorem – je to nejjednodušší ze všech možných prostředí procesorů.
- $P_m$ Systém s $m$ paralelně uspořádanými procesory. Dávka v takovém systému vyžaduje pouze jednu operaci a může být zpracována na kterémkoli z $m$ procesorů, jelikož jsou tyto procesory pro systém identické.
- $Q_m$ Systém s $m$ paralelně uspořádanými procesory s různými rychlostmi.
- $R_m$ Systém s $m$ paralelně uspořádanými různými procesory.
- $O_m$ Systém s $m$ sériově uspořádanými procesory „open shop“. Pro soubor operací neexistují žádné podmínky přednosti, dávky v takovém systému mohou být zpracovány v libovolném pořadí (Blazewicz, 1994).

- $F_m$ Systém s *m* sériově uspořádanými procesory „flow shop“. Na rozdíl od předchozího modelu uspořádaní procesorů, je v daném modelu stanoveno přesné pořadí procesorů, na kterých budou dávky zpracovány. Předpokládá se postupné vícestupňové plnění každého požadavku na každém procesoru předepsaným způsobem. První operace se provede na prvním procesoru, pak bude zahájena druhá operace na druhém procesoru, a tak dále až do *n*-té operace (Brucker, 2007).

- $J_m$ Systém s *m* sériově uspořádanými procesory „job shop“, ve kterém má každá dávka svou vlastní předem stanovenou cestu, po které se má vydat. Úlohy jsou vázány relacemi následnosti a liší se pro různé dávky (Brucker, 2007).

Podle Leunga (2004) poskytuje pole *β* podrobnosti o charakteristikách zpracování dávek a omezeních, které model má, a může obsahovat tyto parametry:

- $r_j$ – pro každou dávku je nastaven okamžik zahájení jejího zpracování. V případě, že se tato podmínka objeví v poli *β*, úloha pro *j*-tou dávku nemůže být zahájena dříve, než hodnota $r_j$. Pokud tato podmínka není uvedena v tomto poli, zahájení zpracování jednotlivých dávek není časově omezeno.

- $d_j$ - omezení okamžiku požadovaného dokončení zpracování dávky.

- *prmp* – tato podmínka umožňuje přerušení provádění jakýchkoli dávek na procesoru. Často se využívá k možnosti zastavení zpracování dávky na jednom procesoru, převodu a její dokončení na jiném procesoru. Množství zpracované předem přijaté dávky již není ztraceno. Když je předem zadaná dávky znovu vložena na procesor, potřebuje ho pouze po zbývající dobu zpracování. Základní podmínkou je také celočíselné dělení dávky.

- *prec* – precedenční omezení. Jednu nebo více dávek je třeba dokončit před povolením další dávky k jejímu zpracování. Omezení se mohou objevit v jednoprocesorovém modelu či víceprocesorovém s paralelně uspořádanými procesory.

- *batch* - mnoho dávek je rozděleno do skupin a postupné provádění úloh z různých skupin na jednom procesoru vyžaduje jeho následné uzpůsobení.

- *brkdwn* – selhání procesorů znamená, že procesor nemusí být neustále k dispozici.
- $M_j$ – podmínka, která v sobě zahrnuje určité omezení ve vztahu k procesoru. Pokud existuje omezení $M_j$, ne všechny $m$ procesory jsou schopny zpracovat $j$-tou dávku.
- *prmu* – jedná se o omezení, které se může objevit v prostředí flow shopu $F_m$. Představuje fronty před každým procesorem, které jsou řízeny podle pravidel FIFO. To znamená, že pořadí, ve kterém dávky procházejí prvním procesorem, je udržováno v celém systému.
- *rcrc* – recirkulace může nastat v prostředí job shopu $J_m$, když dávka může projít procesorem nebo pracovním centrem více než jednou.

Pole $\beta$ není limitováno pouze uvedenými procesními charakteristikami. Existují další omezení a podmínky, které mohou být použity při problému rozvrhování, dokonce existují modely, ve kterých může být více než jedna taková podmínka.

Poslední z tripletu notace je pole $\gamma$, které používá dvě skupiny kritérií, vycházejících buď z maximalizace ukazatelů, nebo z průměrných hodnot. Atributy pozice $\gamma$, které označuje vybrané kritérium pro hodnocení rozvrhů, jsou následující (Fábry, 2019):

- $C_{max}$ - čas dokončení poslední dávky. Odpovídá době dokončení poslední úlohy, která systém opustila. Minimální čas dokončení poslední dávky obvykle znamená dobré využití procesoru. Účelem modelu s takovým atributem je minimalizace celkové délky rozvrhu.
- $F_{max}$ – nejdelší doba pobytu dávky v systému.
- $\bar{F}$ – průměrná doba pobytu dávky v systému.
- $L_{max}$ – největší časová diference mezi časem dokončení a plánovaným termínem dokončení dávky. Účelová funkce v daném modelu se snaží o minimalizaci zpoždění.
- $T_{max}$ – největší zpoždění dávky.
- $N$ – počet zpožděných dávek.
- $\sum C_j$ – minimalizace součtu časů dokončení všech úloh.

Jak již bylo zmíněno, modely se liší počtem procesorů. Když je v systému k dispozici pouze jeden procesor, jedná se o jednoprocesorový model. Pokud je počet procesorů větší než jeden, jde o víceprocesorový model, který se dále dělí na modely se sériově či paralelně uspořádanými procesory.

## 1.2 Základní modely s jedním procesorem

Prostředí s jedním procesorem je relativně jednoduché a některé z jednoprocesorových modelů mají často vlastnosti, které nejsou vždy charakteristické pro modely s paralelně či sériově seřazenými procesory. Nicméně, výsledky, které lze získat v modelech s jedním procesorem, poskytují nejen vhled do prostředí jednoho procesoru, ale také základy, které jsou použitelné ve složitějších procesorových prostředích, protože tyto komplikovanější rozvrhovací problémy můžeme rozdělit na jednotlivé menší problémy, zabývající se jednotlivými procesory.

Ve většině základních modelů s jedním procesorem neexistují žádné podmínky z pole $\beta$ (Pinedo, 2016). Pro takové modely platí rozvrhy bez přerušení a bez prostojů, přičemž pořadí jednotlivých dávek je pevně stanoveno.

### Model 1||*Z*

První základní model s jedním procesorem je model 1||*Z*, kde hodnota *Z* zastupuje jakékoliv z kritérií pole $\gamma$ z notace. Mohou to být například $C_{max}$, $\bar{F}$ nebo $F_{max}$. V tomto modelu platí předpoklad, že v systému je pouze jeden procesor, který má na zpracování *n* množství dávek, z nichž každá je tvořena jednou operací. Kromě toho platí u daného základního modelu další stanovené podmínky (Fábry, 2019):

1. Je přesně definováno *n* množství dávek, které má být zpracováno na existujícím jednom procesoru, přičemž každá z definovaných dávek má dobu realizace $t_j$ a termín nejdříve možného zahájení *j*-té dávky je stanoven v čase $r_j = 0$.

2. Dávky jsou vzájemně nezávislé. To znamená, že mezi nimi neexistuje žádné precedenční relace, totiž není zde návaznost dávek při jejich zpracování. Zpracování následující dávky může začít ihned po dokončení zpracování libovolné předchozí dávky.

3. Doby realizace existujících dávek v procesoru nejsou závislé na pořadí jejich realizace.

Za předpokladu rozvrhu bez přerušení a bez prostojů, jsou rozvrhy dány pořadím jednotlivých dávek. Přičemž počet možných rozvrhů je určen počtem všech permutací *n!,* a jelikož termín zahájení je roven nule, termín dokončení poslední dávky je stejný jako doba pobytu poslední dávky v systému:

$$F_j = C_j - r_j, \tag{4}$$

kde $r_j = 0$.

Tím pádem hodnoty $C_j$ a $F_j$ se rovnají a je možné sledovat pouze jedno z těchto dvou kritérií.

### Model 1||$F_{max}$

Dalším základním modelem je model 1||$F_{max}$, kde $F_{max}$ znamená nejdelší dobu pobytu dávky v systému (Fábry, 2019). Jelikož nejdříve možné zahájení je rovno nule a dávky nemají precedenční relaci, tj. jsou vzájemně nezávislé, všechny rozvrhy jsou tady rovnocenné, a hodnotu $F_{max}$ získáme podle následujícího vzorce:

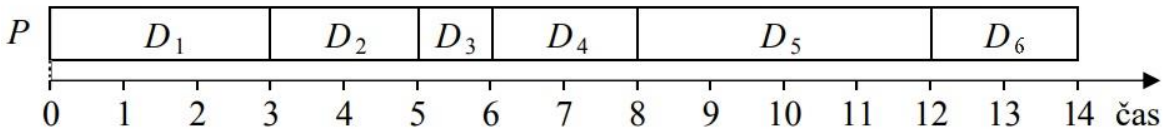
$$F_{max} = \sum_{j=1}^{n} t_j \tag{5}$$

Pro vysvětlení daného a modelu je uveden následující příklad. Existuje jeden procesor, na který je nutné zařadit šest dávek, u nichž jsou k dispozici následující vstupní údaje (v minutách):

***Tab. 3 Příklad 2 – část 1, doby realizace dávek***

| $D_j$ | $t_j$ |
|---|---|
| **D1** | 3 |
| **D2** | 2 |
| **D3** | 1 |
| **D4** | 2 |
| **D5** | 4 |
| **D6** | 2 |

Ganttův diagram by vypadal následovně:




***Obr. 2 Ganttův diagram, 1||$F_{max}$***

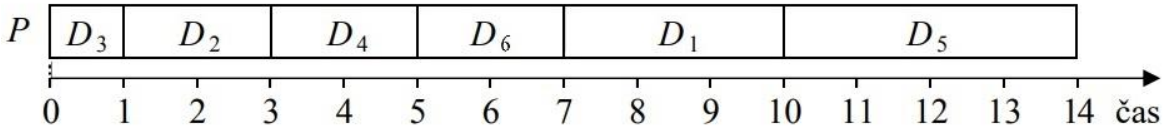
Hodnota $F_{max}$ v daném příkladu činí 14 minut. V tomto modelu jakákoli změna v uspořádání dávek vede k optimálnímu rozvrhu.

## Model 1|| $\overline{F}$

Dalším základním modelem je model 1|| $\overline{F}$, kde se řeší průměrná doba pobytu dávky v systému (Fábry, 2019). Při hledání optimálního rozvrhu je zásadní podmínkou uspořádání dávek dle doby jejich zpracování od nejdelší po nejkratší:

$$t_1 \leq t_2 \leq \cdots \leq t_n \tag{6}$$

Pro tvorbu Ganttova diagramu jsou použity údaje z předchozího příkladu. Po uspořádání jednotlivých dávek podle doby realizace v klesající posloupnosti bude mít diagram následující podobu:




***Obr. 3 Ganttův diagram, 1|| $\overline{F}$***

Pro výpočet optimální hodnoty kritéria účelové funkce slouží následující vzorec:

$$\overline{F} = \frac{1}{n}\sum_{j=1}^{n} F_j \tag{7}$$

Optimální hodnota kritéria $\overline{F}$ se rovná 6,66 minut.

## Model *1*|| $\overline{F}$(*w*)

Dalším základním modelem je model *1*|| $\overline{F}$(*w*), kde se již bere v úvahu relativní důležitost (váha) jednotlivých dávek (Fábry, 2019). Cílem tohoto modelu je minimalizovat váženou průměrnou dobu pobytu dávky v systému. Optimální rozvrh

bude mít následující vlastnosti:

$$\frac{t_1}{w_1} \leq \frac{t_1}{w_2} \leq \cdots \leq \frac{t_n}{w_n} \tag{8}$$

Pro vysvětlení tohoto modelu je použit předchozí příklad, jehož základní údaje jsou rozšířeny o relativní důležitost dávek v podobě váhy. Tyto údaje jsou představeny v tabulce 4:

***Tab. 4 Příklad 2 – část 2, doby realizace dávek a jejich váhy***

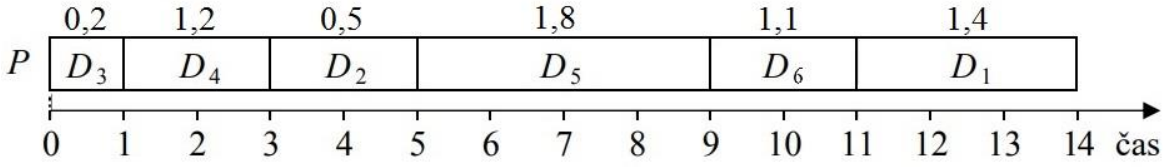
| $D_j$ | $t_j$ | $w_j$ |
|---|---|---|
| **D1** | 3 | 0,1 |
| **D2** | 2 | 0,1 |
| **D3** | 1 | 0,2 |
| **D4** | 2 | 0,4 |
| **D5** | 4 | 0,2 |
| **D6** | 2 | 0,1 |

První krok řešení uvedeného modelu spočívá v seřazení dávek dle vzorce (8). Výsledky tohoto kroku jsou zobrazeny v tabulce 5.

***Tab. 5 Příklad 2 – část 3, seřazené dávky dle vah***

| $D_j$ | $t_j$ | $w_j$ | $t_j / w_j$ |
|---|---|---|---|
| **D1** | 3 | 0,1 | 30 |
| **D2** | 2 | 0,1 | 20 |
| **D3** | 1 | 0,2 | 5 |
| **D4** | 2 | 0,4 | 5 |
| **D5** | 4 | 0,2 | 20 |
| **D6** | 2 | 0,1 | 20 |

Ganttův diagram bude vypadat následujícím způsobem:




***Obr. 4 Ganttův diagram, 1|| $\bar{F}(w)$***

Dávky na diagramu jsou seřazeny podle hodnot $t_j$ / $w_j$ v rostoucí posloupnosti. Ke každé dávce je vypočítána průměrná vážená doba (9) jejího pobytu v systému a je znázorněna nad jednotlivými dávkami.

$$\bar{F}(w) = \sum_{j=1}^{n} w_j F_j \tag{9}$$

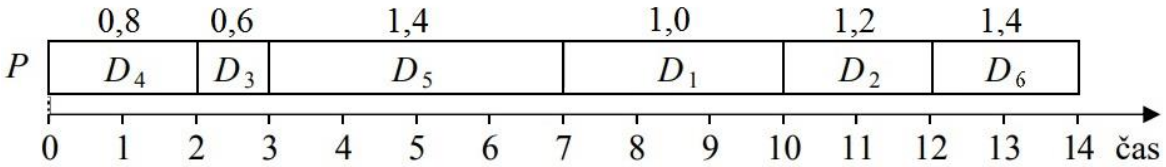
Hodnota $\bar{F}(w)$ činí 6,2 minuty.

## Model 1||$F_{max}(w)$

V modelu 1||$F_{max}(w)$ se také bere v úvahu relativní důležitost (váha) jednotlivých dávek, ale účelem tohoto modelu je minimalizace vážené doby pobytu dávky v systému. Pro zjištění optimálního rozvrhu je třeba uspořádat dávky do klesající posloupnosti jejich vah:

$$w_1 \geq w_2 \geq \cdots \geq w_n \tag{10}$$

Dávky jsou seřazeny podle jejich vah bez ohledu na dobu realizace. S umístěním dávek na procesor podle klesající posloupnosti jejich vah bude Ganttův diagram vypadat následujícím způsobem (údaje jsou použité z předchozího příkladu):




***Obr. 5 Ganttův diagram, 1||$F_{max}(w)$***

Ke každé dávce je vypočítána vážená doba jejího pobytu v systému, znázorněna nad jednotlivými dávkami. Jedná se o hodnotu, vypočítanou jako součin váhy dávky

a její poslední časové jednotky na ose Ganttova diagramu (viz vzorec 11).

$$F_{max}(w) = max\{w_j F_j\}, j = 1,2,\dots,n \tag{11}$$

Hodnota $F_{max}(w)$ je 1,4 minuty.

### Modely 1||$L_{max}$ a 1||$T_{max}$

Dalšími základními modely jsou modely 1||$L_{max}$ a 1||$T_{max}$, kde $L_{max}$ je největší časová diference mezi časem dokončení $j$-té dávky $C_j$ a plánovaným termínem dokončení dávky $d_j$. Hodnota $T_{max}$ představuje největší zpoždění dávky. Optimálním zde bude ten rozvrh, jehož dávky jsou uspořádány do rostoucí posloupnosti jejich požadovaného termínu dokončení:

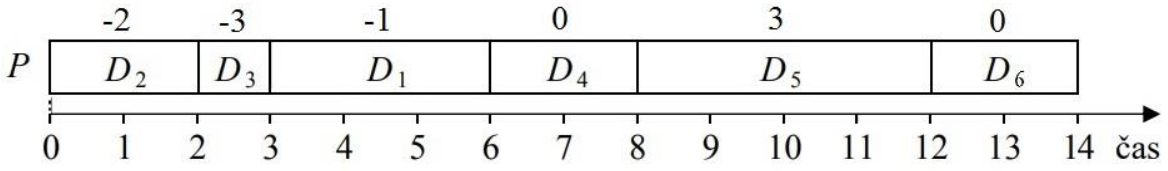
$$d_1 \leq d_2 \leq \cdots \leq d_n \tag{12}$$

Základní údaje z předchozího příkladu jsou rozšířeny o hodnoty $d_j$ a jsou znázorněny v tabulce 6.

***Tab. 6 Příklad 2 – část 4, doby realizace dávek, požadované termíny jejich dokončení a váhy***

| $D_j$ | $t_j$ | $d_j$ | $w_j$ |
|---|---|---|---|
| **D1** | 3 | 7 | 0,1 |
| **D2** | 2 | 4 | 0,1 |
| **D3** | 1 | 6 | 0,2 |
| **D4** | 2 | 8 | 0,4 |
| **D5** | 4 | 9 | 0,2 |
| **D6** | 2 | 14 | 0,1 |

Dávky, seřazené dle požadovaných termínů dokončení $d_j$ do rostoucí posloupnosti, jsou zobrazeny na obrázku 6.




***Obr. 6 Ganttův diagram, 1||$L_{max}$ a 1||$T_{max}$***

Hodnoty nad jednotlivými dávkami představují předstih, či zpoždění dávek oproti požadovanému termínu jejich dokončení. Záporné hodnoty znamenají, že dávka je dokončena před požadovaným termínem dokončení. Kladné hodnoty napovídají o zpoždění v dokončení dávky. Na tomto příkladu je vidět, že hodnota zpoždění dávky $D_5$ je 3 minuty a je optimální hodnotou kritéria $L_{max}$ a zároveň kritéria $T_{max}$.

### Algoritmy v základních jednoprocesorových modelech

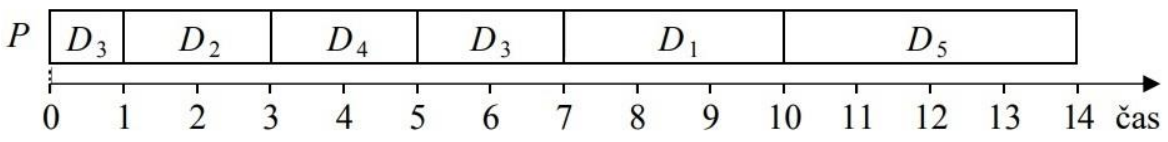
Pro nalezení optimálního rozvrhu v základních modelech se také používá celá řada různých algoritmů, představujících určité postupy, které vedou k řešení konkrétních rozvrhovacích úloh. Jedním z takových modelů je model $1||L_{max}(w)$ a model $1||T_{max}(w)$, jejichž optimální rozvrh se dá zjistit Lawlerovým algoritmem (Brucker, 2007). Tento algoritmus spočívá v přesměrování v rozvrhu dávek s nižším váženým zpožděním na pozdější dobu než dávky s vyšším váženým zpožděním. Algoritmům bude věnována další část práce, kde je ukázána jejich aplikace v modelech s paralelně řazenými procesory.

## 1.3 Jednoprocesorové modely s parametrem *β*

Ne vždy však u základních modelů platí, že neexistují žádné podmínky z pole *β*. Mohou zde být uvolňovány stanovené předem podmínky. Nemusí tady platit, že termín nejdříve možného zahájení dávek $r_j$ je roven nule, a zároveň může dojít k přerušení dávek, prostojům a dalším charakteristickým odlišnostem oproti podmínkám, které jsou obecně platné pro základní modely.

### Model $1|r_j;prmp|\overline{T}$

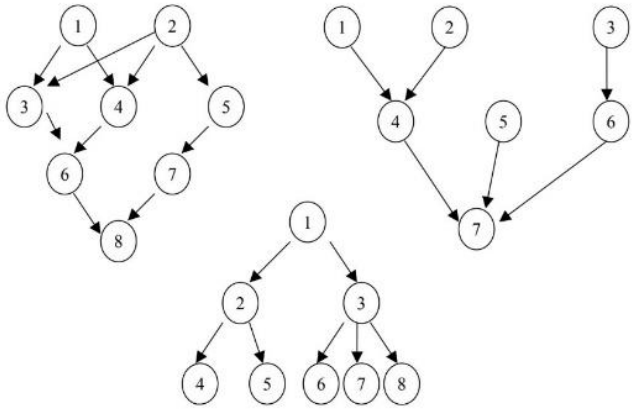
Typickým příkladem je model $1|r_j;prmp|\overline{T}$ , který již nepředpokládá termíny nejdříve možného zahájení dávek $r_j$ v čase 0 a zároveň je zde možnost *prmp*, která dovoluje přerušení procesu zpracování jakýchkoliv dávek (Pinedo, 2016). Kritérium $\overline{T}$ vyjadřuje průměrnou hodnotu zpoždění dávky a cílem v tomto modelu je tuto hodnotu minimalizovat. Ganttův diagram optimálního rozvrhu s přerušením by vypadal následovně (pouze ilustračně, nevychází z údajů předchozího příkladu):




***Obr. 7 Ganttův diagram,*** $1|r_j;prmp|\overline{T}$

### Model 1|*prec*|$Z_{max}$

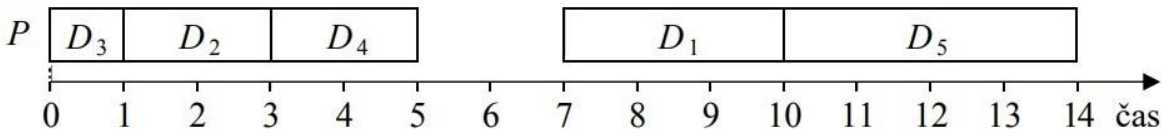
V tomto modelu je stanovena závislost mezi dávkami, která je vyjádřena pomocí precedenčních relací (Pinedo, 2019). To znamená, že jednu nebo více dávek je nutné dokončit před povolením další úlohy k jejímu zpracování. Parametr $Z_{max}$ představuje jakékoli kritérium, v němž chceme minimalizovat maximum některé z veličin. Precedenční relaci lze ukázat pomocí speciálních grafů:




***Obr. 8 Příklady precedenčních relací mezi dávkami***

### Model 1|*brkdwn*|$Z_{max}$

V modelu 1|*brkdwn*|$Z_{max}$ existuje předpoklad, že může dojít k selhání procesoru, a proto nebude po celou dobu realizace všech dávek k dispozici (Pinedo, 2016). V tomto případě může na časové ose vzniknout interval, kdy nebude procesor obsazen žádnou z dávek. Ganttův diagram optimálního rozvrhu s omezením *brkdwn* může vypadat takto (pouze ilustračně, nevychází z údajů předchozího příkladu):




***Obr. 9 Ganttův diagram, 1|brkdwn|$Z_{max}$***

## 1.4 Modely se sériově řazenými procesory

V mnoha výrobních a montážních procesech musí každá zakázka podstoupit řadu operací. Tyto operace se často provádějí ve všech úlohách ve stejném pořadí, což znamená, že úlohy musí následovat stejnou cestu. Tím pádem se předpokládá, že procesory jsou uspořádány sériově.

Pokud se jedná o sériově řazené procesory, rozlišují se různé variace a podmínky, které naznačují, existuje-li podmínka přednosti, předpokládá-li se postupné vícestupňové plnění každého požadavku na každém procesoru předepsaným způsobem, má-li každá dávka svou vlastní předem stanovenou cestu apod. Podle těchto omezení lze také dále určit, zda prostředí procesorů má scénář open shopu, flow shopu, job shopu či jinou z jejich variací.

### Model $F_m||C_{max}$

Stejně jako u základních jednoprocesorových modelů, modely se sériově řazenými procesory mají určité vlastnosti. Například model $F_m||C_{max}$, kde je v systému $m$ procesorů, na nichž se má zpracovat $n$ množství dávek, každá z kterých se dále skládá z $m$ množství operací. Tyto operace probíhají na každém z procesorů separátně. Procesory nemohou najednou zpracovávat více než jednu dávku, a jednotlivé dávky nemohou být současně zpracovávány na více procesorech. Zároveň zde platí další podmínky a omezení, které mají základní modely s jedním procesorem (Fábry, 2019):

1. Dávky jsou vzájemně nezávislé, neexistuje mezi nimi precedenční relace, tj. není nutná návaznost dávek při jejich zpracování.

2. Doby realizace existujících dávek v procesoru jsou nezávislé na pořadí jejich realizace.

3. Všechny dávky a procesory jsou dostupné v čase $t = 0$.

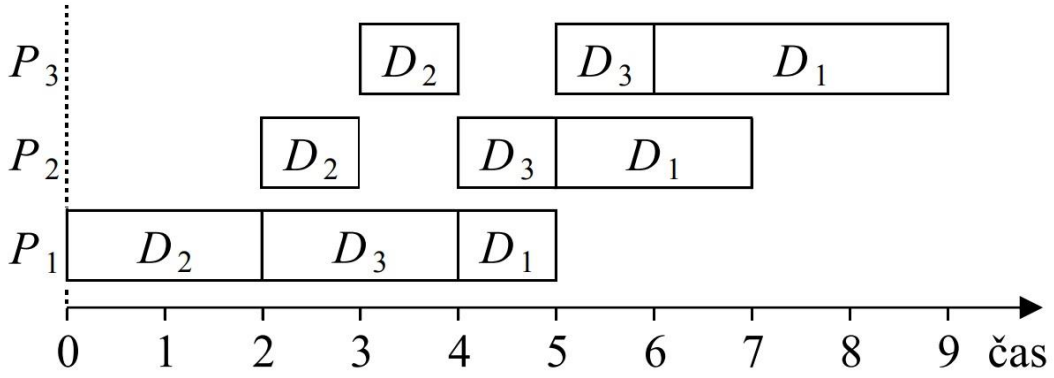
4. Je zakázáno jakékoli přerušení operace.

Pro vysvětlení daného modelu se sériově řazenými procesory je níže uveden příklad. Existují tři sériově řazené procesory, na kterých je třeba zpracovat tři dávky, u nichž je k dispozici doba jejich realizace (v minutách).

*Tab. 7 Příklad 3, doby realizace operací dávek na procesorech*

| | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|
| $t_{1j}$ | 1 | 2 | 2 |
| $t_{2j}$ | 2 | 1 | 1 |
| $t_{3j}$ | 3 | 1 | 1 |

Ganttův diagram bude mít následující tvar:




*Obr. 10 Ganttův diagram, $F_m||C_{max}$*

Jedná se o přípustný rozvrh v délce trvání 9 minut.

# 2 Modely s paralelními procesory

Není ojedinělé, že se s modely s paralelními procesory setkáváme, neboť výskyt paralelních procesorů je v reálném světě běžný a představuje z praktického hlediska velmi důležitou složku. Různé techniky a modely paralelních procesorů se často používají jako součásti vícestupňových systémů. Nejen že mají modely s paralelními procesory význam z praktického hlediska, ale také z hlediska teoretického, kde takové procesory představují zobecnění jednoprocesorových modelů.

Dávky, zpracovávající se na paralelních procesorech, mohou být řazeny vedle sebe, a lze je tím pádem zpracovávat současně. K tomu se přidávají mnohem častěji různé vlastnosti a charakteristiky procesorů (pole $\beta$ z notace), což vede k rozmanitosti úloh a navíc k větší obtížnosti problematiky modelů s paralelními procesory.

Tato kapitola je věnována vybraným typům úloh rozvrhování produkce na paralelní procesory a algoritmům jejich řešení. Tyto modely jsou následující:

- Model $P_m||F_{max}$,
- Model $P_m|prmp|F_{max}$,
- Model $P_m||\bar{F}$.

## 2.1 Model $P_m||F_{max}$ a metody řešení

Jako první je rozebrán model $P_m||F_{max}$. Hodnota $F_{max}$ představuje nejdelší dobu pobytu dávky v systému (Brucker, 2007). Účelem tohoto modelu je minimalizace hodnoty proměnné $F_{max}$:

$$z = F_{max} \rightarrow min \tag{13}$$

V tomto modelu platí určité omezující podmínky, které musí být dodrženy při sestavení optimálního rozvrhu. Tyto podmínky jsou následující (Fábry, 2019):

1. Celková doba realizace dávek, které jsou přiřazeny na procesor, nesmí překročit délku nejkratšího rozvrhu, odpovídajícího hodnotě $F_{max}$.
2. Každá dávka může být přiřazena právě na jeden procesor.
3. Proměnná $F_{max}$ musí být celočíselná hodnota.

Daný model bude vysvětlen na následujícím příkladu. K dispozici jsou čtyři identické procesory, na které je potřeba zařadit osm dávek. Doby realizace jednotlivých dávek $t_j$ jsou uvedeny v tabulce 8.

***Tab. 8 Příklad 4 - část 1, doby realizace dávek***

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ |
|---|---|---|---|---|---|---|---|---|
| $t_j$ | 5 | 3 | 2 | 1 | 8 | 5 | 4 | 8 |

Hodnota $F_{max}$ (13) činí 9 minut (po zaokrouhlení). Avšak není možné sestavit přípustný rozvrh, v němž by nejdelší doba pobytu dávky v systému $F_{max}$ odpovídala hodnotě, získané dle vzorce. Toto je způsobeno nemožností přerušení dávek v systému. Pro hledání optimálního rozvrhu se často používá heuristický algoritmus, který u většiny úloh může poskytnout optimální řešení. Tento algoritmus zahrnuje tři kroky (Fábry, 2019):

1. V prvním kroku je nezbytné uspořádat dávky podle doby jejich realizace od nejdelší k nejkratší:

$$t_1 \geq t_2 \geq \cdots \geq t_n \,. \tag{14}$$

2. Z nově uspořádaných dávek se vybere první množství dávek, odpovídající počtu procesorů $m$, a přiřadí se na volné procesory v čase $t = 0$.

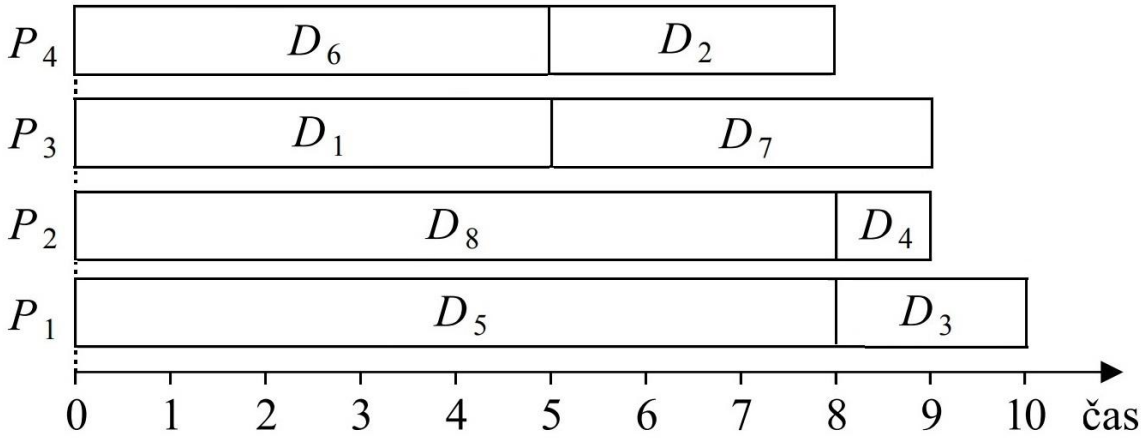
3. Poslední krok spočívá v přiřazování zbylých dávek na procesor, který se uvolní jako první. Tento krok se opakuje, dokud nebudou všechny dávky přiřazeny na procesory.

Po aplikaci vzorce (14) bude pořadí dávek z příkladu 4 vypadat následujícím způsobem:

***Tab. 9 Příklad 4 - část 2, uspořádané dávky dle doby realizace od nejdelší k nejkratší***

| | $D_5$ | $D_8$ | $D_1$ | $D_6$ | $D_7$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|---|---|---|---|
| $t_j$ | 8 | 8 | 5 | 5 | 4 | 3 | 2 | 1 |

Podle dalších dvou kroků budou všechny další dávky přiřazeny na procesory. Optimální rozvrh této úlohy v podobě Ganttova diagramu má následující podobu:




***Obr. 11 Ganttův diagram, $P_m||F_{max}$***

Hodnota $F_{max}$ činí 10 minut.

## 2.2 Model $P_m|prmp|F_{max}$ a metody řešení

U paralelních procesorů hrají přerušení důležitější roli než u modelů s jedním procesorem. U modelu s jedním procesorem přerušení hrají obvykle roli pouze tehdy, když mají dávky nejdříve možný termín zahájení v různých časových okamžicích. Naproti tomu u modelů s paralelními procesory jsou přerušení důležitá, i kdyby všechny dávky měly nejdříve možný termín zahájení v čase nula ($r_j = 0$).

V případě modelu s paralelními procesory $P_m|prmp|F_{max}$ se předpokládá existence $n$ dávek, které se mohou zpracovávat na kterémkoliv z $m$ procesorů. Přičemž, jak již bylo zmíněno, nejdříve možný termín zahájení dávek je v čase nula ($r_j = 0$). Dávky jsou nezávislé a není tady návaznost dávek při jejich zpracování, tj. neexistují precedenční relace. Jedinou vlastnost, kterou tento model má, je charakteristika *prmp*, která dovoluje přerušení procesu zpracování jakýchkoliv dávek po dokončení realizované operace, přičemž množství zpracování předem přijaté dávky již není ztraceno, a lze se zpracováním dávky pokračovat na kterémkoli z $m$ procesorů. Hodnota $F_{max}$ představuje nejdelší dobu pobytu dávky v systému (Fábry, 2019).

Pro délku nejkratšího rozvrhu platí:

$$F_{max} = max\left\{\frac{1}{m}\sum_{j=1}^{n} t_j \; ; \; \underset{j=1,2,\ldots,n}{max}\, t_j\right\}, \tag{15}$$

kde první výraz v závorce vyjadřuje průměrnou dobu obsazení procesoru, která se vypočítá jako součet dob realizace dávek dělena počtem procesorů (Pinedo, 2016). Jelikož lze dávky přerušit prakticky kdykoli během jejich realizace, nemusí výsledek vyjít celočíselný. I když je potřeba v praktické úloze respektovat veškerá omezení, je nezbytné předpokládat, že jednotlivé operace mají jednotkovou dobu trvání. Tím pádem se v případě neceločíselného výsledku provede jeho zaokrouhlení směrem nahoru. Druhá hodnota představuje nejdelší dobu realizace dávky v systému.

Jeden ze základních algoritmů, které se používají k získání optimálního rozvrhu pro tento model, je McNaughtonův algoritmus. Tento algoritmus se skládá ze tří kroků, které jsou popsány takto (Fábry, 2019):

1. První krok spočívá ve zjištění hodnoty $F_{max}$, která se zjistí dle vzorce 14. Tato hodnota $F_{max}$ pak bude odpovídat délce nejdelšího rozvrhu, tzn. že zpracování poslední dávka na *i*-tém procesoru bude dokončeno v čase $F_{max}$. Potom se dávka $D_1$ zařazuje na procesor $P_1$ v čase $t = 0$.
2. Druhý krok spočívá v přiřazování dalších dávek z množiny dávek *n* podle jejich pořadových čísel na stejný procesor. Během přiřazování zbývajících dávek musí být respektována vypočítaná hodnota $F_{max}$, tzn. že tato hodnota nesmí být překročena a může nastat jedna ze tří situací:
    a) Na daný procesor jsou přiřazeny všechny zbývající dávky a hodnota $F_{max}$ nebyla překročena. Poslední přiřazenou dávkou je dávka $D_n$;
    b) poslední přiřazená dávka $D_k$ na tento procesor je dokončena přesně v čase $F_{max}$;
    c) při přiřazování dávky $D_k$ dojde k překročení hodnoty $F_{max}$.

3. V závislosti na tom, ke kterému z výše uvedených případů došlo, se rozlišují tři další možné postupy:

   a) všechny dávky jsou rozvrženy na procesory, rozvrh je sestaven. Konec algoritmu;

   b) na další volný procesor se zařadí dávka $D_{k+1}$ v čase $t$=0. Následně se opakuje krok 2;

   c) jelikož došlo k překročení hodnoty $F_{max}$, je potřeba stávající dávku $D_k$ rozdělit na dvě části, kde její první část bude dokončena v čase $F_{max}$ na současném procesoru, na kterém bylo zahájeno její zpracování, a druhá část se převede na další volný procesor v čase t=0. Po dokončení zpracování druhé části dávky $D_k$ se postupuje podle kroku 2.

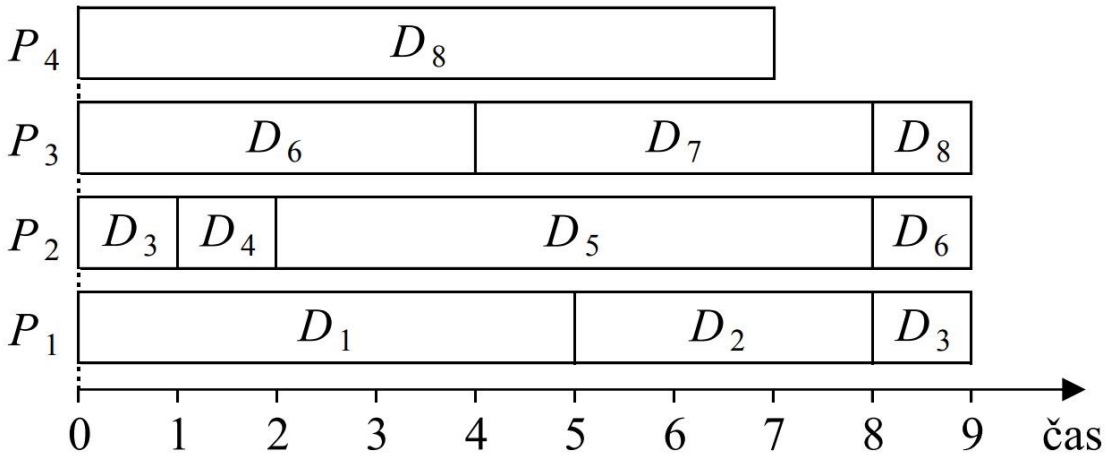
Pro vysvětlení principu fungování tohoto algoritmu je uveden následující příklad – je k dispozici osm dávek a čtyři procesory. U dávek jsou známy doby jejich realizace. Úkolem je rozvrhnout tyto dávky na procesory tak, aby byly realizovány v minimálním čase. Doby realizace dávek jsou uvedeny v tabulce 10 (v minutách). Podmínkou v takovém modelu je, že jednotlivé operace prováděné na dávkách, mají jednotkovou dobu trvání 1 min, tzn. že dávky smí být přerušeny pouze po dokončení jednotlivé operace.

***Tab. 10 Příklad 5, doby realizace dávek***

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ |
|---|---|---|---|---|---|---|---|---|
| $t_j$ | 5 | 3 | 2 | 1 | 6 | 5 | 4 | 8 |

Řešení této úlohy pomocí McNaughtonova algoritmu podle tří výše uvedených kroků bude následující – nejdříve se vypočítá hodnota $F_{max}$, která bude maximem ze dvou hodnot - součtu dob realizace dávek děleným počtem procesorů nebo maximální doby realizace $j$-té dávky. První hodnota se rovná 8,5 minuty, druhá 8 minut. Vzhledem k podmínce, která zakazuje přerušovat nedokončenou operaci, musí být hodnota $F_{max}$ zaokrouhlena na celočíselnou hodnotu směrem nahoru. V tomto případě bude hodnota $F_{max}$ činit 9 minut.

Po zjištění hodnoty $F_{max}$ lze zahájit přiřazení jednotlivých dávek na procesory. Optimální rozvrh této úlohy bude mít následující tvar:




***Obr. 12 Ganttův diagram, $P_m|prmp|F_{max}$***

Jak je vidět na obrázku 11, došlo k přerušení tří dávek $D_3$, $D_6$ a $D_8$. Od toho lze odvodit, že maximální možný počet přerušení dávek je $m$-1. Tato přerušení jsou možná pouze při dodržování předpokladu, že všechny procesory jsou identické.

## 2.3 Model $P_m||\bar{F}$ a metody řešení

Poslední model s paralelně seřazenými procesory, který bude rozebrán v této kapitole, řeší rozvrhovací problém minimalizace průměrné doby pobytu dávky v systému. Pro zjištění optimálního rozvrhu se používá optimalizační algoritmus se stejným principem jako algoritmus, který byl rozebrán pro obdobný model s jedním procesorem $1||\bar{F}$ . Tento optimalizační algoritmus se skládá ze tří kroků (Pinedo, 2016):

1. První krok spočívá ve vytvoření pořadí, ve kterém jsou dávky uspořádány podle doby realizace od nejkratší po nejdelší:

$$t_1 \leq t_2 \leq \cdots \leq t_n \,. \tag{16}$$

2. Ve druhém kroku, stejně jako v heuristickém algoritmu pro nalezení optimálního řešení modelu $P_m||F_{max}$, se zařadí první $m$ množství dávek na procesory v čase $t = 0$.

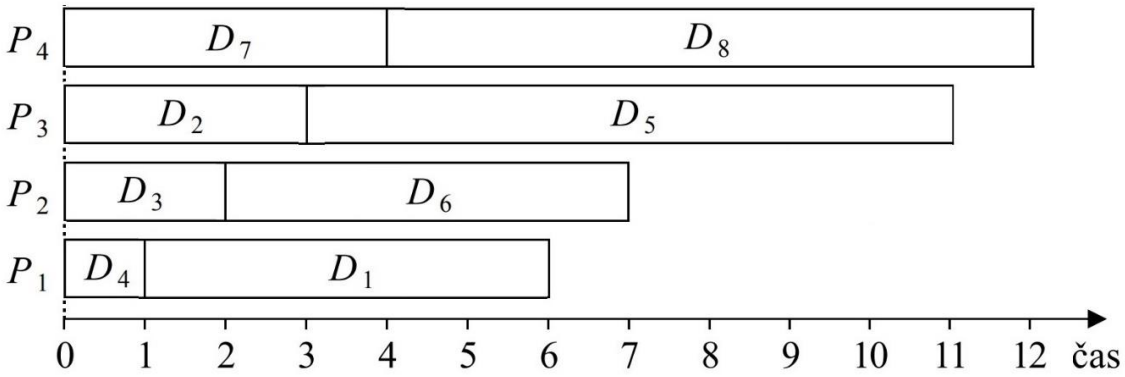
3. Třetí krok spočívá v přiřazení zbývajících dávek na procesory za dávky, jejichž zpracování bude dokončeno co nejdříve.

Pro aplikaci daného algoritmu se použije příklad 4 s údaji, které jsou zobrazené v tabulce 8. Po seřazení dávek do rostoucí posloupnosti lze získat následující pořadí:

***Tab. 11 Příklad 4 - část 3, uspořádané dávky dle doby realizace od nejkratší k nejdelší***

| | $D_4$ | $D_3$ | $D_2$ | $D_7$ | $D_1$ | $D_6$ | $D_5$ | $D_8$ |
|---|---|---|---|---|---|---|---|---|
| $t_j$ | 1 | 2 | 3 | 4 | 5 | 5 | 8 | 8 |

Optimální rozvrh je zobrazen na obrázku 13:




***Obr. 13 Ganttův diagram, $P_m||\bar{F}$***

Pro výpočet optimální hodnoty kritéria $\bar{F}$ se používá vzorec (7). Tato hodnota představuje průměrnou dobu pobytu dávky v systému a pro tento příklad činí 6 minut.

# 3 Algoritmizace vybraných metod v prostředí VBA for Excel

Řešení určitých rozvrhovacích úloh je ve většině případů časově náročné. Kromě toho je čas potřebný pro nalezení optimálního rozvrhu přímo závislý na množství vstupních údajů. Čím více proměnných vstupuje do modelu, tím více času bude potřeba věnovat řešení úlohy. Aby bylo možné rychle a bezchybně získávat optimální řešení rozvrhovacích úloh, zejména pokud se tato činnost opakuje, lze použitý algoritmus naprogramovat. K tomuto účelu slouží celá řada programů a softwarů, které nabízí vlastní programovací jazyky (Microsoft Excel).

Algoritmizací se rozumí postup, který se používá při tvorbě programu určeného k řešení nějakého problému prostřednictvím algoritmu (Bechyňová, 2012). Proces algoritmizace se skládá z několika etap: definice problému, analýza úlohy, tvorba algoritmu, sestavení programu a jeho odladění. První tři etapy byly rozebrány v předchozích kapitolách. Z představených úloh jsou vybrány tři modely s paralelně seřazenými procesory, jejichž algoritmy bude nezbytné naprogramovat ve zvoleném prostředí VBA for Excel.

## VBA for Excel

Visual Basic for Application (neboli VBA) je programovací jazyk, který vyvinula společnost Microsoft. Původně byl tento jazyk určen pro uživatele, kteří nemají hodně zkušenosti s programováním, a proto je často označován jako user-friendly, neboli uživatelsky přátelský. Další výhodou VBA je to, že tento jazyk je integrován do všech hlavních softwarů z řady MS Office (Word, Excel, Acces, PowerPoint, Outlook, FrontPage, InfoPath), jiných programů společnosti Microsoft (Vision, Microsoft Project) a také do velkého množství softwarů a aplikací od jiných společností (CorelDraw, AutoCAD apod.). Není se třeba stárat se o separátní prostředí jazyka a propojovat dva různé programy – kódy a algoritmy, naprogramované v VBA, lze spouštět rovnou z jednoho z výše zmíněných softwarů.

Kód, který je naprogramován v prostředí VBA, se nazývá makro. Po spuštění makra provede program výpočet podle zvoleného uživatelem algoritmu. Makro lze vyvolávat opakovaně – výsledky se budou měnit pouze při zadání různých vstupů. Jelikož VBA je součástí jádra MS Excel, lze kód psát přímo ve Visual Basic, který je integrován do tohoto programu. Kromě toho, lze kód nejenom psát, ale také ho nahrávat. K tomu slouží funkce makro recorderu.

## Programování algoritmů vybraných metod

Pro automatizaci procesu nalezení optimálního řešení rozvrhovacích úloh a tvorbu optimálního rozvrhu jsou zvoleny tři modely s paralelně řazenými procesory. Metody jejich řešení již byly představeny a popsány v předchozí kapitole. Jedná se o tyto modely:

- Model $P_m||F_{max}$,
- Model $P_m|prmp|F_{max}$,
- Model $P_m||\bar{F}$.

Ke každému z těchto modelů bude vytvořen kód, který bude moci uživatel využit k řešení konkrétní rozvrhovací úlohy.

# 3.1 Přípravné kroky pro programování algoritmů řešení rozvrhovacích úloh

Před zahájením psaní kódů algoritmů pro řešení rozvrhovacích úloh v prostředí VBA je nutné rozhodnout, jak budou tyto kódy zpracovávat zadané uživatelem vstupy a přeměňovat je na požadované výstupy. Proto je potřeba použít správné nástroje, které VBA nabízí, a připravit určité funkce a procedury, které budou využity v rámci realizace algoritmů.

## Pole dávek a procesorů

Vstupy představují pro uživatele proměnné, které mají pouze jeden atribut – dávku $D_j$, která má dobu trvání $t_j$. Samotné VBA nedovoluje deklarovat tyto vstupy pouze jako proměnné. Důvodem pro to je řada podmínek, které jsou „předepsány“ pro tvorbu grafu. Jedna z těchto podmínek je logické uspořádání dávek a procesorů při sestavení rozvrhu. Pokud by uživatel hledal optimální řešení manuálně, tzn. bez využití maker, dodržoval by při tvorbě rozvrhu následující logiku:

1. Pokud některé z dávek mají stejný čas na jejich zpracování na procesoru, tj. $t_j = t_{j+k}$, přednost z těchto dvou dávek má dávka s menším pořadovém číslem, tj. dávka $D_j$. Tato dávka bude zařazena na procesor $P_m$ jako první. Pak bude následovat dávka $D_{j+k}$.

2. Dávky se přiřazují na procesor podle pořadového čísla procesorů. To znamená, že by jako první byl využit procesor *P1*, pak *P2* atd.

Stejně jako pro vstupy, výstup algoritmu v podobě Ganttova diagramu se také opírá o více než jednu vlastnost procesorů, počítá tedy jak s dobou realizace obsazených dávek, tak i se správním pořadím, do kterého musí být tyto procesory ve finálním řešení uspořádány. Proto je nutné k proměnným dávek a procesorům dodatečné atributy, ke kterým se bude přihlížet při sestavení rozvrhu.

Vhodné je v tomto případě použít pole. „Pole je indexovaná skupina dat, která se chová jako jedna proměnná“ (Král, 2012, str. 29). Jinými slovy, pole je proměnná, která v prostředí VBA má více než jednu vlastnost. Tato pole budou definována jak pro dávky, tak i pro procesory. Poli dávek je pro účely programování přiřazen název „Tasks“. Schematické zobrazení podoby pole dávek „Tasks“ lze vidět v tabulce 12.

***Tab. 12 Pole dávek - Tasks***

| *Value* | *Index_Original* | *Index_Sort* | *Name* |
|---|---|---|---|
| $t_j$ | $j$ | | $D_j$ |
| $t_{j+1}$ | $j+1$ | | $D_{j+1}$ |
| … | … | | … |
| $t_n$ | $n$ | | $D_n$ |

Atributy prvního sloupce „Value“ představují doby realizace jednotlivých dávek, posledního sloupce „Name“ – název dávky. Tyto atributy se přenášejí z údajů, které zadá uživatel do aplikace. Sloupec „Index_Original“ se vyplní automaticky podle původního pořadí, do kterého jsou uspořádány dávky. Tento sloupec bude nezbytný v okamžiku, kdy algoritmus bude muset rozhodnout, kterou dávku z množství dávek se stejnou dobou realizace zařadí na procesor jako první. Sloupec „Index_Sort“ je využit pro modely, ve kterých jeden z kroků je uspořádání dávek dle doby realizace do klesající či rostoucí posloupnosti. Toto pole bude deklarováno v VBA následujícím kódem:

```
ReDim Tasks(1 To n, 1 To 4) As Variant,
```

kde počet sloupců je vždy konstantní a počet řádků je závislý na proměnné *n*, která je dána množstvím dávek v systému. Pole dávek „Tasks“ slouží pouze jako základ pro řešení algoritmu a nebude pro uživatele viditelné.

Dalším polem, které je potřeba deklarovat, je pole procesorů s názvem „Result“. Každý řádek tohoto pole představuje jednotlivý procesor, který bude mít *n+2* sloupců. Pro model se čtyřmi paralelně seřazenými procesory, na které je potřeba zařadit pět dávek, má v poli „Result“ čtyři řádky a 7 sloupců (viz tab. 13).

***Tab. 13 Pole procesorů „Result“ – část 1***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| | | | | | | *i* |
| | | | | | | *i+1* |
| | | | | | | … |
| | | | | | | *m* |

Na začátku algoritmu bude první *n* množství sloupců prázdné, neboť žádná z dávek ještě není zařazena na procesor. Postupně se budou obsazovat hodnotami doby realizace jednotlivých dávek podle toho, jaká dávka bude přiřazena na jaký procesor. Sloupec „Sum“ představuje celkovou dobu zpracování dávek na jednom procesoru. Tato hodnota se počítá průběžně a aktualizuje se po každé nově zařazené dávce. V modelech $P_m||F_{max}$ a $P_m||\bar{F}$ zařazení zbývajících dávek probíhá na základě toho, jaký procesor se uvolní jako první. Proto celková doba zpracování dávek na jednom procesoru a její průběžná aktualizace umožňuje algoritmu rozhodnout, na jaký procesor se zařadí další *j*-tá dávka.

Do sloupce „Index“ se zapíše pořadové číslo od 1 do *m* podle toho, jaký počet procesorů má konkrétní úloha. Na základě výše uvedeného lze pole procesorů „Result“ deklarovat v VBA následovně:

```
ReDim Result(1 To m, 1 To n + 2) As Variant,
```

kde počet řádků bude roven množství procesorů a počet sloupců se bude odvíjet od množství dávek v systému plus dva sloupce pro dodatečné atributy procesorů. Po ukončení algoritmu sestaví makro výsledný rozvrh podle hodnot z pole procesorů „Result“. Samotné pole zůstane pro uživatele neviditelné. Podrobněji

o interakci polí „Tasks“ a „Results“ a jejich roli v algoritmu, bude projednáno v další části této kapitoly.

### Procedura „sumResult“

Pro průběžnou sumarizaci celkové doby zpracování dávek na procesoru $P_m$ bude využita procedura, která je pojmenována jako „sumResult“. Tato procedura bude vyvolána algoritmem jak pro model $P_m||F_{max}$, tak i pro $P_m||\bar{F}$, neboť algoritmy těchto modelů obsahují podobné kroky. Z tohoto důvodu lze napsat proceduru „sumResult“ mimo oblast maker samotných algoritmů řešení a deklarovat ji jako věřejnou (Public), aby jiné moduly aplikace ji byly schopny „vidět“. Kód této procedury vypadá následovně:

```
Public Sub sumResult()
    Dim i As Double
    Dim j As Double

    For i = 1 To m
    Result(i, n + 1) = 0
        For j = 1 To n
            Result(i, n + 1) = Result(i, n + 1) + Result(i, j)
        Next j
    Next i
End Sub
```

V tomto kódu jsou deklarovány dvě proměnné. První proměnná *i* slouží k označení procesorů, resp. řádku pole procesorů „Result“ a proměnná *j* představuje *j*-tou dávku, která je umístěná na *i*-tém procesoru. Tato procedura je zacyklována a bude se opakovat, dokud nezpracuje všechny dostupné procesory a zařazené dávky.

### Procedura „ExcelSort“

Další společnou procedurou, kterou budou využívat jednotlivá makra algoritmů, je procedura „ExcelSort“, která seřazuje proměnné do zadané posloupnosti (rostoucí či klesající) na základě dvou či více kritérií. Jak již bylo zmíněno v předchozí části této podkapitoly, mohou nastat situace, kde dvě či více dávek budou mít stejnou dobu realizace a rozhodnutí, ve kterém pořadí budou dávky zařazeny na procesor, bude přijato na základě původní posloupnosti, do které byly dávky původně uživatelem zavedeny do systému. Proto je potřeba nakódovat

postup, kterého se bude algoritmus v takovém případě držet. Kód této procedury je uveden v příloze 1.

### Funkce „max“

Kromě sestavení optimálního rozvrhu, je za další výstup považována hodnota kritéria $F_{max}$, která se v modelu $P_m|prmp|F_{max}$ vypočítá dle vzorce (15) jako největší ze dvou hodnot. Samotný VBA nedisponuje integrovanou funkcí, která by uměla ze dvou hodnot definovat maximální hodnotu, proto je potřeba tuto jednoduchou funkci naprogramovat:

```
Function max(val1, val2)
    If val1 >= val2 Then
        max = val1
    Else
        max = val2
    End If
End Function
```

### Procedura „buildGraph“

Procedura „buildGraph“ je určena pro tvorbu rozvrhu v podobě Ganttova diagramu. Tato procedura bude opět společná pro všechny algoritmy, a proto je deklarována jako „veřejná“. Prvním krokem v dané proceduře bude vytvořit graf a odstranit všechny nepotřebné části grafu, které mohou vzniknout defaultně (například řádek nebo sloupec). Je nutné mít prázdný graf před tím, než se do něj začnou obsazovat jednotlivé dávky umístěné na procesorech. Toto je zajištěno následující částí kódu:

```
ActiveSheet.Shapes.AddChart2(297, xlBarStacked).Select
    Set ch = ActiveChart
    For Each srs In ch.FullSeriesCollection
        srs.Delete
    Next
```

Po provedení tohoto kroku lze přistoupit k nahrávání údajů jednotlivých řádků (procesorů). Tyto údaje jsou přebírány z pole procesorů „Result“. Správné pořadí zařazených dávek na procesory je zajištěno pomocí atributů „Index_Original“ a „Index_Sort“ z pole dávek „Tasks“. Jednotlivé dávky s jejich časem realizace jsou pak seřazeny v tomto pořadí na řádcích (procesorech). Kód pro nahrávání údajů

o dávkách je naprogramován takto:

```
For j = 1 To n
        ch.SeriesCollection.NewSeries
        ch.FullSeriesCollection(j).Name ="=""" & Tasks(j, 4)& """"
        rowData = "={"
        For i = 1 To m
            rowData = rowData & Result(i, Tasks(j, 2)) & ","
        Next i
        rowData = Left(rowData, Len(rowData) - 1) & "}"
        ch.FullSeriesCollection(j).Values = rowData
    Next j
```

Dalším krokem je definice popisku osy Y, na kterém jsou zobrazeny názvy jednotlivých procesorů (*P1*, *P2*, *P3*, …, *Pm*). Tyto popisky jsou také naprogramovány ve formě řádků. Část kódu, která definuje os Y, vypadá následujícím způsobem:

```
rowLabels = "={"
    For i = 1 To m
        rowLabels = rowLabels & """P" & i & ""","
    Next
    rowLabels = Left(rowLabels, Len(rowLabels) - 1) & "}"
    ch.FullSeriesCollection(1).XValues = rowLabels
```

Po vytvoření grafu s údaji o procesorech, na nichž jsou již zařazeny jednotlivé dávky, je nutné také provést grafické formátování diagramu. Tato nastavení se týkají určení polohy grafu na sešitu Excel, definice délky a šířky grafu, formátování jednotlivých řádků (procesorů) a bloků (dávek), názvu těchto bloků, písma, ohraničení, osy X a Y a dalších atributů. Kód pro grafickou úpravu je podrobně představen v příloze 2 včetně komentářů k jednotlivým částem kódu, označených apostrofem a zelenou barvou.

### Procedura „clean“

Procedura „clean“ slouží k odstranění výsledků předchozích výpočtů, tzn. odstranění již sestaveného rozvrhu a hodnoty účelové funkce předchozí úlohy, a také vstupních údajů o dávkách, které uživatel do modelu vložil. Tato procedura se skládá ze dvou částí – „clean1“ a „clean2“. První část „clean1“ slouží pouze

k odstranění grafu a hodnoty $F_{max}$, resp. $\bar{F}$. Tato část procedury je vyvolána každým algoritmem, aby nedocházelo k shromáždění výsledků předchozích výpočtů. Kód této části je zapsán v VBA následně:

```
Sub clean1()
    Dim ch As Object
    For Each ch In ActiveSheet.ChartObjects
        ch.Delete
    Next
    Cells(10, 6).Value = ""
End Sub
```

Druhá část procedury „clean2“ umožňuje odstranit údaje o dávkách. Tato část procedury není vyvolána algoritmem, ale může ji uživatel spustit sám. Může využít celou proceduru, pokud bude potřebovat odstranit všechny údaje z konkrétního modelu a začít řešit novou úlohu. Kód procedury „clean2“ vypadá následujícím způsobem:

```
Sub clean2(modeln As Integer)
    Range("TasksInput" & modeln).Cells(1, 2).Value = ""
    Range("TasksInput" & modeln).Cells(1, 1) = "=$B$9 & ROW()-9"
    If Range("TasksInput" & modeln).Rows.Count > 1 Then
        Range(Cells(11, 2), Cells(Range("TasksInput" &
modeln).Rows.Count + 9, 3)).clear
        ActiveSheet.ListObjects("TasksInput" & modeln).Resize
Range("$B$9:$C$10")
    End If
End Sub
```

Všechny procedury a funkce popsané v této podkapitole jsou nezbytné pro programování algoritmů rozvrhovacích úloh a jsou relevantní pro každý z modelů, proto se nacházejí v separátním modulu VBA této aplikace. Makra samotných algoritmů jednotlivých modelů mají vlastní moduly.

## 3.2 Programování algoritmu modelu $P_m||F_{max}$

První model, jehož algoritmus je naprogramován v prostředí VBA for Excel, je model $P_m||F_{max}$. Účelem tohoto modelu je minimalizovat nejdelší dobu pobytu dávky v systému ($F_{max}$). Algoritmus pro řešení této rozvrhovací úlohy již byl popsán v předchozí kapitole.

### Kontrola vstupních hodnot

Každý model začíná kontrolou přítomnosti vstupních údajů v programu. Jedná se o informace, které musí do modelu zadat uživatel. K tomu patří počet procesorů *m* a počet dávek *n*. Kontrola těchto údajů je definována pomocí následujícího kódu:

```
If Cells(6, 3).Value = "" Then MsgBox "Zadej počet procesorů":
Exit Sub

    m = Cells(6, 3).Value

    n = Range("tasksinput1").Rows.Count

    If n <= 1 Then MsgBox "Zadej informaci o davkach": Exit Sub

    If m > n Then MsgBox "Počet procesorů je větší než počet
dávek. Zkontroluj zadání": Exit Sub
```

V rámci realizaci tohoto kódu probíhá kontrola třech podmínek:

1. Vložil-li uživatel do modelu počet procesorů *m*.
2. Vložil-li uživatel údaje o dávkách (název dávek a doby jejich realizace).
3. Nepřesahuje-li počet procesorů *m* počet dávek *n*.

Pokud nebude alespoň jedna z těchto podmínek splněna, bude uživatel po spuštění makra informován o problému pomocí hlášky „MsgBox“ s příslušným komentářem. V případě, že jsou všechny podmínky splněny, přechází algoritmus k další etapě.

### Vyplnění polí „Tasks“ a „Results“

Po zjištění vstupních informací přistoupí makro k vyplnění hodnot v poli dávek „Tasks“ a přiřazení pořadových čísel procesorům v poli „Result“. Výsledkem této části makra jsou vyplněné hodnoty „Value“, „Index_Original“ a „Name“ v poli „Tasks“. Pro vysvětlení fungování polí v tomto algoritmu bude uveden následující jednoduchý příklad. K dispozici jsou dva paralelně seřazené procesory, na které je potřeba zařadit pět dávek. Informace o dávkách jsou již rovnou zapsány do pole dávek „Tasks“ (viz tab. 14).

*Tab. 14 Příklad 6 - část 1, pole dávek „Tasks“*

| Value | Index_Original | Index_Sort | Name |
|---|---|---|---|
| 7 | 1 | | D1 |
| 3 | 2 | | D2 |
| 4 | 3 | | D3 |
| 3 | 4 | | D4 |
| 1 | 5 | | D5 |

Buňky v sloupci „Index_Sort“ zůstanou prázdné, neboť není zatím definováno algoritmem, do jakého pořadí musí být dávky uspořádány. Zadání hodnot do pole „Tasks“ je tímto způsobem uskutečněno pomocí následujícího kódu:

```
For j = 1 To n
        Tasks(j, 1) = CDbl(Range("tasksinput1").Cells(i,
2).Value)
        Tasks(j, 4) = Range("tasksinput1").Cells(i, 1).Value
        Tasks(j, 2) = i
Next j

For i = 1 To m
        Result(i, n + 2) = i
Next
```

Poslední část kódu přiřadí pořadové číslo každému řádku, který představuje jednotlivé procesory, v poli „Result“. Výsledkem toho bude pole „Result“ (viz tab. 15), které je vyplněno podle údajů z výše uvedeného příkladu. Jelikož jsou v zadání k dispozici pouze dva procesory, bude mít pole procesorů „Result“ dva řádky.

*Tab. 15 Příklad 6 - část 2, pole procesorů „Result“*

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| | | | | | | 1 |
| | | | | | | 2 |

### Zařazení dávek na procesory

Další část kódu je zaměřena na řešení konkrétní úlohy tohoto modelu podle kroků heuristického algoritmu, popsaného v druhé kapitole. Prvním krokem je seřazení dávek podle dob realizace od největší hodnoty k nejmenší. Výsledkem tohoto seřazení jsou hodnoty ve sloupci „Index_Sort“, které představují nová pořadová čísla jednotlivých dávek. Zároveň bude makro v průběhu tohoto řazení brát v potaz původní pořadové číslo, které bylo *j*-té dávce přiděleno (sloupec „Index_Original“ v poli dávek „Tasks“). To znamená, že ze dvou či více dávek se stejnou dobou realizace nejmenší pořadové číslo bude nově přiděleno té dávce, kterou uživatel vložil do systému dříve, než ostatní dávky (Index_Sort $D_k$ < Index_Sort $D_{k+1}$). Seřazení dávek podle těchto dvou kritérií proběhne pomocí procedury „ExcelSort“, která se nachází v služebním modulu. Makro modelu $P_m||F_{max}$ vyvolá tuto proceduru a určí, jak budou dávky seřazeny. Kód pro tuto část algoritmu bude následující:

```
Call ExcelSort(Tasks, 1, xlDescending, 2, xlAscending)
    For i = 1 To n
        Tasks(i, 3) = i 'zapis hodnot dle serazeni
    Next i
```

V tomto případě se provede sestupné řazení (Descending) podle hodnot v prvním sloupci pole „Tasks“ a vzestupné řazení (Ascending) podle hodnot v druhém sloupci tohoto pole. Výsledné pořadí se propíše do sloupce „Index_Sort“ v poli dávek „Tasks“ (viz tab. 16).

***Tab. 16 Příklad 5 - část 3, pole dávek „Tasks“ s uspořádanými dávkami***

| *Value* | *Index_Original* | *Index_Sort* | *Name* |
|---|---|---|---|
| 7 | 1 | 1 | D1 |
| 3 | 2 | 3 | D2 |
| 4 | 3 | 2 | D3 |
| 3 | 4 | 4 | D4 |
| 1 | 5 | 5 | D5 |

Dalším krokem algoritmu je zařazení *m* počtu dávek na volné procesory v čase $t = 0$. Zde začne makro vyplňovat údaje do pole „Result“, neboť na základě těchto

informací bude sestaven graf. Hodnoty ze sloupce „Value“ pole „Tasks“ se přenesou do příslušných buněk pole „Reslut“. Makro ukončí tuto operaci v okamžiku, kdy poslední *m*-tý procesor bude obsazen *m*-tou dávkou. Tento krok je zajištěn krátkým kódem, který je zapsán níže:

```
For i = 1 To m
        Result(i, Tasks(i, 2)) = Tasks(i, 1)
    Next i
```

Pole procesorů „Result“ po zařazení prvních *m* dávek bude vypadat následujícím způsobem:

***Tab. 17 Příklad 5 - část 6, pole procesorů „Result“ po zařazení prvních m dávek***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| 7 | | | | | | 1 |
| | | 4 | | | | 2 |

Zařazení zbylých dávek na procesory probíhá dle algoritmu takto – každá další dávka je přiřazena na procesor, který se uvolní jako první. Proto je potřeba před každým dalším zařazení *j*-té dávky provádět zjištění celkové doby zpracování již zařazených dávek na jednotlivých procesorech. K tomu slouží sloupec „Sum“ v poli procesorů „Result“. Makro vyvolá proceduru „sumResult“, která zapíše zjištěné hodnoty do tohoto sloupce (viz tab. 18).

***Tab. 18 Příklad 5 – část 5, pole procesorů „Result“ s celkovou dobou zpracování dávek***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| 7 | | | | | 7 | 1 |
| | | 4 | | | 4 | 2 |

Na základě těchto zjištěných hodnot se pomocí procedury „ExcelSort“ provede řazení procesorů od nejkratší celkové doby zpracování dávek na *i*-tém procesoru do nejdelší. Kromě toho, musí být dodržena stejná podmínka, která platí pro řazení dávek – pokud dva či více procesorů budou mít stejné hodnoty v sloupci „Sum“, zvolí se jako první ten procesor, který má nejnižší hodnotu v sloupci „Index“, jedná

se tedy opět o uspořádání na základě dvou kritérií. Podle tohoto kroku se změní pořadí řádků v poli „Result“. Výsledek tohoto kroku je představen na příkladu 5 v tabulce 19.

***Tab. 19 Příklad 5 – část 6, pole procesorů „Result“ s uspořádanými procesory dle „Sum“***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| | | 4 | | | 4 | 2 |
| 7 | | | | | 7 | 1 |

Podle nového uspořádání procesorů je patrné, že další dávka *D3* bude přiřazena na procesor $P_2$ v čase 4. Pak následuje procedura pro výpočet hodnot v sloupci „Sum“ a následné uspořádání procesorů dle těchto hodnot. Tento krok se provádí, dokud nebude poslední dávka $D_n$ zařazena na *n*-tý procesor. Kód pro tuto část makra vypadá takto:

```
For i = m + 1 To n
        Call sumResult
        Call ExcelSort(Result, n + 1, xlAscending, n + 2, xlAscending)
        Result(1, Tasks(i, 2)) = Tasks(i, 1)
    Next i
    Call sumResult
    Call ExcelSort(Result, n + 2, xlAscending)
```

Poslední dva řádky kódu zajistí vrácení procesorů do jejich původní posloupnosti ($P_1$, $P_2$, …, $P_m$). Potom bude mít pole procesorů „Result“ podobu, zobrazenou v tabulce 20.

***Tab. 20 Příklad 5 – část 7, pole procesorů „Result“ po zařazení všech dávek***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| 7 | | | 3 | | 10 | 1 |
| | 3 | 4 | | 1 | 8 | 2 |

### Formátování pole „Result“

Před tím, než se spustí procedura kreslení diagramu „buildChart“, je nutné po každém algoritmu provést úpravy pole procesorů „Result“. První úprava se týká prázdných buněk. Jelikož každá dávka se zpracovává pouze na jednom procesoru, doba jejího trvání $t_j$ se může objevit pouze jednou v každém ze sloupců $D_1$ až $D_n$. Proto ostatní buňky zůstávají prázdné. Pokud budou ponechány v tomto stavu, VBA neumožní přejít k proceduře kreslení grafu, neboť se do prázdných buněk musí dosadit hodnota 0. Jedná se o nutnou podmínku VBA. Po tomto kroku bude pole procesorů „Result“ mít tuto podobu:

***Tab. 21 Příklad 5 - část 8, pole procesorů „Result“ po formátování***

| D1 | D2 | D3 | D4 | D5 | Sum | Index |
|---|---|---|---|---|---|---|
| 7 | 0 | 0 | 3 | 0 | 10 | 1 |
| 0 | 3 | 4 | 0 | 1 | 8 | 2 |

Další úpravou je výměna znaků, se kterými se sčítají hodnoty z pole „Result“. Defaultním nastavením je znak „čarka“, VBA ale požaduje znak „tečka“. Tyto dva kroky jsou realizovány pomocí následujícího kódu:

```
For i = 1 To m
        For j = 1 To n
            If Result(i, j) = "" Then Result(i, j) = 0
            Result(i, j) = Replace(Result(i, j), ",", ".")
        Next j
    Next i
```

### Výpočet hodnoty $F_{max}$

Hodnota $F_{max}$ v tomto modelu se počítá jako nejdelší celková doba trvání dávek na procesorech. Kód algoritmu vyhledá tuto hodnotu z pole „Result“ a zobrazí ji uživateli v aplikaci jako výsledek.

```
Fmax = Application.WorksheetFunction.max(Result)
    Cells(10, 6).Value = Fmax
```

## 3.3 Programování algoritmu modelu $P_m|prmp|F_{max}$

Dalším modelem, pro který je naprogramován algoritmus jeho řešení, je model $P_m|prmp|F_{max}$. Tento model, na rozdíl od předchozího, umožňuje přerušení dávek, které se zpracovávají na jednom procesoru a přenášení části dávky na jiný procesor. Část kódu, který sloužil ke kontrole vstupních údajů a vyplnění polí dávek a procesorů, je totožný a bude použit znovu pro toto makro.

### Zařazení dávek na procesor

Při hledání optimálního řešení této rozvrhovací úlohy se aplikuje McNaughtonův algoritmus, který se skládá ze třech kroků. První krok spočívá v zjištění hodnoty $F_{max}$ dle vzorce (15). To znamená, že optimální rozvrh bude mít nejdelší dobu pobytu dávky v systému jako maximum ze dvou hodnot – nejdelší doba zpracování *j*-té dávky nebo součet všech dob dávek dělený počtem procesorů. V této části makra pole dávek „Tasks“ již obsahuje vstupní data, proto lze vypočítat hodnotu $F_{max}$ pomocí tohoto kódu:

```
Fmax = Application.WorksheetFunction.RoundUp(max(TasksSum / m,
TasksMax), 0)
```

V této části kódu je využita procedura ze služebního modulu. Po zjištění hodnoty $F_{max}$ první dávka $D_1$ se zařadí na první procesor $P_1$ v čase $t = 0$. Další zařazení dávky je závislé na hodnotě $F_{max}$, která nesmí být překročena. Proto musí makro kontrolovat, zda po zařazení další dávky na současný procesor nebude překročena hodnota nejdelší doby pobytu dávky. V průběhu této kontroly mohou nastat tři situace:

1. Po zařazení dávky $D_j$ je celková doba zpracování již zařazených dávek na daný procesor menší než hodnota $F_{max}$. V tomto případě lze *j*-tou dávku zařadit na procesor a pokračovat dále.

2. Po zařazení dávky $D_j$ se celková doba zpracování již zařazených dávek na daný procesor rovná hodnotě $F_{max}$ – dávka bude zařazena na současný procesor a makro bude pak pokračovat s obsazením dalšího procesoru následujícími dávkami.

3. Po zařazení dávky $D_j$ je celková doba zpracování již zařazených dávek na daný procesor menší než hodnota $F_{max}$. Potom kód rozdělí dávku na dvě

části – první bude zařazena na současný procesor s délkou nepřesahující hodnotu $F_{max}$ po ukončení zpracování této dávky a druhá část bude převedena na další procesor.

Tyto kroky se opakují, dokud nebudou všechny dávky rozvrženy na procesory. Kód pro tento algoritmus je napsán následujícím způsobem:

```
i = 1
    For j = 1 To n
        If Result(i, n + 1) + Tasks(j, 1) < Fmax Then
            Result(i, j) = Tasks(j, 1)
        ElseIf Result(i, n + 1) + Tasks(j, 1) = Fmax Then
            Result(i, j) = Tasks(j, 1)
            i = i + 1
        ElseIf Result(i, n + 1) + Tasks(j, 1) > Fmax Then
            Result(i, j) = Fmax - Result(i, n + 1)
            Result(i + 1, j) = Tasks(j, 1) - Result(i, j)
            i = i + 1
        End If
        Call sumResult
    Next
```

## 3.4 Programování algoritmu modelu $P_m||\bar{F}$

Algoritmus modelu $P_m||\bar{F}$ je z hlediska programování velmi podobný heuristickému algoritmu, který se používá pro řešení úloh v modelu $P_m||F_{max}$. Hlavní rozdíl spočívá v uspořádání dávek, které pak budou zařazeny na procesory. Účelem tohoto modelu je minimalizace průměrné doby pobytu dávky v systému. Jednotlivé hodnoty $F_j$ se pak sčítají a dělí se celkovým počtem dávek. Minimální průměrnou dobu pobytu dávky v systému lze dosáhnout, pokud jako první budou zařazeny dávky s nejmenší dobou trvání. Odlišným bude část kódu prvního kroku algoritmu, kde dochází k řazení dávek podle dvou kritérii. V tomto algoritmu budou obě řazení vzestupná (Ascending).

```
Call ExcelSort(Tasks, 1, xlAscending, 2, xlAscending)
```

### Výpočet hodnoty $\overline{F}$

Průměrná doba pobytu dávky v systému se počítá v průběhu zařazení jednotlivých dávek na procesory speciálním kódem. Pro účely tohoto kódu je deklarována proměnná *Psum*, která představuje součet doby zpracování dávek na procesoru. Tato hodnota postupně sčítá časy dokončení zpracování dávek, které se potom dělí počtem dávek v systému *n*. Výsledek je pak zobrazen uživateli. Daný kód je napsán následujícím způsobem:

```
For i = 1 To m
        Psum = 0
        For j = 1 To n
            If Result(i, Tasks(j, 2)) <> 0 Then
                Favg = Favg + Psum + Result(i, Tasks(j, 2))
                Psum = Psum + Result(i, Tasks(j, 2))
            End If
        Next j
    Next i
    Favg = Favg / n
    Cells(10, 6).Value = Favg
```

# 4 Vytvoření finální aplikace pro rozhodování uživatele

V této části práce je představena finální aplikace, která bude sloužit uživateli pro řešení rozvrhovacích úloh vybraných modelů. V rámci samotné aplikace bude mít uživatel možnost zvolit, s jakým modelem potřebuje pracovat, a jakou úlohu bude muset makro řešit. Všechna makra (včetně služebního modulu) jsou integrována do jednoho souboru formátu Excel. Jednotlivé listy v tomto sešitu obsahují konkrétní modely s paralelně řazenými procesory.

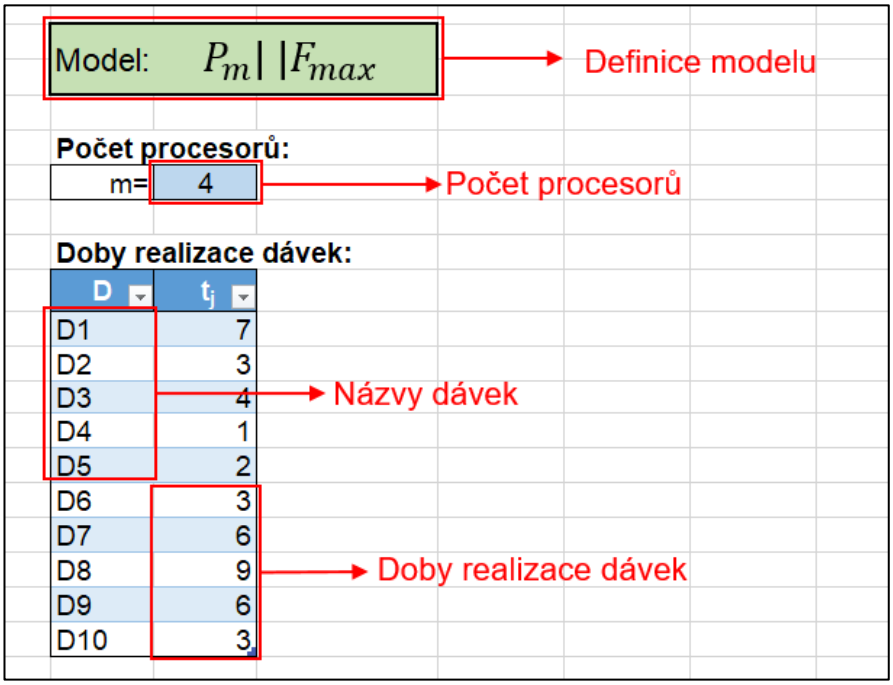
## 4.1 Návrh uživatelského prostředí aplikace

Pro snadnou orientaci uživatele musí být jasně definováno uživatelské prostředí aplikace. Jedná se o vzhled samotné aplikace, v níž bude uživatel pracovat. Uživatelské prostředí se skládá z různých tabulek, polí, tlačítek a ikon. Tyto nabídky jsou závislé na vstupech a výstupech každé úlohy.

Všechny modely budou mít stejné požadavky na vstupy, neboť pro řešení těchto úloh musí uživatel zadat následující údaje: počet procesorů, názvy dávek a doby realizace jednotlivých dávek. Tyto údaje se mohou lišit dle zadání úlohy a jsou nutné pro spuštění algoritmů řešení. Za výstup je považován rozvrh v podobě Ganttová diagramu, na kterém jsou jednotlivé dávky uspořádány na $m$ počtu procesorů. Na vertikální ose budou zobrazeny jednotlivé procesory, na horizontální ose čas, který bude odrážet doby pobytu dávek v systému. Další součástí výstupu je vypočtena hodnota kritéria účelové funkce. V případě těchto modelů se jedná o dvě hodnoty – $F_{max}$ a $\bar{F}$.

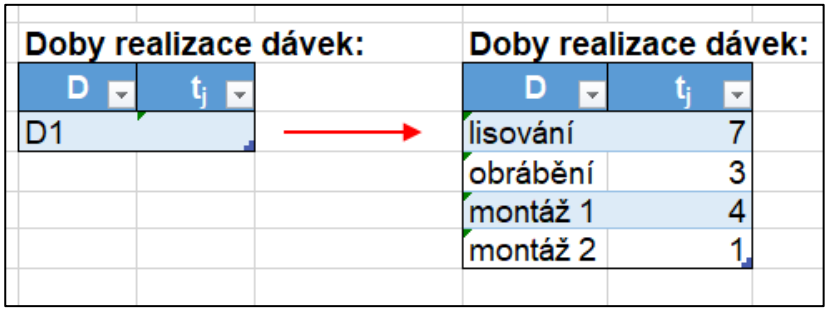
Na obrázku 14 je představena první část návrhu uživatelského prostředí. Pro orientaci uživatele, s jakým modelem momentálně pracuje, je v levém horním rohu uveden název modelu ve formátu notace $P_m \mid \beta \mid \gamma$. Dále má uživatel k dispozici pole, kam musí zadat počet procesorů konkretní úlohy.

Informace o dávkách se neomezuje pouze na jejich počet. Pro řešení rozvrhovacích úloh a sestavení rozvrhu jsou nezbytné další atributy jako doba realizace dávek $t_j$ a jejich názvy. Tyto údaje mohou být zaneseny do tabulky „Doby realizace dávek“.




***Obr. 14 Návrh uživatelského prostředí – zadání vstupů***

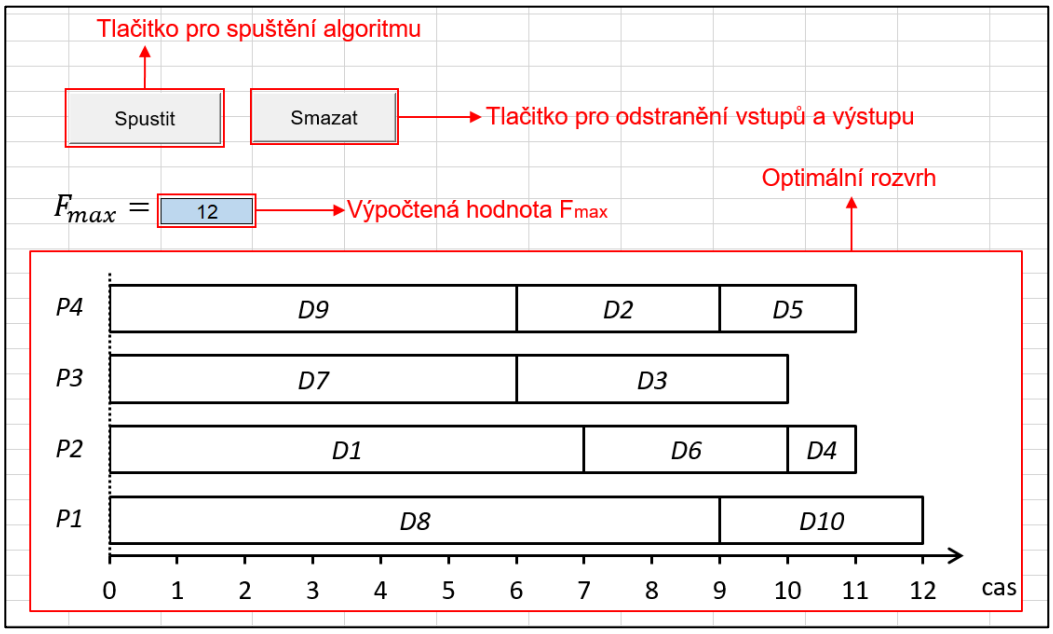
Pro zjednodušení používání tabulky byla zvolená tzv. „chytrá tabulka“ – typ tabulek v Excelu, který se automaticky formaáuje dle uživatelem zadaných či odebíraných dat. Tato tabulka má dva sloupce – název dávky „*D*“ a dobu realizace dávek „$t_j$“. Uživatel má připraven první řádek s prázdnou hodnotou doby realizace první dávky s defaultním názvem „*D1“*. Po vkládání hodnoty $t_j$ do další prázdné buňky v tomto sloupci se bude automaticky měnit ohraničení tabulky a bude se přidavat název ve formátu „*Dj*“. Názvy dávek lze následně upravovat dle potřeb uživatele (viz obr. 15).




***Obr. 15 Návrh uživatelského prostředí – „chytrá tabulka“***

Po zadání všech potřebných vstupních informací pro výpočet optimálního řešení bude uživatel moci spustit algoritmus přes tlačítko „Spustit“ (viz obr. 16). Tím se

spustí makro, které je předepsáno pro konkrétní model a po ukončení kódu se na stejném listu objeví požadované výstupy v podobě hodnoty účelové funkce a Ganttova diagramu. Podle potřeby uživatele lze pomocí tlačítka „Smazat“ odstranit všechny vstupní a výstupní informace, tzn. Ganttův diagram, hodnotu účelové funkce a informace o dávkách („chytrá tabulka“ se vrátí do původního defaultního stavu). Toto umožňuje uživateli přistoupit k řešení další úlohy bez nutnosti manuálního vymazávání vstupů a výsledků předchozí úlohy.

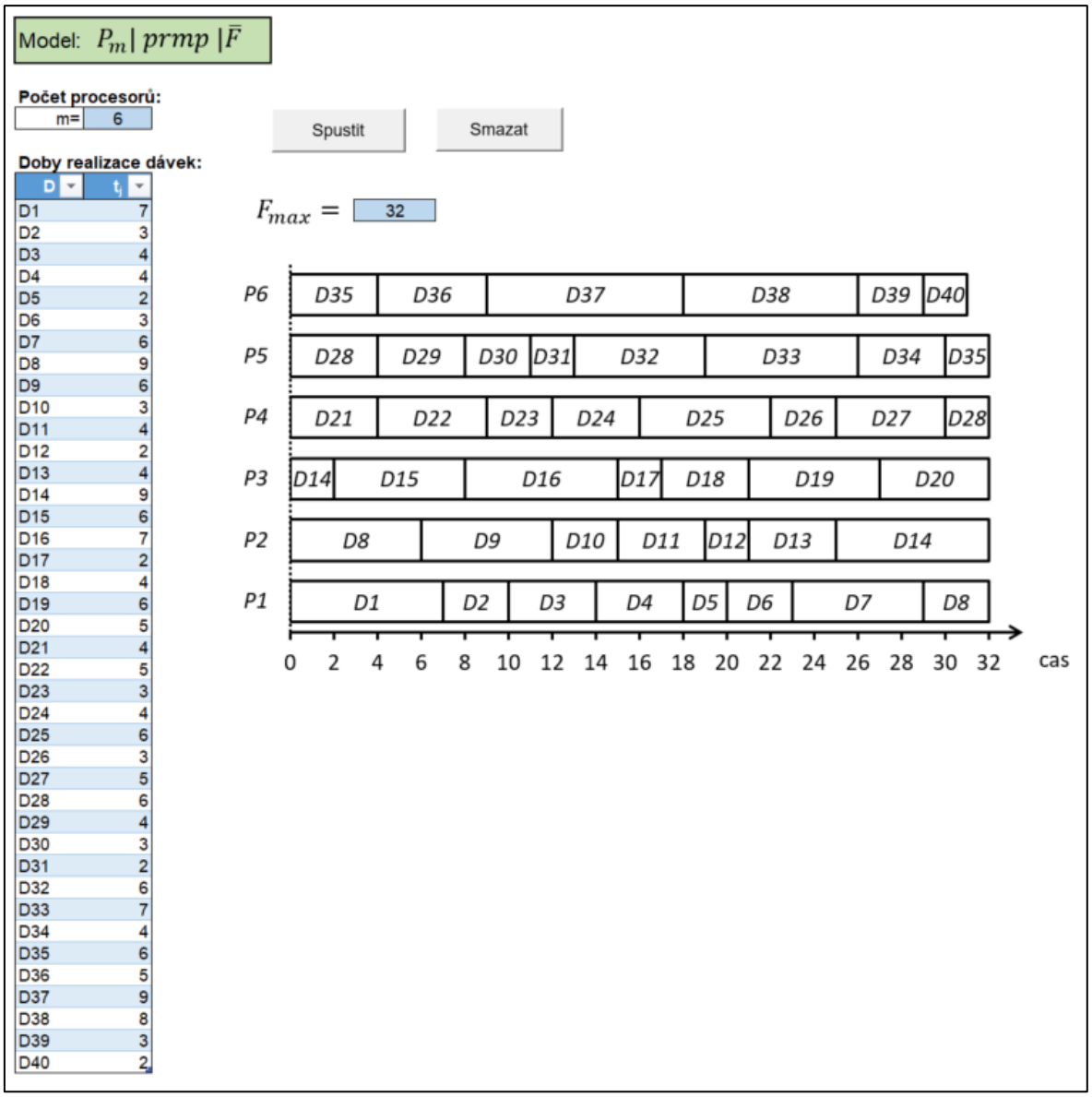



***Obr. 16 Návrh uživatelského prostředí – získání výstupů***

Toto uživatelské prostředí je stejné pro každý z třech vybraných modelů s paralelně seřazenými procesory.

## 4.2 Řešení vybraných rozvrhovacích úloh pomocí aplikace

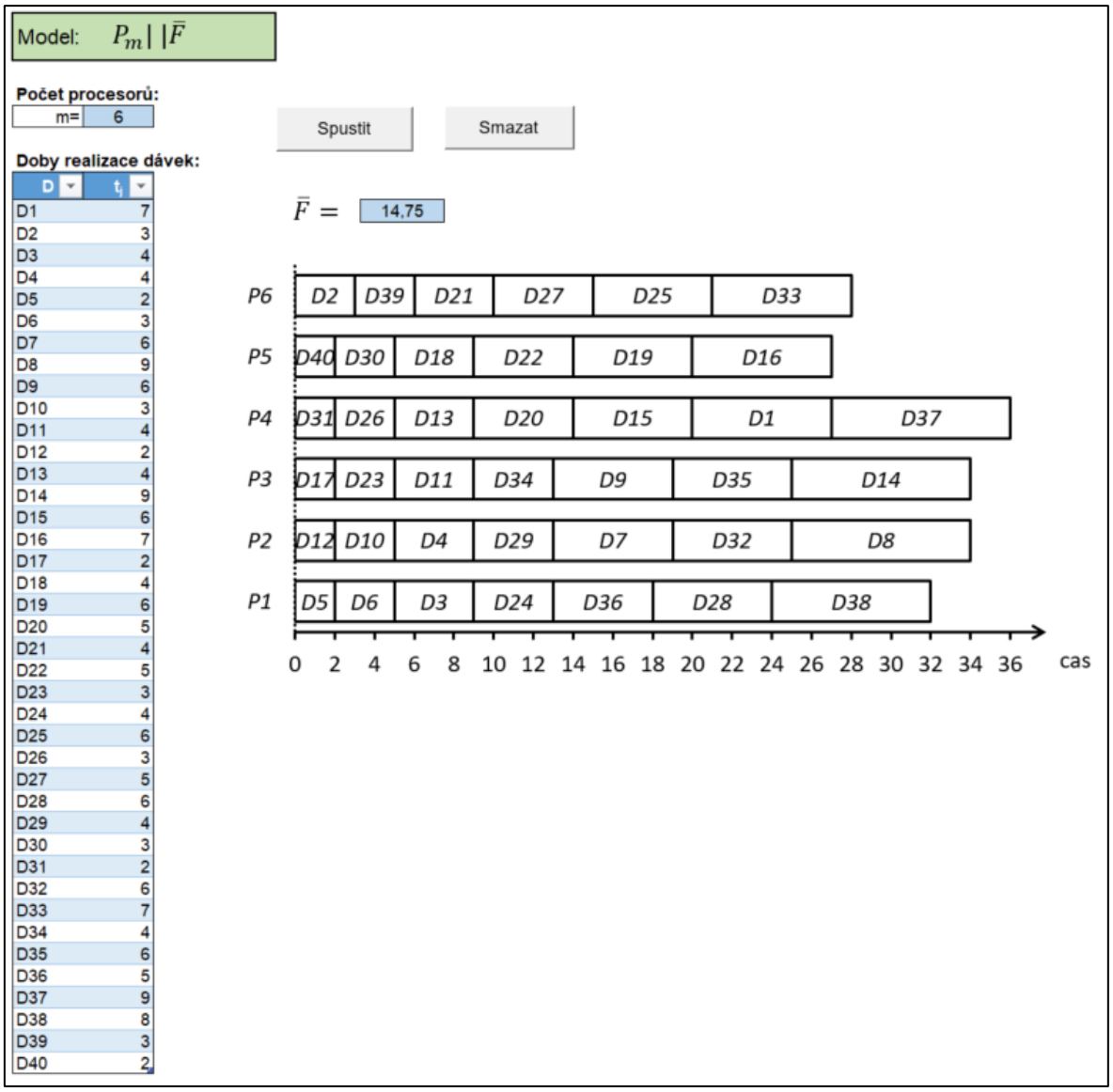
Jednou z hlavních výhod této aplikace oproti manuálnímu řešení rozvrhovacích úloh je přesnost a rychlost, se kterou dokáže makro vyřešit určitou úlohu a sestavit optimální nebo přípustný rozvrh. Toto platí hlavně pro úlohy, které obsahují velké množství vstupů. Pro účely představení výhod aplikace je dále rozebrán následující příklad. Existují šest procesorů, na které je třeba zařadit čtyřicet dávek. Doba trvání jednotlivých dávek je zadána do aplikace (viz obr. 17). Po spuštění příslušného makra pro model $P_m|prmp|F_{max}$ se automaticky provede řešení této úlohy, sestaví se Ganttův diagram a zobrazí se hodnota $F_{max}$.




| D | $t_j$ |
|---|---|
| D1 | 7 |
| D2 | 3 |
| D3 | 4 |
| D4 | 4 |
| D5 | 2 |
| D6 | 3 |
| D7 | 6 |
| D8 | 9 |
| D9 | 6 |
| D10 | 3 |
| D11 | 4 |
| D12 | 2 |
| D13 | 4 |
| D14 | 9 |
| D15 | 6 |
| D16 | 7 |
| D17 | 2 |
| D18 | 4 |
| D19 | 6 |
| D20 | 5 |
| D21 | 4 |
| D22 | 5 |
| D23 | 3 |
| D24 | 4 |
| D25 | 6 |
| D26 | 3 |
| D27 | 5 |
| D28 | 6 |
| D29 | 4 |
| D30 | 3 |
| D31 | 2 |
| D32 | 6 |
| D33 | 7 |
| D34 | 4 |
| D35 | 6 |
| D36 | 5 |
| D37 | 9 |
| D38 | 8 |
| D39 | 3 |
| D40 | 2 |

***Obr. 17 Řešení vybrané úlohy v modelu $P_m|prmp|F_{max}$***

Řešení této úlohy lze také provést v modelu $P_m||\bar{F}$. Finální výsledek zjištěný pomocí aplikace je představen na obrázku 18.




| D | $t_j$ |
|---|---|
| D1 | 7 |
| D2 | 3 |
| D3 | 4 |
| D4 | 4 |
| D5 | 2 |
| D6 | 3 |
| D7 | 6 |
| D8 | 9 |
| D9 | 6 |
| D10 | 3 |
| D11 | 4 |
| D12 | 2 |
| D13 | 4 |
| D14 | 9 |
| D15 | 6 |
| D16 | 7 |
| D17 | 2 |
| D18 | 4 |
| D19 | 6 |
| D20 | 5 |
| D21 | 4 |
| D22 | 5 |
| D23 | 3 |
| D24 | 4 |
| D25 | 6 |
| D26 | 3 |
| D27 | 5 |
| D28 | 6 |
| D29 | 4 |
| D30 | 3 |
| D31 | 2 |
| D32 | 6 |
| D33 | 7 |
| D34 | 4 |
| D35 | 6 |
| D36 | 5 |
| D37 | 9 |
| D38 | 8 |
| D39 | 3 |
| D40 | 2 |

***Obr. 18 Řešení vybrané úlohy v modelu $P_m||\bar{F}$***

# Závěr

Při zpracování této práce byly naprogramovány tři různé algoritmy pro řešení rozvrhovacích úloh modelů s paralelně řazenými procesory v prostředí VBA for Excel. Byly získány zkušenosti a rozšířeny znalosti programovacího jazyka VBA, které lze v budoucnu uplatnit při programování podobných algoritmů. Jednotlivé části kódu vyžadovaly důkladné zkoušení a kontrolu, což vedlo k lepšímu porozumění práce v daném prostředí a získání dalších znalostí o problematice rozvrhování v produkčním systému.

Vytvořená aplikace může být využita uživatelem pro řešení rozvrhovacích úloh a pomoci mu lépe rozumět fungování jednotlivých algoritmů. Tuto aplikaci lze také použit v rámci výuky předmětu „Modelování produkčních a logistických systémů“ v navazujícím magisterském studiu na vysoké škole ŠKODA AUTO VYSOKÁ ŠKOLA o.p.s.

Po vytvoření aplikace byla jednotlivá marka odzkoušena na řadě rozvrhovacích úloh a prokázalo se jejich bezpochybné sestavení. Způsob jejich programování umožňuje použít je do budoucna jako základ pro algoritmizaci dalších modelů s paralelně řazenými procesory a tím rozšířit nabídku algoritmů pro rozhodování uživatele.

## Seznam literatury


BLAZEWICZ, Jacek, Klaus H. ECKER, Günter SCHMIDT a Jan WEGLARZ. *Scheduling in Computer and Manufacturing Systems*. 2nd Edition. Springer-Verlag Berlin Heidelberg, 1994. ISBN 978-3-642-79036-2.

BRUCKER, Petr. *Scheduling Algorithms*. 5th Edition. Leipzig: Springer-Verlag Berlin Heidelberg, 2007. ISBN 978-3-540-69515-8.

FÁBRY, Jan. *Modelování produkčních a logistických systémů pro prezenční a kombinovanou formu studia*. Mladá Boleslav: ŠKODA AUTO VYSOKÁ ŠKOLA, o.p.s., 2019. ISBN 978-80-87042-85-4.

FIALA, Petr. *Modely produkčních systémů*. 2. vyd. Praha: Oeconomica, 2013. ISBN 978-80-245-1966-1.

KRÁL, Martin. *Excel VBA: výukový kurz*. Brno: Computer Press, 2010. ISBN 978-80-251-2358-4.

LEUNG, Josef Y-T. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. Florida: CRC Press, 2004. ISBN 1-58488-397-9.

PINEDO, Michael L. *Scheduling: Theory, Algorithms, and Systems*. 5th Edition. Springer, 2016. ISBN 978-3-319-26578-0.

*Stránky k výuce informatiky* [online]. Vlašim: Ing. Marta Bechyňová, 2012 [2020-11-14]. Dostupné z: http://www.ivt.mzf.cz/algoritmizace-a-programovani/uvod-do-algoritmu/2-algoritmizace/

# Seznam obrázků a tabulek

## Seznam obrázků

## Seznam tabulek

## Seznam příloh

## Příloha 1 Kód procedury „ExcelSort“

```
Sub ExcelSort(vArray As Variant, sortColumn1 As Long, Order1,
Optional sortColumn2 As Long, Optional Order2, Optional
sortColumn3 As Long, Optional Order3, Optional sortColumn4 As
Long, Optional Order4, Optional sortColumn5 As Long, Optional
Order5)
'rostouci posloupnost: xlAscending, klesajici posloupnost:
xlDescending
'serazeni pres Excel na zaklade dvou hodnot - doba realizace
davek a puvodniho poradi davek

    Dim ws As Worksheet
    Dim R As Range

    Set ws = ThisWorkbook.Worksheets.Add
    Set R = Range("A1").Resize(UBound(vArray) -
LBound(vArray) + 1, UBound(vArray, 2) - LBound(vArray, 2) +
1)
    R.Value = vArray

    ws.Sort.SortFields.clear
                              ws.Sort.SortFields.Add Key:=R(1,
sortColumn1), Order:=Order1
    If sortColumn2 <> 0 Then ws.Sort.SortFields.Add Key:=R(1,
sortColumn2), Order:=Order2
    If sortColumn3 <> 0 Then ws.Sort.SortFields.Add Key:=R(1,
sortColumn3), Order:=Order3
    If sortColumn4 <> 0 Then ws.Sort.SortFields.Add Key:=R(1,
sortColumn4), Order:=Order4
    If sortColumn5 <> 0 Then ws.Sort.SortFields.Add Key:=R(1,
sortColumn5), Order:=Order5

    ws.Sort.SetRange R
    ws.Sort.Header = xlNo
    ws.Sort.Apply

    vArray = R

    ws.Delete

End Sub
```

## Příloha 2 Kód pro grafickou úpravu grafu

```
    'urceni polohy grafu na sesitu
    ch.Parent.Left = 172
    ch.Parent.Top = 146
    'sirka je konstantni, delka grafu zavisla na poctu
procesoru
    ch.Parent.Height = 30 + m * 38
    ch.Parent.Width = 500
    'Odstraneni ohraniceni, legendy a nazvu
    ch.Parent.ShapeRange.Line.Visible = msoFalse
    ch.SetElement (msoElementLegendNone)
    ch.Axes(xlValue).MajorGridlines.Delete
    ch.HasTitle = False
    'urceni tloustky jednotlivych radku
    ch.ChartGroups(1).GapWidth = 50

    'formatování jednotlivych bloku (davek) na diagramu
    For Each srs In ch.FullSeriesCollection
        'pridani a nastaveni popisku zobrazenych udaju na
blokech (pouze nazev jednotlivych bloku ve forme "Dn")
        srs.ApplyDataLabels
        srs.DataLabels.ShowSeriesName = True
        srs.DataLabels.ShowValue = False
        'pridani a formatovani ohraniceni, nastaveni pisma
nazvu bloku
        For Each pnt In srs.Points
            pnt.Format.Fill.Visible = msoFalse
            pnt.Format.Line.Visible = msoTrue
            pnt.Format.Line.ForeColor.RGB = RGB(0, 0, 0)
            pnt.Format.Line.Weight = 1.5

pnt.DataLabel.Format.TextFrame2.TextRange.Font.size = 14

pnt.DataLabel.Format.TextFrame2.TextRange.Font.Italic =
msoTrue

pnt.DataLabel.Format.TextFrame2.TextRange.Font.Fill.ForeColor
.RGB = RGB(0, 0, 0)
        Next pnt
        'odstraneni vsech popisku pro "nulove" bloky
        For i = 1 To UBound(srs.Values)
            If srs.Values(i) = 0 Then
srs.Points(i).DataLabel.Delete
        Next
    Next srs

    'Osa Y: nastaveni formatu pisma - velikost, kurziva,
cerna barva
    ch.Axes(xlCategory).TickLabels.Font.size = 14
```

```
    ch.Axes(xlCategory).TickLabels.Font.Italic = True
    ch.Axes(xlCategory).TickLabels.Font.Color = RGB(0, 0, 0)
    'Osa X: nastaveni formatu pisma - velikost, kurziva,
cerna barva
    ch.Axes(xlValue).TickLabels.Font.Color = RGB(0, 0, 0)
    ch.Axes(xlValue).TickLabels.Font.size = 14

    'Minimalni hodnota osy X je vzdy 0, maximalni vzdy vetsi
nez hodnota Fmax
    ch.Axes(xlValue).MinimumScale = 0
    ch.Axes(xlValue).MaximumScale = Fmax * 1.05

    'Definovani rozdeleni osy X v zavislosti od hodnoty Fmax
    If Fmax > 40 And Fmax <= 100 Then
ch.Axes(xlValue).MajorUnit = 5
    If Fmax > 20 And Fmax <= 40 Then
ch.Axes(xlValue).MajorUnit = 2
    If Fmax > 5 And Fmax <= 20 Then
ch.Axes(xlValue).MajorUnit = 1
    If Fmax > 1.5 And Fmax <= 5 Then
ch.Axes(xlValue).MajorUnit = 0.5
    If Fmax > 0.5 And Fmax <= 1.5 Then
ch.Axes(xlValue).MajorUnit = 0.1

    'Osa X: tloustka cary, barva, deleni, sipka na konci
    ch.Axes(xlValue).Format.Line.Weight = 1.5
    ch.Axes(xlValue).Format.Line.Visible = True
    ch.Axes(xlValue).MajorTickMark = xlOutside
    ch.Axes(xlValue).Format.Line.EndArrowheadStyle =
msoArrowheadOpen
    'Osa Y: tloustka cary, barba
    ch.Axes(xlCategory).Format.Line.Visible = True
    ch.Axes(xlCategory).Format.Line.Weight = 1.5
    ch.Axes(xlCategory).Format.Line.ForeColor.RGB = RGB(0, 0,
0)
    ch.Axes(xlCategory).Format.Line.DashStyle = msoLineSysDot

    'Nazev osy X: "cas", velikost pisma, barva, poloha v
zavislosti od hodnoty Fmax
    ch.Axes(xlValue).HasTitle = True
    ch.Axes(xlValue).AxisTitle.Characters.Text = "cas"
    ch.Axes(xlValue).AxisTitle.Font.size = 14
    ch.Axes(xlValue).AxisTitle.Font.Color = RGB(0, 0, 0)
    ch.Axes(xlValue).AxisTitle.Left = 475
    ch.Axes(xlValue).AxisTitle.Top = m * 38
    'Uprava velikosti oblasti grafu (pridani nazvu osy X
zpusobi zmenu oblasti)
    ch.PlotArea.Height = ch.Parent.Height - 15

End Sub
```

# ANOTAČNÍ ZÁZNAM

| | | | |
|---|---|---|---|
| **AUTOR** | **Bc. Roman Matern** | | |
| **STUDIJNÍ PROGRAM/OBOR/SPECIALIZACE** | **6208R088 Podniková ekonomika a management provozu** | | |
| **NÁZEV PRÁCE** | **Algoritmy rozvrhování produkce na paralelní procesory v prostředí VBA for Excel** | | |
| **VEDOUCÍ PRÁCE** | **doc. Ing. Jan Fábry, Ph.D** | | |
| **KATEDRA** | **KRVLK - Katedra řízení výroby, logistiky a kvality** | **ROK ODEVZDÁNÍ** | **2020** |

| | |
|---|---|
| **POČET STRAN** | **63** |
| **POČET OBRÁZKŮ** | **18** |
| **POČET TABULEK** | **21** |
| **POČET PŘÍLOH** | **2** |

| | |
|---|---|
| **STRUČNÝ POPIS** | Diplomová práce je zaměřena na problematiku rozvrhovacích úloh v produkčních systémech. Cílem práce je zpracovat v prostředí VBA for Excel optimalizační algoritmy i heuristické metody pro řešení základních typů rozvrhovacích úloh v produkčních systémech.<br>Pomocí softwaru Microsoft Excel byla v prostředí VBA for Excel naprogramována makra pro vybrané modely s paralelními procesory pro účely rozhodování uživatele této aplikace.<br>Na základě vytvořených maker lze dále vytvořit kódy pro řešení dalších typů rozvrhovacích úloh. |
| **KLÍČOVÁ SLOVA** | **Produkční systémy, rozvrhovací úlohy, paralelní procesory, VBA for Excel, makro** |

# ANNOTATION

| | | | |
|---|---|---|---|
| **AUTHOR** | **Bc. Roman Matern** | | |
| **FIELD** | **6208T088 Business Administration and Operations** | | |
| **THESIS TITLE** | **Scheduling algorithms in production systems for parallel machines in VBA for Excel environment** | | |
| **SUPERVISOR** | **doc. Ing. Jan Fábry, Ph.D** | | |
| **DEPARTMENT** | **KRVLK - Department of Production, Logistics and Quality Management** | **YEAR** | **2020** |

| | |
|---|---|
| **NUMBER OF PAGES** | **63** |
| **NUMBER OF PICTURES** | **18** |
| **NUMBER OF TABLES** | **21** |
| **NUMBER OF APPENDICES** | **2** |

| | |
|---|---|
| **SUMMARY** | These diploma thesis is focused on scheduling tasks in production systems. The aim of this thesis is to compose optimization algorithms and heuristic methods in VBA for Excel environment for solving basic types of scheduling tasks in production systems.<br>By using the software Microsoft Excel macros for selected models with parallel machines were programmed in VBA for Excel environment with for user's decisions.<br>On the basis of programmed macros it is possible to create codes for solving of further types of scheduling tasks. |
| **KEY WORDS** | **Production systems, scheduling, parallel machines, VBA for Excel, macro** |