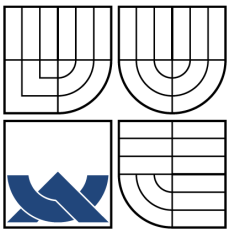
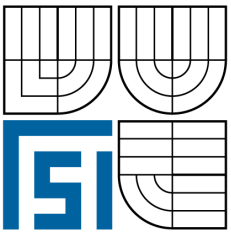


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

# OPTIMALIZAČNÍ MODELY S APLIKACEMI V ORGANIZACI VÝROBY

OPTIMIZATION MODELS IN PRODUCTION LOGISTICS

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR
TOMÁŠ MAUDER

VEDOUCÍ PRÁCE
SUPERVISOR
RNDr. PAVEL POPELA, Ph.D.

BRNO 2008

**Abstrakt**
Diplomová práce se zabývá lineární celočíselnou optimalizací v problémech organizace výroby pomocí matematického programování. Tento nástroj je použit pro hledání optimálních rozvrhů výroby pro průchod více zakázek firmou s omezenými zdroji. Práce řeší problém ve spojení s firmou TOSHULIN, a.s., která projevila o řešení tohoto problému zájem. Výsledkem je potom softwarové řešení, jejímž výstupem je tvorba Ganttova diagramu.

**Summary**
The diploma thesis deals with linear integer optimization in production logistics via mathematical programming. This tool is used for optimization of the time production schedule with a number of various jobs performed by a company with limited resources. The thesis solves the problem in conjunction with TOSHULIN, a.s. company, which is interested in solving the problem. As a result is the software implementation in which Gantt chart is created as its output.

**Klíčová slova**
lineární optimalizace, celočíselná optimalizace, Ganttovy diagramy

**Keywords**
linear optimization, integer optimization, Gantt chart

MAUDER, T. *Optimalizační modely s aplikacemi v organizaci výroby.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 67 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D.

Prohlašuji, že jsem diplomovou práci *Optimalizační modely s aplikacemi v organizaci výroby* vypracoval samostatně pod vedením RNDr. Pavla Popely, Ph.D., s použitím materiálů uvedených v seznamu literatury.

Tomáš Mauder

Děkuji svému školiteli panu RNDr. Pavlu Popelovi, Ph.D. za vedení, odbornou pomoc při vytváření diplomové práce a za pomoc při sestavování programových implementací modelu. Dále děkuji panu Doc. Dr. Ing. Jiřímu Markovi za zasvěcení do problému. Na závěr chci také poděkovat svým rodičům za finanční a morální podporu po čas studia a také přátelům, kteří mi v průběhu studia pomáhali.

Tomáš Mauder

# Obsah

# Úvod

Tato práce se zaměřuje na řešení reálného problému firmy TOSHULIN, a.s. pomocí matematického programování. Jde o problémy souběžného průchodu více zakázek firmou s omezenými zdroji. Podle povahy problému jsou použity matematické modely lineárního celočíselného programování. Cílem práce je vytvoření matematického modelu popisující reálný problém, programová implementace pomocí modelovacího jazyku GAMS, původní vizualizace výstupních dat pomocí Ganttových diagramů v MS EXCEL. Také je kladen důraz na přátelské uživatelské rozhraní pomocí programovacího jazyku Microsoft Visual Basic, a pomocí vzájemného propojení programů GAMS a MS EXCEL. Tento text se skládá ze šesti kapitol, jejichž skladbu určuje povaha problému a vymezení cílů.

První kapitola krátce popisuje historii a současnost firmy TOSHULIN, a.s. Obsahuje formulaci reálného problému a potřebu řešení na matematické úrovni. Dále popisuje historii, podstatu Ganttových diagramů a jejich hojné použití v průmyslové výrobě. Závěr kapitoly je věnován tvorbě Ganttových diagramů v prostředí tabulkového procesoru MS EXCEL za pomoci programovacího jazyku Microsoft Visual Basic.

Druhá kapitola zařazuje problém mezi problémy operačního výzkumu, konkrétně jako problém plánování a rozvrhování výroby. Díky tomu je zde uvedena krátká historie operačního výzkumu a plánování a rozvrhování výroby. V dalším kapitola vymezí základní pojmy, které jsou použity v celé práci. Stručně ještě uvede matematické programování jako jednu z metod řešení operačního výzkumu, kterou používá tato práce.

V kapitole třetí formulujeme problém průchodu jedné zakázky firmou. V úvodu kapitoly uvedeme modelový příklad, na kterém bude budována matematická teorie. Pomocí poznatků z teorie grafů zavedeme pojem analýza kritické cesty. Budeme hledat řešení tohoto problému pomocí lineárního programování. Vytvoříme obecný matematický model a ukážeme si jeho implementaci v programu GAMS. Vytvoříme propojení programu GAMS a MS EXCEL a z vypočtených dat sestrojíme Ganttův diagram.

Čtvrtá kapitola rozšiřuje problém na průchod dvou zakázek firmou. Na modelovém příkladě ukážeme matematickou teorii. Matematický model rozšíříme o indikátorovou proměnnou a tím se dostaneme na problém celočíselného matematického programování. Při jeho tvoření ukážeme postupy, které zachovají linearitu modelu. Zároveň si řekneme jak rozšířit GAMS model a kapitolu uzavře výsledný Ganttův diagram.

V páté kapitole ukážeme, jak bude vypadat situace, kdy zakázek procházejících firmou je libovolně mnoho. Opět v úvodu kapitoly ukážeme modelový příklad. Následuje úprava matematického modelu a GAMS modelu, který příklad spočítá. Kromě toho budeme hodnotit výsledky použití různých účelových funkcí. Poslední část kapitoly ukáže finální softwarové zpracování problému a popíše jeho používání.

Závěrečná kapitola obsahuje pár možných směrů vylepšení modelu pro jeho rozšíření a přiblížení k realitě.

# Kapitola 1

# Firma TOSHULIN, a.s., vybrané problémy

## 1.1. Firma TOSHULIN, a.s.

Tradice firmy sahá k roku 1949, kdy byla zahájena výstavba strojírenského závodu ve městě Hulíně, který se nachází ve východní části České republiky. Po roce 1951 firma zaměřila svoji činnost na výrobu obráběcích strojů, zejména svislých soustruhů obrázek (1.2). Firma začala jako jedna z prvních na světě vyrábět generaci strojů s plynulými posuvy, NC pravoúhlým řízením a kopírováním. Osvědčené svislé soustruhy s automatickou výměnou nástrojů z patnáctipolohového zásobníku byly vyráběny již v roce 1974. Za dobu své existence dodala firma přes 13 000 obráběcích strojů do 58 zemí světa a jejím cílem je i nadále zachovat a rozvíjet spolupráci se zahraničními prodejci a zákazníky. Společnost TOSHULIN, a.s. dnes vyrábí osm základních typů svislých soustruhů a také patří mezi přední světové výrobce svislých soustruhů. V současné době zaměstnává TOSHULIN, a.s. 370 odborníků a naplánovaná roční výroba je kolem 40-ti nových strojů. Více o firmě lze nalézt na internetových stránkách firmy http://www.toshulin.cz.

Obrázek 1.1: *TOSHULIN, a.s.*

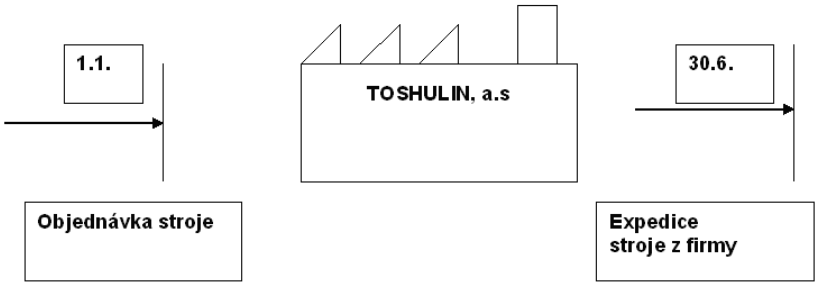
Tak jako každá firma, musí i firma TOSHULIN, a.s. modernizovat svoji výrobu a zavádět nové inženýrské postupy, aby co nejlépe uspokojila přání a požadavky zákazníků. Proto TOSHULIN, a.s. své postupy inovuje a spolupracuje s vysokými školami, mezi které patří i Vysoké učení technické v Brně. Můžeme například citovat článkny pana Doc. Dr. Ing. Jiřího Marka technického ředitele firmy TOSHULIN, a.s. [25],[26].

Obrázek 1.2: *Vybrané druhy soustruhů*

## 1.2. Formulace problému

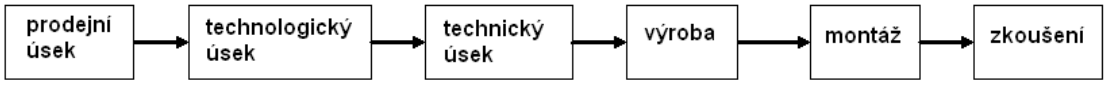
TOSHULIN, a.s. vyrábí v režimu "Projekt na zakázku". Ačkoliv je produkce založena na standardních návrzích a komponentách, stroje jsou často upravovány konkrétním zákaznickým potřebám. Někdy se stane, že při podepsání smlouvy nahlásí zákazník pouze několik požadavků s detailními specifikacemi, které později rozpracuje ve spolupráci s konstrukčním oddělením TOSHULIN, a.s. V důsledku toho se pracovní zatížení v podniku hodně mění. Jak již bylo řečeno, firmou prochází ročně kolem 40-ti nových strojů, které se rozdělují do osmi základních typů. Každý typ má svá výrobní specifika, jak v oblasti časové náročnosti, tak v oblasti použitého materiálu apod. Také každou z těchto zakázek si můžeme představit jako soubor komponent, z nichž se některé mohou vyrábět paralelně, další musí na svoji výrobu čekat až budou jiné kompletní. Komponenty můžeme rozdělit do dvou skupin, na ty u kterých je dokumentace známá a na ty, které si zákazník upravil podle potřeby a kde dokumentaci musí firma vyrobit. Takovýchto komponent může být i kolem dvě stě u jedné zakázky. Navíc firma musí stanovit přesné datum expedice stroje z firmy již ve fázi objednávky stroje, neboť překročení stanoveného data způsobuje nepříjemnosti jak firmě, která přichází o zisk spojený s opožděním zakázky, tak i zákazníkovi, který na stroj čeká a může tak přijít o svůj případný zisk jeho používáním.

Když se podrobněji podíváme na průchod jedné zakázky, například která byla objednána k 1.1. a její expedice z firmy je naplánována k 30.6.,



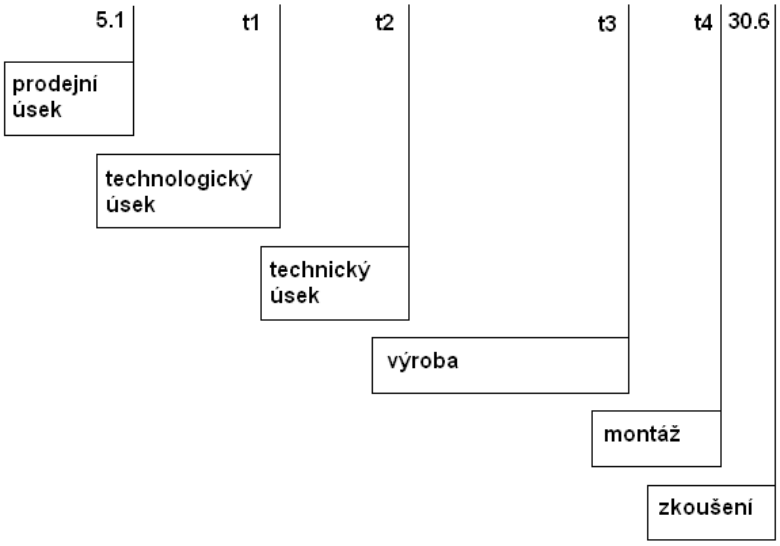

Obrázek 1.3: *Objednávka a expedice stroje z firmy TOSHULIN, a.s.*

musí taková zakázka projít fázemi jako je prodejní úsek, technologický úsek, technický úsek(konstrukce), výrobou, montáží a zkoušením viz obrázek (1.4).




Obrázek 1.4: *Blokové schéma průchodu zakázky firmou*

Každá z těchto fází potřebuje časový vymezený úsek závislý na druhu stroje. V plánu se pak tyto úseky vzájemně časově překrývají z důvodu nejistoty doby trvání v jednotlivých úsecích viz obrázek (1.5).




Obrázek 1.5: *Časové rozvržení zakázky*

Každý z těchto úseků je navíc kapacitně omezen, ať už lidskou silou, nebo výrobními prostory. Kdyby tyto úseky takto omezeny nebyly, mohli bychom vyrábět všechny aktuální zakázky paralelně a problém by byl vyřešen. Na druhou stranu náklady na kapacitu oddělení by přerostly zisk ze zakázky. Proto při souběžném průchodu více zakázek firmou ve stejném období, musí být rozhodnuto o vhodném pořadí. S těmito problémy se nesetkává pouze firma TOSHULIN, a.s., jde o obecně rozšířený problém mnoha firem. V kapitole 2 budeme tento problém klasifikovat a v dalších kapitolách řešit.

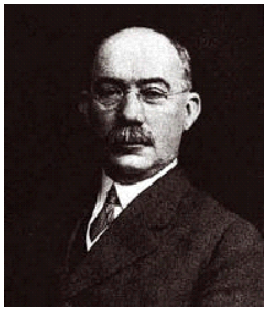
## 1.3. Matematické modely

Dosud všechny tyto problémy řešil lidský faktor na základě historických firemních dat. Pan Doc. Dr. Ing. Jiří Marek technický ředitel firmy TOSHULIN, a.s. navázal spolupráci s výzkumným Centrem pro jakost a spolehlivost výroby Doc. RNDr. Zdeňka Karpíška, CSc. na Fakultě strojního inženýrství Vysokého učení technického v Brně. Jeho přání je zmatematizovat právě ten lidský faktor, který rozhoduje o plánu a rozvrhu výroby. Diplomová práce je součástí řešení projektu MŠMT České republiky čís. 1M06047 Centrum pro jakost a spolehlivost výroby a projektu GA ČR reg. čís. 103/08/1658.

Práce se zabývá řešením problémů plánování a rozvrhování výroby pomocí matematických modelů. K řešením těchto modelů používá počítačový modelovací jazyk GAMS. Pro firmu je též důležité typické grafické zpracování v podobě Ganttova diagramu. Proto si ukážeme, jak lze tento diagram sestrojit v MS EXCELu pomocí programovacího jazyka Microsoft Visual Basic. Také pomocí tohoto jazyka a GAMS utility GDXXRW ukážeme propojení programu MS EXCEL a programu GAMS.

## 1.4. Řešení a jeho vizualizace pomocí Ganttových diagramů

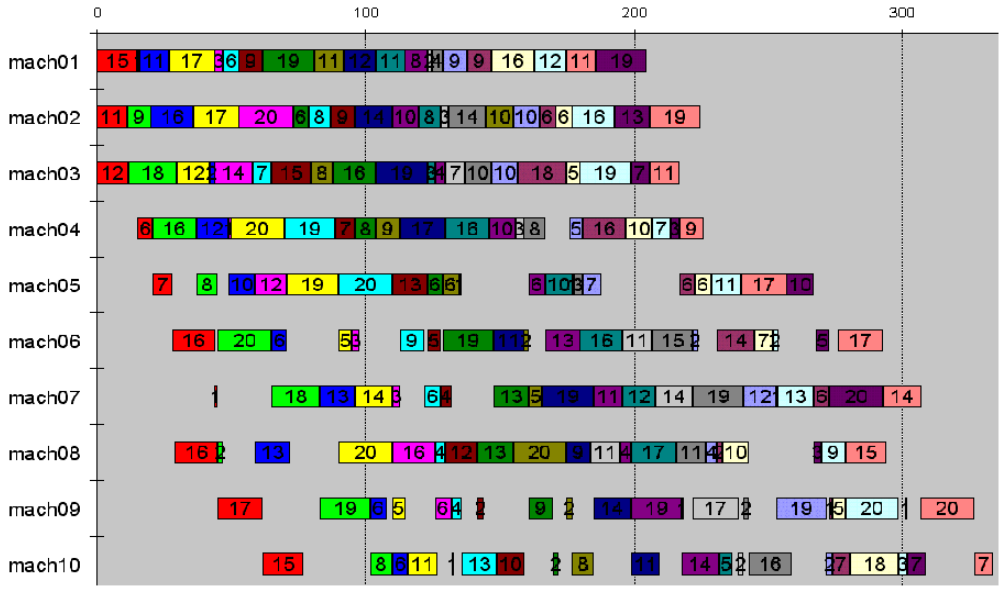
Protože Ganttovy diagramy jsou grafickým řešením kterého chceme dosáhnout a budou na nich v celé práci ukazovány modelové příklady, řekneme si o jejich historii a použití už nyní. Matematický model nám může poskytnout informaci o optimálním časovém výsledku a o jednotlivých dobách dokončení všech zakázek a pořadí jejich zpracování. Kromě těchto číselných údajů by byla vhodná i jejich vizualizace. V praxi se pro oblast plánování a rozhodování používají tzv. *Ganttovy diagramy (angl. Gantt chart)*, podle Henryho Laurence Gantta (1861 - 1990) strojního inženýra, který tento diagram rozvíjel a používal už kolem roku 1910 na základě analýzy pracovních postupů v průmyslové výrobě. Více v [15]. Na jeho počest se také každoročně uděluje cena Henryho Laurence Gantta za významný úspěch v průmyslu a službu veřejnosti.



Obrázek 1.6: *Henry Laurence Gantt*

S vývojem moderních nástrojů pro plánování a řízení projektů došlo k účelnému rozšíření tohoto původně pruhového diagramu zejména pro různé prezentace síťových grafů a hierarchických datových struktur.
Ganttův diagram umožňuje přehledně prezentovat aktuální stav na projektu, směrný a aktuální plán, zejména údaje časového rozvrhu, práce, nákladů, financování a zisku na projektu. K aktualizaci a prezentaci závislostí mezi úkoly poskytuje Ganttův diagram strukturu na časové stupnici (ose), zejména ke znázornění důležitých termínů. Prezentace souhrnných úkolů mohou poskytovat požadované sumarizace hodnot směrných, aktuálních a současných plánovaných ukazatelů podle reálné situace projektu. Ukázka Ganttova diagramu je na obrázku (1.7).

Na vertikální ose jsou tedy znázorněny jednotlivé stroje (mach01 - mach10 z angl. machine), kterými musí zakázka projít a na vertikální ose čas. Z diagramu je dobře vidět, v jakém časovém období budou jednotlivé stroje (zdroje) v nečinném stavu, a kdy naopak budou zakázky zpracovávat. Snadno vidíme doby dokončení jednotlivých zakázek a jejich rozvržení. Podle takovéhoto diagramu potom můžeme například plánovat pracovní směny, náklady na provoz strojů, plán jejich údržby apod.
Hlavní výhodou Ganttova diagramu je tedy přehlednost projektových ukazatelů na časové ose, a přehlednost hierarchické struktury projektu. Proto také v současné době Ganttův diagram patří k nejpoužívanějším formám prezentace projektových modelů pro plánování a řízení rozsáhlých projektů.




Obrázek 1.7: *Ganttův diagram pro dvacet zakázek a deset strojů*

Na diagramu z obrázku (1.7) je dobře vidět doba dokončení zakázek, která je v tomto případě po dokončení zpracování zakázky na stroji mach10. Hodnoty v obdélnících znamenají čas zpracování zakázky na jednotlivých strojích. Například první dokončená zakázka je tedy zakázka, která je označena červeně.

## 1.5. Implementace v MS EXCEL

Protože MS EXCEL neposkytuje ve svých možnostech tvorbu Ganttových diagramů, ukážeme, jak lze sestrojit tento diagram pomocí makra, které vytvoříme v programovacím jazyce Microsoft Visual Basic (VBA). Použijeme k tomu skládaný pruhový graf, na jehož výrobu budeme potřebovat data poskládat do speciální tabulky, kterou vytvoříme opět pomocí VBA. Makro rozčleníme podle průběhu operací a ukážeme části kódu, které jednotlivé operace umožňují. Celý kód je potom v příloze (B.2.1). Na začátku máme tabulku dat, například pro průchod dvou zakázek pěti stroji

| machine | Star time | End time |
|---|---|---|
| mach01 | 12 | 23 |
| mach02 | 26 | 35 |
| mach03 | 35 | 44 |
| mach04 | 44 | 49 |
| mach05 | 49 | 54 |
| mach01 | 0 | 12 |
| mach02 | 12 | 13 |
| mach03 | 13 | 25 |
| mach04 | 25 | 38 |
| mach05 | 38 | 49 |

Tabulka 1.1: Tabukla časů zpracování zakázek

Makro rozčleníme podle průběhu operací

1. uspořádá vstupní data podle čísla stroje a podle času začátku a konce aktivit,

| machine | Star time | End time |
|---|---|---|
| mach01 | 0 | 12 |
| mach01 | 12 | 23 |
| mach02 | 12 | 13 |
| mach02 | 26 | 35 |
| mach03 | 13 | 25 |
| mach03 | 35 | 44 |
| mach04 | 25 | 33 |
| mach04 | 44 | 49 |
| mach05 | 39 | 49 |
| mach05 | 49 | 54 |

Tabulka 1.2: Setříděná tabukla časů zpracování zakázek

```
'\ {sort data}
   With Worksheets("input").Range("U3").CurrentRegion    '{oblast dat}
       .Sort Key1:="Machine", Key2:="Start Time", Header:=xlYes
       vTimeData = .Value
                       '{uspořádání podle čísla stroje a podle aktivit}
   End With
```

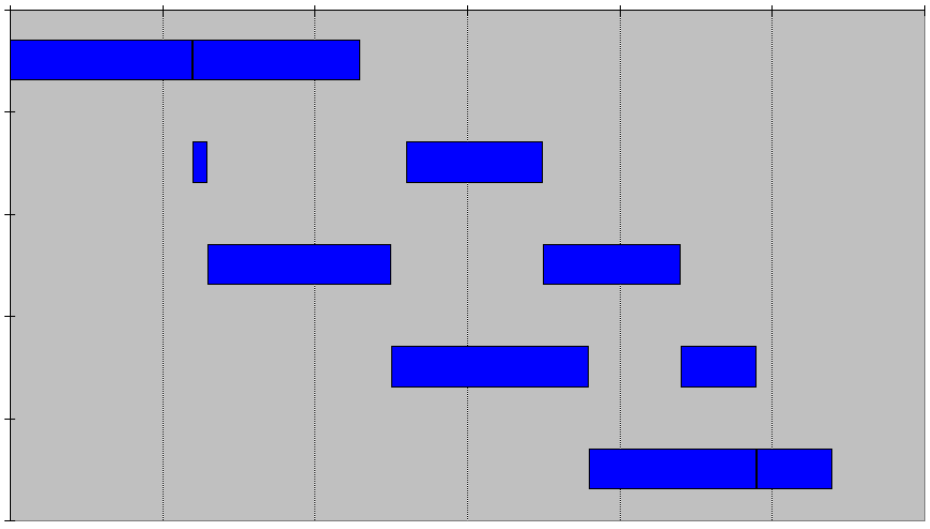
2. vytvoří tabulku dat pro tvorbu diagramu, kde první sloupec obsahuje stroje a v dalších se střídají sloupce činnosti a nečinnosti stroje,

| machine | Star time | Used | Not Used | Used |
|---|---|---|---|---|
| mach01 | 0 | 12 | 0 | 11 |
| mach05 | 12 | 1 | 13 | 9 |
| mach03 | 13 | 12 | 10 | 9 |
| mach04 | 25 | 13 | 6 | 5 |
| mach05 | 38 | 11 | 0 | 5 |

```
'\ {create chart data worksheet}
    With Worksheets("input").Range("O18").CurrentRegion '{oblast dat}
        vTimeData = .Value
        Worksheets.Add
        On Error Resume Next
            Worksheets("ChartData").Delete   '{smazání posledně vytvořenýh dat}
            Charts("TimeChart").Delete       '{smazání diagramu}
        On Error GoTo 0
        ActiveSheet.Name = "ChartData"   '{vytvoření sešitu kde bude
                                          vytvořena tabulka}
        .Columns(1).AdvancedFilter Action:=xlFilterCopy, _
            CopyToRange:=Range("A1"), Unique:=True
    End With
    Range("a1").Select
    For i = 2 To UBound(vTimeData)                  '{zapsání dat do tabulky}
```

```
    If vTimeData(i, 1) <> Selection.EntireRow.Cells(1) Then
        Selection.Offset(1).EntireRow.Cells(1).Select
        Selection.Value = vTimeData(i, 1)
        vLastEndTime = 0
    End If
    Selection.Offset(0, 1).Select
    Selection.Value = vTimeData(i, 2) - vLastEndTime
    Selection.Offset(0, 1).Select
    Selection.Value = vTimeData(i, 3) - vTimeData(i, 2)
    vLastEndTime = vTimeData(i, 3)
Next i
With Selection.CurrentRegion                    '{pojmenování sloupců}
    .Offset(0, 1).NumberFormat = ""
    .Columns(2).Cells(1) = "Start Time"
    For i = 3 To .Columns.Count
        If i Mod 2 <> 0 Then
            .Columns(i).Cells(1) = "Used"
        Else
            .Columns(i).Cells(1) = "Not Used"
        End If
    Next i
End With
```

3. vytvoření diagramu, kde jsou vymazány řady ze sloupců, které byly označeny jako nečinné,

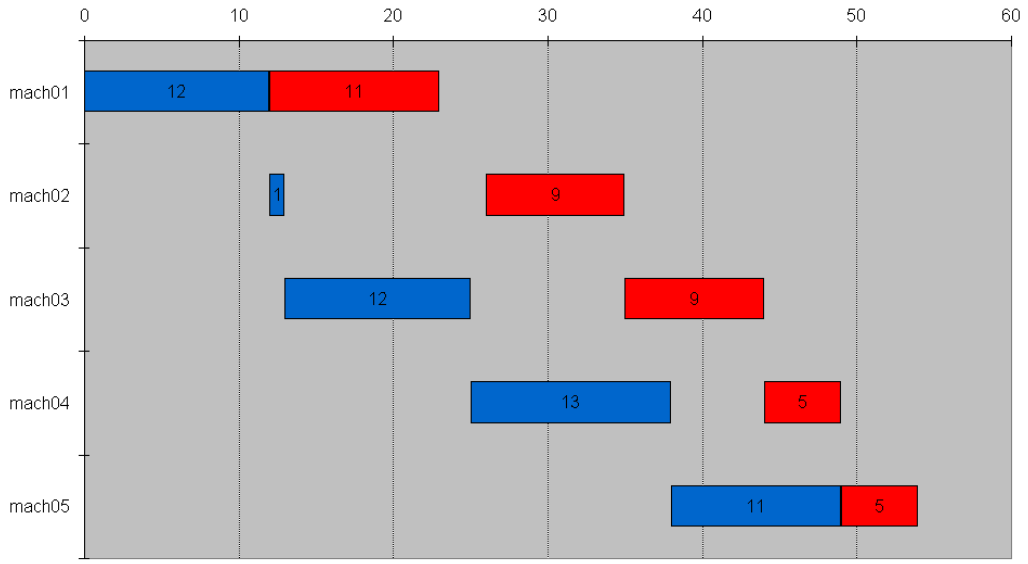


Obrázek 1.8: *Diagram*

```
'\ {create Gantt chart(from chart data)}
Charts.Add                                      '{vytvoření diagramu}
ActiveChart.Name = "TimeChart"
ActiveChart.ChartWizard Source:=Sheets("chartdata")
    .Range("A1").CurrentRegion, _
    Gallery:=xlBar, Format:=3, PlotBy:=xlColumns, CategoryLabels _
    :=1, SeriesLabels:=1, HasLegend:=2          '{tvar nastavení diagramu}

'\ {box properties}
```

```
    For Each oSeries In ActiveChart.SeriesCollection
        If oSeries.PlotOrder Mod 2 <> 0 Then        '{odstranění řad
z~oSeries.Border.LineStyle = xlNone        nečinných sloupců}
            oSeries.Interior.ColorIndex = xlNone
        Else
            oSeries.Border.LineStyle = 1        '{základní barva a ohraničení}
            oSeries.Border.ColorIndex = 1
        End If

    Next oSeries
```

4. časový popis jednotlivých úseků, jejich barevné označení a nastavení os.




Obrázek 1.9: *Kompletní diagram*

```
'\ box description
    ActiveChart.ApplyDataLabels AutoText:=True, LegendKey:=False, _
        HasLeaderLines:=False, ShowSeriesName:=False, ShowCategoryName:=False,_
        ShowValue:=True, ShowPercentage:=False, ShowBubbleSize:=False
      With Selection.Font                    '{popsání pruhů vdiagramu hodnotami
        .Name = "Arial"                       tedy dobou trvání operace}
        .Size = 12
      End With

    cc = Sheets("input").Range("B4").FormulaR1C1     '{načtení počtu strojů}
    bb = Sheets("input").Range("B3").FormulaR1C1     '{načtení počtu zakázek}
    For j = 0 To bb - 1
     ActiveChart.SeriesCollection(2 * j + 1).DataLabels.Select
     Selection.Delete
    Next j
     Dim jj As Integer

'\ {box color}
    For jj = 1 To bb
     For j = 1 To cc
    ActiveChart.SeriesCollection(2 * jj).Points(j).Select '{načtení hodnoty
```

```
   Selection.Shadow = False                                  barvy pro konkrétní
   Selection.InvertIfNegative = False                        pruh}
      With Selection.Interior
         .ColorIndex = Sheets("input")
         .Range("W" + CStr(-bb + 3 + bb * j + jj)).FormulaR1C1
      End With                 '{hodnota barvy musí být zapsána ve sloupci W}
     Next j
   Next jj

'\ {Axes seting}
  With ActiveChart.Axes(xlValue)                        '{základní nastavení os}
       .MajorUnitIsAuto = True
       .TickLabels.NumberFormat = ""
       .HasMajorGridlines = True
        If .HasMajorGridlines Then
           .MajorGridlines.Border.LineStyle = 6
        End If
  End With
```

Výsledkem je tedy grafické zpracování dat z tabulky (1.1). Zatímco tato data nebyla pro orientaci příliš vhodná, grafická podoba je názorná a lehce čitelná. Přesně z ní dokážeme určit, kdy bude daný stroj zpracovávat jakou zakázku, a kdy bude v nečinném stavu.

Kompletní zdrojový kód je v příloze číslo (B.2.1). Další modifikace lze sestrojit pomocí [12],[13],[14].

Už víme, jak vytvořit Ganttův diagram. Otázka ale zní, jak získat data v úvodní tabulce, tedy data, pro přípustný a dokonce optimální rozvrh. Tím se budeme zabývat v následujících kapitolách.

# Kapitola 2

# Modely operačního výzkumu pro plánování a rozvrhování

Kapitola 1.2 popisovala reálný problém firmy TOSHULIN, a.s. Zatím bylo řečeno pouze to, že se jedná o velmi komplexní problém, proto se nyní pokusíme tento problém přesně klasifikovat a vymezit cíle této práce. Máme zakázku na výrobu stroje, která se skládá z předem známého počtu komponent, ze kterých je tento stroj složen. Tyto komponenty jsou buď známé, ve smyslu dokumentace a výrobních časů, nebo neznámé, kde není vyhotovena dokumentace a výrobní časy se pouze odhadují na základě analogie z předešlých let. Výroba části komponent probíhá paralelně, kde tyto komponenty vyrábíme ve stejném čase, další část komponent musí počkat na výrobu předchozích. Skutečnost návaznosti operací výroby jednotlivých komponent bereme jako známou a počítáme s ní. Takových zakázek máme v jednom časovém období více. Navíc máme omezené zdroje, jako jsou lidská síla, výrobní prostory apod., které znemožňují výrobu všech aktuálních zakázek ve stejném čase. Proto chápeme řešením problému optimální plán rozvrhu výroby, tedy plán, při jehož plnění dosáhneme nejlepších možných dob dokončení všech zakázek. Jde tedy o problém plánování výroby různě časově náročných zakázek za případné časové nejistoty.

Díky komplexnosti tohoto problému se práce soustředí na deterministicky zadané výrobní časy, tedy na deterministické modely, jejichž rozšíření může vést k požadovanému cíli. Díky tomu nebude použito reálných, ale pouze modelových dat, na kterých bude demonstrováno řešení problému. Tato problematika podle literatury a prostředků řešení patří do metod operačního výzkumu, přesněji do oblasti plánování a rozvrhování (angl. Scheduling problem) viz [5],[6]. Proto si připomeneme stručnou historii.

Plánování a rozvrhování spadá pod metody operačního výzkumu. *Operační výzkum* je vědecká disciplína zabývající se analýzou operací spjatých s řízením, fungováním a navrhováním složitých společensko-ekonomicko-technických systémů. Příkladem takového systému může být průmyslový podnik, kterýkoliv jeho provoz, systémy dopravy apod.

První práce, které by se daly zařadit do operačního výzkumu, se vyskytly už v roce 1909 a pocházely od dánského matematika Agnera Krarupa Erlanga (1878 - 1929). Byly zaměřeny na teorii hromadné obsluhy. Na ně potom ve dvacátých letech navazovaly práce různých autorů, kteří aplikovali matematiku na oblast řízení zásob. Na ně začíná navazovat tvorba dalších metod s uplatněním na vojenský výzkum britské armády. Zakladatelem tohoto výzkumu byl Robert Watson-Watt (1892 - 1973) ze skupiny vědců a technologů, zabývající se rozvojem radaru. Právě tato skupina poprvé v roce 1940 použila pro svoji

činnost název *Operační výzkum*. První výsledky se tehdy uplatnily při plánování britských odvetných náletů na Německo, rozmístnění protivzdušné obrany Anglie, při výpočtech rozložení ponorkového loďstva, optimální velikosti ochranného doprovodu železničních transportů a optimálního sledu prací při kladení min.

Obrázek 2.1: *vlevo Agner Krarup Erlang a vpravo Robert Watson-Watt*

V současné době se operační výzkum používá i v řízení státní správy, k řešení vztahu mezi ekonomickým růstem a kvalitou životního prostředí, a při pomoci rozvojovým zemím. Dostupnost metod operačního výzkumu roste zvláště v posledním desetiletí s rozvojem výpočetní a sdělovací techniky, což je spjato s tvorbou *systémů na podporu rozhodování*. Ty se např. v průmyslových podnicích používají při strategickém i operativním plánování, ve všech fázích projektového řízení, při řízení výrobních procesů apod. viz [1].

Problémy operačního výzkumu, které se dají klasifikovat jako *problémy rozvrhování*, chápeme nejčastěji ve smyslu rozvrhování výrobního procesu. Typickými plánovacími problémy bývají například plánování výroby a výrobních postupů v továrnách, plánování rozvrhů pracovních směn, tvorba pracovních rozvrhů, plánování využití specializovaných zařízení, plánování provozu na letištích, plánování složitých výpočtů a experimentů, nebo plánováním úloh na počítačových systémech. Teorie plánování a rozvrhování definované pomocí analýzy kritické cesty se začíná vyvíjet od šedesátých let dvacátého století. V 1956-1957 začali James Kelly a Morgan Walker vývojem algoritmů a metodologií v rozvrhování. Více z historie plánování a rozvrhování lze nalézt v ([27]).

## 2.1. Vymezení pojmů

Obecně se dá říci, že plánování a rozvrhování jsou procesy zabývající se rozvržením úloh nebo činností v čase, jejichž cílem je použitím matematických technik a heuristických metod docílit rozvržení potřebných úloh na *limitované zdroje*. Toto rozvržení pak většinou sleduje nějaký záměr, jehož účelem bývá minimalizace nákladů, nebo maximalizace zisku. Nejčastěji bývá snaha o minimalizaci nákladů časových, finančních, materiálních, energetických či lidských. Ziskem pak můžeme rozumět zvýšení finančního profitu, zkvalitnění produkce, vyšší produktivitu práce, lepší ekologické parametry a účinnější využití surovin, lepší informace atd. Pojmy plánování a rozvrhování se často zaměňují, proto uvádíme jejich obecné vymezení.

- Úloha je objekt, který plánujeme. Úloha je charakterizována svými vlastnostmi a vnitřní strukturou. Operace nebo také podúloha je dílčí částí celé úlohy. Úloha

se skládá z jedné nebo více operací, jež mohou být prováděny na jednom nebo více zdrojích. Ve výrobě je potom úloha často nazývána zakázkou.

- Zdroj zpracovává nebo provádí jednotlivé operace, případně je prostředkem k realizaci operace. Zdroje bývají v systému omezené.

- Plánování je dlouhodobý proces vytváření množiny vhodných aktivit, aby bylo dosaženo stanovených cílů.

- Rozvrhování je alokace zdrojů umístěných v časoprostoru za daných podmínek na objekty (úlohy) tak, aby se splnila zadaná kritéria, například minimalizace celkové ceny daných zdrojů. Důraz je kladen na uspořádání objektu v čase. Rozvrh je výstupem procesu rozvrhování. Jedná se o datovou strukturu obsahující informace o umístění objektu (úloh) v čase na zdrojích.

Tyto a další pojmy lze nalézt v [7].

## 2.2. Struktura úlohy, omezení úloh a charakteristika zdrojů

Struktura, parametry a omezení úloh mohou být různé. Některé úlohy nebo jejich operace musí být provedeny v určitém pořadí v závislosti na daných precedenčních podmínkách. Tyto podmínky mohou vytvářet lineární posloupnost operací, stromovou strukturu, obecný orientovaný acyklický graf nazývaný graf precedencí. Plánovací proces musí tuto skutečnost respektovat a správně směřovat jednotlivé operace úlohy ze zdroje na zdroj. V našem případě navíc mluvíme o *multioperačních úlohách*, tedy úlohách, které běží postupně na více zdrojích. Typickým rozdělením v rozvrhovacích problémech jsou úlohy *preemptivní (přerušitelné)*, které lze během jejich zpracování na zdroji přerušit a poté opět po nějaké době spustit, a úlohy *nepreemptivní (nepřerušitelné)*, které nelze přerušit poté, co je spuštěno jejich provádění a musí doběhnout až do konce. Tato práce se zabývá nepreemptivními úlohami.

Z vymezení pojmu zdroje je zřejmé, že se jedná o dosti obecný pojem. Vzhledem k zaměření této diplomové práce bude v dalším textu zdroj představovat nějakou množinu strojů, které slouží ke zpracovávání úloh nebo poskytují nějaké služby. Takový zdroj pak má své charakteristiky a omezení. Podobně struktura zdroje může být různá. Zdroj může být tvořen jedním strojem nebo více stroji. Pokud jeden stroj chápeme jako zdroj, jenž v daném časovém okamžiku může zpracovávat maximálně jednu úlohu, pak hovoříme o *disjunktním zdroji*. Je-li zdroj tvořen více identickými stroji, mluvíme o tzv. *identických paralelních strojích*. Taky si můžeme identické paralelní stroje představit jako jeden zdroj, který může zpracovávat více zakázek v jednom časovém okamžiku. Počet těchto zakázek je potom roven počtu paralelních (stejných) strojů. Pokud tedy zdroje umožňují v jeden časový okamžik zpracovávat více úloh, mluvíme o *kumulativních zdrojích*. Náš systém může obsahovat jak disjunktní zdroje, tak kumulativní zdroje. Což je v praxi nejčastější situace.

## 2.3. Základní rozvrhovací problémy

Problém rozvrhování se dělí na několik podskupin, podle složitosti systému a na základě povahy úloh.

1. Plánování na jednom zdroji představuje situaci, kdy veškeré úlohy jsou zpracovány právě jedním zdrojem. Rozvrhem pak rozumíme přiřazení jednotlivých úloh tomuto zdroji v čase.

2. Plánování na více zdrojích řeší situaci, kdy je k dispozici více paralelně pracujících zdrojů, které mohou jednotlivé úlohy zpracovávat. Rozvrhem pak rozumíme přiřazení jednotlivých úloh na konkrétní zdroje v čase.

*Multi-operační (shop) problémy* představují plánování takových úloh, kdy jedna úloha je postupně zpracovávána na více strojích, tzn. skládá se z jednotlivých operací. Rozlišujeme následující varianty.

1. *Flow shop*, představuje situaci, kdy každá úloha musí být prováděna na všech strojích ve stejném pořadí.

2. *Flexible Flow Shop*, jedná se o zobecnění předcházejícího případu. Jednotlivé stroje jsou však paralelní a úloha musí být zpracovávána na každém z nich. Na příslušném paralelním stroji pak může být úloha zpracovávána libovolným strojem.

3. *Job shop*, v této situaci musí být úloha prováděna na všech strojích podle předem daného pořadí. Tento typ je v praxi nejběžnější.

4. *Open shop*, úloha nemusí být prováděna na všech strojích. Pořadí provádění úlohy na strojích určí plánovač.

Ve spojitosti s prací budeme používat variantu Job shop viz [5],[6], protože se nejlépe hodí k popisu našeho problému. Máme tedy definovaný problém, víme jaký očekáváme výsledek a chceme tohoto výsledku dosáhnout pomocí matematického zpracování.

## 2.4. Matematické programování

Ve třicátých letech se začíná rozvíjet *matematické programování* jako ucelený soubor optimalizačních metod, zaměřený na hledání vázaných extrémů funkcí více proměnných. Extrémy funkcí jsou popsány v dodatku A.1 nebo v [16],[17].

Matematických metod na řešení problémů operačního výzkumu je několik. Nás budou v této práci zajímat především metody *matematického programování*, jejichž vznik byl vyvolán výskytem optimalizačních problémů velkého rozsahu. Tvorba *matematického modelu* zahrnuje převedení věcného problému do matematické formulace. Jsou přitom vymezeny závislé a nezávislé proměnné modelu a relace, popisující jejich vzájemný vztah. *Výsledková* proměnná je taková závislá proměnná, která charakterizuje úroveň efektivnosti systému. Účelová funkce potom vyjadřuje výsledkovou proměnnou pomocí nezávislých proměnných. *Řiditelná* (rozhodovací) proměnná je potom taková nezávislá proměnná, kterou může ovlivňovat řešitel modelu. Je-li pro náročnost nutné zjednodušení modelu, nesmí toto zjednodušení znemožnit ty prvky reality, které jsou pro řešení problémů podstatné. Matematické programování ještě můžeme rozdělit podle matematického zápisu na

1. *lineární matematické programování*, což je takové, kde účelová funkce a omezující podmínky mají tvar lineárních rovnic a nerovností,

2. *nelineární matematické programování*, kde se vyskytuje nelineární výraz buď v účelové funkci, nebo v některé z omezujících podmínek,

3. *celočíselné matematické programování*, kde alespoň jedna rozhodovací proměnná nabývá pouze celočíselných hodnot.

Tato práce matematicky řeší problémy lineárního a celočíselného programování. Obecnou formulaci těchto úloh naleznete v příloze (A.2), (A.3) nebo v [1],[2],[4]. V následujících kapitolách budeme vždy uvádět o jaký typ omezení se v konkrétním případě jedná.

# Kapitola 3

# Matematický model průchodu jedné zakázky

Jak předesílá název kapitoly omezíme se nyní na průchod jedné zakázky firmou. Na jejím průchodu si ukážeme matematickou realizaci následnosti operací pomocí lineárního programování. Zakázku rozdělíme na komponenty, které se zpracovávají na různých strojích. Tato posloupnost operací je pro firmu TOSHULIN, a.s. známa, proto ji bereme jako vstupní údaj problému. Cílem kapitoly bude potom určit čas, potřebný k dokončení celé zakázky.

V úvodu kapitoly si ukážeme modelový příklad, matematickou teorii, řešení příkladu, obecný matematický model, softwarovou implementaci modelu a grafický výstup pomocí Ganttova diagramu.

**Příklad 1:** Představme si zakázku, která se skládá z osmi komponent (k1 - k8). Výrobní doby těchto komponent jsou předem známy a dány tabulkou (3.1).

| | k1 | k2 | k3 | k4 | k5 | k6 | k7 | k8 |
|---|---|---|---|---|---|---|---|---|
| čas zpracování | 4 | 12 | 7 | 2 | 10 | 5 | 3 | 4 |

Tabulka 3.1: Výrobní doby komponent

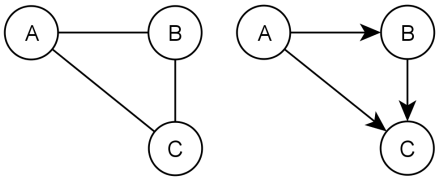
Předpokládáme, že každá z těchto komponent se vyrábí na jiném stroji, a máme proto osm strojů (mach01 - mach08). Dále tedy místo označení komponenta (k) budeme mluvit o stroji (mach). V praxi to tak však platit nemusí, tedy jeden stroj může zpracovávat více komponent. Na druhou stranu můžeme brát komponenty, které by vyráběl jeden stroj jako celek a tím splníme náš předpoklad. Návaznost zdrojů dobře popisuje acyklický orientovaný graf bez vícenásobných hran. Proto si uvedeme pár základních poznatků z teorie grafů a vysvětlíme pojem kritická cesta v grafu, který budeme dále používat.

## 3.1. Analýzy kritické cesty

Definice 3.1 *Graf $G$ je uspořádaná dvojce $G = (V, E)$, kde $V$ je neprázdná množina vrcholů a $E$ je množina hran.*

- *Pro neorientovaný graf $E \to V \otimes V$, kde $V \otimes V$ označuje množinu všech neuspořádaných dvojic prvků z množiny $V$.*

- *Pro orientovaný graf $E \to V \times V$, kde $V \times V$ označuje kartézský součin množinu všech uspořádaných dvojic prvků z množiny $V$.*




Obrázek 3.1: *Neorientovaný a orientovaný graf*

*Poznámka:* Pokud se mezi dvěma vrcholy $u$ a $v$ nachází více hran, nazýváme tyto hrany vícenásobné.

Definice 3.2 *Sled je posloupnost ne nutně různých vrcholů $\{v_0, v_1, \ldots v_n\}$, kde každé dva sousední jsou spojeny hranou, tedy $v_i \in V$ a $\{v_i, v_{i+1}\} \in E$.*

Definice 3.3 *Tah je sled, ve kterém se neopakují hrany, tedy $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\}$ pro $i \neq j$.*

Definice 3.4 *Cesta je tah, v němž se neopakují vrcholy, tedy $v_i \neq v_j$ pro $i \neq j$.*

Věta 3.1 *Mezi dvěma vrcholy existuje cesta, právě tehdy když mezi nimi existuje tah, a právě tehdy když mezi nimi existuje sled.*

Definice 3.5 *Řekneme, že graf $G$ je souvislý, jestliže mezi každými jeho dvěma vrcholy existuje cesta.*

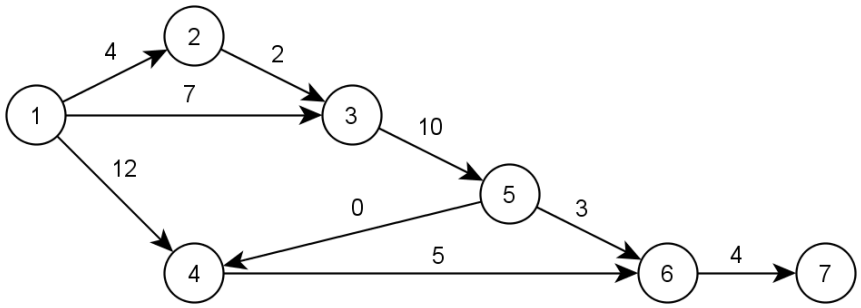
Definice 3.6 *Vzdáleností $d(u, v)$ dvou uzlů $u$ a $v$ v souvislém grafu nazveme délku nejkratší z cest mezi vrcholy $u$ a $v$.*

Tyto a další poznatky lze nalézt například v [9],[10].

Kritická cesta je nejdelší cestou v grafu. Analýza kritické cesty (angl. critical path analysis) je tedy vlastně algoritmus pro hledání nejdelší možné cesty. Cílem všech metod tohoto typu je stanovení doby trvání projektu na základě délky kritické cesty. Svou délkou indikuje minimální dobu potřebnou k dokončení projektového díla. Nejdelší cestě se říká kritická, protože každé zpoždění činností na kritické cestě způsobí zpoždění celého projektu. Každý projekt má kritickou cestu. Existují rozdíly ve sledování kritických a nekritických úkolů. Zatímco úkoly, které leží na kritické cestě, je třeba vyhodnocovat na bázi jednotlivých úkolů, úkoly ležící mimo tuto cestu postačí z pohledu času sledovat podle míry čerpání rezervy. Ta představuje čas, který zbývá do okamžiku, než se z cesty nekritické stane kritická. Kritická cesta a její metody řešení jsou například v [11].

*Poznámka:* Problém kritické cesty bude ale v našem případě pouze informací o délce trvání práce a nebudeme se proto kritickou cestou zabývat ve smyslu sledováním činností na této cestě a jejich optimalizováním.

Acyklický orientovaný graf v našem případě popisuje závislosti mezi činnostmi projektu. Hrany odpovídají činnostem a ohodnocení hran době, jak dlouho bude daná činnost probíhat. Pro náš případ průchodu zakázky firmou budeme jako uzel brát časový okamžik a jako hranu stroj, který zakázku zpracovává. Zadaní příkladu můžeme doplnit o graf z obrázku (3.2).




Obrázek 3.2: *Graf návazností operací pro příklad 1*

Například operace z uzlu 3 může začít, až jsou všechny operace, které do uzlu vchází dokončené. V zadání příkladu mluvíme o osmi strojích, ale hran je v grafu devět. Hrana, která vede z uzlu 5 do uzlu 4 má časové ohodnocení 0 a je to tzv. fiktivní činnost (angl. dummy activity), která uvádí logický vztah mezi uzlem 5 a 4 a říká, že můžeme vyjít z uzlu 4 až je hotové vše v uzlu 5. Pokud takový vztah mají jakékoli dva uzly, vytvoří se fiktivní hrana (stroj) která tyto uzly spojuje. Opět tedy vidíme, že pojem stroj je spíše intuitivní. Vstupním parametrem příkladu je tedy tabulka časů operací, kde musíme také připsat fiktivní činnost, a informace o návaznosti operací.

| | mach1 | mach2 | mach3 | mach4 | mach5 | mach6 | mach7 | mach8 | mach9 |
|---|---|---|---|---|---|---|---|---|---|
| t | 4 | 12 | 7 | 2 | 10 | 0 | 5 | 3 | 4 |

## 3.2. Matematický model

Matematický model, jak uváděla kapitola 2.4 bude vždy obsahovat účelovou funkci, kterou budeme minimalizovat a sadu omezení. Účelovou funkci označíme $z$ a proměnné v jednotlivých uzlech $x_1, x_2, \ldots, x_7$.

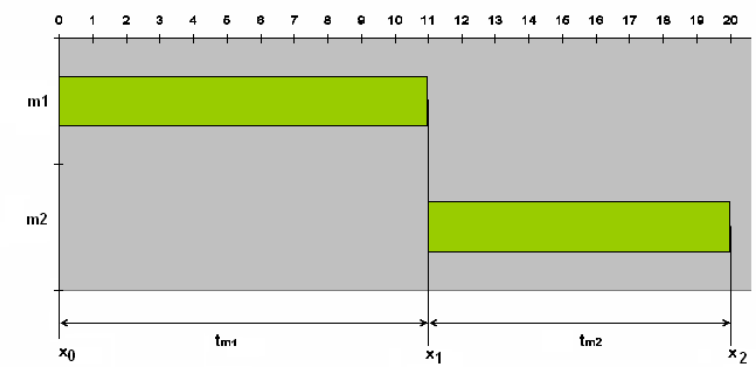
### Účelová funkce

V našem příkladě chceme aby byla zakázka co možná nejdříve hotová, tedy hodnota v posledním uzlu grafu co nejmenší.

$$z_{\min} = \min x_7$$

### Omezení

Zatímco rovnice účelové funkce byla zcela zřejmá, hlavním krokem je matematický zápis omezení. Ty musí popsat cestu grafem, tedy návaznosti operací matematicky. Obrázek (3.3) popisuje situaci, kdy musí zakázka projít dvěma stroji. Zaměřme se na stroj m2.




Obrázek 3.3: *Digram pro následnost operací jedné zakázky*

Z diagramu je zřejmé, že pokud k časové hodnotě $x_1$ připočítáme dobu $t_{m2}$ dostaneme hodnotu $x_2$. Omezení přesto musíme napsat ve tvaru nerovnice, protože v situaci, kdy bude vstupovat do uzlu $x_2$ další hrana, bude hodnota $x_2$ vždy maximem těchto hodnot. Kdyby ne, mohla by nastat situace nedodržení čekací doby začátku operace na předchozí, což je hlavním smyslem tohoto modelu. Podmínku pro stroj m2 tedy můžeme napsat jako

$$-x_1 + x_2 \geq t_{m2}.$$

Omezení je tedy tolik, kolik je hran v grafu. V našem případě to tedy bude následujících devět nerovnic.

$$\begin{aligned}
-x_1 &+ x_2 \geq t_1 \\
-x_1 &+ x_4 \geq t_2 \\
-x_1 &+ x_3 \geq t_3 \\
-x_2 &+ x_3 \geq t_4 \\
-x_3 &+ x_5 \geq t_5 \\
-x_5 &+ x_4 \geq t_6 \\
-x_5 &+ x_6 \geq t_7 \\
-x_4 &+ x_6 \geq t_8 \\
-x_6 &+ x_7 \geq t_9
\end{aligned}$$

Předpokládáme, že čas nemůže nabývat záporných hodnot, proto klademe požadavek na nezápornost proměnných.

$$x_1, x_2, \ldots, x_7 \geq 0$$

Máme tedy účelovou funkci, devět omezení ze sedmi podmínkami nezápornosti proměnných. I když problém na začátku nevypadal příliš složitě, vyřešit ho pomocí tužky a papíru není jednoduché. Tento model budeme řešit pomocí matematického modelovacího jazyka GAMS. Psát ale pro každou sebemenší změnu nový model je zbytečné, proto si nejdříve ukážeme zobecněný matematický model, který budeme v programu GAMS řešit při zadání vstupních dat z příkladu 1.

## 3.3. Obecný matematický model

Rovnice účelové funkce i všech podmínek jsou lineární. Pro stavbu matematického modelu použijeme lineární programování. Konkrétně použijeme lineární program ve tvaru

$$\min\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\},$$

viz příloha (A.2.1) nebo [1],[2],[4].

Návaznost pomocí grafu je sice názorná, ale pro tvorbu obecného matematického modelu nevhodná. Maticový popis grafu vychází ze dvou základních vztahů v grafu. Tím prvním je vztah mezi hranou a jejím koncovým uzlem, který se nazývá vztah incidence viz [9]. Ten popisuje matice incidence grafu. Druhým vztahem je sousednost uzlů viz [9],[10]. Ten popisuje matice sousednosti grafu. Pro naši soustavu rovnic se více hodí první z těchto vztahů, tedy budeme používat incidenční matici.

Definice 3.7 *Incidenční matice (incidence matrix) grafu $G = (V, E, \varepsilon)$, kde $V = \{v_1, \ldots, v_n\}$, $E = \{e_1, \ldots, e_m\}$ a $\varepsilon$ indikuje jestliže existuje orientovaná hrana z uzlů $v$ do $u$ nebo z $u$ do $v$, je $n \times m$ matice $\mathbf{B} = (b_{i,j})$, kde každý řádek odpovídá vrcholu, každý sloupec hraně a*

$$b_{i,j} = \begin{cases} 1, & \textit{jestliže} \quad \exists u \in V : \varepsilon(e_j) = (v_i, u) \\ -1, & \textit{jestliže} \quad \exists u \in V : \varepsilon(e_j) = (u, v_i) \\ 0, & \textit{jinak.} \end{cases}$$

*Poznámka:* Tedy jinak řečeno $b_{i,j} = -1$ pokud j-tá hrana vychází z i-tého uzlu, $b_{i,j} = 1$ pokud j-tá hrana vchází do i-tého uzlu a $b_{i,j} = 0$ pokud j-tá hrana nevychází ani nevchází do i-tého uzlu. Pro náš konkrétní příklad má incidenční matice tvar

$$\mathbf{B} = \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Zadání úlohy je tedy určeno dobou výroby jednotlivých komponent a grafem návazností operací resp. incidenční maticí.

Matematický model obsahuje

- počet strojů $j = 1, \ldots, m$,
- počet uzlů $k = 1, \ldots, q$
- doba zpracování zakázky na $j$-tém stroji $t_j$,
- doba kdy může zakázka pokračovat z uzlu $k$ $x_k$,
- incidenční matice $\mathbf{B}$,

- doba dokončení zakázky na posledním stroji $x_q$,
- účelová funkce $z$.

Tedy rozhodovací nezávislé proměnné jsou $x$ a závislá proměnná je reálná proměnná $z$.

**Účelová funkce**

Účelovou funkci stále chápeme jako minimalizaci hodnoty v posledním uzlu grafu $x_q$, který chápeme jako časový okamžik dokončení zakázky. Účelovou funkci potom zapíšeme ve tvaru

$$z_{\min} = \min x_q. \tag{3.3.1}$$

V příkladě 1 byl stanoven poslední uzel $x_7$ konkrétně. Můžeme ho také stanovit obecně jako $x_q \geq x_k$ kde $k = 1, \ldots, q$. Kdybychom toto zobecnění neudělali, museli bychom pro každou změnu vstupních dat měnit tvar účelové funkce.

**Omezení**

Omezení je tedy tolik, kolik je hran v grafu (strojů). Toto omezení zobecníme pro každé dva uzly $h$ a $p$, které v grafu spojuje hrana $j$

$$x_h - x_p \geq t_j. \tag{3.3.2}$$

Omezení můžeme za pomocí incidenční matice $\mathbf{B}$ rovněž napsat v maticovém tvaru

$$\mathbf{x}^T\mathbf{B} \geq \mathbf{t}^T, \tag{3.3.3}$$

tedy

$$(x_1, \ldots, x_k, \ldots, x_q) \begin{pmatrix} b_{11} & & \ldots & & b_{1m} \\ & \ddots & & & \\ \vdots & & b_{kj} & & \vdots \\ & & & \ddots & \\ b_{q1} & & \ldots & & b_{qm} \end{pmatrix} \geq \begin{pmatrix} t_1 \\ \vdots \\ t_j \\ \vdots \\ t_m \end{pmatrix}^T$$

Předpokládáme, že čas nemůže nabývat záporných hodnot, proto máme požadavek na nezápornost proměnných

$$x_k \geq 0. \tag{3.3.4}$$

Spojením (3.3.1), (3.3.2) nebo (3.3.3) a (3.3.4) dostáváme lineární model pro průchod jedné zakázky grafem.

| | | |
|---|---|---|
| $z_{\min} = \min x_q$ | | |
| $x_q \geq x_k$ | $k = 1, \ldots, q$ | |
| $x_h - x_q \geq t_j$ | $j = 1 \ldots, m$ | $1 \leq p, h \leq q$ |
| $x_k \geq 0$ | $k = 1, \ldots, q$ | |

## 3.4. Implementace modelu v programu GAMS

Modelovací jazyk GAMS (General Algebraic Modeling Systém), vytvořený vývojovým týmem Světové banky v osmdesátých letech, je široce používán na mnohých počítačových

platformách a stal se světovým standardem ve výzkumu a aplikacích. Byl navržen zejména pro úlohy lineárního, nelineárního a smíšeného celočíselného programování, pro které nabízí široký výběr řešičů. Systém byl naprogramován v PASCALu. Soubory vytvořené vnějším editorem jsou zpracovány GAMSem v dávkovém režimu. Více informací lze nalézt na internetových stránkách http://www.gams.com/ nebo v [19].

Ve vývoji optimalizačních programů představuje využití částečně či úplně předkompilovaných procedur tzv. řešičů. Uživatel je využívá, aniž by byl nucen znát implementační detaily a může se zabývat pouze jejich vstupem a výstupem. Tyto řešiče jsou však obvykle dostupné v plné verzi na komerční bázi. V této práci použijeme jak na lineární, tak na celočíselné programování řešič CPLEX. Tedy vždy když budeme hovořit o výsledcích z programu GAMS, budeme hovořit o výsledcích řešiče CPLEX. Tento řešič používá tzv. Simplexovou metodu, která vychází z Gaussovy eliminační metody a stručně popsána v příloze (A.2.2) nebo v [4].

Je dobré zdůraznit, že GAMS je programovací (modelovací) jazyk a program musí být zapsán v jazyce, který používá. Základní strukturu programů můžeme rozdělit na

- množiny (statické, dynamické..),
- parametry (skaláry, tabulky..),
- proměnné (celočíselné, reálné..),
- rovnice (lineární, nelineární..).

Konkrétně pro naše hodnoty vypadá GAMS model

```
$Title  one_job                                          *{jméno modelu}
sets                                                     *{nastavení množin}
         n nodes /1*7/                                   *{množina uzlů}
         m activities  /mach1*mach9/ ;                   *{množina strojů}
parameter                                                *{vstupní parematry}
          t(m) production time                           *{čas zpracování zakázky}
          /mach1 4,mach2 12,mach3 7,mach4 2,mach5 10,
          mach6 0,mach7 5,mach8 3,mach9 4/;
table                                      *{tabulka návazností (incidenční matice)}
       incidence_matrix(n,m)
     mach1 mach2 mach3 mach4 mach5 mach6 mach7 mach8 mach9
  1   -1    -1    -1     0     0     0     0     0     0
  2    1     0     0    -1     0     0     0     0     0
  3    0     1     0     0     0     1    -1     0     0
  4    0     0     1     1    -1     0     0     0     0
  5    0     0     0     0     1    -1     0    -1     0
  6    0     0     0     0     0     0     1     1    -1
  7    0     0     0     0     0     0     0     0     1 ;
variable                                                 *{zavedení proměnných}
          x(n) start time
          z    finish time;
positive variable                                        *{požadavek na nezápornost proměnné}
          x(n);
equations                                                *{zápis rovnic}
         finish_time     finish total time
         balance         balance equations;
finish_time ..  z =e= x('7');
balance(m) ..   sum(n,x(n)*incidence_matrix(n,m)) =g= t(m);
```

```
model one_job one_job /all/ ;
solve one_job  minimizing z using lp;            *{zavolání výpočtu, minimalizace z}
display  x.l;                                    *{výpis proměnných x}
```

Potom po kompilaci v řešiči CPLEX dostaneme výsledek

```
MODEL STATISTICS
BLOCKS OF EQUATIONS           2     SINGLE EQUATIONS           10
BLOCKS OF VARIABLES           2     SINGLE VARIABLES            8
NON ZERO ELEMENTS            20

GENERATION TIME      =        0.000 SECONDS      4 Mb  WIN225-148 May 29, 2007

EXECUTION TIME       =        0.000 SECONDS      4 Mb  WIN225-148 May 29, 2007
GAMS Rev 148  x86/MS Windows                            05/08/08 09:57:59 Page 5
one_matrix
Solution Report     SOLVE one_matrix Using LP From line 39

               S~O~L V~E      S~U~M M A~R Y

     MODEL   one_matrix          OBJECTIVE
z~TYPE    LP                   DIRECTION  MINIMIZE
     SOLVER  CPLEX                FROM LINE  39

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE               26.0000     *{výsledná hodnota účelové funkce
                                               (doba dokončení zakázky)}

 RESOURCE USAGE, LIMIT          0.031      1000.000
 ITERATION COUNT, LIMIT         0         10000

GAMS/Cplex    Jun  1, 2007 WIN.CP.CP 22.5 034.037.041.VIS For Cplex 10.2
Cplex 10.2.0, GAMS Link 34

----     40 VARIABLE x.L  start time                    *{vypsané proměnné x}
2  4.000,    3 17.000,    4  7.000,    5 17.000,    6 22.000,    7 26.000
```

GAMS spočítal výsledek který říká, že zakázka bude hotova v čase 26, a ukázal nám i pro jaké hodnoty $x$ tento výsledek nastane. Díky tomu můžeme spočítat přesné časové vytížení jednotlivých strojů.

I když už máme výsledek, tento příklad ještě nekončí. V minulé kapitole bylo řečeno, že přijatelným výsledkem je Ganttův diagram. Ten máme ovšem v programu MS EXCEL. Přepisování dat po každém výpočtu je z uživatelského hlediska naprosto nepřijatelné. Proto ukážeme, jak se dají poměrně snadno vypočtená data převést z GAMSu do MS EXCELu. Stejně tak pro časté změny vstupních dat není GAMS příliš uživatelsky přívětivý, proto si rovněž ukážeme, jak propojit GAMS a MS EXCEL v opačném směru. Asi nejlepší varianta je zadat hodnoty v MS EXCELu, spustit v pozadí GAMS pomocí makra vytvořeného v Microsoft Visual Basicu a nechat po výpočtu zapsat automaticky výstupní data do MS EXCELu.

Doplňkem GAMSu jsou tzv. GDX (GAMS data exchange) příslušenství, která mají řadu prostředků na propojení programu GAMS s jinými platformami. Nás konkrétně zajímá utilita GDXXRW kvůli možnosti propojení GAMSu s MS EXCELem. Pomoci

ní můžeme přenášet množiny, parametry, proměnné, rovnice, výstupní údaje apod. mezi GAMSem a MS EXCELem. Rozbor tohoto problému však přesahuje rámec práce a v manuálu k programu GAMS je napsaná celá příloha, která se této problematiky týká, proto se zde odkazujeme na [19]. Ukážeme si rovnou jak musíme GAMS model upravit.

```
$Title one_job_scheduling
$onecho > one_job_excel.txt       *{zapíše do souboru .txt z jakých buněk MS EXCEL
set=m rng=b12:j12 cdim=1           sešitu má GAMS načítat vstupní parametry}
set=n rng=a13:a19 rdim=1
par=t rng=a2 rdim=1
par=incidence_matrix rng=a12 cdim=1 rdim=1
$offecho
$call gdxxrw.exe one_job_excel.xls @one_job_excel.txt   *{zavolání utility GDXXRW}
$gdxin one_job_excel.gdx

sets                                                    *{množiny které GAMS načte}
         n(*) nodes
         m(*) activities ;
$load n m
display n,m;
parameter t(m) production time                          *{množiny které GAMS načte}
          incidence_matrix(n,m) incidence_matrix;
$LOAD t incidence_matrix

Display t, incidence_matrix;
$GDXin

variable
          x(n) start time
          z~finish time;
positive variable
          x(n);
equations
         finish_time     finish total time
         balance         balance equations;

finish_time ..  z~=e= x('6');
balance(m) ..   sum(n,x(n)*incidence_matrix(n,m)) =g= t(m);

model scheduling one_job_ /all/ ;
solve scheduling minimizing z~using lp;
display  x.l;

execute_unload 'one_job_excel.gdx',x z~;                *{zapsání hodnot do MS EXCEL}
execute 'gdxxrw.exe one_job_excel.gdx var=x.l rng=b23 var=z rng=b26' ;
                                                        *{umístění hodnot do MS EXCEL}
execute '=shellexecute one_job_excel.xls';              *{zpuštění MS EXCEL}
```

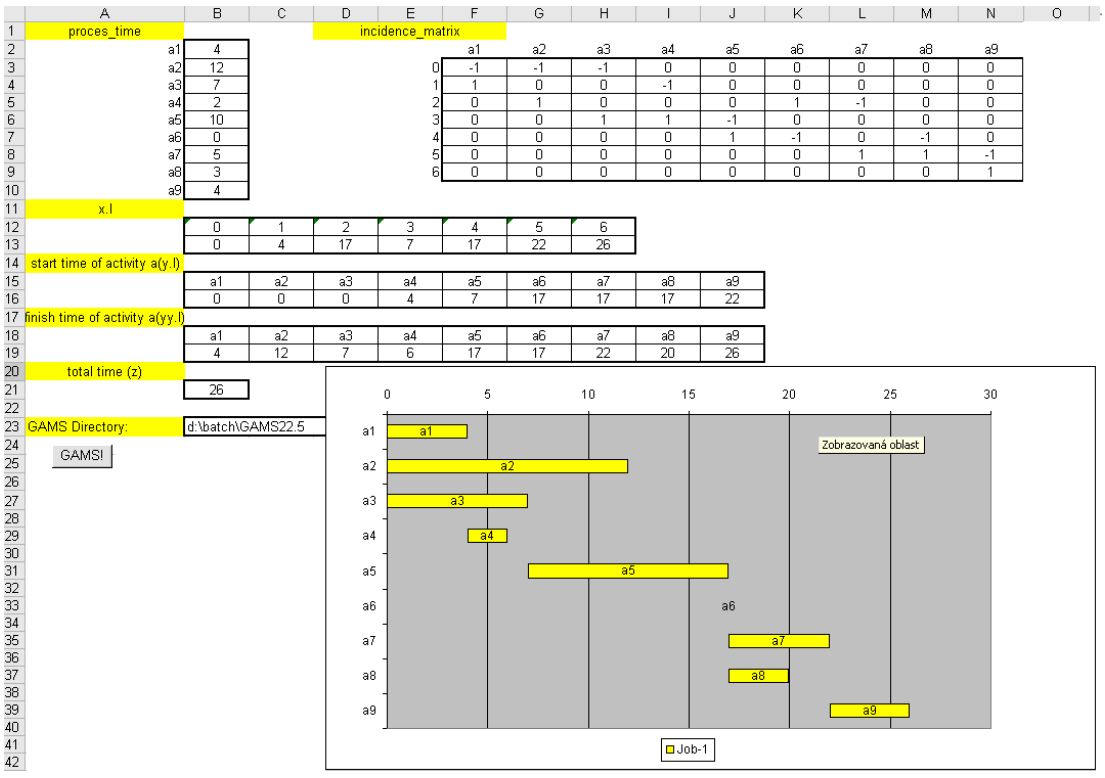
Požadavek na výpočet v programu GAMS můžeme také zavolat z příkazového řádku, nebo pomocí makra i z MS EXCELu. Příkazy k volání výpočtu jsou rovněž napsány v příloze (B.2.2) nebo v [19]. Zde si ukážeme pouze makro vytvořené v VBA pro toto zavolání.

```
Sub GAMS_sum_xf()                           '{jméno makra}
ActiveWorkbook.Save                         '{uložení změn}
```

```
    Path1 = Cells(2, "N").Value                 '{načtení cesty ke GAMS adresáři
                                                 z~buňky N:2}
    Path2 = Cells(2, "P").Value                 '{načtení cesty k~licenčnímu souboru
                                                 z~buňky P:2}
If Path2 = "" Then Path2 = "demo"               '{pokud není nic v~buňce P:2 je
                                                 spuštěna demo verze GAMSu}
    Path = Path1 + "\gams scheduling1.gms license =" + Path2
                                                '{zavolání výpočtu v~programu GAMS}
   ActiveWorkbook.Close                         '{uzavření sešitu z~důvodu zápisu dat}
End Sub                                         '{konec}
```

Prostředí v MS EXCEL je zobrazeno na obrázku (3.4). Makro, které načte cestu k programu GAMS, spouštíme pomocí tlačítka GAMS!. Cesta musí být napsána v buňce B:32. Toto makro zároveň ukládá změny a zavírá aktuální sešit, protože aby mohl zapsat GAMS data do souboru .xls, nesmí tento soubor používat jiný program v době zápisu dat. Na druhou stranu poslední příkaz v GAMS modelu (execute '=shellexecute...) znovu otevře sešit v programu MS EXCEL.

Posledním krokem je zavedení Ganttova diagramu. Data musí být ve tvaru doby začátku a doby konce operace na jednotlivých strojích. Proto si navíc zavedeme pomocné proměnné, kde pro každý uzel a každou hranu které z něj vede, spočítáme dobu začátku operace $x_j^-$ jako $x_k = x_j^-$ pro všechny hrany $j$, které z uzlu vedou. Spočítáme také dobu konce operace $x_j^+$ jako $x_k + t_j = x_j^+$ pro všechny hrany $j$, které do uzlu vchází. Hodnot $x_j^-$ a $x_j^+$ je tedy tolik, kolik je strojů. Z takto poskládaných dat můžeme sestrojit Ganttův diagram podle návodu z kapitoly 1.5.




Obrázek 3.4: *Prostředí MS EXCEL*

Máme tedy model, který počítá potřebnou dobu k zhotovení zakázky, pro různá vstupní data jako je čas operací nebo tabulka návazností operací. Kromě výpočtu má-

me i grafické zpracování v podobě Ganttova diagramu, což bylo hlavním cílem tohoto příkladu. Zdrojové kódy pro GAMS a pro MS EXCEL jsou v příloze (B.1),(B.2).

V dalších kapitolách problém obohatíme o počet zakázek, nejdříve o dvě a potom obecně o libovolný počet. Věci z této kapitoly jako jsou vymezené pojmy, rovnice pro hledání kritické cesty, poznatky z teorie grafů a propojení programů GAMS a MS EXCEL budeme dále používat.

# Kapitola 4

# Matematický model průchodu dvou zakázek

V této kapitole si ukážeme jak se změní náš model, když k němu přibude navíc další zakázka. Ukážeme si matematickou realizaci rozvržení výroby pro dvě zakázky pomocí celočíselného programování. Tato kapitola bude základem pro kapitolu 5, kde těchto zakázek bude libovolně mnoho. Jak bylo řečeno v kapitole 1 a 2, zdroje (stroje) které zakázky zpracovávají jsou omezené ve smyslu počtu zakázek, které mohou zpracovávat zároveň. Mluvíme o problému omezených zdrojů, které jsou reálným problémem jak firmy TOSHULIN, a.s., tak i mnohých dalších. V našich úvahách předpokládáme, že jeden zdroj může zpracovávat právě jednu zakázku v čase. Cílem této kapitoly je tedy tvorba optimálního rozvrhu průchodu dvou zakázek firmou a grafické znázornění pomocí Ganttova diagramu.

Ukážeme si opět modelový příklad, matematickou teorii, matematický model, softwarovou implementaci modelu a grafický výstup pomocí Ganttova diagramu.

**Příklad 2:** Budeme pokračovat v příkladu 1 s tím rozdílem, že grafem posloupností operací budou procházet dvě zakázky. Našim úkolem je rozhodnout o vhodném pořadí těchto zakázek na jednotlivých strojích. Výrobní doby operací jsou dány tabulkou (4.1). Mezi vstupní parametry přibyla navíc množina zakázek, i když v tomto případě

| | mach1 | mach2 | mach3 | mach4 | mach5 | mach6 | mach7 | mach8 | mach9 |
|---|---|---|---|---|---|---|---|---|---|
| $t_{j1}$ | 1 | 24 | 1 | 1 | 6 | 0 | 5 | 3 | 4 |
| $t_{j2}$ | 4 | 5 | 5 | 7 | 4 | 2 | 7 | 4 | 4 |

Tabulka 4.1: Výrobní doby operací

dvoučlenná. Graf návaznosti operací budeme uvažovat stejný jako v příkladě 1. Tedy incidenční matice má tvar

$$\mathbf{B} = \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Musíme vytvořit speciální proměnnou, která bude určovat, jaká zakázka bude zpracována jako první na j-tém stroji a která bude muset na své zpracování počkat. Takováto proměnná se nazývá indikátorová proměnná (angl. indicator variable) a je speciálním případem celočíselné proměnné. Proto v tomto příkladě přejdeme na problém celočíselného programování viz příloha (A.3) nebo [1].

## 4.1. Indikátorové proměnné

V rámci úloh celočíselného programování tvoří speciální skupinu úlohy bivaletního (nula-jedničkového) programování viz [1]. Ty se vyznačují tím, že $D \in \{0,1\}, j = 1,2,\ldots,n$. Pro své použití se také těmto proměnným říká indikátorové proměnné. Indikují například $\delta = 1$ jestliže stroj zpracovává zakázku nebo $\delta = 0$ pokud je volný. Tyto proměnné nevystupují v modelech samostatně, ale spíše v úlohách s reálnými proměnnými, tedy v částečně celočíselných. Logické operace s indikátorovými proměnnými:

| | | |
|---|---|---|
| $X_1 \vee X_2$ | je ekvivalentní s | $\delta_1 + \delta_2 \geq 1$ |
| $X_1 \wedge X_2$ | je ekvivalentní s | $\delta_1 + \delta_2 = 2$ |
| $\neg X_1$ | je ekvivalentní s | $\delta_1 = 0$ |
| $X_1 \rightarrow X_2$ | je ekvivalentní s | $\delta_1 - \delta_2 \leq 0$ |
| $X_1 \leftrightarrow X_2$ | je ekvivalentní s | $\delta_1 - \delta_2 = 0$ |

Využití indikátorových proměnných pro modely plánování a rozvrhování výroby lze nalézt v článcích [20],[21],[22].

V rámci rozsahu rovnic píšeme matematický model rovnou pro obecný případ. Po formulaci modelu ukážeme změny pro původní GAMS model, který po dosazení zadaných dat zkompilujeme a zhodnotíme výsledek.

## 4.2. Matematický model

Pro stavbu matematického modelu použijeme lineární celočíselný program ve tvaru

$$\min\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in D\right\},$$

kde $D \subseteq \mathbb{Z}$ resp. $D \in \{0,1\}$ viz příloha (A.3.1) nebo [1]. Kromě stávajících lineárních rovnic omezení vytvoříme nové rovnice omezení, které budou obsahovat celočíselnou proměnnou, konkrétně indikátorovou proměnnou. Tuto proměnnou označíme $\delta_{i,j}$ pro $i = 1,2$ a bude indikovat jakou zakázku daný stroj právě zpracovává. Tato proměnná se dá interpretovat jako $\delta_{1,j} = 1$ resp. $\delta_{2,j} = 0$ pokud zakázka 1 předchází zakázku 2 na j-tém stroji. Analogicky $\delta_{2,j} = 1$ resp. $\delta_{1,j} = 0$, když 2 předchází zakázku 1. U problému jedné zakázky hodnota proměnné $x_k$ říkala, že v tomto čase skončila poslední operace, která do uzlu $k$ vcházela. Také to byla hodnota, která rovněž uváděla počáteční čas operací, které z uzlu $k$ vycházely. V tomto případě, ale nečeká zpracování zakázky na stroji $j$ na době dokončení předchozích operací, ale i na dostupnosti stroje $j$. Může totiž zpracovávat pouze jednu zakázku v čase. Musíme proto zavést pomocné proměnné, které nám řeknou opravdový počáteční a konečný čas zpracování zakázky $i$ na stroji $j$. Tyto proměnné označíme $x^-_{i,j}$ a $x^+_{i,j}$. Zatímco v minulém případě tyto proměnné byly pouze pomocné na vytvoření Ganttova diagramu, nyní je musíme začlenit do matematického modelu a tedy

i do modelu pro program GAMS.

Matematický model obsahuje

- počet zakázek $i = 1, 2,$
- počet strojů $j = 1, \ldots, m,$
- počet uzlů $k = 1, \ldots, q$
- doba zpracování zakázky $i$ na stroji $j$ $t_{i,j},$
- doba kdy může zakázka $i$ pokračovat z uzlu $k$ $x_{i,k},$
- doba začátku zpracování zakázky $i$ na stroji $j$ $x^-_{i,j},$
- doba konce zpracování zakázky $i$ na stroji $j$ $x^+_{i,j},$
- incidenční matice $\mathbf{B},$
- doba dokončení zakázky na posledním stroji $x_{i,q},$
- indikátorová proměnná $\delta_{i,j},$
- účelová funkce $z,$
- velké kladné číslo $G.$

**Účelová funkce**

Chceme aby součet časů dokončení zakázek byl minimální. Proto píšeme účelovou funkci (3.3.1) upravíme na tvar

$$z_{\min} = \min \sum_{i=1}^{2} x_{i,q}, \tag{4.2.1}$$

kde stejně jako u obecného modelu jedné zakázky $x_{i,q} \geq x_{i,k}$ kde $k = 1, \ldots, q$.

**Omezení**

Lineární omezení (3.3.2) popisující návaznost operací upravíme pro pomocné proměnné, kde pro každé dva uzly $h, p$, které v grafu spojuje hrana $j$.

$$x^-_{i,j} + t_{i,j} = x^+_{i,j} \tag{4.2.2}$$
$$x_{i,h} \leq x^-_{i,j} \tag{4.2.3}$$
$$x_{i,p} \geq x^+_{i,j} \tag{4.2.4}$$

Tyto rovnice jsou pro jednu zakázku ekvivalentní s rovnicí (3.3.2). Dostali jsme sice tři rovnice namísto jedné, ale kdybychom tuto úpravu neudělali, mohlo by se v mnohých výpočtech stát, že by zakázky zbytečně čekaly i když už by mohly být zpracovávány. Když incidenční matici $\mathbf{B}$ rozdělíme na $\mathbf{B}^-$ a $\mathbf{B}^+$, kde pro $b_{kj} > 0$ píšeme $b^+_{kj} = 1$ a $b^-_{kj} = 0$, pro $b_{kj} < 0$ píšeme $b^+_{kj} = 0$ a $b^-_{kj} = 1$ a pro $b_{kj} = 0$ píšeme $b^+_{kj} = 0$ a $b^-_{kj} = 0$, můžeme napsat rovnice v maticovém tvaru

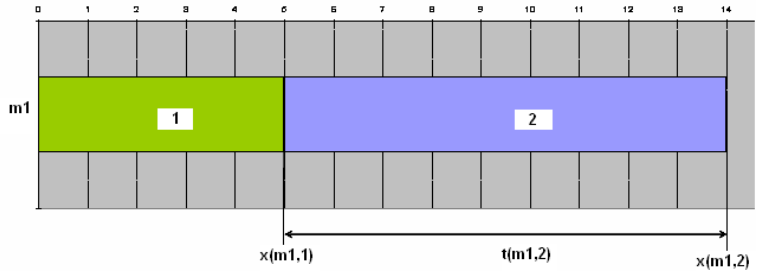
$$\mathbf{x}^{-T} + \mathbf{t}^T = \mathbf{x}^{+T}$$

$$\mathbf{x}^T\mathbf{B}^- \leq \mathbf{x}^{-T}$$
$$\mathbf{x}^T\mathbf{B}^+ \geq \mathbf{x}^{+T}.$$

Celočíselné omezení

$$\sum_{i=1}^{2} \delta_{i,j} = 1 \tag{4.2.5}$$

říká, že pokud zakázka číslo 1 předchází zakázku číslo 2 na stroji $j$, nemůže na tomto stroji zároveň zakázka číslo 2 předcházet zakázce číslo 1. Tato podmínka se dá nazvat jako podmínka pro omezené zdroje a zaručuje zpracování pouze jedné zakázky na stroji $j$ v daném čase.




Obrázek 4.1: *Digram pro pořadí dvou zakázek na jednom stroji*

Podle obrázku 4.1 potřebujeme další omezení, které bude vyjadřovat návaznost zakázek na jednom stroji

$$\delta_{1,j} = 1 \leftrightarrow x^+_{1,j} \leq x^-_{2,j} \tag{4.2.6}$$

resp.

$$\delta_{2,j} = 1 \leftrightarrow x^+_{2,j} \leq x^-_{1,j}, \tag{4.2.7}$$

když $\delta_{1,j} + \delta_{2,j} = 1$ pro $\forall j = 1, \ldots, m$. To můžeme zapsat jako nelineární podmínku ve tvaru

$$(x^+_{1,j} - x^-_{2,j})\delta_{1,j} \leq 0. \tag{4.2.8}$$

Pro $\delta_{1,j} = 1$ rovnice platí a pro $\delta_{1,j} = 0$ nikoliv. Nelineární programování je však výpočtově mnohem náročnější než programování lineární. Existuje proto řada postupů, jak původně nelineární omezení korektně upravit na lineární, tedy tak, aby omezení nebylo porušeno. Ukážeme si zde postupy, jak pomocí lineárního programování za použití indikátorových proměnných nahradit omezení (4.2.8). Tyto postupy jsou ukázány v [23].

Jedním z nejjednodušších případů použití indikátorové proměnné může být $\delta = 1$ jestliže $x > 0$. Jinými slovy $x > 0 \rightarrow \delta = 1$, což potom můžeme zapsat omezením typu

$$x - M\delta \leq 0, \tag{4.2.9}$$

kde $M$ je konstantní koeficient reprezentující známou horní mez proměnné $x$. Vztah $\delta = 1 \rightarrow x > 0$, nemůžeme vyjádřit jako lineární podmínku. Zvolíme proto spodní hranici $m$ pro proměnnou $x$. Dostaneme $\delta = 1 \rightarrow x \geq 0$ což můžeme vyjádřit jako omezení

$$x - m\delta \geq 0, \tag{4.2.10}$$

Podobně můžeme indikátorovou proměnnou použít pro nerovnost $\sum_i a_i x_i \leq b$. Nejdříve si ukážeme příklad $\delta = 1 \rightarrow \sum_i a_i x_i \leq b$. To nám umožní omezení

$$\sum_i a_i x_i + M\delta \leq M + b, \tag{4.2.11}$$

kde $M$ je horní mez výrazu $\sum_i a_i x_i - b$.
Implikace $\sum_i a_i x_i \leq b \rightarrow \delta = 1$ je pouze jinak vyjádřený vztah $\delta = 0 \rightarrow \sum_i a_i x_i > b$. Výraz $\sum_i a_i x_i > b$ upravíme na $\sum_i a_i x_i \geq b+\varepsilon$, kde $\varepsilon$ je malá toleranční hodnota. Omezení má potom tvar

$$\sum_i a_i x_i - (m - \varepsilon)\delta \geq b + \varepsilon. \tag{4.2.12}$$

Analogicky pro výraz $\sum_i a_i x_i \geq b$ dostaneme pro případ (4.2.11) a (4.2.12) omezení

$$\sum_i a_i x_i + m\delta \geq m + b, \tag{4.2.13}$$

$$\sum_i a_i x_i - (M - \varepsilon)\delta \leq b + \varepsilon. \tag{4.2.14}$$

Konečně pro případ $\delta = 1 \leftrightarrow \sum_i a_i x_i \leq b$ musí být použity obě omezení (4.2.11), (4.2.12) a pro případ $\delta = 1 \leftrightarrow \sum_i a_i x_i \geq b$ omezení (4.2.13) a (4.2.14).

Podle předchozího postupu můžeme vyjádřit omezení (4.2.8) jako dvě lineární omezení. Za "vykoupení" z nelinearity sice zaplatíme jedním omezením navíc, na druhou stranu bude i tak model lépe výpočetně řešitelný. Náhrada omezení (4.2.8)

$$x^+_{1,j} - x^-_{2,j} \leq G(1 - \delta_{1,j}) \tag{4.2.15}$$
$$x^+_{1,j} - x^-_{2,j} - (-G - 1)\delta_{1,j} \geq 1. \tag{4.2.16}$$

Kromě jedné rovnice navíc nese tato náhrada ještě jeden problém a to je vhodná volba kladného čísla $G$, aby vyhovovalo všem rovnicím. Mělo by být dostatečně velké, aby rovnice kde bude $G$ obsaženo neměla z optimálního hlediska smysl. Pojem velké číslo je ale dosti neurčitý. Pokud se budou pohybovat doby operací v desítkách, bude stačit když $G \cong 1000$. Když ale budou jednotlivé doby zpracování operací ve stovkách, hodnota takového $G$ stačit nebude a budeme muset tuto hodnotu zvýšit.

Proměnné představují opět čas, proto vyžadujeme podmínky nezápornosti a nula-jedničkovou podmínku pro indikátorovou proměnnou.

$$x^-_{i,j} \geq 0, \tag{4.2.17}$$
$$x^+_{i,j} \geq 0, \tag{4.2.18}$$
$$x_{i,k} \geq 0, \tag{4.2.19}$$
$$\delta_{i,j} \in \{0, 1\}. \tag{4.2.20}$$

Spojením účelové funkce (4.2.1), tří typů lineárních omezení (4.2.2-4), tří typů celočíselných omezení (4.2.5) a (4.2.15-16) s podmínkami nezápornosti (4.2.17-19) a nula-jedničkovou podmínkou (4.2.20) dostaneme matematický model

| | |
|---|---|
| $z_{\min} = \min \sum_{i=1}^{2} x_{i,q}$ | |
| $x_{i,q} \geq x_{i,k}$ | $i = 1,2 \quad k = 1,\ldots,q$ |
| $x_{i,j}^{-} + t_{i,j} = x_{i,j}^{+}$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $x_{i,h} \leq x_{i,j}^{-}$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $x_{i,p} \geq x_{i,j}^{+}$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $\sum_{i=1}^{2} \delta_{i,j} = 1$ | $j = 1,\ldots,m$ |
| $x_{1,j}^{+} - x_{2,j}^{-} \leq G(1 - \delta_{1,j})$ | $j = 1,\ldots,m$ |
| $x_{1,j}^{+} - x_{2,j}^{-} - (-G-1)\delta_{1,j} \geq 1$ | $j = 1,\ldots,m$ |
| $x_{i,j}^{-} \geq 0$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $x_{i,j}^{+} \geq 0$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $x_{i,k} \geq 0$ | $i = 1,2 \quad j = 1,\ldots,m$ |
| $\delta_{i,j} \in \{0,1\}$ | $i = 1,2 \quad j = 1,\ldots,m$ |

## 4.3. Implementace modelu v programu GAMS

Model v jazyce GAMS se v tomto případě rozrostl o množinu zakázek, parametr G (velké číslo), celočíselnou proměnnou $\delta$, proměnné které určují dobu začátku $x^{-}$ a dobu dokončení operace $x^{+}$ a o pět nových nerovností. I v případě celočíselného programování můžeme využít řešiče CPLEX, který řeší celočíselné programování pomocí metody větví a mezí (Branch and bound) viz příloha (A.3.2). GAMS model pro dvě zakázky je mnohem obsáhlejší, proto je uveden pouze v příloze (B.1.2). Po kompilaci řešičem CPLEX dostaneme tyto výsledky

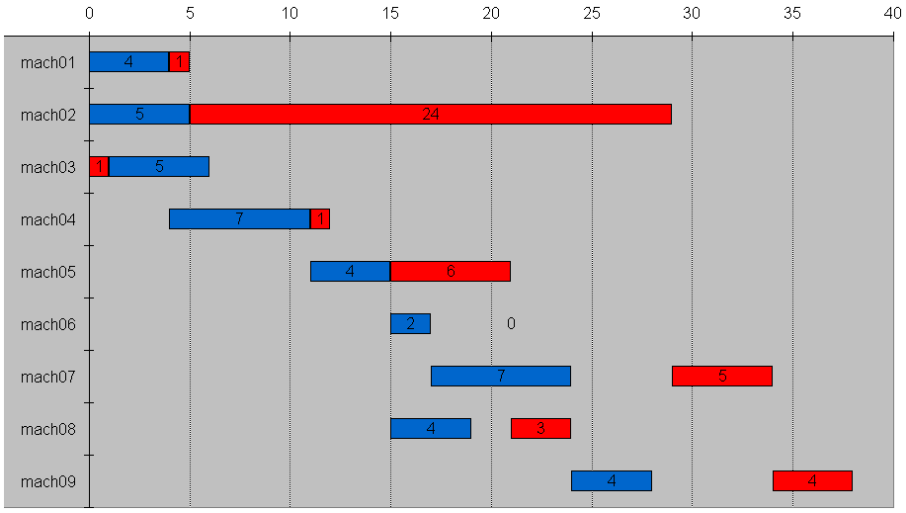
```
MODEL STATISTICS
BLOCKS OF EQUATIONS           7     SINGLE EQUATIONS          100
BLOCKS OF VARIABLES           5     SINGLE VARIABLES           69
NON ZERO ELEMENTS           237     DISCRETE VARIABLES         18

GENERATION TIME      =        0.016 SECONDS      4 Mb  WIN225-148 May 29, 2007

EXECUTION TIME       =        0.016 SECONDS      4 Mb  WIN225-148 May 29, 2007
GAMS Rev 148  x86/MS Windows                            05/08/08 13:26:49 Page 7
two_jobs_scheduling
Solution Report     SOLVE scheduling Using MIP From line 121

               S~O~L V~E      S~U~M M A~R Y

     MODEL   scheduling          OBJECTIVE
z~TYPE    MIP                 DIRECTION  MINIMIZE
     SOLVER  CPLEX               FROM LINE  121

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE               66.0000

 RESOURCE USAGE, LIMIT          0.109      1000.000
 ITERATION COUNT, LIMIT        49         10000

GAMS/Cplex    Jun  1, 2007 WIN.CP.CP 22.5 034.037.041.VIS For Cplex 10.2
Cplex 10.2.0, GAMS Link 34
```

```
----    122 VARIABLE y.L  Indicator variable for job precedence
         mach1       mach2       mach3       mach4       mach5       mach6
j1                                1.000
j2       1.000       1.000                   1.000       1.000       1.000
 +       mach7       mach8       mach9
j2       1.000       1.000       1.000
```

| začátek | mach1 | mach2 | mach3 | mach4 | mach5 | mach6 | mach7 | mach8 | mach9 |
|---|---|---|---|---|---|---|---|---|---|
| $j1$ | 4 | 5 | 0 | 11 | 15 | 21 | 29 | 24 | 34 |
| $j2$ | 0 | 0 | 1 | 4 | 11 | 15 | 17 | 20 | 24 |
| konec | mach1 | mach2 | mach3 | mach4 | mach5 | mach6 | mach7 | mach8 | mach9 |
| $j1$ | 5 | 29 | 1 | 12 | 21 | 21 | 34 | 27 | 38 |
| $j2$ | 4 | 5 | 6 | 11 | 15 | 17 | 24 | 24 | 28 |

Tabulka 4.2: Výsledné doby začátků a konce operací na jednotlivých strojích

Výsledná doba dokončení zakázky j1 je 38 a zakázky j2 28, což dává součet 66. Můžeme také vidět podle výsledku indikátorové proměnné, jaká zakázka je dříve zpracována na konkrétním stroji. Podle výsledků je pouze u stroje mach3 zakázka j1 zpracovávána dříve. V případě dvou zakázek je situace docela přehledná, ale u většího počtu zakázek už tabulka indikátorových proměnných působí velice nepřehledným dojmem, proto zavádíme Ganttův diagram. Pro příklad 2 je Ganttův diagram na obrázku (4.2).




Obrázek 4.2: *Ganttův diagram k příkladu 2*

Červeně je označena zakázka j1 a modře zakázka j2. Ganttův diagram vlastně zobrazuje tabulku (4.2). Oproti tabulce (4.2) ale můžeme okamžitě vidět časové úseky v době, kdy na strojích neprobíhají žádné operace.

# Kapitola 5

# Matematický model průchodu více zakázek

Předešlé dvě kapitoly připravily potřebnou teorii k sestavení modelu pro průchod libovolného počtu zakázek. Ukázali jsme obecný matematický model pro řešení průchodu dvou zakázek firmou, který v této kapitole zdokonalíme. Množina zakázek už nebude obsahovat pouze dvě zakázky, ale libovolný počet. Stále požadujeme aby model počítal s návazností operací a s omezenými zdroji.

Ukážeme si opět modelový příklad, matematický model doplníme o problém více zakázek a s použitim softwarové implementace modelu příklad vyřešíme. Samozřejmě ukážeme i grafický výstup pomocí Ganttova diagramu. Na závěr představíme kompletní softwarové řešení problému rozvrhování výroby s návazností operací a omezenými zdroji pomocí tabulkového editoru MS EXCEL a modelovacího jazyka GAMS, které je doplněno o uživatelsky příjemnější prostředí díky programovacímu jazyku Microsoft Visual Basic.

**Příklad 3:** Představme si 10 zakázek, které se zpracovávají na devíti strojích (mach1 - mach9). Výrobní doby jsou předem známy a dány následující tabulkou.

| | mach1 | mach2 | mach3 | mach4 | mach5 | mach6 | mach7 | mach8 | mach9 |
|---|---|---|---|---|---|---|---|---|---|
| $t_{j1}$ | 11 | 9 | 9 | 5 | 5 | 12 | 1 | 12 | 13 |
| $t_{j2}$ | 11 | 1 | 7 | 13 | 12 | 6 | 15 | 14 | 1 |
| $t_{j3}$ | 15 | 6 | 8 | 12 | 1 | 9 | 8 | 5 | 10 |
| $t_{j4}$ | 10 | 4 | 5 | 13 | 13 | 9 | 15 | 14 | 4 |
| $t_{j5}$ | 11 | 15 | 4 | 9 | 2 | 15 | 11 | 1 | 9 |
| $t_{j6}$ | 2 | 2 | 12 | 5 | 1 | 5 | 6 | 5 | 15 |
| $t_{j7}$ | 15 | 7 | 5 | 3 | 3 | 10 | 7 | 7 | 11 |
| $t_{j8}$ | 5 | 10 | 4 | 3 | 9 | 2 | 7 | 14 | 4 |
| $t_{j9}$ | 12 | 6 | 5 | 14 | 10 | 10 | 7 | 2 | 9 |
| $t_{j10}$ | 11 | 14 | 13 | 1 | 9 | 14 | 7 | 11 | 8 |

Graf návaznosti operací budeme uvažovat stejný jako v příkladě 1 a 2. Tedy incidenční matice má opět tvar

$$\mathbf{B} = \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

## 5.1. Matematický model

Struktura lineárních omezení zůstane stejná jako u předchozího modelu, ale musíme upravit indikátorovou proměnnou. Tato proměnná bude mít o jednu dimenzi více, protože zatím co v předchozí části udávala pouze jestli zakázka 1 předchází zakázku 2 pro daný stroj nebo nikoliv, teď bude muset indikovat závislost pro každé dvě zakázky $l, g$ na každém stroji $j$ tak, že $1 \leq l, g \leq n$, $l \neq g$. Protože se tato proměnná nachází ve třech rovnicích vzroste počet rovnic počítaných GAMSem mnohonásobně a s tím i výpočtová náročnost modelu. Upravená indikátorová proměnná bude vypadat

$$\delta_{l,g,j} = \begin{cases} 1, & \text{jestliže zakázka } l \text{ předchází zakázce } g \text{ na stroji } j, \\ 0, & \text{jinak.} \end{cases}$$

Matematický model obsahuje

- počet zakázek $i = 1, \ldots, n$,
- počet strojů $j = 1, \ldots, m$,
- počet uzlů $k = 1, \ldots, q$
- doba zpracování zakázky $i$ na stroji $j$ $t_{i,j}$,
- doba kdy může zakázka $i$ pokračovat z uzlu $k$ $x_{i,k}$,
- doba začátku zpracování zakázky $i$ na stroji $j$ $x^-_{i,j}$,
- doba konce zpracování zakázky $i$ na stroji $j$ $x^+_{i,j}$,
- incidenční matice $\mathbf{B}$,
- doba dokončení zakázky na posledním stroji $x_{i,q}$,
- indikátorová proměnná $\delta_{l,g,j}$,
- účelová funkce $z$,
- velké kladné číslo $G$.

**Účelová funkce**

V předchozích modelech nebylo třeba sledovat druhy účelových funkcí, ale pro tento model můžeme zaznamenat výrazný rozdíl ve volbě účelové funkce. Ukážeme si dvě účelové funkce a na výsledcích modelového příkladu vysvětlíme jejich opodstatnění pro konkrétní problémy. Další druhy účelových funkcí, které se často v problematice plánování a rozvrhování používají lze nalézt v příloze (A.4) nebo v [7],[24].

$$z_{\min} = \min \sum_{i=1}^{n} x_{i,q} \tag{5.1.1}$$

$$z_{\min} = \min(\max_{i} x_{i,q}), \tag{5.1.2}$$

kde $x_{i,q} \geq x_{i,k}$ pro $k = 1, \ldots, q$.

**Omezení**

Lineární omezení mají stejný tvar jako v případě dvou zakázek s tím rozdílem, že $i = 1, \ldots, n$.

$$x_{i,j}^{-} + t_{i,j} = x_{i,j}^{+}$$

$$x_{i,h} \leq x_{i,j}^{-}$$

$$x_{i,p} \geq x_{i,j}^{+}$$

Celočíselné omezení (4.2.5) změní svůj tvar na

$$\delta_{l,g,j} + \delta_{l,g,j} = 1,$$

které říká, že pokud zakázka číslo l předchází zakázku číslo g na stroji $j$, nemůže na tomto stroji zároveň zakázka číslo g předcházet zakázce číslo l. Tedy opět podmínka pro omezené zdroje která říká, že na jednom stroji lze zpracovávat pouze jednu zakázku v daném čase. Další dvě omezení jsou opět analogií rovnic (4.2.15-16) z modelu pro dvě zakázky.

$$x_{l,j}^{+} - x_{g,j}^{-} \leq G(1 - \delta_{l,g,j})$$

$$x_{l,j}^{+} - x_{g,j}^{-} - (-G - 1)\delta_{l,g,j} \geq 1$$

Podmínky nezápornosti a nula-jedničková podmínka

$$x_{i,j}^{-} \geq 0,$$

$$x_{i,k} \geq 0,$$

$$x_{i,j}^{+} \geq 0,$$

$$\delta_{l,g,j} \in \{0, 1\}.$$

Kompletní matematický model má tvar

| | |
|---|---|
| $z_{\min} = \min \sum_{i=1}^{n} x_{i,q}(z_{\min} = \min(\max_i x_{i,q}))$ | |
| $x_{i,q} \geq x_{i,k}$ | $i = 1, \ldots, n \quad k = 1, \ldots, q$ |
| $x_{i,j}^{-} + t_{i,j} = x_{i,j}^{+}$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $x_{i,h} \leq x_{i,j}^{-}$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $x_{i,p} \geq x_{i,j}^{+}$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $\delta_{l,g,j} + \delta_{l,g,j} = 1$ | $j = 1, \ldots, m \quad 1 \leq l, g \leq n, l \neq g$ |
| $x_{l,j}^{+} - x_{g,j}^{-} \leq G(1 - \delta_{l,g,j})$ | $j = 1, \ldots, m \quad 1 \leq l, g \leq n, l \neq g$ |
| $x_{l,j}^{+} - x_{g,j}^{-} - (-G - 1)\delta_{l,g,j} \geq 1$ | $j = 1, \ldots, m \quad 1 \leq l, g \leq n, l \neq g$ |
| $x_{i,j}^{-} \geq 0$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $x_{i,j}^{+} \geq 0$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $x_{i,k} \geq 0$ | $i = 1, \ldots, n \quad j = 1, \ldots, m$ |
| $\delta_{l,g,j} \in \{0, 1\}$ | $j = 1, \ldots, m \quad 1 \leq l, g \leq n, l \neq g$ |

## 5.2. Implementace modelu v programu GAMS

Právě tím, že používáme v modelu různé kombinace množiny zakázek, musíme rovnice zapsat v režimu dynamických množin. Viz následující ukázka zápisu množin a části rovnic v GAMS modelu. Podrobně je používání dynamických rovnic popsáno v [19].

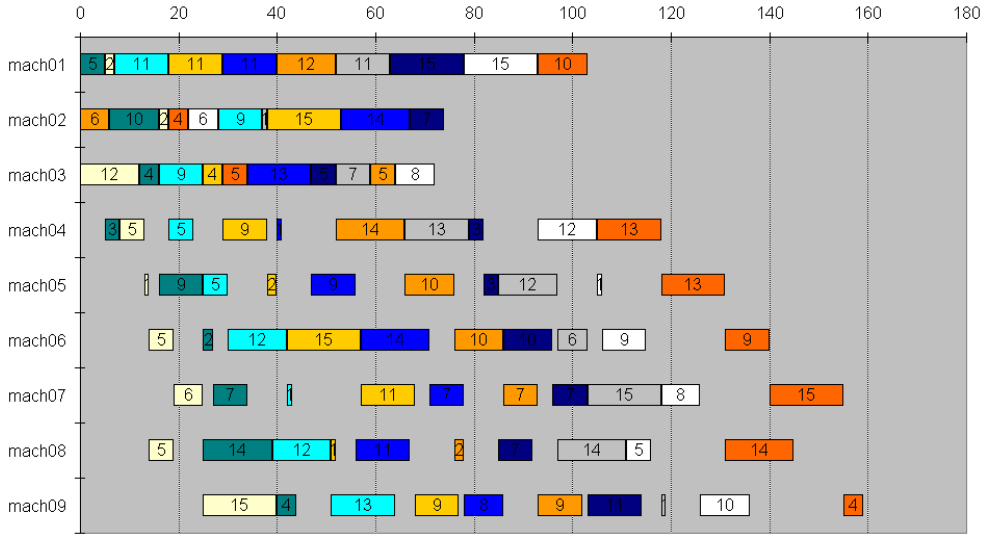
```
sets     m(*)     machines
         j(*)     job
         n(*)     nodes  ;

alias (j, p); *{nové pojmenování množiny j kvůli dynamickému zápisu rovnic}
alias (j, k);
*-------------------------------------------------------------------------------------
binary_condition(m,k,p)$(not ord(k) ge ord(p)).. y(k,p,m) + y(p,k,m) =e= 1;

sequential4(m,k,p)$(not ord(k) eq ord(p))..uu(k,m) - u(p,m) =l= g*(1-y(k,p,m));

sequential5(m,k,p)$(not ord(k) eq ord(p))..uu(k,m) - u(p,m) - (-g-1)*(y(k,p,m)) =g= 1;

*{použití dinamických rovnic za pomocí operátoru $}
```

Výsledná doba po kompilaci modelu s účelovou funkcí (5.1.1) řešičem CPLEX je 941. Jednotlivé doby dokončení zakázek jsou v tabulce (5.1).

| | |
|---|---|
| j1 | 64 |
| j2 | 119 |
| j3 | 136 |
| j4 | 159 |
| j5 | 77 |
| j6 | 40 |
| j7 | 114 |
| j8 | 44 |
| j9 | 102 |
| j10 | 86 |

Tabulka 5.1: Tabulka časů dokončení zakázek

Výsledná tabulka indikátorových proměnných je vlastně trojdimenzionální a v tomto případě obsahuje devět sloupců a devadesát řádků. Z takové tabulky by opravdu rozvrh asi nikdo neposkládal, zato prezentace pomocí Ganttova diagramu z obrázku (5.1) vypadá zcela zřejmě.



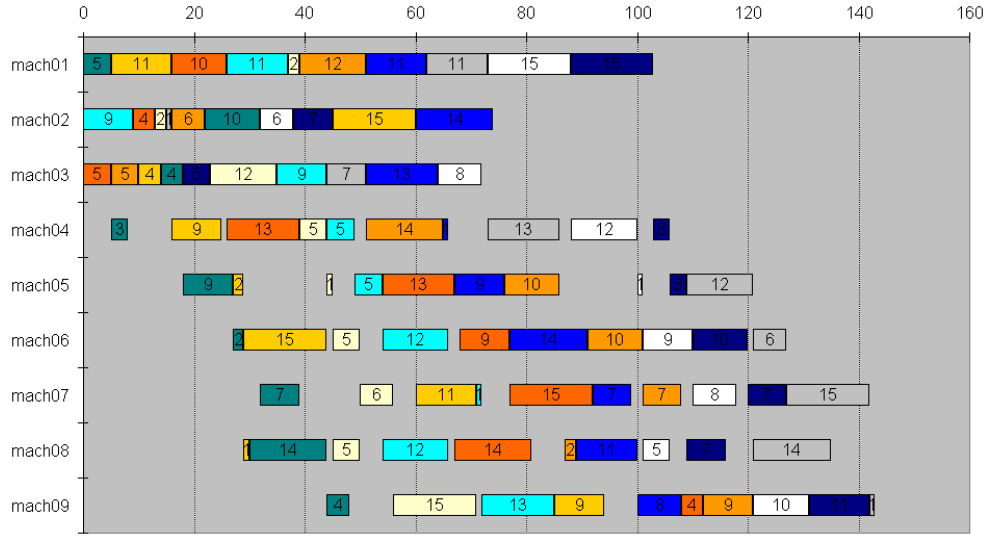

Obrázek 5.1: *Ganttův diagram k příkladu 3 pomocí účelové funkce (5.1.1)*

Když ve stejném modelu při použití stejných vstupních parametrů nahradíme účelovou funkci (5.1.1) účelovou funkcí (5.1.2) výsledek bude zcela odlišný. Hodnota účelové funkce teď neudává součet, ale dobu dokončení poslední zakázky, ta je 143. Abychom mohli porovnávat účelové funkce sečteme doby dokončení všech zakázek, které jsou v tabulce (5.2) a dostaneme hodnotu 1055.

| | |
|---|---|
| j1 | 85 |
| j2 | 143 |
| j3 | 131 |
| j4 | 112 |
| j5 | 94 |
| j6 | 71 |
| j7 | 142 |
| j8 | 48 |
| j9 | 121 |
| j10 | 108 |

Tabulka 5.2: Tabulka časů dokončení zakázek

Ještě vykreslíme Ganttův diagram obrázek (5.2), abychom mohli zakázky porovnat i graficky.




Obrázek 5.2: *Ganttův diagram k příkladu 3 pomocí účelové funkce (5.1.2)*

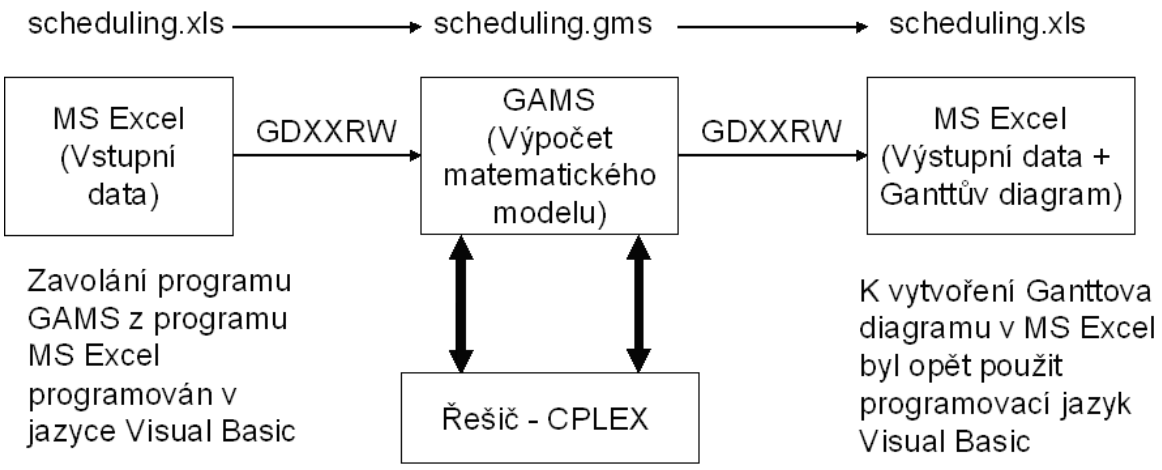
Zhodnocení výsledků

- účelová funkce (5.1.1), dokončení poslední zakázky 159, součet dokončení zakázek 941,
- účelová funkce (5.1.2), dokončení poslední zakázky 143, součet dokončení zakázek 1055.

V druhém případě jsme sice skončili s výrobou všech zakázek o 114 časových jednotek dříve. Na druhou stranu to zpozdilo zakázky, které už mohly být podle prvního plánu hotovy. Co je tedy lepší? Výsledky neříkají, který model je lepší, pouze ukazují důležitost výběru účelové funkce pro konkrétní případ. V prvním případě se snažíme udělat všechny zakázky co nejrychleji (čím dříve opustí firmu tím lépe). Druhý model zase popisuje situaci, kdy se musíme pokusit do nějakého určitého data dokončit všechny zakázky, přičemž je jedno, jestli do tohoto data budou nějaké hotovy dříve nebo ne. Například kdybychom vyráběli produkty, které musíme koncem každého měsíce odeslat, bylo by jedno, jestli některé vyrobíme na začátku, nebo až na konci měsíce. Hlavně aby byly hotovy všechny do konečného termínu.

## 5.3. Další uživatelská vylepšení a uživatelský přístup

Prostředí MS EXCEL, zdrojový kód k programu GAMS a zdrojové kódy maker vytvořené v VBA nazveme dohromady jako program `SCHEDULING`. Tento program dokáže řešit úlohy rozvrhování a plánování, které byly v kapitolách 3-5. Byly v něm spočítány všechny příklady a jsou v něm vytvořeny všechny Ganttovy diagramy v celé práci. Tento program se dá představit jako blokové schéma z obrázku (5.3).




Obrázek 5.3: *Blokové schéma programu* `SCHEDULING`

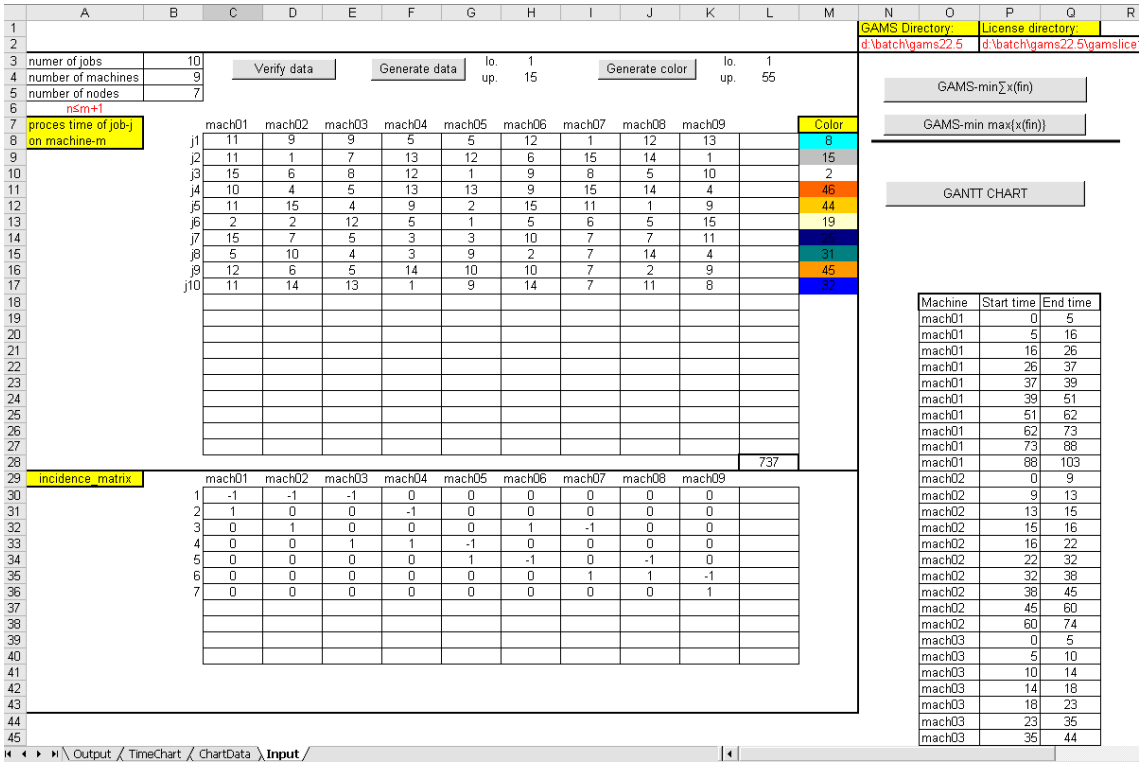
První blok (MS EXCEL vstupní data) představuje soubor v MS EXCEL formátu scheduling.xls, kde zadáváme vstupní údaje o počtu zakázek, strojů, tabulku časů apod. Dále je zprogramováno makro (B.2.2), díky kterému můžeme zavolat program GAMS. Ten převede pomocí utility GDXXRW vstupní data do souboru v GAMS formátu scheduling.gms (blok GAMS), kde je ve zdrojovém kódu napsán matematický model (B.1.2). Tento model se vstupními daty vyřeší řešič CPLEX a výsledek předá GAMSu. Opět pomocí utility GDXXRW převede GAMS výsledky do souboru scheduling.xls (blok MS EXCEL výstupní data). Poslední věcí je makro na tvorbu Ganttova diagramu (B.2.1). Je dobré poznamenat, že je výpočet prováděn v pozadí, tedy uživatel nemusí mít znalosti o programu GAMS.

Dále si ukážeme pár drobných vylepšení v programu MS EXCEL, které program `SCHEDULING` obsahuje popíšeme si ho z uživatelského hlediska. Obrázek (5.4) ukazuje poslední verzi softwarového řešení problému.

V buňkách B:3-5 sešitu Input uživatel vyplní počet zakázek, strojů a uzlů. Z těchto buněk čtou makra své údaje, proto je vyplnění těchto buněk nezbytností. Tlačítko Verify data podle těchto buněk označí sloupce a řádky tabulek. Pokud se rozhodneme testovat problém na náhodných číslech, dá se použít tlačítko Generate data, které vyplní tabulku náhodnými čísly v rozmezí hodnot načtených z buněk H:3 a H:4. Stejně tak barevné označení zakázek generuje tlačítko Generate color. Jak uváděla kapitola, mohli jsme použít k výpočtu dvě účelové funkce. Abychom nemuseli měnit údaje v GAMS modelu, použijeme tyto modely dva, lišící se právě v účelové funkci. Na jejich zavolání jsou proto použita dvě tlačítka označená právě tvarem účelové funkce. Stejně tak jako v případě jedné zakázky musíme načíst cestu k hlavnímu GAMS adresáři. Zároveň načítáme i cestu k licenčnímu souboru, který nemusí být umístěn v hlavním adresáři. Po kompilaci se znovu otevře sešit v programu MS EXCEL se zapsanými výsledky v sešitu Output. Poslední tlačítko s názvem GANTT CHART generuje Ganttův diagram z vypočtených hodnot.

Prostředí MS EXCEL obsahuje čtyři sešity. Input, kde zadáváme vstupní data, Output, kde jsou po výpočtu vloženy data výstupní, ChartData, kde jsou uspořádána výstupní data pro tvorbu diagramu a GanttChart, kde je vykreslen Ganttův diagram.

Když jsme mluvili o načtení vstupních dat do programu GAMS, nevysvětlili jsme co




Obrázek 5.4: *Prostředí MS EXCEL*

dosazujeme za parametr $G$. Pouze jsme zmínili, že se jedná o velké číslo. Kdyby bylo v GAMS kódu konstantně zadané, při řádové změně dat by bylo potřeba tento parametr změnit přímo v GAMSu. Z uživatelského hlediska je proto lepší i hodnotu parametru $G$ nečíst z MS EXCELu. Použijeme k tomu sumu dob trvání operací přes všechny stroje a všechny zakázky z buňky L:28. Hodnota $G$ potom vystihuje nejhorší možný scénář, který nemá z optimálního hlediska smysl použít.

Všechny zdrojové kódy k programu GAMS a použitých maker vytvořených VBA jsou v příloze (B.1),(B.2). Program `SCHEDULING` je potom na přiloženém CD. K jeho spuštění je nutná verze MS EXCEL 2002 nebo vyšší a GAMS 22.5 obsahující soubor gdxxrw.exe, který je také součástí CD.

# Kapitola 6

# Možné rozšíření modelu

V této kapitole uvedeme další možné kroky pro přiblížení matematického modelu k realitě.

## 6.1. Vícekriteriální úlohy

Z praxe často musíme rozhodovat současně podle několika kritérií. Při optimalizaci ve výrobním podniku můžeme často chtít minimalizovat náklady na výrobu a přitom minimalizovat výrobní časy zakázek z důvodu dodržení termínu. Máme tedy *množinu variant* a máme z ní vybrat takovou, která bude co možná nejlepší vzhledem k *množině kritérií*. Uvažovaná kritéria bývají obvykle konfliktní v optimálním smyslu. Musíme se proto spokojit s nějakým kompromisem. Metody jsou v podstatě založeny na jednorázovém nebo opakovaném použitím metod pro jednokriteriální optimalizaci viz [1].

Model se dá pro případy vícekriteriální optimalizace upravit tak, že účelová funkce bude obsahovat součet více účelových funkcí, které bude násobit váhová funkce. Potom se dá volbou této funkce zkoumat více scénářů a vybrat takový, který se pro danou situaci hodí nejlépe. Další možností, jak upravit model je vložit účelovou funkci jako další omezení, které se bude snažit držet řešení mezi hodnotami, které požadujeme.

## 6.2. Heuristické metody

Výpočtová náročnost metod operačního výzkumu je uvedena v příloze (A.5) nebo v [3],[7]. Problém plánování a rozvrhování výroby patří mezi NP-úplné problémy. Proto náš model nebude pro komplexní problémy schopen nalézt řešení v rozumném čase.

Pojem *heuristika* je odvozen z řeckého slova *heuriskein*, což znamená nalézt, objevit. Příkladem jednoduchého heuristického přístupu je řešení problému "od oka". V operačním výzkumu je *heuristická metoda* charakterizována jako metoda pro hledání "dobrých" řešení (tj. řešení blízkých k optimálnímu) pomocí intuitivního přístupu. Z toho tedy vyplývá, že heuristická metoda nemůže zaručit nalezení globálního optima a obvykle ani stanovit, jak blízko k optimu se její přípustné řešení nachází. Vznik heuristických metod a rozvoj zájmu o ně má řadu příčin. Je to zapříčiněno především náročností exaktních metod a jejich použitím na úlohy komplikované svým rozměrem, nebo strukturou. Dalším argumentem pro používání heuristických metod je skutečnost, že neřešíme skutečný problém, ale pouze jeho model a není žádná záruka toho, že optimální řešení

modelu bude také optimálním řešením skutečného problému. Proto také můžeme v posledních letech pozorovat vývoj a zefektivňování moderních heuristických metod. Mezi nejznámější heuristické metody patří

- *Lokální hledání* (angl. Local search), které je základní jednoduchou metodou. Počátečním stavem je náhodné zvolení přípustného řešení. V dalším kroku se vybírá sousední "lepší" řešení obvykle metodou nejstrmějšího, nebo náhodného spádu. V této metodě není řešením obvykle globální optimum, ale lokální. K překonání tohoto nedostatku se například metoda vícekrát opakuje od začátku a je šance nalézt alespoň lepší lokální optimum.

- *Simulované žíhání*, bylo původně zavedeno jako analogie termodynamických procesů. Tento algoritmus je podstatě vylepšením předešlé metody, kde jsou povoleny i heuristické kroky směřující k horšímu řešení. Nabízí se tak možnost překonat bariéru lokálního optima.

- *Tabu prohledávání* (angl. Tabu search) obsahuje třídu technik lokálního prohledávání a zvyšuje jejich efektivitu používáním paměťových struktur. Již navštívené řešení je označeno jako "tabu" a algoritmus toto řešení znovu nenavštíví.

- *Genetické algoritmy* pochází od analogie mezi reprezentací složité struktury pomocí vektoru složek a genetickou strukturou *chromozómu*. Základní myšlenka tohoto algoritmu stojí na hledání lepších řešení pomocí kombinací částí existujících řešení.

Zmínění heuristických metod je pro tuto práci pouze okrajové, více lze nalézt v [1]. Přesto pro budoucí rozvíjení této práce jsou heuristické algoritmy nepostradatelnou součástí. Problémy plánování a rozvrhování za pomocí genetických algoritmů jsou například řešeny v [18]. Do našeho modelu by se dala také zahrnout heuristika, například kdybychom za některé indikátorové proměnné pevně dosadili konstantu. Tím by se snížila výpočtová náročnost problému a mohli bychom potom vypočítat různé scénáře pro různou volbu parametrů. Z nich dále vybrat pro nás ten nejlepší.

## 6.3. Dynamické programování a dynamické plánování

Řešení problému dynamickým programováním rozumíme řešení dílčích problémů a ty po jejich vyřešení spojit dohromady. Tento pojem byl poprvé použit v padesátých letech dvacátého století Richardem Bellmanem. Postup našel uplatnění v systémovém řízení a inženýrských otázkách. Nachází také uplatnění v matematickém programování.

Při dynamickém plánování nejsou na rozdíl od statického známy všechny úlohy, omezení a zdroje před začátkem tvorby rozvrhu. V dynamickém plánování se naopak snažíme řešit tu skutečnost, že v reálných aplikacích jsou málokdy dopředu známy a dostupné všechny úlohy, pro které se má stanovit rozvrh. Dynamické plánování pak usiluje o nalezení rozvrhu pro dynamicky přibývající úlohy v dynamicky se měnícím prostředí. Kromě přibývání úloh může docházet i ke ztrátě či zneplatnění některých úloh, případně výpadkům, nebo přibývání zdrojů a tím k modifikacím v rozvrhu. Poté je nutné přistoupit k tzv. on-line plánování a řešit nově vzniklou situaci.

On-line plánování představuje proces tvorby rozvrhu během provádění úloh na zdrojích, tedy za běhu systému. Rozvrh je tedy tvořen inkrementálně v čase. Častým požadavkem bývá dostatečná rychlost on-line plánování.

Další nepříjemnou skutečnost představuje fakt, že v dynamicky měnícím se prostředí nelze zaručit optimalitu rozvrhu a jeho robustnost, neboť ta je garantována pouze pokud se systém a prostředí nemění. Situace, které většinou naruší rozvrh, jsou trojího typu. První z nich je změna množiny plánovaných aktivit, zejména tedy příchod nové úlohy k naplánování. Další je změna množiny dostupných zdrojů a posledním faktorem pak bývají časové změny, např. zpoždění úloh, které způsobí nekonzistenci rozvrhu.

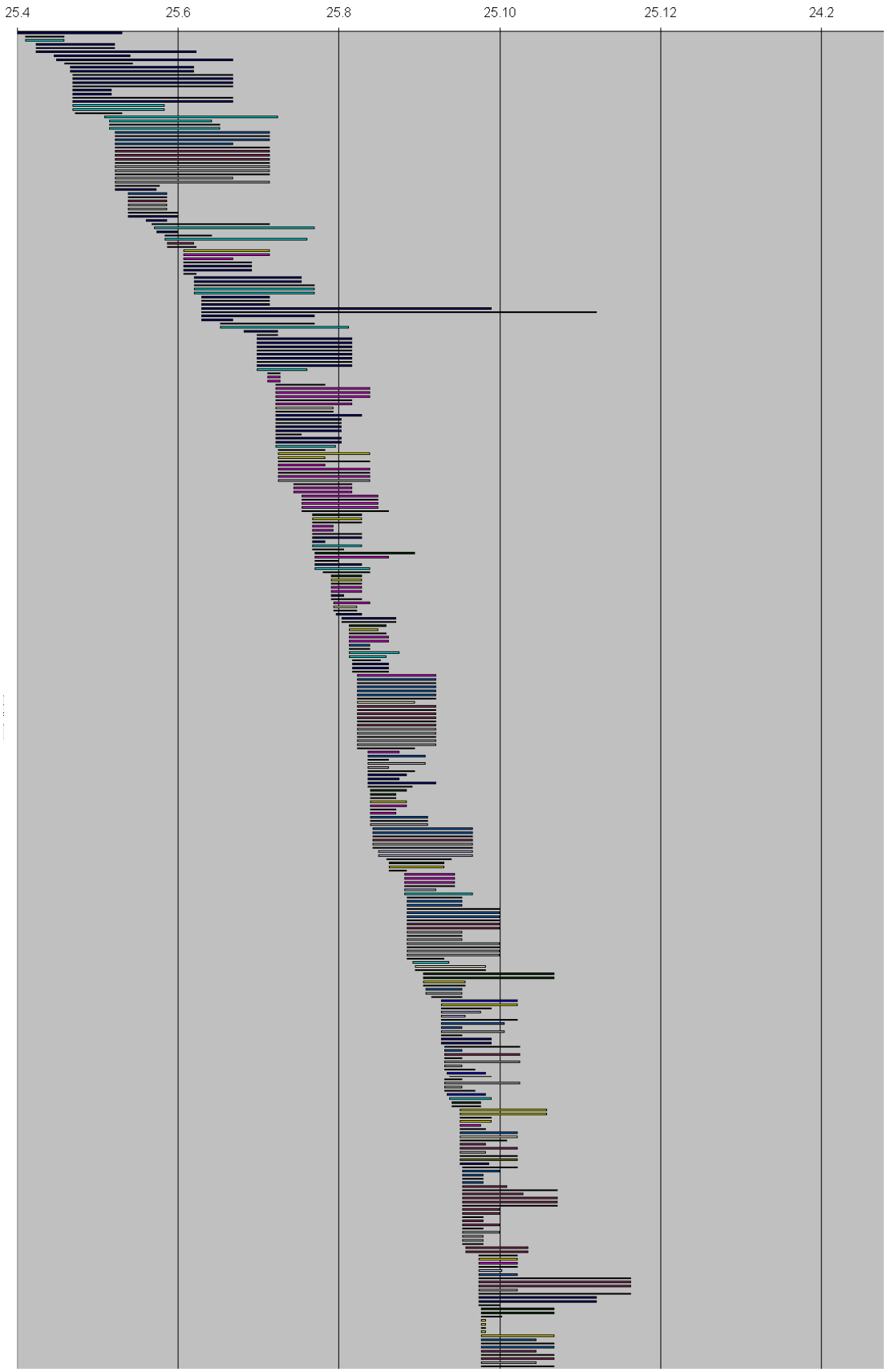
Způsoby, jakými se s on-line plánováním algoritmy vyrovnávají, jsou různé. První způsob spočívá v tom, že pro každou novou úlohu v systému, nebo pro jinou změnu je celý algoritmus spuštěn znovu pro všechny rozvrhované úlohy. Toto řešení nebývá časté, neboť je časově náročné a ztrácí se při něm dosažené znalosti z předchozího rozvrhu. Druhý způsob spočívá ve snaze uplatnit již spočtený rozvrh a provádět pouze lokální změny v již hotovém rozvrhu, aniž by bylo nutné celý problém řešit od začátku. Lze také problém dekomponovat na subproblémy a ty vyřešit zvlášť a pokusit se z nich sestavit konzistentní řešení. Více o dynamickém programování a plánování v [1],[7].

## 6.4. Stochastické programování

Prakticky ve všech reálných problémech vystupují členy (parametry), které jsou svým chováním náhodné. V některých problémech se dají tyto parametry zanedbat bez větších problémů. Někdy jsou ale velice důležité a jejich zanedbání by vedlo k porušení reality modelu a výsledek takového modelu by byl bezcenný.

Již v úvodu když byl formulován problém firmy TOSHULIN, šlo o problém s velkou účastí náhodnosti. Tato náhodnost byla prezentována jako časová neurčitost doby zpracování zakázek. Kvůli komplexnosti samotného problému byla řešena pouze verze s deterministickými časem. Proto by mělo být dalším rozšířením modelu směr k stochastickému programování. Definice stochastického programování a další informace lze nelézt například v [8].

Složitost reálného problému ukazuje zlomek reálných historických dat na obrázku (6.1), které popisují průchod zakázek jedním oddělením firmy. Diagram ukazuje rozvržení zakázek, které jsou rozlišeny barevně, během osmiměsíčního období. Z diagramu je například vidět postupné přibývání zakázek což vyžaduje dynamické plánování. Navíc optimální plán musí být optimální ve smyslu průchodu všemi odděleními a ne pouze jedním čímž opět vzroste složitost problému. Výrobní časy některých komponent nejsou předem známy a problém se bude muset řešit pomocí stochastického programování.




Obrázek 6.1: *Ukázka reálných dat*

# Závěr

Cílem práce bylo představit reálný problém firmy TOSHULIN, a.s. a pomocí optimalizačních metod tento problém řešit. Díky komplexnosti problému byla práce zaměřena na problémy deterministického charakteru, konkrétně na paralelní průchod více zakázek firmou s omezenými zdroji. Cíl práce tedy můžeme rozdělit na tvorbu matematického modelu, na vytvoření původního programového zpracování modelu v modelovacím jazyce GAMS a na grafický výstup v podobě Ganttových diagramů, které jsou velice rozšířené v průmyslové výrobě.

Práce potom ukazovala na jednoduchých modelových příkladech matematickou teorii a postupně vybudovávala obecný matematický model. Tento model byl zpracován v původní implementaci modelovacím jazykem GAMS. Na tvorbu Ganttových diagramů byl použit programovací jazyk Microsoft Visual Basic v prostředí MS EXCEL. Pro uživatelsky příjemné prostředí byl navíc propojen program GAMS a MS EXCEL.

Práce ukázala, že tvorba optimálního rozvrhu výroby bez použitých metod matematického programování je i pro jednodušší příklady velice obtížná až nemožná. Přesto optimální rozvrh může firmě ušetřit nemalé náklady. Právě proto se v oblasti plánování a rozvrhování výroby stále vyvíjí nové a účinnější metody.

Současně je třeba poznamenat, že vzhledem k náročnosti reálného problému je práce prvním krokem a může sloužit jako základní kámen řešení problému. Vhodným rozšířením modelu je nahrazení fixní a dopředu známé výpočetní délky úloh, kterou ve skutečnosti většinou přesně neznáme. Model potom může být přeformulován na problém stochastické optimalizace. Výpočet bude složitější a můžou nastat komplikace s časovou náročností výpočtu. Pokud se ale spokojíme s dobrým, ne však nutně s optimálním řešením, může práce směřovat na implementaci heuristických metod. Konkrétně se pro tyto problémy používají genetické algoritmy.

Závěrem ještě znovu podotýkáme, že diplomová práce je součástí řešení projektu MŠMT České republiky čís. 1M06047 Centrum pro jakost a spolehlivost výroby a projektu GA ČR reg. čís. 103/08/1658. Bude se tedy pokračovat v řešení problému firmy TOSHULIN, a.s.

# Literatura


[1] J. Klapka, J. Dvořák, P. Popela: *Metody operačního výzkumu*, VUTIUM (2001), ISBN 80-214-1839-7.

[2] J. Plesník, J. Dupačová, M.Vlach: *Lineárne programovanie*, Alfa Bratislava (1990).

[3] L. Čapatý: *Rozvrhování výroby pomocí metod lokálního hledání*, Diplomová práce, VUT FSI, Brno (1998).

[4] Thomas S. Ferguson: *Linear Programming*. A Concise Introduction, University of California at Los Angeles (2005).

[5] F. Cottet, J. Delacroix, C. Kaiser, Z. Mammeri: *Scheduling in Real-Time Systems*, JOHN WILEY & SONS, West Sussex PO19 8SQ, England (2002).

[6] Thomas E. Uher: *Programing and Scheduling Techniques*, University of New South Wales Press Ltd, UNSW Sydney NSW 2052 Australia (2003).

[7] D. Klusácek: *Plánování úloh v paralelním a distribuovaném prostredí*, Diplomová práce,Masarykova Univerzita Fakulta Informatiky, Brno (2006).

[8] J. Dupačová: *Stochastické programování*, VN MON, Praha (1986).

[9] M. Šeda: *Teorie grafů*, VUT FSI, Brno (2003).

[10] A. Večerka: *Grafy a grafové algoritmy*, Univerzita Palackého Přírodovědecká Fakulta Katerda Informatiky, Olomouc (2007).

[11] Kerzner, Harold: *Systems Approach to Planning, Scheduling, and Controlling*, Wiley Publishing Inc. (2003).

[12] J. Green, S. Bullen, R. Bovey, M. Alexander: *Excel® 2007 VBA Programmer's Reference*, Wiley Publishing Inc., Indianapolis, Indiana (2007), ISBN: 978-0-470-04643-2.

[13] Steven M. Hansen: *Mastering$^{TM}$ Excel 2003 Programming with VBA*, SYBEX Inc., Marina Village Parkway, Alameda (2004), ISBN: 0-7821-4281-8.

[14] D. Birnbaum: *Microsoft Excel VBA Programming for the Absolute Beginner*, Premier Press, United States of America (2002), ISBN: 1-931841-04-7.

[15] Henry L. Gantt: *Organizing for Work*, Hive Publishing Company, Easton, Maryland (1973).

[16] Z. Došlá, J. Kuben: *Diferenciální počet funkcí jedné proměnné*, Masarykova Univerzita, Brno (2003), ISBN 80-210-3121-2.

[17] Z. Došlá, O. Došlý: *Diferenciální počet funkcí více proměnných*, Masarykova Univerzita, Brno (2003), ISBN 80-210-2052-0.

[18] D. Todd: *Multiple Criteria Genetic Algorithms in Engineering Design and Operation*, PhD Thesis, Engineering Design Centre, University of Newcastle Upon Tyne (1997).

[19] Richard E. Rosenthal: *GAMS — A User's Guide*, GAMS Development Corporation, Washington, DC, USA (2007).

[20] S. Ólafsseon, L. Shi: *A method for scheduling in parallel manufacturing systems with flexible resources*, Transactions (2000) 32, 135-146, University of Wisconsin-Madison, Madison, USA (1998).

[21] J. M. Y. Leung, G. Zhang, X. Yang, R. Mak, K. Lam: *Optimal Cyclic Multi-Hoist Scheduling: A Mixed Integer Programming Approach*, OPERATIONS RESEARCH Vol. 52, No. 6, November–December 2004, pp. 965–976, (2003).

[22] H.A.J. Crauwels, P. Beullens, D. Van Oudheusden: *Parallel Machine Scheduling by Family Batching with Sequence-independent Set-up Times*, International Journal of Operations Research Vol. 3, No. 2, 144-154 (2006).

[23] H.P Williams: *Model Building for Mathematical Programming*, Wiley Publishing Inc. (1993), 3. edition, ISBN: 978-0-471-99788-7.

[24] Subhash C. Sarin, J. E. Kobza, H. D. Sherali: *Bi-criteria Scheduling Problems on Parallel Machines*, Virginia Polytechnic Institute and State University, Blacksburg, Virginia (1997).

[25] P. Blecha, J. Marek: *Expertní systémy a zabezpečování jakosti*, MM Průmyslové spektrum, Vol.1999, (1999), No.10, pp.68-69, ISSN 1212-2572.

[26] J. Marek, Z. Kolíbal, V.Dokoupil, R. Knoflíček: *Využití analýzy způsobilosti provozu a důsledků poruch ve strojních zařízeních*, Znalectvo, Vol.1997, (1997), No.3, pp.23, ISSN 1335-1133

[27] P. Weaver: *A Brief History of Scheduling*, Mosaic Project Services Pty Ltd, South Melbourne, Australia (2006).

# Dodatek A

# Teorie

## A.1. Účelová funkce a její extrémy

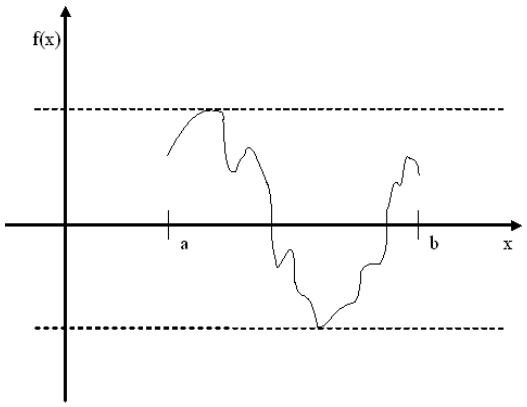
Mírou dosažení cíle je v modelech operačního výzkumu *účelová (kriteriální) funkce*. Řešení problému často spočívá v nalezení extrému této funkce. V nejjednodušším případě se jedná o maximalizaci zisku ze systému, nebo o minimalizaci nákladů na systém. V případě, kdy existuje více rozhodovacích kritérií mluvíme o vícekriteriální účelové funkci. Bohužel jenom výjimečně se extrémy pro jednotlivá kritéria shodují. Snaha o minimalizaci nákladu v jedné oblasti jde často proti snaze minimalizovat náklady v oblasti druhé. Proto často volíme mezi těmito oblastmi vhodný kompromis. Samotnou existenci a dělení extrémů na *lokální* a *globální*, vysvětlují následující definice a věty.

Věta A.1 *(1. Weierstrassova) Nechť $f$ je spojitá funkce na uzavřeném intervalu $\langle a, b\rangle$, pak $f$ je omezená na tomto intervalu.*

Věta A.2 *(2. Weierstrassova) Nechť $f$ je spojitá funkce na uzavřeném intervalu $\langle a, b\rangle$, pak $f$ nabývá na tomto intervalu svého maxima a minima.*

Důkaz těchto vět lze nalézt např. v [16].

*Poznámka:* V další definici znamená $D(f)$ definiční obor funkce, což je množina všech hodnot, pro které je funkce $f$ definována.

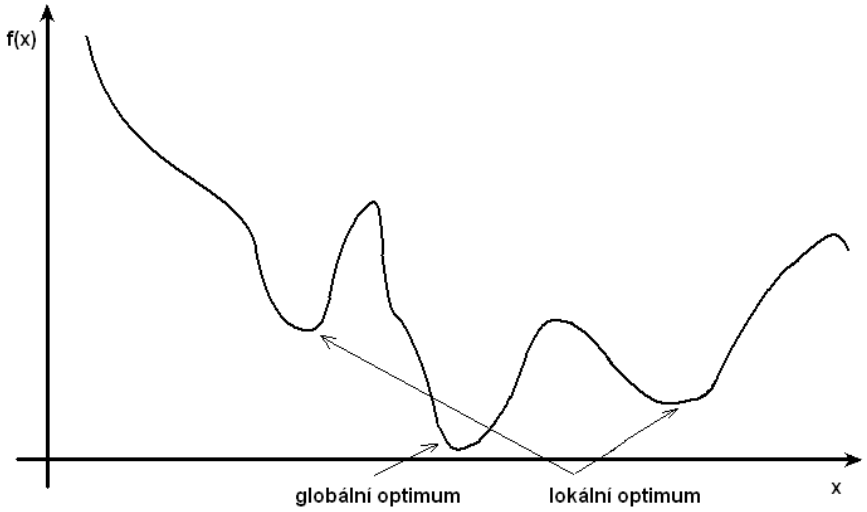



Obrázek A.1: *2. Weierstrassova věta, existence minima a maxima funkce*

Definice A.1 *Řekneme, že funkce $f$ má v bodě $x_0 \in \mathbb{R}$ lokální maximum (resp. minimum) právě když existuje ryzí okolí $O(x_0) - \{x_0\}$ takové že $O(x_0) - \{x_0\} \in D(f)$ a zároveň pro $\forall x \in O(x_0) - \{x_0\}$ platí $f(x) \leq f(x_0)$ (resp. $f(x) \geq f(x_0)$).*

Definice A.2 *Nechť $f(x)$ je funkce a $M \in D(f)$. Jestliže existuje bod $x_0 \in M$ tak, že pro všechna $x \in M$ platí $f(x) \leq f(x_0)$ (resp. $f(x) \geq f(x_0)$), pak řekneme, že funkce $f(x)$ nabývá v bodě $x_0$ globálního maxima (resp. globálního minima) na množině $M$.*

*Poznámka:* Globální extrém může být na množině M více než jeden. Lokální minima a maxima funkce $f$ se nazývají lokální extrémy a globální minima a maxima funkce $f$ se nazývají globální extrémy.




Obrázek A.2: *Extrémy funkce jedné proměnné*

Omezující podmínky, které musí systém při dosahování cíle respektovat, jsou v modelech vyjádřeny pomocí nerovností nebo rovnic. Extrém funkce, který musí vyhovovat dodatečným podmínkám zadaných rovnicemi nebo nerovnostmi, označujeme jako *extrém vázaný*. Hledáme tedy extrémy funkce $z = f(x, y)$, přičemž tyto extrémy musí vyhovovat dodatečné podmínce zadané rovnicí $g(x, y) = 0$.

Definice A.3 *Pokud v okolí bodu $[x_0, y_0]$ platí $f(x, y) \leq f(x_0, y_0)$ a současně $g(x_0, y_0) = 0$, pak se v bodě $[x_0, y_0]$ nachází vázané lokální maximum funkce $f$. Pokud v okolí bodu $[x_0, y_0]$ platí $f(x, y) \geq f(x_0, y_0)$ a současně $g(x_0, y_0) = 0$, pak se v bodě $[x_0, y_0]$ nachází vázané lokální minimum funkce $f$.*

Definice platí analogicky i v případě funkcí více proměnných, lze ji a další matematickou teorii o extrémech funkce jedné nebo více proměnných nalézt [16],[17].

## A.2. Lineární programování

Lineární programování se zabývá problémy souvisejícími s hledáním vázaných extrémů lineárních funkcí více proměnných, jejichž omezující podmínky mají tvar lineárních rovnic

a nerovností. Tyto problémy řešil nejprve teoretický fyzik J. B. J. Fourier v letech 1826-1888 v souvislosti s analytickou mechanikou a teorií pravděpodobnosti. V třicátých letech dvacátého století byly řešeny lineární optimalizační problémy v ekonomice, ve čtyřicátých potom dopravní problémy. Rozvoj efektivních lineárních metod dovršil G. B. Dantzig, který vytvořil tzv. simplexovou metodu. V současné době neustává snaho o další zefektivňování, především ve spojení s rozvojem výpočetní techniky.

## A.2.1. Obecná formulace úlohy lineárního programování

Problém lineární programování můžeme definovat jako problém maximalizování nebo minimalizování lineární funkce podmíněné lineárními omezeními viz [1],[2],[4].

DEFINICE A.4 *Definujeme lineární maximalizační program ve tvaru*

$$\max\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\}. \tag{A.2.1}$$

*Kde* $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$.
*Lineární maximalizační úlohu můžeme napsat ve tvaru (A.2.1), kde* $a_{i,j}, b_i, c_j$, *pro* $i = 1, 2, \ldots, m$ *a* $j = 1, 2, \ldots, n$, *jsou daná reálná čísla a nechť* $I_1 \subset I = \{1, 2, \ldots, m\}$ , $J_1 \subset J = \{1, 2, \ldots, n\}$.

$$\sum_{j=1}^{n} c_j x_j = c_1x_1 + c_2x_2 + \cdots + c_nx_n, \tag{A.2.2}$$

*na množině řešení soustavy lineárních rovnic a nerovností*

$$\sum_{j=1}^{n} a_{i,j}x_j \leq b_i \qquad i \in I_1 \tag{A.2.3}$$

$$\sum_{j=1}^{n} a_{i,j}x_j = b_i \qquad i \in I - I_1 \tag{A.2.4}$$

$$x_j \geq 0 \qquad j \in J_1. \tag{A.2.5}$$

*Úlohu nazýváme maximalizační úlohou lineárního programování ve smíšeném tvaru, jestliže* $I \neq \emptyset$,$I_1 \neq I$ *nebo* $J_1 \neq J$. *V případě lineárního programování pro* $I \neq \emptyset$ *a* $J_1 = J$, *odpadnou podmínky typu (A.2.3) a maximalizační úlohu nazýváme maximalizační úlohou lineárního programování v rovnicovém tvaru.*

$$\max\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\}. \tag{A.2.6}$$

*V případě* $I_1 = I$ *a* $J_1 = J$ *odpadá podmínka typu (A.2.4) a maximalizační úlohu nazýváme maximalizační úlohou lineárního programování ve tvaru nerovností. Koeficienty* $a_{i,j}$ *se obvykle nazývají strukturální koeficienty, koeficienty* $b_i$ *nazýváme kapacitní limity a koeficienty* $c_j$ *cenovými koeficienty.*

Každou úlohu lineárního programování ve smíšeném tvaru nebo ve tvaru nerovností můžeme převést na úlohu v rovnicovém tvaru těmito úpravami:

1. Podmínky typu
$$\sum_{j=1}^{n} a_{i,j}x_j \geq b_i \quad i \in I_1$$
a
$$\sum_{j=1}^{n} a_{i,j}x_j + x_{n+i} = b_i \qquad x_{n+i} \geq 0 \quad i \in I_1,$$
vymezují stejnou množinu n-rozměrných vektorů o složkách $x_1, x_2, \ldots, x_n$. Zavedeme proměnné $x_{n+i}$ pro všechny nerovnosti (A.2.3), které nazýváme doplňkové proměnné. Doplňkové proměnné mají v kriteriální funkci (A.2.2) koeficienty $c_{n+i}, i \in I_1$.

2. Každou proměnnou $x_j$ pro $j \notin J_1$ můžeme zapsat ve tvaru
$$x_j = x_j^+ - x_j^-,$$
kde $x_j^+ \geq 0, x_j^- \geq 0$. Pro $j \notin J_1$ dosadíme proměnnou $x_j$ rozdíl dvou nezáporných proměnných $x_j^+ - x_j^-$ do podmínek (A.2.3), (A.2.4) i do kriteriální funkce (A.2.2). I když v tomto případě není vztah mezi původní proměnnou $x_j$ a proměnnými $x_j^+ - x_j^-$ vzájemně jednoznačný, není to na překážku řešení úlohy.

   Podobně můžeme proměnnou $x_j$ v případě $I_1 \neq I$ vyjádřit pomocí jedné z rovnic (A.2.4) a toto její vyjádření dosadit do všech ostatních podmínek i do účelové funkce. Dostaneme tak úlohu lineárního programování s n-1 proměnnými a m-1 omezeními.

Definice A.5 *Množinu* $M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ *nazveme množinou přípustných řešení, její prvky pak přípustnými řešeními úlohy (A.2.1).*

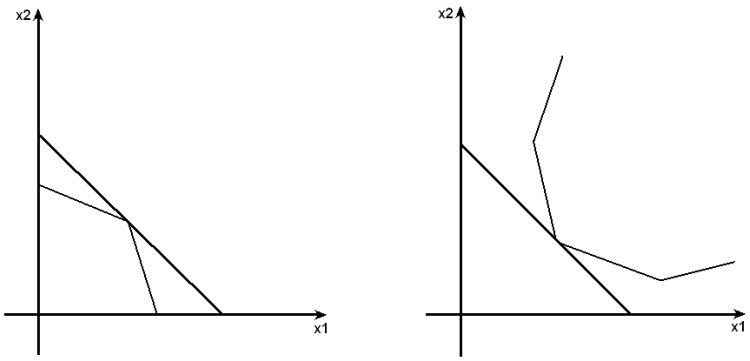
*Poznámka:* Definice 1.2 platí i pro maximalizační úlohou lineárního programování v rovnicovém tvaru (A.2.6).

Definice A.6 *Přípustné řešení* $\mathbf{x}^* \in M$ *nazveme optimálním řešením úlohy (A.2.1) jestliže*
$$\mathbf{c}^T\mathbf{x}^* \geq \mathbf{c}^T\mathbf{x} \qquad \forall x \in M.$$

*Poznámka:* Kromě existence a stanovení maximální hodnoty účelové funkce nás také zajímá existence, vlastnosti a výpočet hodnot vektoru $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$, ve kterých maximum nastává.

Věta A.3 *Pro libovolnou množinu* $M \subset \mathbb{R}^n$ *a libovolnou funkci* $z : M \to \mathbb{R}^1$ *platí*
$$\min_{\mathbf{x}\in M} z(\mathbf{x}) = -\max_{\mathbf{x}\in M}(-z(\mathbf{x}))$$

*Pokud extrém existuje můžeme převést maximalizační úlohu na minimalizační úlohu lineárního programování. Nerovnosti typu*
$$\sum_{j=1}^{n} a_{i,j}x_j \leq b_i$$
*upravíme vynásobením číslem -1.*




Obrázek A.3: *Maximalizace a minimalizace funkce*

Dále uvedená teorie slouží ke konstrukci metod řešení lineárního programování. Věty jsou uvedeny bez důkazu, ty lze nalézt například v [2].

DEFINICE A.7 *Řekneme, že množina $S \subset \mathbb{R}^n$ je konvexní, pokud s každými dvěmi body $x, y \in M$ obsahuje i celou úsečku spojující body $x$ a $y$. Jinými slovy jsou-li $x, y \in M$ a $\lambda \in \langle 0, 1\rangle$ tak i $\lambda x + (1 - \lambda)y \in M$.*

DEFINICE A.8 *Konvexní polyedrická množina $M \subset \mathbb{R}^n$ je taková množina, kterou lze vyjádřit jako průnik konečného počtu uzavřených poloprostorů.*

VĚTA A.4 *Množina přípustných řešení úlohy lineárního programování ve tvaru nerovností i ve smíšeném nebo rovnicovém tvaru je konvexní polyedrická množina.*

VĚTA A.5 *Množina $M^*$ optimálních řešení úlohy*

$$\max\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\right\}$$

*je konvexní polyedrická množina.*

DEFINICE A.9 *Nechť $S \subset \mathbb{R}^n$ je libovolná množina. Bod $s \in S$ nazveme krajním bodem množiny $S$, jestliže neexistují body $x, y \in S$ a číslo $\lambda \in (0, 1)$ tak, že $x \neq y$ a $s = \lambda x + (1 - \lambda)y$.*

VĚTA A.6 *Konvexní polyedrická množina má konečný počet krajních bodů.*

DEFINICE A.10 *Přípustné řešení $\mathbf{x} \in M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ nazveme základním řešením úlohy lineárního programování v rovnicovém tvaru, jestliže jsou sloupce matice $\mathbf{A}$ s indexy odpovídajícími nenulovým složkám $\mathbf{x}$ lineárně nezávislé.*

DEFINICE A.11 *Bod $\mathbf{x} \in M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ je krajním bodem množiny $M$ právě tehdy, je-li základním řešením.*

VĚTA A.7 *(základní věta lineárního programování) Pro úlohu lineárního programování maximalizovat $\mathbf{c}^T\mathbf{x}$ na množině $M = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ platí jedna ze tří možností:*

1. *$M = \emptyset$,*

2. $M \neq 0 \wedge \sup_{\mathbf{x}\in M} \mathbf{c}^T\mathbf{x} = +\infty$
   *(tj.* $M^* = \left\{\mathbf{x} \in M \mid \mathbf{c}^T\mathbf{x}^* = \max_{\mathbf{x}\in M} \mathbf{c}^T\mathbf{x}\right\} = \emptyset$*),*

3. $M^* \neq \emptyset$

*Kromě toho platí:*

1. *Je-li* $M \neq \emptyset$*, pak existuje základní přípustné řešení,*

2. *Je-li* $M^* \neq \emptyset$*, pak existuje základní optimální řešení.*

*Poznámka:* Pro optimální řešení z předešlé věty plyne, že má-li lineární programování optimální řešení, pak lze toto optimální řešení najít mezi základními řešeními. Jinak řečeno, jestliže existují optimální řešení, pak mezi nimi existuje alespoň jeden krajní bod.

### A.2.2. Základní idea simplexové metody

Tuto metodu využil G. B. Dantzig s využitím myšlenek Jordanovi modifikace Gaussovy eliminační metody pro řešení soustav lineárních algebraických rovnic. Lze ji snadno popsat geometricky. Předpokládejme, že známe krajní bod $x_0$ množiny přístupných řešení $M$. Z tohoto krajního bodu vychází konečné množství hran množiny $M$, z nichž každá buď obsahuje jediný další krajní bod množiny $M$, nebo je neomezená. Jestliže na některé neomezené hraně existuje bod, pro který je hodnota účelové funkce menší než $c^T x_0$, nemá úloha optimální řešení a postup končí. V opačném případě je hledán sousední krajní bod, pro který je hodnota účelové funkce menší než $c^T x_0$. Nechť je to krajní bod $x_1$. Pokud neexistuje krajní bod $x$ s vlastností $c^T x < c^T x_0$, je hledaným optimálním řešením.

## A.3. Celočíselné programování

Celočíselné programování je takové matematické programování, kde alespoň jedna rozhodovací proměnná nabývá pouze celočíselných hodnot. Tyto problémy zaznamenaly výrazný rozvoj od začátku devadesátých let dvacátého století. Z praktických důvodů často potřebujeme celočíselné hodnoty rozhodovacích proměnných. Je to především z důvodu, že základní jednotky těchto proměnných jsou fyzicky nedělitelné. Celočíselné programování má ale dobré využití při jiných problémech, například pro kombinatorické problémy. Klasickým příkladem kombinatorických úloh je tzv. Problém obchodního cestujícího, nebo úlohy rozvrhování výroby.

### A.3.1. Obecná formulace úlohy celočíselného programování

Definice A.12 *Definujeme celočíselný matematický program ve tvaru*

$$\textit{minimalizovat } f(x_1, x_2, \ldots x_n)$$

*za podmínek*

$$g(x_1, x_2, \ldots x_n) \leq 0 \qquad i = 1, 2, \ldots m$$
$$x_j \in D_j \subseteq \mathbb{Z} \qquad j \in J,$$

*kde $J \neq \emptyset$ a $\mathbb{Z}$ je množina celých čísel. Pokud jsou podmínkou celočíselnosti vázány všechny proměnné, jedné se o úplně (ryze) celočíselnou úlohu. Jestliže se podmínka na celočíselnost týká pouze některých proměnných, hovoříme o částečně (smíšené) celočíselné úloze. Dále členíme celočíselné programování podle charakteru funkcí $f, g_1, g_2, \ldots, g_m$ na lineární a nelineární.*

Definice A.13 *Lineární program (1.1.1) s přidáním podmínky celočíselnosti nazveme lineární celočíselný program*

$$\max\left\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in D\right\},$$

*kde $D \subseteq \mathbb{Z}$.*

### A.3.2. Základní idea metody větví a mezí

Je interační metoda pro nalezení globálního extrému funkce $f(x_1, x_2, \ldots, x_n)$ na množině přípustných řešení $M$. Tato metoda je založena na opakovaném sledu dvou operací

- větvení - zprvu se množina $M$ a následně její vybraná podmnožina rozkládá na po dvou disjunktní podmnožiny,
- omezování - pro každou podmnožinu získanou předchozí operací určuje dolní (při minimalizaci), resp. horní (při maximalizaci) mez hodnot funkce $f(x_1, x_2, \ldots, x_n)$ na této podmnožině.

Pro další rozklad se volí podmnožina z nejnižší dolní, resp. nevyšší horní mezí. Cílem je najít takové přípustné řešení, pro něž hodnota účelové funkce není větší než dolní meze, resp. není menší než horní meze u všech dosud nerozložených podmnožin, neboť takové řešení je optimální.

## A.4. Optimalizační kritéria

zakázka $j = 1, \ldots n$
rozvrh $S$
čas dokončení zakázky $C_j(S)$
žádaný čas dokončení zakázky (due date) $d_j$, po jehož překročení může následovat penalizace
poslední termín dokončení zakázky (deadline) $D_j$, jehož překročení nemůže nastat
váha $w_j$, která určuje důležitost zakázky
čas připuštění $r_j$, před kterým není zakázka k dispozici

Při plánování úloh na zdroje používáme pro rozhodování tzv. optimalizační kritéria. Tato kritéria nám umožňují stanovit kvalitu dosaženého řešení a porovnat jej s ostatními generovanými rozvrhy. Tato kritéria se dělí do skupin podle toho, které parametry chceme optimalizovat.

- *Maximální čas konce úloh (makespan)*, $C_{max} = \max\{C_1, \ldots, C_n\}$. Minimalizace makespan často maximalizuje výkon a zajišťuje rovnoměrné zatížení strojů.

- *Zpoždění (lateness)*, $L_j = C_j - d_j$, minimalizace zpoždění snižuje makespan a zvyšuje výkon.
- *Nezáporné zpoždění (tardiness)*, $T_j = \max\{C_j - d_j, 0\}$, minimalizace maximálního nezáporného zpoždění je častým optimalizačním kritériem pro úlohy s termíny dostupnosti a dokončení.
- *Smluvní pokuta* $U_j = 1$ jestliže $C_j \geq d_j$, $U_j = 1 = 0$ jinak.
- $\sum_{j=1}^{n} T_j$ *celkové zpoždění úloh* představuje kritérium, kdy je snahou minimalizovat celkové nezáporné zpoždění úloh.
- $\sum_{j=1}^{n} w_j T_j$ *vážené zpoždění úloh* zohledňuje váhy úloh při minimalizaci celkového nezáporného zpoždění.
- $\sum_{j=1}^{n} C_j$ *součet času dokončení úloh* představuje minimalizaci součtu časů konce všech úloh.
- $\sum_{j=1}^{n} w_j C_j$ *vážený součet časů dokončení úloh* zohledňuje váhy úloh při minimalizaci součtu času konce úloh.
- $\sum_{j=1}^{n} w_j U_j$ *vážený počet zpožděných zakázek*

Více optimalizačních kritérii lze nalézt např. v [24].

## A.5. Výpočtová náročnost úloh operačního výzkumu

S rozvojem výpočetní techniky byl spojen velký rozvoj úloh operačního výzkumu. Řešení těchto úloh se rozděluje do dvou způsobů, a to jednak *exaktní matematické metody*, jednak *heuristické metody*. Zatím co exaktní matematické metody hledají optimální řešení (globální optimum), metody heuristické se snaží hledat dobré, ne vždy optimální řešení (lokální optimum). Je to dáno především podle časové náročností výpočtů problémů. Proto existuje náročnostní rozdělení problémů operačního výzkumu. Základem této teorie byl článek S. Cooka v roce 1971. Je dokázáno, že určité přesně definované matematické problémy jsou nerozhodnutelné a tudíž principiálně nemůže existovat algoritmus schopný řešit všechny případy těchto problémů. Cook analyzoval třídy jednotlivých problémů, které se dnes označují jako P a NP. Třída P obsahuje všechny problémy, které lze řešit v reálném čase a třída NP obsahuje všechny problémy, pro které v reálném čase řešení nalézt nelze. Např. problémy plánování a rozvrhování patří obecně mezi NP-úplné problémy, což znamená, že přesné optimalizační metody jsou prakticky použitelné pouze pro úlohy omezeného rozsahu.

Pro tuto práci je zajímavá výpočetní náročnost exaktních matematických metod v problémech lineárního a celočíselného matematického programování. Pro lineární programování efektivní software může vyřešit problémy s více než 1 000 000 proměnnými a omezeními. Pro celočíselné programování můžeme v rozumném čase vyřešit problémy s několika tisíci celočíselnými proměnnými.

# Dodatek B

# Zdrojové kódy

## B.1. GAMS

### B.1.1. Průchod jedné zakázky firmou

```
$Title one_job_scheduling
================================NAČÍTÁNÍ DAT Z MS EXCEL==================================
$onecho > one_job_excel.txt
set=m rng=b12:j12 cdim=1
set=n rng=a13:a19 rdim=1
par=t rng=a2 rdim=1
par=incidence_matrix rng=a12 cdim=1 rdim=1
$offecho

$call gdxxrw.exe one_job_excel.xls @one_job_excel.txt
$gdxin one_job_excel.gdx

sets
         n(*) nodes
         m(*) activities ;

$load n m
display n,m;

parameter t(m) activity time
          incidence_matrix(n,m) incidence_matrix;

$LOAD t incidence_matrix

Display t, incidence_matrix;

$GDXin

variable
          x(n) start time
          z~finish time;

positive variable
          x(n);

equations
         finish_time     finish total time
         balance         balance equations;
================================MATEMATICKÝ MODEL=======================================
finish_time ..  z~=e= x('6');
balance(m) ..   sum(n,x(n)*incidence_matrix(n,m)) =g= t(m);
================================ZALOVÁNÍ VÝPOČTU========================================
model scheduling one_job_ /all/ ;
solve scheduling minimizing z~using lp;
display  x.l;
=======================NAČTENÍ DAT DO MS EXCELU A~OTEVŘENÍ SEŠITU=======================
```

```
execute_unload 'one_job_excel.gdx',x z~;
execute 'gdxxrw.exe one_job_excel.gdx var=x.l rng=b23 var=z rng=b26' ;
execute '=shellexecute one_job_excel.xls';
```

## B.1.2. Průchod více zakázek firmou

```
$Title  scheduling
==============================NAČÍTÁNÍ DAT Z MS EXCEL================================
$onecho > scheduling.txt
set=m rng=Input!c7:l7 cdim=1
set=n rng=Input!b30:b40 rdim=1
set=j rng=Input!b8:b27 rdim=1
par=t rng=Input!b7:l27 cdim=1 rdim=1
par=incidence_matrix1 rng=Input!b29 cdim=1 rdim=1
par=g rng=Input!L28 Dim=0
$offecho

$call gdxxrw.exe scheduling.xls @scheduling.txt
$gdxin scheduling.gdx
*-------------------------------------------------------------------------------------------
sets
         m(*)     machines
         j(*)     job
         n(*)     nodes  ;

alias (j, p);
alias (j, k);

$load n m j
display m,n,j;

parameter t(j,m)                 activity time
          incidence_matrix1(n,m) from_node-to_node
          g                      largenumber ;

$LOAD t incidence_matrix1 g

Display t, incidence_matrix1, g;

$GDXin

parameter
          incidence_matrix2(n,m)  node_to;

incidence_matrix2(n,m)$(incidence_matrix1(n,m) > 0) = 1 ;
incidence_matrix2(n,m)$(incidence_matrix1(n,m)  <= 0) = 0 ;
Display incidence_matrix2;

parameter
          incidence_matrix3(n,m) node_from ;

incidence_matrix3(n,m)$( incidence_matrix1(n,m) < 0) = 1 ;
incidence_matrix3(n,m)$(incidence_matrix1(n,m)  >= 0) = 0 ;

Display incidence_matrix3;

variable z~total lead time(makespan)
         x(j,n)    max complete time of job j in node n
         xx(j)     finish time of job j
         y(k,p,m)  indicator variable for job precedence
         u(j,m)    start procesing time on machine x
         uu(j,m)   finish procesing time on machine x;

binary variable
        y(k,p,m);

positive variable
        x(j,n)
        u(j,m)
        uu(j,m)
```

```
       xx(j);
*-------------------------------------------------------------------------------------------------------
equations
  total_time                total lead time(makespan) - objective function
  finish_time               finish time of job j
  binary_condition          condition for binary variables
  sequential1               sequential operations on different machines
  sequential2               sequential operations on different machines
  sequential3               sequential operations on different machines
  sequential4               sequential operations on one machine
  sequential5               sequential operations on one machine;

====================================MATEMATICKÝ MODEL===================================
total_time .. z~=e= sum(j,xx(j));

finish_time(j,n) .. xx(j) =g= x(j,n);

*-------------------------------------------------------------------------------------------------------

sequential1(m,j).. u(j,m) + t(j,m) =e= uu(j,m);

sequential2(m,j).. sum(n,x(j,n)*incidence_matrix3(n,m)) =l=  u(j,m);

sequential3(m,j)..  sum(n,x(j,n)*incidence_matrix2(n,m)) =g= uu(j,m);

*-------------------------------------------------------------------------------------------------------

binary_condition(m,k,p)$(not ord(k) ge ord(p)).. y(k,p,m) + y(p,k,m) =e= 1 ;

*-------------------------------------------------------------------------------------------------------

sequential4(m,k,p)$(not ord(k) eq ord(p)).. uu(k,m) - u(p,m) =l= g*(1-y(k,p,m));

sequential5(m,k,p)$(not ord(k) eq ord(p)).. uu(k,m) - u(p,m) - (-g-1)*(y(k,p,m)) =g= 1 ;

*-------------------------------------------------------------------------------------------------------
====================================NASTAVENÍ VÝPOČTU====================================
option limrow = 10000;

option optcr=0 ;

option optca=0 ;

*-------------------------------------------------------------------------------------------------------
====================================ZALOVÁNÍ VÝPOČTU====================================
model scheduling n_jobs /all/ ;

option mip = cplex;
solve scheduling minimizing z~using mip;
display  x.l,u.l,uu.l,y.l,xx.l,scheduling.resusd,scheduling.iterusd,scheduling.numnz;
======================NAČTENÍ DAT DO MS EXCELU A~OTEVŘENÍ SEŠITU======================
execute_unload 'scheduling.gdx',x u~uu y z~xx;
execute 'gdxxrw.exe scheduling.gdx SQ=N var=x.l rng=Output!b4 var=u.l rng=Output!b26 var=uu.l
rng=Output!b48 var=y rng=Output!o2 var=z rng=Output!b2 var=u.l rng=Input!T4 cdim=0 var=uu.l
rng=Input!X4 cdim=0 var=xx.l rng=Output!b71 cdim=0 ';
execute '=shellexecute scheduling.xls';
```

# B.2. Microsoft Visual Basic

## B.2.1. Ganttův diagram

```
Sub GANTT_CHART()

  Range("W4:W100").Select
    Selection.ClearContents

j = Cells(3, "B").Value
m = Cells(4, "B").Value

 For jjj = 1 To j
```

```
  For mm = 1 To m

   x = jjj * m - m

 Range("W" + CStr(3 + x + mm)).Select
  ActiveCell.FormulaR1C1 = Sheets("input").Range("M" + CStr(7 + jjj)).FormulaR1C1

Next mm

Next jjj

  j = Cells(3, "B").Value
  m = Cells(4, "B").Value

  c = j * m

  Range("O19:Q300").Select
    Selection.ClearContents
   Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    Selection.Borders(xlEdgeLeft).LineStyle = xlNone
    Selection.Borders(xlEdgeTop).LineStyle = xlNone
    Selection.Borders(xlEdgeBottom).LineStyle = xlNone
    Selection.Borders(xlEdgeRight).LineStyle = xlNone
   Selection.Borders(xlInsideVertical).LineStyle = xlNone
   Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

  Range("O19").Select
  ActiveCell.FormulaR1C1 = "=R[-15]C[6]"

  Range("O19").Select
    Selection.AutoFill Destination:=Range("O19:O" + CStr(18 + c)), Type:=xlFillDefault

  Range("P19").Select
    ActiveCell.FormulaR1C1 = "=R[-15]C[6]"

  Range("P19").Select
    Selection.AutoFill Destination:=Range("P19:P" + CStr(18 + c)), Type:=xlFillDefault

    Range("Q19").Select
    ActiveCell.FormulaR1C1 = "=R[-15]C[9]"

   Range("Q19").Select
    Selection.AutoFill Destination:=Range("Q19:Q" + CStr(18 + c)), Type:=xlFillDefault
 Range("O19:Q" + CStr(18 + c)).Select
 Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .Weight = xlThin
```

```
        .ColorIndex = xlAutomatic
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With

Range("A1").Select

    Dim vTimeData As Variant
    Dim i As Integer
    Dim sRoom As String
    Dim vLastEndTime As Variant
    Dim oSeries As Series

    '\ set up
    Application.ScreenUpdating = False
    Application.DisplayAlerts = False

    '\ sort data
    With Worksheets("input").Range("U3").CurrentRegion
        .Sort Key1:="Machine", Key2:="Start Time", Header:=xlYes
        vTimeData = .Value
    End With

    With Worksheets("input").Range("Y3").CurrentRegion
         .Sort Key1:="Machine", Key2:="end Time", Header:=xlYes
         vTimeData = .Value
    End With

    '\ create chart data worksheet
    With Worksheets("input").Range("O18").CurrentRegion
        vTimeData = .Value
        Worksheets.Add
        On Error Resume Next
            Worksheets("ChartData").Delete
            Charts("TimeChart").Delete
        On Error GoTo 0
        ActiveSheet.Name = "ChartData"
        .Columns(1).AdvancedFilter Action:=xlFilterCopy, _
            CopyToRange:=Range("A1"), Unique:=True
    End With
    Range("a1").Select
    For i = 2 To UBound(vTimeData)
        If vTimeData(i, 1) <> Selection.EntireRow.Cells(1) Then
            Selection.Offset(1).EntireRow.Cells(1).Select
            Selection.Value = vTimeData(i, 1)
            vLastEndTime = 0
        End If
        Selection.Offset(0, 1).Select
        Selection.Value = vTimeData(i, 2) - vLastEndTime
        Selection.Offset(0, 1).Select
        Selection.Value = vTimeData(i, 3) - vTimeData(i, 2)
        vLastEndTime = vTimeData(i, 3)
    Next i
    With Selection.CurrentRegion
        .Offset(0, 1).NumberFormat = ""
        .Columns(2).Cells(1) = "Start Time"
        For i = 3 To .Columns.Count
            If i Mod 2 <> 0 Then
                .Columns(i).Cells(1) = "Used"
            Else
                .Columns(i).Cells(1) = "Not Used"
            End If
        Next i
    End With

    '\ create Gantt chart(from chart data)
    Charts.Add
    ActiveChart.Name = "TimeChart"
```

```
   ActiveChart.ChartWizard Source:=Sheets("chartdata").Range("A1").CurrentRegion, _
       Gallery:=xlBar, Format:=3, PlotBy:=xlColumns, CategoryLabels _
       :=1, SeriesLabels:=1, HasLegend:=2

  '\ box properties
   For Each oSeries In ActiveChart.SeriesCollection
       If oSeries.PlotOrder Mod 2 <> 0 Then
           oSeries.Border.LineStyle = xlNone
           oSeries.Interior.ColorIndex = xlNone
       Else
           oSeries.Border.LineStyle = 1
           oSeries.Border.ColorIndex = 1

       End If

    Next oSeries

  '\ box description
   ActiveChart.ApplyDataLabels AutoText:=True, LegendKey:=False, _
       HasLeaderLines:=False, ShowSeriesName:=False, ShowCategoryName:=False, _
       ShowValue:=True, ShowPercentage:=False, ShowBubbleSize:=False
     With Selection.Font
       .Name = "Arial"
       .Size = 12
     End With

   cc = Sheets("input").Range("B4").FormulaR1C1
   bb = Sheets("input").Range("B3").FormulaR1C1

   For j = 0 To bb - 1
    ActiveChart.SeriesCollection(2 * j + 1).DataLabels.Select
    Selection.Delete
   Next j

   Dim jj As Integer

   '\ box color
   For jj = 1 To bb
    For j = 1 To cc

   ActiveChart.SeriesCollection(2 * jj).Points(j).Select
   Selection.Shadow = False
   Selection.InvertIfNegative = False
      With Selection.Interior
         .ColorIndex = Sheets("input").Range("W" + CStr(-bb + 3 + bb * j + jj)).FormulaR1C1
      End With

     Next j
   Next jj

      '\ Axes seting
  With ActiveChart.Axes(xlValue)
       .MajorUnitIsAuto = True
       .TickLabels.NumberFormat = ""
       .HasMajorGridlines = True
        If .HasMajorGridlines Then
           .MajorGridlines.Border.LineStyle = 6
        End If

  End With

  With ActiveChart.Axes(xlCategory)
      .ReversePlotOrder = True
  End With

Dim B As Integer

B = Sheets("output").Range("B2").FormulaR1C1

  '\ chart name
   With ActiveChart
```

```
        .HasTitle = True
        .ChartTitle.Characters.Text = "GANTT CHART"
    ActiveChart.ChartArea.Select
    End With


End Sub
```

## B.2.2. Volání GAMS z MS EXCEL

```
Sub GAMS_sum_xf()
ActiveWorkbook.Save
    Path1 = Cells(2, "N").Value
    Path2 = Cells(2, "P").Value
If Path2 = "" Then Path2 = "demo"

    Path = Path1 + "\gams scheduling1.gms license =" + Path2

    Ret = Shell(Path, 3)

   ActiveWorkbook.Close

End Sub
```

## B.2.3. Potvrzení vstupních dat

```
Sub Verify_data()

j = Cells(3, "B").Value
m = Cells(4, "B").Value
n = Cells(5, "B").Value

If j < 1 Then j = 1
If j > 20 Then j = 20
If m < 1 Then m = 1
If m > 10 Then m = 10
If n < 2 Then n = 2
If n > 11 Then n = 11
If n > m + 1 Then n = m + 1

mm = m + 66

Range("B9:B27").Select
    Selection.ClearContents
Range("D7:L7").Select
    Selection.ClearContents
Range("B32:B40").Select
    Selection.ClearContents
Range("C29:L29").Select
    Selection.ClearContents

Range("B8").Select
ActiveCell.FormulaR1C1 = "j1"
Range("C7").Select
ActiveCell.FormulaR1C1 = "mach01"
Range("C29").Select
ActiveCell.FormulaR1C1 = "mach01"
Range("B30").Select
ActiveCell.FormulaR1C1 = "1"
Range("B31").Select
ActiveCell.FormulaR1C1 = "2"
Range("O18").Select
ActiveCell.FormulaR1C1 = "Machine"
Range("P18").Select
ActiveCell.FormulaR1C1 = "Start time"
Range("Q18").Select
ActiveCell.FormulaR1C1 = "End time"
Range("U3").Select
```

```
ActiveCell.FormulaR1C1 = "Machine"
Range("V3").Select
ActiveCell.FormulaR1C1 = "Start time"
Range("W3").Select
ActiveCell.FormulaR1C1 = "Color"
Range("Y3").Select
ActiveCell.FormulaR1C1 = "Machine"
Range("Z3").Select
ActiveCell.FormulaR1C1 = "End time"

If j > 1 Then
Range("B8").Select
    Selection.AutoFill Destination:=Range("B8:B" + CStr(7 + j)), Type:=xlFillDefault
End If

If m > 1 Then
Range("C7").Select
   Selection.AutoFill Destination:=Range("C7:" + Chr(mm) + CStr(7)), Type:=xlFillDefault
End If

If m > 1 Then
Range("C29").Select
   Selection.AutoFill Destination:=Range("C29:" + Chr(mm) + CStr(29)), Type:=xlFillDefault
End If

If n > 2 Then
Range("B30:B31").Select
    Selection.AutoFill Destination:=Range("B30:B" + CStr(29 + n)), Type:=xlFillDefault
  End If
Range("B5").Select
ActiveCell.FormulaR1C1 = n
Range("B4").Select
ActiveCell.FormulaR1C1 = m
Range("B3").Select
ActiveCell.FormulaR1C1 = j

Range("L28").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-20]C[-9]:R[-1]C)"

Range("A1").Select
End Sub
```

## B.2.4. Generování dat

```
Sub Generate_data()

 Range("C8:L27").Select
    Selection.ClearContents

 lo = Cells(3, "H").Value
 up = Cells(4, "H").Value


 j = Cells(3, "B").Value
 m = Cells(4, "B").Value

 mm = m + 66

 For jj = 0 To j - 1
 For mmm = 0 To m - 1
 Range(Chr(67 + mmm) + CStr(8 + jj)).Select
       ActiveCell.FormulaR1C1 = Int((up - lo + 1) * Rnd + lo)
   ' ActiveCell.FormulaR1C1 = Int((20 - 5 + 1) * Rnd + 5)

Next mmm
Next jj

  Range("C8:L27").Select
   With Selection
        .HorizontalAlignment = xlCenter
```

```
   End With

Range("A1").Select

End Sub
```

## B.2.5. Generování barvy

```
Sub Generate_color()
 Range("M8:M27").Select
    Selection.ClearContents
    Selection.Interior.ColorIndex = xlNone

 lo = Cells(3, "L").Value
 up = Cells(4, "L").Value

 j = Cells(3, "B").Value

' Range("M8").Select
 '      ActiveCell.FormulaR1C1 = Int((up - lo + 1) * Rnd + lo)
 Dim x As Boolean
 For jj = 0 To j - 1


 Do
 x = True
   Range("M" + CStr(8 + jj)).Select
       ActiveCell.FormulaR1C1 = Int((up - lo + 1) * Rnd + lo)
```