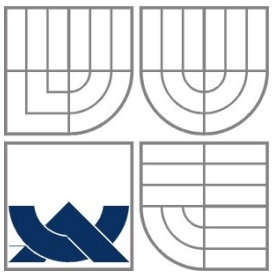


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPTIMALISACE VÝROBNĚ-MONTÁŽNÍ LINKY

OPTIMIZATION OF THE PRODUCTION-ASSEMBLY LINE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **BC. ZBYNĚK DOLEŽAL**
AUTHOR

VEDOUCÍ PRÁCE **DOC. RNDR. JINDŘICH KLAPKA, CSC.**
SUPERVISOR

BRNO 2011

1. ZADÁNÍ ZÁVĚREČNÉ PRÁCE

(na místo tohoto listu vložte originál a nebo kopii zadání Vaší práce)

2. LICENČNÍ SMLOUVA

(na místo tohoto listu vložte vyplněný a podepsaný list formuláře licenčního ujednání)

3. ABSTRAKT

Tato práce se zabývá způsobem stanovování konfigurace výrobně-montážní linky, který by minimalisoval náklady na její sestavení. Linka má umožnovat sestavování více vzájemně podobných variant téhož výrobku. Doba trvání každé operace je konstantní, ale může se vzájemně lišit mezi jednotlivými variantami výrobku. Doba trvání cyklu je vztažena k variantě výrobku.

ABSTRACT

This work deals with the way of balancing production-assembly line, which would minimalize set up costs. The production-assembly line shall allow assembling more mutually similar variants of the same product. The time of duration of each operation is constant, but can differs between individual variants of the product. Cycle time is related to the variant of the product.

KLÍČOVÁ SLOVA

Výrobně montážní linka, optimalizace, metoda větví a mezí.

KEYWORDS

Mixed-model product-assembly line, optimization, branch and bound.

4. BIBLIOGRAFICKÁ CITACE

DOLEŽAL, Z. *Optimalisace výrobně-montážní linky*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. 65 s. Vedoucí diplomové práce doc. RNDr. Jindřich Klapka, CSc..

5. PROHLÁŠENÍ O PŮVODNOSTI PRÁCE

1. Autor této diplomové práce prohlašuje, že toto dílo vytvořil samostatnou vlastní tvůrčí činností.
2. Autor potvrzuje, že listinná a elektronická verze díla je identická.

V Brně dne 27.5.2011

Autor ….....

6. PODĚKOVÁNÍ

Na tomto místě bych rád vyjádřil své poděkování doc. RNDr. Jindřichu Klapkovi, CSc. za jeho odborné vedení, podněty a teoretické poznatky nezbytné k vyhotovení této diplomové práce.

Obsah:

7. ÚVOD

Assembly line balancing, neboli stanovování konfigurace montážní linky, je úloha, která spočívá v přiřazení operací určitým pracovním stanicím na montážní lince takovým způsobem, aby toto přiřazení bylo optimální podle určitého kritéria. Od doby, kdy Henry Ford zavedl montážní linku, se stanovování konfigurace montážní linky stalo důležitým optimalizačním problémem, protože rozdíly mezi různě nastavenými montážními linkami můžou znamenat velké rozdíly z ekonomického hlediska.

Výzkum v oblasti stanovování konfigurace montážní linky probíhá několik desetiletí. Za tuto dobu bylo v této oblasti vyvinuto mnoho algoritmů. Navzdory praktické důležitosti tohoto problému však existuje velmi málo komerčně dostupného softwaru zaměřeného na tento problém. Podle studie [12] z roku 2004 existují pouze dva komerčně dostupné programy pro stanovování konfigurace montážní linky, které zároveň obsahují uživatelsky přívětivé rozhraní pro práci s daty. Jeden z těchto programů je určen pouze pro Simple Assembly Line Balancing Problem (SALBP), pro stanovování konfigurace jednomodelové montážní linky.

8. ROZBOR PROBLÉMU

V této kapitole budou popsány vlastnosti výpočetního modelu [3], který je použit pro určení konfigurace vícemodelové montážní linky a matematický popis řešeného problému.

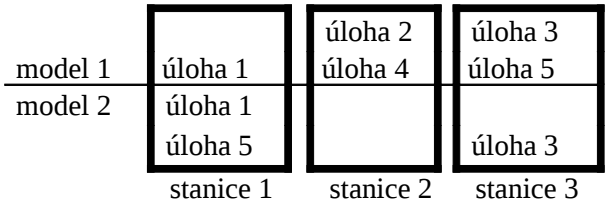
Výrobně-montážní linky jsou speciální výrobní systémy, které se typicky vyskytují v průmyslové výrobě standardizovaných výrobků, vyráběných ve velkých objemech. Během montážního procesu se výrobek pohybuje po montážní lince od stanice k stanici a v každé stanice je provedena určitá předdefinovaná množina operací. Každá operace je jednoduchá, dále nedělitelná pracovní jednotka (úloha), která obvykle vyžaduje určité pracovní nástroje a dovednost. Návrh montážní linky zahrnuje přiřazení těchto úloh stanicím, při zachování daného pořadí priorit mezi úlohami.

Nejjednodušší konfigurace montážní linky je jednomodelová (single model) montážní linka, ve které se vyrábí jedna varianta výrobku. Nejčastější cíl u navrhování jednomodelových montážních linek (single model assembly line-balancing problem, SALB-P), je maximalizace efektivity montážní linky minimalizováním požadované kapacity na jednotkové množství výrobku. Toho může být dosaženo buď minimalizací počtu pracovních stanic při dané době trvání cyklu, nebo minimalizací doby trvání cyklu při daném počtu pracovních stanic.

Vícemodelová (mixed model) montážní linka je komplexnější zařízení, ve kterém je současně montováno několik variant výrobku, označovaných jako modely. Navrhování vícemodelové montážní linky (line-balancing problem in a mixed-model environment, MALB-P) v sobě zahrnuje přiřazování úloh všech modelů pracovním stanicím. Tento problém je mnohem složitější, protože je nutné uvažovat další souvislosti mezi jednotlivými modely. MALB-P lépe popisuje moderní montážní linku, kde poptávka je charakterizována vysokou proměnlivostí a poměrně malými objemy pro každý model. Tento fenomén je možné vidět ve vzrůstajícím počtu modelů osobních automobilů, televizorů, osobních počítačů, mobilních telefonů a mnoha dalších výrobků.

Vzhledem ke vzrůstající důležitosti vícemodelové montážní linky v moderním průmyslu, bylo tomuto tématu v posledních letech věnováno několik odborných prací. Většina výzkumu na toto téma použila přístup, ve kterém každá úloha, která je společná pro několik modelů, je omezena pouze na jednu stanici. Tím se toto zadání přiblíží SALB-P. Van Zante-de Fokkert a De Kok [6] rozlišují mezi dvěma přístupy v literatuře, která se zabývá nastavením vícemodelové montážní linky: přístup s využitím kombinovaného grafu priorit a přístup s využitím upravené doby trvání úlohy. Oba přístupy transformují problém na nastavení jednomodelové montážní linky.

V této práci bude odstraněna podmínka přiřazení úlohy. Ačkoliv úloha jednotlivého modelu má být přiřazena jedné stanici, úlohy společné pro více modely můžou být přiřazeny různým stanicím. Toto budeme nadále označovat jako duplikace úlohy. S duplikací úlohy jsou spojené některé dodatečné náklady, jako např. náklady na duplikaci strojů a nástrojů, náklady spojené se zvýšením složitosti provozu linky (výcvik, učení) a náklady na řízení zásob. Proto je také změněna kriteriální funkce a cíl je minimalizovat celkové náklady na seřízení linky, místo samotného počtu stanic. V kriteriální funkci se uvažují dvě složky nákladů: náklady na stanici, které jsou přímo úměrné počtu stanic na montážní lince a náklady na stanici, které zahrnují dodatečné náklady na duplikaci úloh.



Obr. 1 Ukázka vícemodelové montážní linky.

8.1 Popis výpočetního modelu MALB

V následujícím odstavci bude specifikován výpočetní model MALB, který má za cíl minimalizovat cekové náklady spojené s nastavením montážní linky. Předpoklady výpošetního modelu jsou následující:

1. Několik podobných modelů stejného výrobku je montováno na stejné lince.
2. Doba trvání úlohy je konstantní, ale může se lišit mezi jednotlivými modely.
3. Pro každý typ modelu je daná doba trvání cyklu.
4. Doba trvání úlohy není větší než doba trvání cyklu příslušného modelu (Nebudou použity paralelní stanice).
5. Celkový součet doby trvání úloh každého modelu nesmí překročit dobu trvání cyklu daného modelu.
6. Každá úloha je pro každý model přiřazena právě jedné stanici.

Vstupy výpočetního modelu jsou následující:

1. počet modelů
2. doba trvání každé úlohy daného modelu
3. množina prioritních omezení (množina prioritních omezení může být pro lepší znázornění reprezentována jako orientovaný acyklický graf, kde každý uzel reprezentuje úlohu a každá orientovaná hrana reprezentuje prioritu mezi úlohami)
4. doba trvání cyklu
5. náklady na otevření každé stanice v montážní lince
6. náklady na duplikaci pro každou úlohu

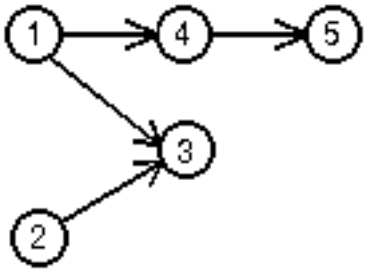
Výstup tohoto výpočetního modelu je takové nastavení montážní linky, ve kterém všechny úlohy všech modelů jsou přiřazeny pracovním stanicím a pro každý model jsou splněny podmínky doby trvání cyklu a podmínky priorit úloh.

8.1.1 Graf priority úloh

V následujícím obrázku je znázorněn příklad grafu priority úloh.
Význam tohoto uspořádání je následující:

1. úloha č. 5 může být přiřazena až po úloze č. 4 nebo zároveň s touto úlohou
2. úloha č. 4 může být přiřazena až po úloze č. 1 nebo zároveň s touto úlohou
3. úloha č. 3 může být přiřazena až po úloze č. 1 a 2 nebo zároveň s těmito úlohami

Z toho vyplývá, že úlohy č.1 a 2 musí být přiřazeny jako první.



Obr. 2 Ukázka grafu priority úloh.

8.2 Matematický popis problému

V této kapitole je uveden seznam proměnných, které jsou použity při matematickém popisu problému. Pojmenování proměnných vychází z [3].

n celkový počet odlišných úloh
m počet modelů, které se mají sestavit na montážní lince
t_{ij} doba trvání úlohy *i (i = 1, … , n)*, jestliže je prováděna na modelu *j (j = 1, …, m)*
IP_{ij} množina bezprostředních předchůdců úlohy *i* modelu *j* (graf priority úloh)
c_j požadovaný čas modelu *j* (doba trvání cyklu)
SC náklady na stanici – konstantní částka spojená s každou stanicí
TC_i náklady na úlohu – konstantní částka spojená s každou stanicí, ke které je přiřazena úloha *i*

rozhodovací proměnné:

z počet stanic, které se použijí na montážní lince

$$x_{ijk} = \begin{cases} 1 \text{ když úloha } i \text{ modelu } j \text{ je přiřazena stanici } k \\ 0 \text{ v ostatním případě} \end{cases}$$

$$\tau_{ik} = \begin{cases} 1 \text{ když úloha } i \text{ libovolného modelu je přiřazena stanici } k \\ 0 \text{ v ostatním případě} \end{cases}$$

matematická formulace problému:

$$Min\left\{SC * z + \sum_{i=1}^{n} TC_i \sum_{k=1}^{n} \tau_{ik}\right\} \quad (1)$$

Za těchto podmínek:

$$\sum_{k=1}^{n} x_{ijk} = 1 \; \forall i, j \quad (2)$$

$$\sum_{k=1}^{n} k * x_{gjk} \le \sum_{l=1}^{n} l * x_{hjl} \; \forall j, g, h, g \in IP_{hj} \quad (3)$$

$$\sum_{i=1}^{n} x_{ijk} * t_{ij} \le c_j \; \forall j, k \quad (4)$$

$$z \ge \sum_{k=1}^{n} k * x_{ijk} \; \forall i, j \quad (5)$$

$$\tau_{ik} \ge \frac{1}{m} * \sum_{j=1}^{m} x_{ijk} \; \forall i, k \quad (6)$$

$$x_{ijk}, \tau_{ik} \in \{0,1\} \quad (7)$$

Kriteriální funkce (1) minimalizuje celkové náklady (součet nákladů na stanici a na úlohy). Podmínka (2) zaručuje, že každá úloha příslušného modelu je přiřazena právě jedné stanici. Priority mezi úlohami jsou vyjádřeny množinou podmínek (3), které zajišťují, že úloha bude přiřazena určité stanici *k*, jestliže všichni její předcůdci byli přiřazeni do této stanice, nebo do předchozí stanice. Množina podmínek (4) zajišťuje, aby celkový čas úloh každého modelu na každé stanici nepřesáhl dobu trvání cyklu příslušného modelu. Podmínka (5) určuje celkový počet stanic použitých v montážní lince (tato hodnota je rovna největšímu indexu stanice všech úloh a všech modelů). Množina omezení (6) určuje, jestli úloha *i* jakéhokoliv modelu je přiřazena stanici *k*. Množina omezení (7) definuje všechny rozhodovací proměnné jako binární.

V MALB-P se může vyskytnout situace, ve které úloha určitého modelu, která může být přiřazena nějaké předchozí stanici *k*, bude přiřazena některé z následujících stanic $k + \Delta$. Tímto vznikne nevyužitý čas, prostoj nebo prázdná stanice pro tento model. Tato situace bude označována jako gapping (gap: mezera, přestávka, pauza).

SALB-P je problém se složitostí NP-Hard [7]. Protože SALB-P je speciální případ MALB-P, MALB-P je také problém se složitostí NP-Hard.

9. ANALÝZA SOUČASNÉHO STAVU

V této kapitole bude analyzován dosavadní stav výzkumu v oblasti řešení konfigurace vícemodelové výrobně-montážní linky a podobných problémů.

Vícemodelová (mixed-model) výrobně-montážní linka je speciální druh výrobně-montážní linky. Výrobně-montážní linky je možné rozdělit do čtyř skupin:

1. single-model – na montážní lince je vyráběn pouze jeden model výrobků
2. mixed-model – na montážní lince je vyráběno více modelů výrobků.
3. multi-model – na montážní lince je vyráběno více modelů výrobků. Výrobky stejného modelu jsou seskupovány do dávek po dvou nebo více výrobcích.
4. U-model – speciální případ jednomodelové linky, která je ve tvaru písmene „U“. Díky tomu může montážní dělník pracovat buď na jedné nebo druhé protilehlé straně montážní linky, nebo se může během montážního procesu přesouvat mezi stanicemi.

9.1 Metody řešení

9.1.1 Exaktní metody

1. metoda větví a mezí – patří mezi nejpoužívanější exaktní metody. Protože tento problém je NP-těžký, tato metoda může být použita jen pro malý počet vstupních parametrů.
2. matematické programování

9.1.2 Přibližné metody

Stručný přehled vybraných heuristických a metaheuristických algoritmů:

1. hladový algoritmus
2. lokální hledání
3. greedy randomized adaptive search procedure – kombinace hladového algoritmu a lokálního hledání
4. horolezecký algoritmus - nejjednodušší informovaná metoda prohledávání stavového prostoru.
5. simulované žíhání - metoda prohledávání stavového prostoru založená na simulaci žíhání oceli.
6. zakázané hledání – algoritmus založený na lokálním hledání
7. genetické algoritmy
8. ranked positional weight
9. largest-candidate rule
10. metoda Kilbridge a Wester

V následujících podkapitolách budou podrobněji popsány dvě metody pro stanovování konfigurace vícemodelové výrobně-montážní linky. Název kapitoly je odvozen od názvu práce, kde je příslušná metoda popsána.

9.1.2.1 A Balancing Problem for Mixed Model Assembly Lines with a Paced Moving Conveyor

Tato práce je v seznamu použité literatury uvedena jako [4]. V tomto problému se předpokládá vícemodelová montážní linka s pohyblivým pásem, po kterém se pohybují montované výrobky.

Jsou dány tyto vstupní údaje:

1. posloupnost modelů, které mají být denně smontovány

2. úlohy příslušné každému modelu
3. priority mezi úlohami
4. rychlost pohybu pásu
5. rychlost pohybu pracovníka
6. doba trvání cyklu
7. počet stanic
8. délka každé stanice
9. doba střídání pracovní směny nebo přestávky na oběd

algoritmus dále bere v úvahu následující parametry:

1. množina úloh, které musí být provedeny společně na určité stanici
2. výchozí pozice pracovníka na montážní lince, kde začíná provádět úkoly
3. koncová pozice pracovníka na montážní lince, kde dokončí poslední úkol a vrací se do výchozí pozice
4. prostoj pracovníka, který vznikne po dokončení poslední úlohy a návratu na výchozí pozici

Při stanovování konfigurace vícemodelové výrobně-montážní linky bývá obvykle jako měřítko posouzení provozní výkonnosti linky použit součin doby trvání cyklu modelu a počtu stanic.

V této práci je jako měřítko posouzení provozní výkonnosti linky použit total overload time. Overload time je dodatečný čas, který pracovník musel vynaložit na provedení úkolu.

Je zřejmé, že algoritmy založené na metodě větví a mezí, celočíselném a dynamickém programování jsou výpočetně velmi náročné i pro jednodušší úlohy. Tento algoritmus je založený na heuristické vyvažovací proceduře (heuristic balancing procedure), která pracuje s dolní mezí celkového dodatečného času. Tato procedura v jednom průchodu projde postupně všechny stanice od počáteční po koncovou a provede přiřazení úloh podle příslušných výpočtů.

Mezi hlavní výstupy algoritmu patří:

1. optimalizace celkového dodatečného času namísto součinu doby trvání cyklu modelu a počtu stanic
2. odhad odchylky celkového dodatečného času jakékoliv zadané konfigurace montážní linky od celkového dodatečného času optimálně konfigurované montážní linky bez nutnosti znát aktuální optimální konfiguraci montážní linky. Aktuální optimální konfiguraci montážní linky je obvykle velmi obtížné získat.

Byly provedeny numerické testy tohoto algoritmu pro 10 modelů a pro počet úloh od 1 do 300, což několikanásobně přesahuje možnosti metody větví a mezí [3]. Výpočetní náročnost tohoto algoritmu roste lineárně s počtem stanic a je tedy vhodný pro rozsáhlé příklady.

9.1.2.2 A Two-Stage Heuristic Method for Balancing Mixed-Model Assembly lines with Parallel Workstations

Tato práce je v seznamu použité literatury uvedena jako [5]. Algoritmus popsaný v této práci je určen pro vícemodelovou montážní linku s paralelními stanicemi a pásmovými omezeními (zoning constraint). Pásmové omezení buď nařizuje, nebo zakazuje přiřazení určitých úloh stejné stanici.

Většina metod použitých k stanovování konfigurace výrobně-montážní linky vyžaduje, aby každá úloha byla přiřazena jedné stanici. Tímto je ovšem rychlost výroby omezená dobou trvání nejdelší úlohy. Toto omezení je v této práci odstraněno zavedením paralelní stanice, kdy dvě nebo více replik stanice může zpracovat stejnou množinu úloh na různých sestavách.

Tato metoda minimalizuje počet stanic při zadané době trvání cyklu. Dále tato metoda optimalizuje pracovní zátěž mezi stanicemi a uvnitř stanice. Optimalizace pracovní zátěže mezi stanicemi znamená rozložení času prostoje mezi stanicemi pro každý model co nejvíc rovnoměrně. Optimalizace pracovní zátěže uvnitř stanice znamená rozložení celkového času prostoje pro každou

stanici napříč všemi modely co nejvíc rovnoměrně. Přínos optimalizace pracovní zátěže mezi stanicemi a uvnitř stanice je ten, že všichni pracovníci montážní linky vykonávají přibližně stejné množství práce pro každý model a že v každé stanici je vykonáno přibližně stejné množství práce bez ohledu na právě montovaný model.

Pro výpočet je použita metoda simulovaného žíhání ve dvou krocích, kdy v prvním kroku se hledá optimální počet stanic a ve druhém kroku se optimalizuje pracovní zátěž mezi stanicemi a uvnitř stanice. Metoda simulovaného žíhání je použita zejména z toho důvodu, že umožňuje pružně reagovat na změnu účelové funkce nebo na různá omezení.

Simulované žíhání je prohledávací metoda založená na simulaci žíhání oceli. Začíná se s počáteční hodnotou - „teplotou“, tato teplota se v průběhu hledání snižuje podle určitých pravidel. Při změně teploty je zároveň generováno aktuální řešení.

Nejčastější kritéria používané pro ukončení simulovaného žíhání:

1. celkový počet generovaných řešení
2. předem definovaná „teplota tuhnutí“
3. poměr mezi počtem přípustných a celkově generovaných řešení

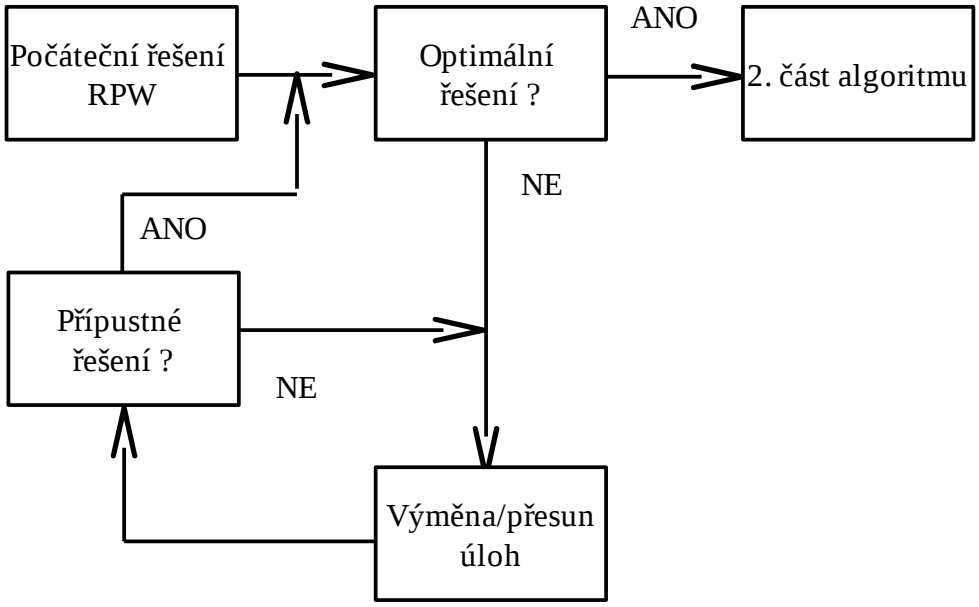
První část algoritmu se skládá z těchto kroků:

1. inicializace počátečního řešení pomocí Rank Positional Weight [11].
2. hledání minimálního počtu stanic
3. výměna dvou úloh nebo přesun úlohy do jiné stanice; Tento krok probíhá tak dlouho, dokud nejsou všechna řešení přípustná
4. kontrola omezení (z hlediska priority úloh, kapacity, pásmová omezení)

Druhá část algoritmu se skládá z těchto kroků:

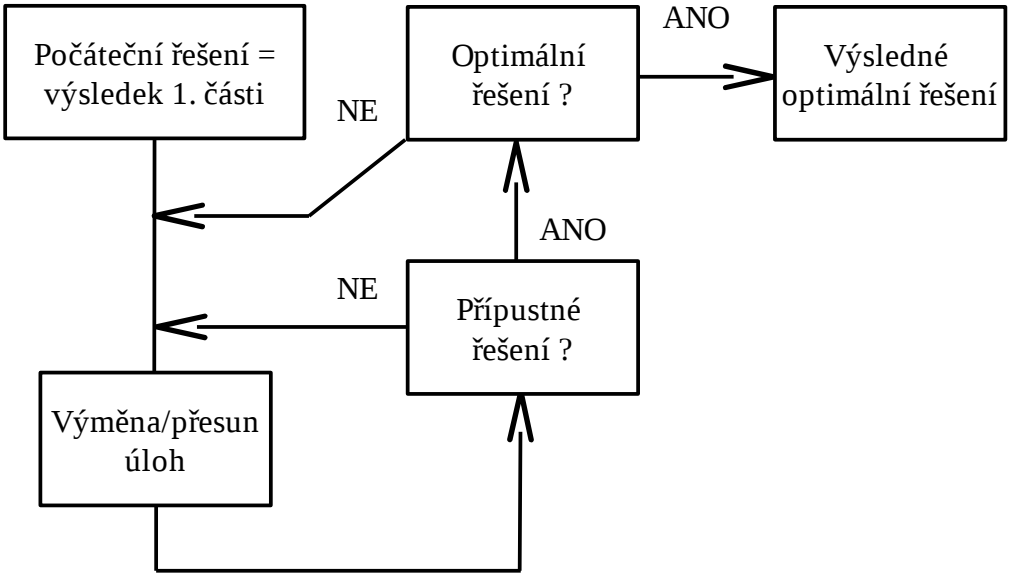
1. inicializace počátečního řešení pomocí výsledného řešení první části algoritmu
2. výměna dvou úloh nebo přesun úlohy do jiné stanice s ohledem na optimalizaci pracovní zátěže mezi stanicemi a uvnitř stanice; Tento krok probíhá tak dlouho, dokud nejsou všechna řešení přípustná
3. kontrola omezení (z hlediska priority úloh, kapacity, pásmová omezení)

Grafické znázornění první části algoritmu:



Obr. 3 Grafické znázornění 1. části algoritmu

Grafické znázornění druhé části algoritmu:



Obr. 4 Grafické znázornění 2. části algoritmu

Byly provedeny numerické testy tohoto algoritmu v rozsahu od 8 úloh a 2 modelů do 70 úloh a 3 modelů. Doba trvání výpočtu příkladu, který se skládal z 70-ti úloh a 3 modelů, byla přibližně 20-ti násobná oproti nejjednoduššímu příkladu, který se skládal z 8-mi úloh a 2 modelů.

Touto metodou je tedy možné řešit příklady s větším počtem úloh než s metodou větví a mezí [3], avšak tato metoda neumožňuje řešit příklady s tak velkým počtem úloh jako metoda od Xao [4].

9.1.3 Programování s omezujícími podmínkami

Programování s omezujícími podmínkami (anglicky Constraint programming) je programovací paradigma, ve kterém jsou vztahy mezi proměnnými popsány pomocí omezujících podmínek.

10. METODA VĚTVÍ A MEZÍ

Při stanovování konfigurace vícemodelové výrobně-montážní linky bude použita metoda větví a mezí s metodou zpětného prohledávání podle [3]. Na začátku běhu algoritmu je rychle nalezeno přípustné řešení a hodnota této účelové funkce je použita pro stanovení počáteční horní mez celkových nákladů. Během prohledávání jsou postupně získávána další řešení a dochází k zlepšování (snižování) hodnoty horní meze. Větve, které obsahují částečné řešení s dolní mezí větší nebo rovno dosud nejlepší nalezené horní mezi, jsou odstraněny. Algoritmus je ukončen, jestliže jsou prohledány všechny větve a je nalezeno optimální řešení. Nejprve budou představeny klíčové části metody větví a mezí, potom budou vysvětleny jednotlivé kroky algoritmu.

10.1 Klíčové části metody větví a mezí

Při popisu budou použity tyto proměnné:

UA_j množina úloh, které dosud nebyly přiřazeny modelu *j*.

AM_i množina modelů, pro které už byla přiřazena úloha *i*.

UAM_i množina modelů, pro které úloha *i* existuje a ještě nebyla přiřazena.

os_j index poslední stanice, které byla přiřazena úloha pro model *j*.

s_{ij} index stanice, které byla přiřazena úloha *i* modelu *j*.

ms_{ij} index nejnižší stanice, které může být přiřazena úloha *i* modelu *j*.

ns_j minimální počet dodatečných stanic, které bude nutné otevřít pro model *j*, aby se přiřadily jeho zbývající nepřiřazené úlohy, včetně úlohy *i*.

gl_{ij} limit pro maximální gapping pro vytváření replik kandidátské úlohy *i* modelu *j:* $gl_{ij} = 1 + n - ns_j - ms_{ij}$ (*n* je maximální počet stanic v montážní lince).

ata_i počet stanic, kterým je přiřazena úloha *i*.

$$nta_i = \begin{cases} 1 & \text{když úloha } i \text{ modelu } j \text{ bude nutně přiřazena více než jedné stanici} \\ 0 & \text{v ostatním případě} \end{cases}$$

10.1.1 Struktura stromu metody větví a mezí

V každém uzlu stromu metody větví a mezí je úloha *i* modelu *j* přiřazena nějaké stanici *k*. V kořenu stromu není dosud přiřazena žádná úloha, odtud $AM_i = \{\} \forall i$. Na konci každé větve je získáno kompletní řešení, ve kterém jsou všechny úlohy všech modelů přiřazeny stanicím: $UAM_i = \{\} \forall i$. Každý uzel stromu představuje částečné řešení obsahující částečné přiřazení úloh stanicím. Toto částečné řešení může být dále rozšířeno přiřazením kandidátské úlohy, která splňuje následující podmínky:

1. nemá žádné předchůdce nebo všichni její předchůdci už byli přiřazeni stanicím
2. volný čas stanice je větší nebo roven době trvání úlohy

10.1.2 Výpočet mezí

Dolní mez, LB (lower bound) celkových nákladů se skládá ze dvou dolních mezí: dolní mez nákladů spojená s počtem stanic, LB_S a dolní mez nákladů spojená s duplikací úlohy, LB_D. Každá z těchto dolních mezí se skládá ze dvou částí: z nákladů, které dosud vznikly přiřazením v částečném řešení a z dolních mezí zbývajících přiřazení potřebných k dokončení konfigurace linky.

Nechť úloha *p* modelu *q* je aktuální kandidát, jehož dolní mez má být vypočtena.

10.1.2.1 Výpočet dolní meze nákladů na stanici LB_S

Po přiřazení úlohy *p* modelu *q* je os_q aktualizováno podle stanice, které byla přiřazena úloha *p*. Počet stanic otevřených v částečném řešení je tedy dáno vztahem

$$os = Max_j\{os_j\} \tag{8}$$

a příslušné náklady na stanici jsou rovny *SC * os.* Postup pro výpočet dolní meze zbývajících přiřazení je následující: Nejprve je vypočtena dolní mez hodnoty ns_q. Tato dolní mez je rovna maximální hodnotě z následujících dolních mezí počtů stanic, vypočtených pro SALB-P.

První dolní mez LB_1 je rovna zbývající době úloh, které ještě zbývá přiřadit modelu *q*, děleno časem modelu *q* a zaokrouhleno na nejbližší vyšší celé číslo:

$$ns_q = \left\lceil \frac{1}{c} * \sum_{i \in UA_q} t_{iq} \right\rceil \tag{9}$$

Druhou a třetí dolní mez LB_2 a LB_3 vyvinul Johnson [8].

obecná rovnice pro výpočet dolní meze:

$$LB_h = \left\lceil \sum_{i=1}^{h-1} \frac{počet\ úloh\ s\ t_i > \left(\frac{i*c}{h}\right)}{(h-1)} + \sum_{i=1}^{h} i * \frac{počet\ úloh\ s\ t_i = \frac{i*c}{h}}{h} - \sum_{i=1}^{h-1} i * \frac{počet\ úloh\ s\ t_i = \frac{(i+1)*c}{h}}{h-1} \right\rceil \tag{10}$$

kde

t_i doba trvání úlohy *i*
c požadovaný čas modelu (doba trvání cyklu)

potom

$$LB_2 = \left\lceil \sum_{i=1}^{1} počet\ úloh\ s\ t_i > \frac{i*c}{2} + \sum_{i=1}^{2} i * \frac{počet\ úloh\ s\ t_i = \frac{i*c}{2}}{2} - \sum_{i=1}^{1} i * počet\ úloh\ s\ t_i = \frac{(i+1)*c}{2} \right\rceil \tag{11}$$

$$LB_3 = \left\lceil \sum_{i=1}^{2} \frac{počet\ úloh\ s\ t_i > (\frac{i*c}{3})}{2} + \sum_{i=1}^{3} i * \frac{počet\ úloh\ s\ t_i = \frac{i*c}{3}}{3} - \sum_{i=1}^{2} i * \frac{počet\ úloh\ s\ t_i = \frac{(i+1)*c}{3}}{2} \right\rceil \quad (12)$$

Čtvrtou dolní mez LB_4 vyvinul Berger [9]:

V práci [9] je použito označení LB1 a LB2, které neodpovídá výše uvedenému označení dolních mezí LB_1 a LB_2.

Nechť jsou *a* a *b* takové, že platí:

$$0 \le a < T\ , a < b \le T \quad (13)$$

a nechť je dán takový vektor *W(a, b)*, že platí:

$$W(a,b) = \{i \in N : a < t_i \le b\}\ , \quad (14)$$

kde

T je čas modelu
t_i je doba trvání úlohy *i*

Dále předpokládejme, že všechny úlohy ve vektoru *W* jsou uspořádány vzestupně podle jejich doby trvání úlohy.

1. Začíná se u úlohy s nejdelší dobou trvání úlohy, stanicím jsou přiřazeny úlohy, pro které platí $W(\frac{T}{2}, T)$.
2. Postupně jsou uvažovány všechny úlohy, pro které platí $W(\frac{T}{3}, \frac{T}{2})$, počínaje úlohou s nejkratší dobou a všechny stanice, kterým byly přiřazeny úlohy $W(\frac{T}{2}, T)$, setříděné podle volné doby stanice vzestupně. Úlohy $W(\frac{T}{3}, \frac{T}{2})$ jsou postupně přiřazeny do první vhodné stanice. Tento proces je ukončen, jestliže jsou přiřazeny všechny úlohy, nebo nezbyvá žádná volná stanice.
3. Nechť *K* je počet úloh $W(\frac{T}{3}, \frac{T}{2})$, které nebyly přiřazeny v minulém kroku. Potom bude potřeba nejméně $\left\lceil \frac{K}{2} \right\rceil$ dodatečných stanic, protože do stejné stanice můžou být přiřazeny maximálně dvě úlohy z $W(\frac{T}{3}, \frac{T}{2})$.
4. Určení minimálního počtu dodatečných stanic, které budou vyžadovány pro všechny zbývající nepřiřazené úlohy:

pro všechna $c \in \left(0, \frac{T}{3}\right]$, je vypočten *p(c)*, počet dodatečných stanic vyžadovaných pro všechny úlohy, pro které platí $t_i > c$:

$$p(c) = max\left\{0, \left\lceil \left\{ \sum_{i \in W(c, T-c)} t_i - \left(\left| W\left(\frac{T}{2}, T-c\right) \right| + \left\lceil \frac{K}{2} \right\rceil \right) T \right\} \{T\}^{-1} \right\rceil \right\} \tag{15}$$

Výpočet *p(c)* je založen na skutečnosti, že úlohy, pro které platí $t_i > c$, nemůžou být vykonány na stanici, kde se už vykonávají úlohy s dobou $T - c$.

Dolní mez počtu vyžadovaných stanic je potom:

$$LB2(c) = \left| W\left(\frac{T}{2}, T\right) \right| + \left\lceil \frac{K}{2} \right\rceil + p(c) \tag{16}$$

Protože toto platí pro všechna *c*, dolní mez je dána vztahem:

$$LB2 = \max_{0 \le c < \frac{T}{3}} \{LB2(c)\} \tag{17}$$

$$LB_4 = LB2 \tag{18}$$

Poslední, pátou dolní mez LB_5 navrhl Scholl a Klein [10]:

Nejdřívější a nejpozdější stanice pro každou úlohu *i* může být určena výpočtem dolní meze počtu stanic požadovaných pro úlohu *i* a její předchůdce a následníky. Toho může být dosaženo použitím dolní meze LB_1 na redukovaný problém s množinou úloh $\{i\} \cup P_i'$ nebo $\{i\} \cup F_i'$, čímž vznikne následující rovnice pro nejdřívější a nejpozdější stanici úlohy *i*:

$$E_i = \left\lceil \frac{t_i + \sum_{h \in P_i'} t_h}{c} \right\rceil \tag{19}$$

$$L_i(m) = m + 1 - \left\lceil \frac{t_i + \sum_{h \in F_i'} t_h}{c} \right\rceil \quad \text{pro } i = 1, \ldots, n. \tag{20}$$

kde

m počet stanic
n počet úloh
c čas modelu
t_i doba trvání úlohy $i = 1, \ldots, n$
P_i' množina předchůdců úlohy *i*
F_i' množina následníků úlohy *i*
E_i nejdřívější stanice pro úlohu *i*

L_i nejpozdější stanice pro úlohu *i*

Protože každá úloha musí být přiřazena jedné stanici, dolní mez je dána vztahem:

$$LB_5 = min\left\{m | L_i(m) \geq E_i \forall i\right\} \quad (21)$$

Dolní mez počtu stanic, které ještě bude nutné otevřít pro model *q*:

$$ns_q = max_{r=1...n}\left|LB_r\right| \quad (22)$$

Minimální počet dodatečných stanic, které ještě bude nutné otevřít ve výsledném řešení, je tedy:

$$ns = Max_j\left\{ns_j\right\} \quad (23)$$

Odtud je dolní mez nákladů spojených s počtem stanic vyjádřená vztahem:

$$LB_S = SC * (os + ns) \quad (24)$$

10.1.2.2 Výpočet dolní meze nákladů na duplikaci úlohy LB_D

Před výpočtem LB_D je ata_p aktualizováno podle množiny AM_p. Dosavadní náklady na duplikaci úlohy spojené s dosud přiřazenými úlohami:

$$\sum_{i=1}^{n} TC_i * ata_i \quad (25)$$

Výpočet dolní meze nákladů spojených s přiřazením zbývajících úloh je založen na detekci případů, ve kterých bude v budoucnu nevyhnutelná duplikace úloh. Duplikace úloh bude zjišťována pomocí dvou metod:

1. napříč modely
2. napříč úlohami

1. napříč modely

Při této metodě jsou prohledávány modely, pro které kandidátská úloha *p* je vyžadována, ale ještě nebyla přiřazena $(l \in UAM_p)$. Jestliže existuje takový model *l*, že $os_l > os_q$ nebo $ms_{pl} > os_q$, nebude možné v budoucnu přiřadit úlohu *p* do stanice os_q, a tedy úloha *p* bude muset být duplikována. V tomto případě je nta_p nastavena na hodnotu jedna. Dokonce i když existuje více než jeden model, který splňuje výše uvedenou podmínku, budoucí duplikace je uvažována pouze jednou, protože je možné, že úloha *p* bude v budoucnu přiřazena do stejné stanice pro tyto modely. Po přiřazení úlohy *p* modelu *l* do výsledného řešení je nta_p nastavena na hodnotu nula a ata_p bude zvětšena o jedničku.

2. napříč úlohami

Při této metodě jsou prohledávány úlohy, které dosud nebyly přiřazeny modelu *q* $(i \in UA_q)$. Jestliže existuje taková úloha *i*, že platí

$$ms_{iq} > Max_{l \in AM_i}\left\{s_{il}\right\} \quad (26)$$

jako výsledek přiřazení úlohy *p*, to znamená, že v budoucnu nebude možné přiřadit úlohu *i* do kterékoliv stanice, do které už byla přiřazena pro jiné modely, a proto bude duplikována. V tomto případě je nta_i nastavena na hodnotu jedna. Po přiřazení úlohy *i* modelu *q* do výsledného řešení je nta_i nastavena na hodnotu nula a ata_i bude zvětšena o jedničku. Dolní mez nákladů spojených s duplikací úlohy je potom:

$$LB_D = \sum_{i=1}^{n} TC_i * (ata_i + nta_i) \tag{27}$$

10.1.2.3 Dolní mez celkových nákladů

Dolní mez celkových nákladů, skládající se z dvou výše popsaných dolních mezí, je dána vztahem:

$$LB = LB_S + LB_D = SC * (os + ns) + \sum_{i=1}^{n} TC_i * (ata_i + nta_i) \tag{28}$$

Speciální případ dolní meze celkových nákladů je počáteční dolní mez pro celý problém, který bude označován PLB. Část PLB spojená s náklady na stanici je dána stejným vztahem jako ostatní dolní meze spojené se stanicí:

$$SC * (os + ns), \tag{29}$$

pouze zde $os = 0.$

Část PLB spojená s náklady na úlohu je:

$$\sum_{i=1}^{n} TC_i, \tag{30}$$

což představuje ideálně nastavenou montážní linku bez duplikací úloh.

10.1.3 Gapping

Gapping (mezery mezi stanicemi) je důležitým a někdy základním prvkem při vytváření efektivní konfigurace montážní linky. Pro získání optimálního řešení musí být pro každé přiřazení úloh prozkoumány všechny možné mezery mezi stanicemi. Jinými slovy, jakmile je úloha poprvé přiřazena, jsou uvažovány všechny možné stanice, kterým může být tato úloha přiřazena. Proto je vytvořena množina replik pro každou kandidátskou úlohu v tomto rozsahu stanic:

$$k = ms_{ij} + g, (g = 0,1,2,\ldots, gl_{ij}) \tag{31}$$

10.2 Popis algoritmu

Algoritmus použitý v této práci se skládá z následujících kroků:

1. Vytvořit počáteční prázdný uzel (dosud nebyly přiřazeny žádné úlohy) a vyhodnotit PLB. Nastavit počáteční hodnotu horní meze účelové funkce UB na dostatečně velkou hodnotu.
2. Vytvořit seznam primárních kandidátů (zatím bez mezer mezi stanicemi) pro nový uzel,

obsahující všechny úlohy *i* všech modelů *j*, pro které platí:

$$IP_{ij} \cap UA_j = \{\} \tag{32}$$

3. Jestliže je seznam primárních kandidátů prázdný (bylo dosaženo listu, vrcholu bez potomků), přejít na krok č. 9. Jinak přejít na krok č. 4.
4. Pro každého kandidáta vypočítat gl_{ij} podle vzorce:

$$gl_{ij} = 1 + n - ns_j - ms_{ij} \tag{33}$$

a vytvořit množinu replik kandidátů pro přiřazení do stanice *k*, kde *k* se určí podle (31)
5. Vypočítat LB pro všechny kandidáty.
6. Odstranit všechny kandidáty s $LB \geqslant UB$ ze seznamu (větev stromu replik kandidátů je smazána)
7. Jestliže je seznam primárních kandidátů prázdný, přejít na krok č. 10, jinak na krok č. 8.
8. Vybrat kandidáta s nejmenším LB a přidat do výsledného řešení. Návrat na krok č. 2.
9. Nastavit hodnotu UB na LB listu (vrcholu bez potomků). Jestliže UB = PLB, bylo dosaženo optimálního řešení, které již nemůže být dále vylepšeno. V tom případě je činnost programu ukončena, jinak se pokračuje krokem č. 10.
10. Zahájit zpětné hledání do hloubky a pokračovat v něm, dokud:
    a) není dosaženo uzlu s nenulovým počtem primárních kandidátů. Přejít na krok č. 6.
    b) není dosažen prázdný, počáteční uzel (kořen stromu replik kandidátů) a jeho seznam primárních kandidátů je prázdný. V tomto případě je činnost programu ukončena, byl prohledán celý strom a poslední nalezené řešení je optimální řešení.

10.3 Modifikace algoritmu

Jedním z úkolů této diplomové práce bylo pokusit se upravit původní algoritmus takovým způsobem, aby došlo k zlepšení výstupu programu – hodnoty kriteriální funkce, neboli snížení nákladů na konfiguraci montážní linky. V této práci byly navrženy úpravy, které je možné rozdělit do dvou skupin:

1. úpravy založené na mazání primárních kandidátů
2. úprava založená na přiřazování úloh stejného modelu

10.3.1 Úpravy založené na mazání primárních kandidátů

Při vytváření stromu replik kandidátů se používají všichni primární kandidáti, vytvoření z dosud nepřiřazených úloh, kteří splňují podmínku priority úloh. Počet uzlů stromu replik kandidátů a tedy i paměťové nároky a celková doba výpočtu se však velmi rychle zvětšuje s rostoucím počtem primárních kandidátů. Proto byly zavedeny úpravy, které dle různých kritérií vyberou jednoho nebo více primárních kandidátů a vymažou jej ze seznamu primárních kandidátů, ze kterého jsou načítány uzly při vytváření stromu replik kandidátů. Toto mazání se týká pouze aktuálního cyklu běhu programu, tento smazaný kandidát je při příštím cyklu opět k dispozici pro výběr kandidátů ke smazání nebo pro vytváření stromu replik kandidátů.

10.3.1.1 Mazání jednoho primárního kandidáta podle doby trvání úlohy

Při této úpravě jsou primární kandidáti setříděni podle t_{ij} - doby trvání úlohy *i* modelu *j*. Je vymazán primární kandidát s nejkratší dobou trvání úlohy. Princip této úpravy je založen na faktu, že

v [9] se jako první přiřazují úlohy s nejdelší dobou trvání.

10.3.1.2 Mazání jednoho primárního kandidáta podle počtu následníků

Při této úpravě je pro každého primárního kandidáta zjišťen počet všech následníků v grafu priority úloh. Pro vymazání je vybrán kandidát s nejmenším počtem následníků.

10.3.1.3 Mazání jednoho primárního kandidáta podle nákladů na úlohu TC

Při této úpravě jsou primární kandidáti setříděni podle TC_i - nákladů na úlohu *i*. Je to konstantní částka spojená s každou stanicí, ke které je přiřazena úloha *i*. Pro vymazání je vybrán kandidát s nejmenší TC_i .

10.3.1.4 Mazání jednoho primárního kandidáta podle RPW, Ranked Positional Weight

Ranked positional weight pro jednomodelovou montážní linku zavedl Helgeson a Birnie [11]. Ranked positional weight se vztahuje k určité úloze a je to součet doby trvání této úlohy a všech jejich následníků v grafu priority úloh. V této práci je výpočet Ranked positional weight upraven tak, že jsou sčítány doby trvání úloh pro všechny modely. Pro vymazání je vybrán kandidát s nejmenší Ranked positional weight.

10.3.1.5 Mazání více primárních kandidátů podle doby trvání úlohy

Tato úprava vychází z 10.3.1.1, Mazání jednoho primárního kandidáta podle doby trvání úlohy. Rozdíl je ten, že se maže více primárních kandidátů. Počet kandidátů, kteří budou smazáni, se určí jako podíl z celkového počtu primárních kandidátů:

$$Počet\ kandidátů\ ke\ smazání = poměr_{sk} * celkový\ počet\ primárních\ kandidátů \tag{34}$$

Kde proměnná $poměr_{sk}$ (poměr smazaných kandidátů) nabývala obvykle tří hodnot: 0,1; 0,5; 0,9. V případě potřeby, například jestliže výsledky byly příliš podobné a nebylo tedy možné spolehlivě rozhodnout, která metoda poskytuje lepší výsledky, se za proměnnou $poměr_{sk}$ dosadily jiné hodnoty.

10.3.1.6 Mazání více primárních kandidátů podle nákladů na úlohu TC

Tato úprava vychází z 10.3.1.3, Mazání jednoho primárního kandidáta podle nákladů na úlohu TC. Počet kandidátů, kteří budou smazáni, se určí podobně jako v 10.3.1.5.

10.3.2 Modifikace úprav založených na mazání primárních kandidátů

Úpravy uvedené v 10.3.1 je možné dále kombinovat s dalšími úpravami:

1. mazání primárních kandidátů podle modelu
2. mazání primárních kandidátů podle UA_j

10.3.2.1 Mazání primárních kandidátů podle modelu

Mazání primárních kandidátů nebude provedeno pro určitý model, dokud nebudou do výsledného řešení přiřazeny všechny úlohy tohoto modelu.

10.3.2.2 Mazání primárních kandidátů podle UA

Primární kandidát nebude vymazán, jestliže stejná úloha jiného modelu je již přiřazena do výsledného řešení.

10.3.3 Úprava založená na přiřazování úloh stejného modelu

Při této úpravě se do výsledného řešení postupně přiřazují pouze úlohy stejného modelu. Začíná se u modelu, pro který je definováno nejvíce úloh. Po přiřazení všech úloh daného modelu jsou přiřazovány úlohy dalšího modelu. Tyto úpravy je možné kombinovat s úpravami, kde se maže jeden primární kandidát (10.3.1).

10.3.4 Úprava založená na změně výpočtu ms(i, j)

ms_{ij} : index nejnižší stanice, které může být přiřazena úloha *i* modelu *j*.

Tato úprava spočívá v přesnější identifikaci případů, kdy dojde k duplikaci úlohy. Podrobnější popis – viz příloha, soubor 2011_DP_Dolezal_Zbynek_45281.ods, list Bukchin.

11. POUŽITÉ ŘEŠENÍ

Algoritmus pro stanovení konfigurace vícemodelové výrobně-montážní linky podle [3] byl napsán v jazyce C++, jako konzolová aplikace. Mezi hlavní výhody konzolové aplikace patří jednodušší vývoj aplikace, rychlejší běh programu a snadnější přenositelnost mezi různými operačními systémy. Program byl zkompilován pomocí GCC 3.4.5 a 4.2.2.

11.1 Pojmenování proměnných ve zdrojovém kódu

Správné pojmenování proměnných je velmi důležité, protože umožňuje výrazně ulehčit orientaci ve zdrojovém kódu. Ve větším projektu, který obsahuje velké množství proměnných, je to přímo nezbytnost. Někdy bylo nutné v zájmu srozumitelnosti použít název proměnné o větším počtu znaků, než bývá obvyklé. Podle [13] je úsilí nezbytné k odladění programu nejmenší, jestliže se délka názvů proměnných pohybuje mezi 10 až 16 znaků.

Většina proměnných obsahuje předponu, která určuje, o jaký typ se jedná. V následující tabulce jsou uvedeny příklady názvů proměnných a jejich typ. Členské proměnné (třídy) navíc obsahují předponu m_.

Příklad názvu proměnné	Typ proměnné
nModel	int, long
dLB_DolniMez	double
pUzel	ukazatel na typ Uzel
sText	textový řetězec
rUzel	reference na typ Uzel
crUzel	konstantní reference na typ Uzel

Tabulka č. 1: Názvy proměnných a jejich typ

Příklad názvu členské proměnné	Typ členské proměnné
m_nModel	int, long
m_dLB_DolniMez	double
m_pUzel	ukazatel na typ Uzel
m_sText	textový řetězec
m_rUzel	reference na typ Uzel
m_crUzel	konstantní reference na typ Uzel

Tabulka č. 2: Názvy členských proměnných a jejich typ

11.2 Ovládání programu

Protože konzolová aplikace neposkytuje takové možnosti ovládání jako aplikace s klasickým grafickým uživatelským rozhraním, ovládání programu je realizováno pomocí direktiv preprocesoru jazyka C++. Tyto direktivy jsou umístěny v hlavičkovém souboru Bukchin.h.

V následujících kapitolách bude uveden seznam direktiv preprocesoru, které slouží např. k výběru metody výpočtu, vstupních hodnot, atd. a seznam některých proměnných.

11.2.1 Seznam direktiv preprocesoru

Direktiva preprocesoru	Význam
POCET_ULOH	celkový počet odlišných úloh
DATA_BUKCHIN	vstupní hodnoty podle příkladu z [3]
BD_2_MODELY	vstupní hodnoty podle příkladu z [3], počet modelů $m = 2$
BD_3_MODELY_DOBA_ULOHY1	vstupní hodnoty podle příkladu z [3], počet modelů $m = 3$, doba trvání úlohy t podle série č.1
DATA_VILARINHO	vstupní hodnoty podle příkladu z [5]
DATA_PRODUCT_LAYOUT	označení vstupních hodnot PRODUCT_LAYOUT
DATA_BERGER	vstupní hodnoty podle příkladu z [9]
DATA_BERG_PROD	kombinace vstupních hodnot podle příkladu z [9] a DATA_PRODUCT_LAYOUT
DATA_ASLB	vstupní hodnoty podle příkladu z [14]

Tabulka č. 3: Seznam direktiv preprocesoru pro řízení vstupních údajů

Direktiva preprocesoru	Význam
UPRAVA_2	Úpravy založené na mazání primárních kandidátů 10.3.1
UPRAVA2_MAZANI_JEDNOHO_KANDIDATA	Úpravy založené na mazání jednoho primárního kandidáta. Pokud je direktiva zakomentovaná, použijí se úpravy založené na mazání více primárních kandidátů.
UPRAVA2_MAZANI_JEDNOHO_KANDIDATA_PODLE_CASU	Mazání jednoho primárního kandidáta podle doby trvání úlohy 10.3.1.1
UPRAVA2_MAZANI_JEDNOHO_KANDIDATA_PODLE_POCTU_NASLED	Mazání jednoho primárního kandidáta podle počtu následníků 10.3.1.2
UPRAVA2_MAZANI_JEDNOHO_KANDIDATA_PODLE_TC	Mazání jednoho primárního kandidáta podle nákladů na úlohu TC 10.3.1.3
UPRAVA2_MAZANI_JEDNOHO_KANDIDATA_PODLE_RPW	Mazání jednoho primárního kandidáta podle RPW, Ranked Positional Weight 10.3.1.4
UPRAVA2_MAZANI_VICE_KANDIDATU_PODLE_CASU	Mazání více primárních kandidátů podle doby trvání úlohy 10.3.1.5
UPRAVA2_MAZANI_VICE_KANDIDATU_PODLE_TC	Mazání více primárních kandidátů podle nákladů na úlohu TC 10.3.1.6
UPRAVA2_MAZANI_KANDIDATU_PODLE_MODELU	Mazání primárních kandidátů podle modelu 10.3.2.1
UPRAVA2_MAZANI_KANDIDATU_PODLE_UA	Mazání primárních kandidátů podle UA 10.3.2.2
POMER_SMAZANYCH_KANDIDATU_UPRAVA2	$poměr_{sk}$ (poměr smazaných kandidátů) z rovnice (34)
UPRAVA_3	Úprava založená na přiřazování úloh stejného modelu 10.3.3
UPRAVA_4_MS	Úprava založená na změně výpočtu ms(i, j) 10.3.4

Tabulka č. 4: Seznam direktiv preprocesoru pro řízení metody výpočtu

11.2.2 Přehled důležitých proměnných

Název proměnné	Význam
m_nn_PocetUloh	*n*, celkový počet odlišných úloh
m_nm_PocetModelu	*m*, počet modelů, které se mají sestavit na montážní lince
m_dSC	SC, náklady na stanici – konstantní částka spojená s každou stanicí
m_dUB	počáteční hodnotu horní meze účelové funkce UB (viz 10.2, krok 1.)
m_IP_BezprostredniPredchudci	IP_{ij} množina bezprostředních předchůdců úlohy *i* modelu *j* (graf priority úloh). Trojrozměrný vektor celých čísel.
m_TC	TC_i náklady na úlohu – konstantní částka spojená s každou stanicí, ke které je přiřazena. Vektor celých čísel.
m_c_PozadovanyCasModelu	c_j požadovaný čas modelu *j* (doba trvání cyklu). Vektor celých čísel.
m_t_DobaUlohy	t_{ij} doba trvání úlohy *i (i = 1, … , n)*, jestliže je prováděna na modelu *j (j = 1, …, m).* Dvourozměrný vektor celých čísel.

Tabulka č. 5: Přehled důležitých proměnných

12. APLIKACE VYTVOŘENÉHO ŘEŠENÍ

V této kapitole bude uveden seznam příkladů použitých při testování a výsledky těchto testů.

Testovací výpočty byly prováděny v operačním systému Puppy Linux 4.3.1 s nejnovější stabilní verzí jádra. Použité hardwarové vybavení: CPU AMD Athlon XP 2000+ Thoroughbred, RAM 512 MB. Velikost operační paměti se ukázala jako hlavní omezující faktor, kdy při větším počtu úloh došlo k zahlcení operační paměti.

Tento program byl také zkompilován v operačním systému MS Windows XP pomocí vývojového prostředí Code::Blocks s využitím kompilátoru GCC. V operačním systému MS Windows je možné pro kompilaci programu použít také prostředí MSYS.

Ukázka příkazu pro kompilaci v prostředí MSYS:

```
g++ -pedantic D:/DP/main.cpp -o D:/DP/main D:/DP/Bukchin.cpp D:/DP/BukchinData.cpp D:/DP/Vilarinho.cpp
```

12.1 Seznam použitých vstupních dat

Protože metodou větví a mezí je možné řešit příklady s omezeným počtem úloh, grafy priority úloh budou zobrazeny jen pro ty úlohy, pro které bylo nalezeno řešení v reálném čase.

12.1.1 ASLB

Tento příklad je definován v [14]. Protože autor není uveden, název ASLB je odvozen z počátečních písmen nadpisu „Assembly line balancing“.

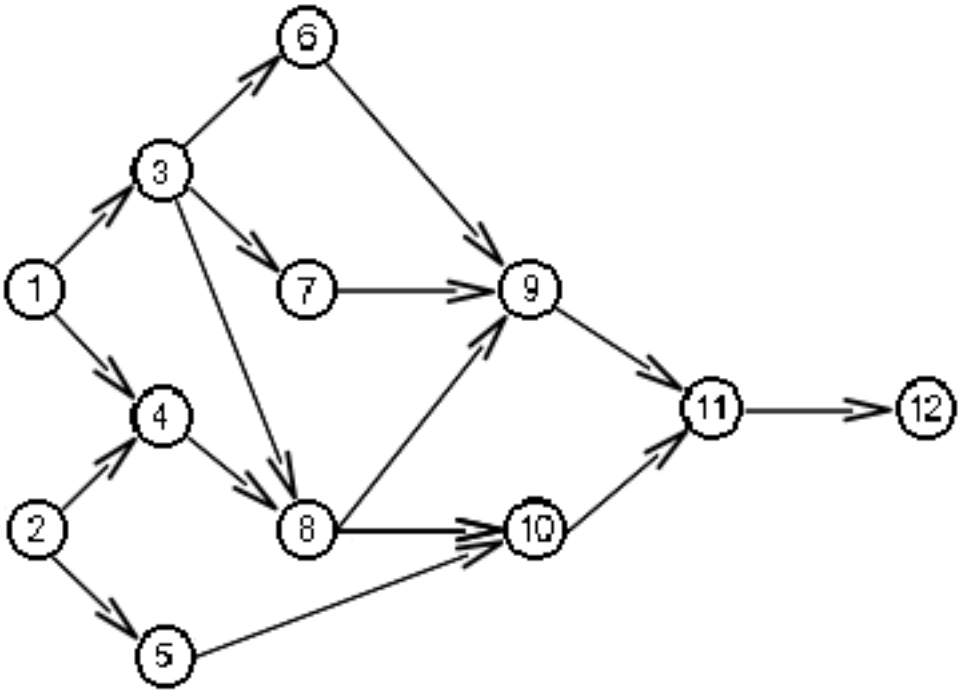
Direktiva preprocesoru pro tento příklad: DATA_ASLB

Doba trvání cyklu (požadovaný čas modelu): 100

V následující tabulce jsou uvedeny doby trvání úloh:

Úloha	Doba trvání úlohy
1	20
2	40
3	70
4	10
5	30
6	11
7	32
8	60
9	27
10	38
11	50
12	12

Tablka č. 6: Doby trvání úloh



Obr. 5 Graf priorit úloh pro příklad ASLB.

12.1.2 Bukchin

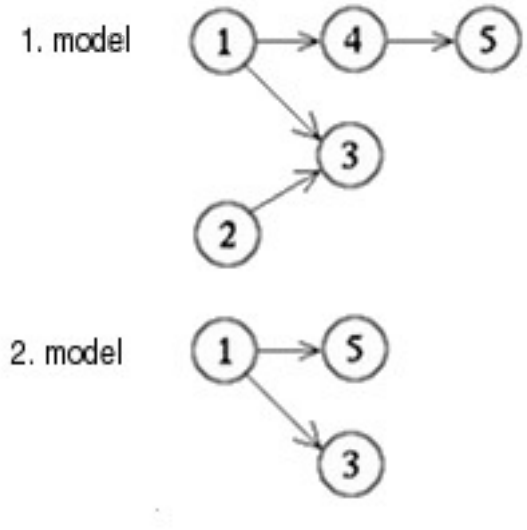
Tento příklad je definován v [3].
Direktiva preprocesoru pro tento příklad: DATA_BUKCHIN.

Doba trvání cyklu (požadovaný čas modelu):
c(1) = 8
c(2) = 5

V následující tabulce jsou uvedeny doby trvání úloh:

Úloha	Model	Doba trvání úlohy
1	1	6
2		4
3		6
4		4
5		2
1	2	2
3		4
5		3

Tablka č. 7: Doby trvání úloh



Obr. 6 Graf priorit úloh pro příklad podle Bukchin.

12.1.3 Vilarinho

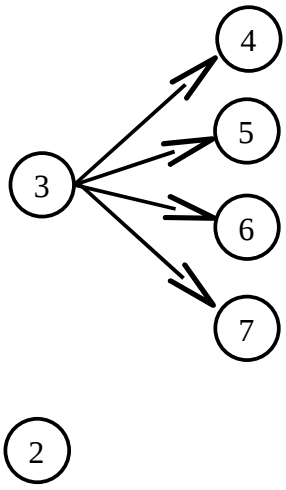
Tento příklad je definován v [5].
Direktiva preprocesoru pro tento příklad: DATA_VILARINHO.

Doba trvání cyklu (požadovaný čas modelu):
c(1) = 150
c(2) = 150

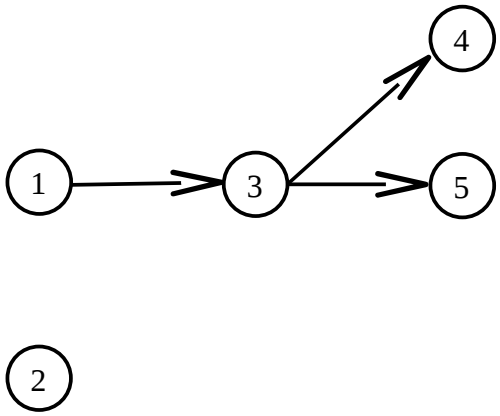
V následující tabulce jsou uvedeny doby trvání úloh:

Úloha	Model	Doba trvání úlohy
1	1	0
2		77
3		73
4		150
5		88
6		62
7		36
1	2	20
2		77
3		73
4		150
5		88
6		0
7		0

Tablka č. 8: Doby trvání úloh



Obr. 7 Graf priorit úloh pro příklad podle Vilarinho – 1. model.



Obr. 8 Graf priorit úloh pro příklad podle Vilarinho – 2. model.

12.1.4 Berger

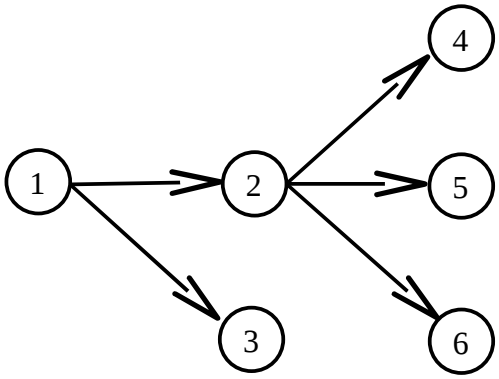
Tento příklad je definován v [9].
Direktiva preprocesoru pro tento příklad: DATA_BERGER.

Doba trvání cyklu (požadovaný čas modelu):
c(1) = 80
c(2) = 94

V následující tabulce jsou uvedeny doby trvání úloh:

Úloha	Model	Doba trvání úlohy
1	1	42
2		20
3		6
4		75
5		49
6		38
1	2	42
2		20
3		6
4		75
5		49
6		38

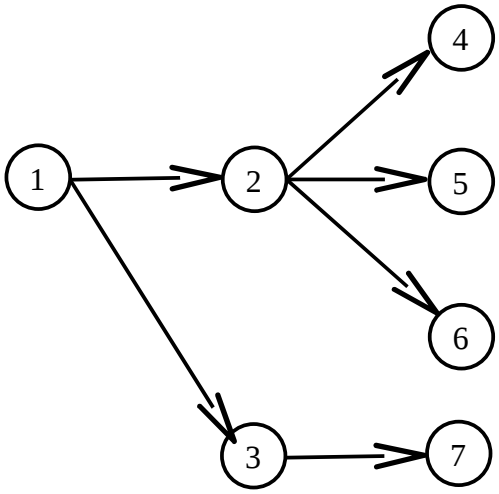
Tablka č. 9: Doby trvání úloh



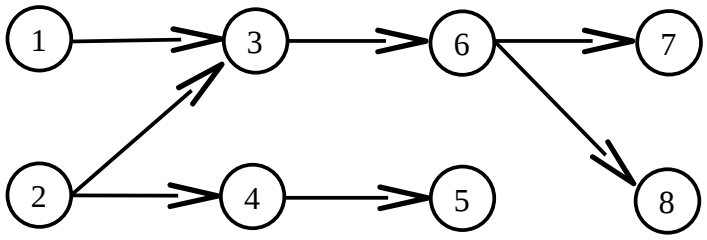
Obr. 9 Graf priorit úloh pro příklad podle Berger.

12.1.5 Berger-Production

DATA_BERG_PROD



Obr. 10 Graf priorit úloh pro příklad podle Berger-Production, 1. model.



Obr. 11 Graf priorit úloh pro příklad podle Berger-Production, 2. model.

12.2 Srovnání kvality výsledků s výsledky získanými tradičními metodami

V této kapitole bude provedeno srovnání kvality výsledků původního algoritmu z [3] a výsledků získanými tradičními metodami [14].

12.2.1 Vstupní data

V tomto srovnání byly použity vstupní hodnoty ASLB 12.1.1 a Bukchin 12.1.2.

12.2.2 Metoda Largest-candidate rule

12.2.2.1 Popis metody

1. Setřídit všechny úlohy sestupně podle jejich doby trvání
2. Počínaje úlohou s nejdelší dobou trvání, postupně přiřazovat úlohy do nejbližší přípustné stanice. Přípustná stanice je taková stanice, ve které nebude překročena doba trvání cyklu a zároveň bude dodržena podmínka priority mezi úlohami.
3. Opakovat krok č. 2

12.2.2.2 Postup výpočtu

V následující tabulce jsou uvedeny všechny úlohy setříděny sestupně podle jejich doby trvání.

Úloha	Doba trvání úlohy
3	70
8	60
11	50
2	40
10	38
7	32
5	30
9	27
1	20
12	12
6	11
4	10

Tablka č. 10: Výpočet – krok č. 1 – data ASLB

Úloha	Model	Doba trvání úlohy
1	1	6
3	1	6
2	1	4
4	1	4
3	2	4
5	2	3
5	1	2
1	2	2

Tablka č. 11: Výpočet – krok č. 1 - data Bukchin

V následující tabulce je uvedeno přiřazení úloh stanicím po provedení kroků 2 a 3.

	data ASLB	data Bukchin	
Stanice	Úloha	Úloha	Model
1	2	1	1
	5	1	2
	1	5	2
	4		
2	3	2	1
	6	4	1
		3	2
3	8	3	1
	10	5	1
4	7		
	9		
5	11		
	12		

Tablka č. 12: Přiřazení úloh stanicí

12.2.3 Metoda Kilbridge a Wester

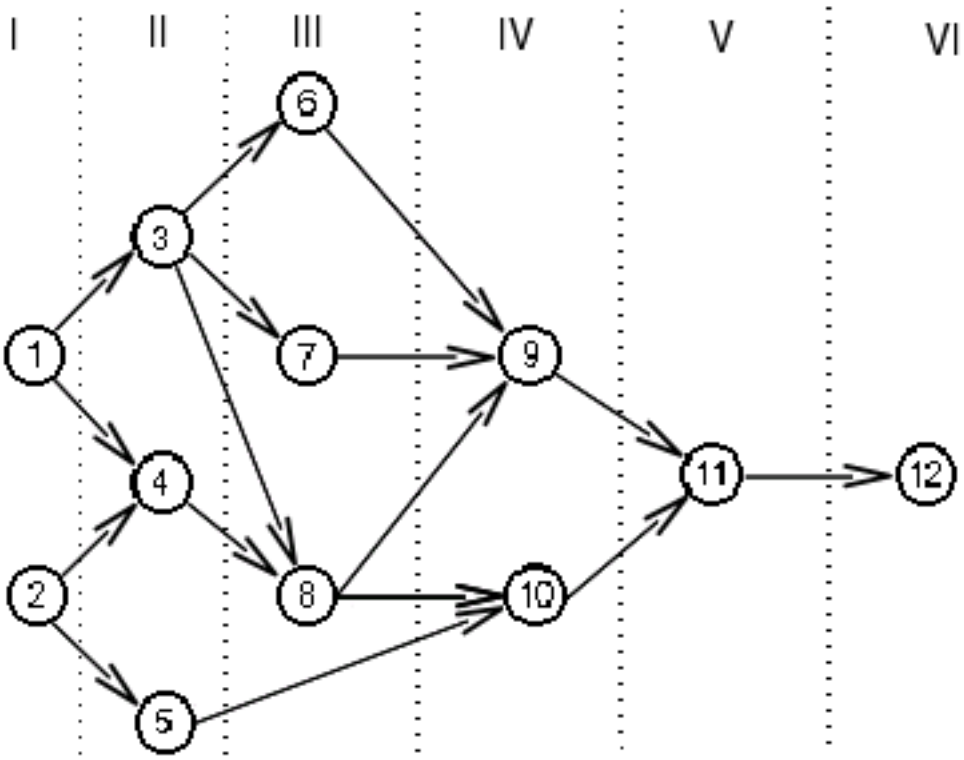
12.2.3.1 Popis metody

Jedná se o heuristickou metodu, která vybírá úlohy pro přiřazení stanicím podle pozice v grafu priority úloh. Tento postup odstraňuje nevýhodu metody largest-candidate rule 12.2.2, ve které může být jako první úloha pro přiřazení vybrána ta úloha, která má sice nejdelší dobu trvání, ale nachází se na konci grafu priority úloh.

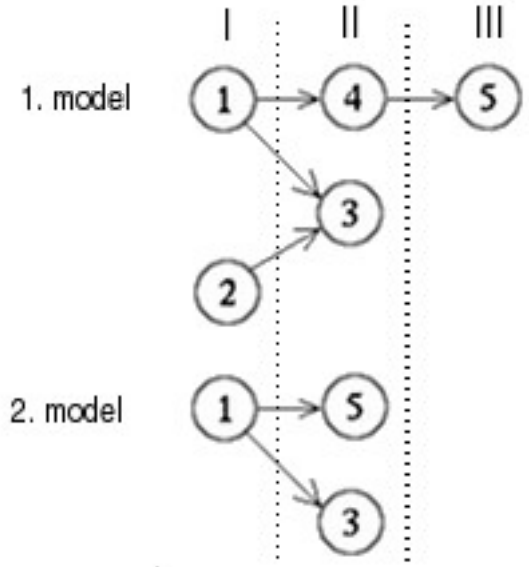
12.2.3.2 Postup výpočtu

1. Sestavit graf priority úloh tak, aby uzly, které představují úlohy o stejné prioritě, byly v grafu priority úloh zobrazeny vertikálně ve sloupcích.
2. Setřídit úlohy podle jejich pořadí ve sloupcích, sloupec I na začátku. Jestliže se nějaká úloha nachází ve více sloupcích, vypsat všechny sloupce.
3. Přiřazení úloh stanicím se začíná od sloupce I. Postupně přiřazovat ostatní úlohy podle pořadí čísla sloupce tak, aby nebyla překročena doba trvání cyklu.

V následujícím obrázku je graf priority úloh zobrazený podle kroku č. 1



Obr. 12 Graf priorit úloh zobrazený podle kroku č. 1. - data ASLB



Obr. 13 Graf priorit úloh zobrazený podle kroku č. 1. - data Bukchin

Seznam úloh setříděný podle kroku č. 2:

Úloha	Sloupec
1	I
2	I
3	II
4	II
5	II, III
6	III
7	III
8	III
9	IV
10	IV
11	V
12	V

Tabulka č. 13: Úlohy setříděné podle sloupců – krok č. 2 - data ASLB

Úloha	Model	Sloupec
1	1	I
2	1	I
1	2	I
3	1	II
4	1	II
3	2	II
5	2	II
5	1	III

Tablka č. 14: Úlohy setříděné podle sloupců – krok č. 2 - data Bukchin

V následující tabulce je uvedeno přiřazení úloh stanicím po provedení kroků 2 a 3.

	data ASLB	data Bukchin	
Stanice	Úloha	Úloha	Model
1	1	1	1
	2	1	2
	4	5	2
	5		
2	3	2	1
	6	4	1
		3	2
3	7	3	1
	8	5	1
4	9		
	10		
5	11		
	12		

Tablka č. 15: Přiřazení úloh stanicím

12.2.4 Metoda Ranked positional weight

Tuto metodu založil Helgeson a Birnie v r. 1961 [11]. Je to kombinace metody Kilbridge a Wester a metody Largest-candidate rule.

12.2.4.1 Popis metody

1. Vypočítat RPW (ranked positional weight) každé úlohy: RPW určité úlohy je rovno součtu doby trvání této úlohy a všech následníků této úlohy v grafu priority úloh.
2. Setřídit úlohy sestupně podle jejich RPW (úloha s největší RPW bude na začátku).
3. Počínaje úlohou s největší RPW, postupně přiřazovat úlohy do nejbližší přípustné stanice. Přípustná stanice je taková stanice, ve které nebude překročena doba trvání cyklu a zároveň bude dodržena podmínka priority mezi úlohami.

12.2.4.2 Postup výpočtu

V následující tabulce jsou úlohy setříděny podle RPW:

Úloha	RPW
1	330
3	300
2	267
4	197
8	187
5	130
7	121
6	100
10	100
9	89
11	62
12	12

Tabulka č. 16: Úlohy setříděné podle RPW – krok č. 2 - data ASLB

Úloha	Model	RPW
1	1	18
2	1	10
1	2	9
3	1	6
4	1	6
3	2	4
5	2	3
5	1	2

Tableau 17: Úlohy setříděné podle RPW – krok č. 2 - data Bukchin

V následující tabulce je uvedeno přiřazení úloh stanicím po provedení kroků 2 a 3.

	data ASLB	data Bukchin	
Stanice	Úloha	Úloha	Model
1	1	1	1
	3	1	2
		5	2
2	2	2	1
	4	4	1
	5	3	2
	6		
3	8	3	1
	7	5	1
4	10		
	9		
5	11		
	12		

Tablka č. 18: Přiřazení úloh stanicím

12.2.5 Původní algoritmus Bukchin

Popis této metody je v 10.

Tento algoritmus nalezl celkem 23 přípustných řešení. V následující tabulce je uvedeno řešení s nejlepší hodnotou kriteriální funkce.

	data ASLB	data Bukchin	
Stanice	Úloha	Úloha	Model
1	2	1	1
	5	1	2
		5	2
2	1	2	1
	3	4	1
	4		
3	6	3	1
	7	5	1
		3	2
4	8		
	9		
5	10		
	11		
	12		

Tablka č. 19: Přiřazení úloh stanicím

12.2.6 Vyhodnocení výsledků

V následující tabulce je uvedeno srovnání výsledků podle počtu stanic a podle počtu duplikovaných úloh.

Metoda	Počet stanic		Počet duplikovaných úloh	
	data ASLB	data Bukchin	data ASLB	data Bukchin
Largest-candidate rule	5	3	-	2
Kilbridge a Wester	5	3	-	2
Ranked positional weight	5	3	-	2
Bukchin	5	3	-	1

Tabulka č. 20: Srovnání výsledků

Počet obsazených stanic je u všech metod stejný pro data ASLB i data Bukchin. Data ASLB definují pouze jeden model, proto nemůže nastat duplikace úlohy. Pouze u příkladu vypočítaného metodou Bukchin došlo k duplikaci jedné úlohy, u ostatních metod byly duplikovány dvě úlohy. Nejlepšího výsledku tedy bylo dosaženo metodou Bukchin. Tento výsledek odpovídá skutečnosti, že Bukchinův algoritmus je navržen pro vícemodelovou montážní linku a ve výpočtu kriteriální funkce zohledňuje mimo jiné i duplikaci úloh. Rozsah této práce neumožňuje uvést výpočty pokročilejších metod, které byly navrženy pro stanovování konfigurace vícemodelové výrobně-montážní linky.

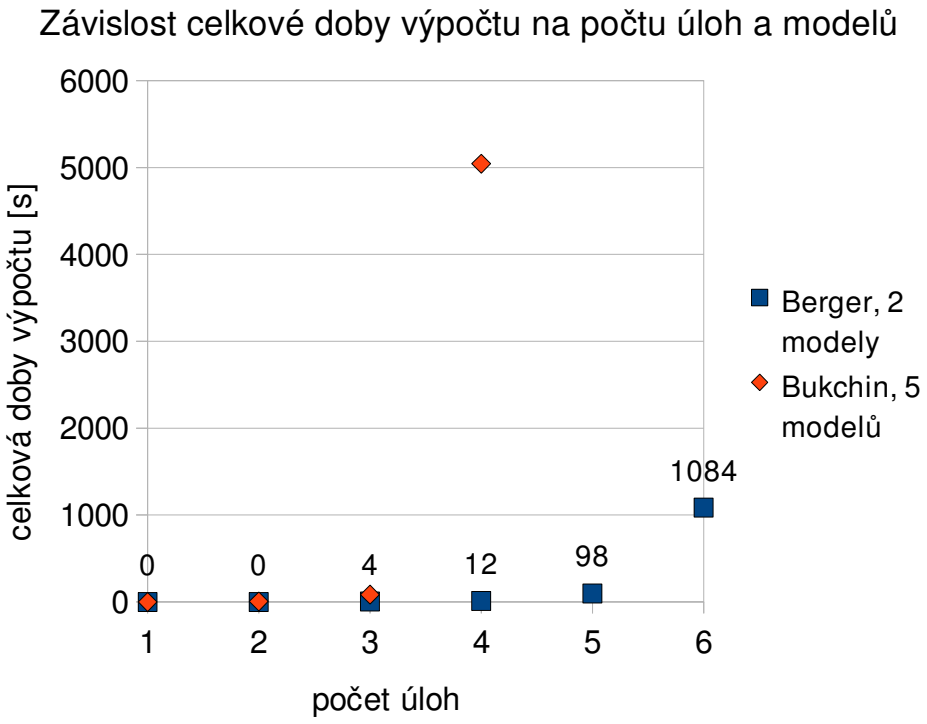
12.3 Měření doby běhu programu v závislosti na počtu úloh a modelů

V následující tabulce jsou uvedeny doby trvání výpočtů v sekundách:

	Doba trvání výpočtů [s]	
	Berger	Bukchin
Počet úloh		
1	< 1	< 1
2	< 1	5
3	4	83
4	12	5045
5	98	-
6	1084	
7	-	

Tablka č. 21: Doby trvání výpočtů

Příklad Berger se skládal ze dvou modelů, doby trvání úloh byly definovány podle direktivy preprocesoru BER_DOBA_ULOHY2. Příklad Berger se skládal z pěti modelů.



Obr. 14 Grafické znázornění doby trvání výpočtu

12.4 Srovnání kvality výsledků s výsledky získanými úpravou původního algoritmu

V této kapitole bude provedeno srovnání kvality výsledků původního algoritmu z [3] a výsledků získanými úpravou původního algoritmu 10.3.

12.4.1 Vstupní data

V tomto srovnání byly použity tyto příklady:

1. Bukchin 12.1.2
2. Vilarinho 12.1.3
3. Berger 12.1.4
4. Berger-Production 12.1.5

12.4.2 Použité úpravy původního algoritmu

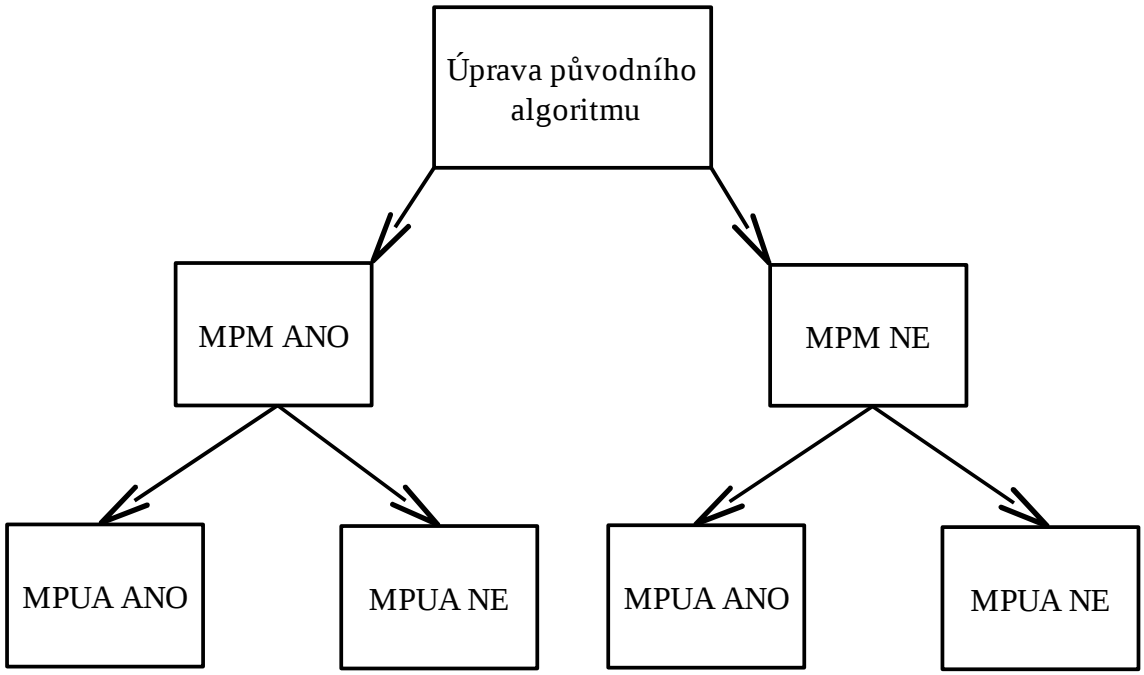
V tomto srovnání byly použity tyto úpravy původního algoritmu:

1. Mazání jednoho primárního kandidáta podle doby trvání úlohy 10.3.1.1
2. Mazání jednoho primárního kandidáta podle počtu následníků 10.3.1.2
3. Mazání jednoho primárního kandidáta podle nákladů na úlohu TC 10.3.1.3
4. Mazání jednoho primárního kandidáta podle RPW, Ranked Positional Weight 10.3.1.4
5. Mazání více primárních kandidátů podle doby trvání úlohy 10.3.1.5 (pro alespoň 3 různé poměry smazaných kandidátů)
6. Mazání více primárních kandidátů podle nákladů na úlohu TC 10.3.1.6 (pro alespoň 3 různé poměry smazaných kandidátů)
7. Úprava založená na přiřazování úloh stejného modelu 10.3.3

Každá úprava původního algoritmu uvedená v předchozím odstavci v bodech 1 až 6 byla dále kombinována s těmito modifikacemi:

1. Mazání primárních kandidátů podle modelu (s výjimkou Mazání jednoho primárního kandidáta podle RPW 10.3.1.4, kde chybí informace o modelu)
2. Mazání primárních kandidátů podle UA

Pro každou úpravu původního algoritmu tedy vznikly 4 kombinace, jak je graficky znázorněno na následujícím obrázku:



Obr. 15 Kombinace úpravy původního algoritmu

Vysvětlivky:

MPM - Mazání primárních kandidátů podle modelu
MPUA - Mazání primárních kandidátů podle UA
ANO – uvedená modifikace je aktivní
NE – uvedená modifikace není aktivní

Pro každý příklad tedy bylo provedeno nejméně 44 testů (24 + 2*10).

12.4.3 Výsledky testů

Výsledky testů je možné hodnotit podle dvou hledisek:

1. podle hodnoty kriteriální funkce
2. podle celkové doby výpočtu

12.4.3.1 Zhodnocení výsledků podle hodnoty kriteriální funkce

Hodnota kriteriální funkce, v tomto případě LB (Lower Bound), dolní mez, je hlavní měřítko výsledné kvality řešení, protože odpovídá nákladům na sestavení výrobně-montážní linky. Pro přehledné zobrazení kvality řešení použitých úprav byly vypočítány odchylky od LB původního algoritmu a tyto odchylky byly zobrazeny v grafech – viz příloha č.1. Čím menší odchylka, tím lepší kvalita řešení. Záporná odchylka znamená, že hodnota kriteriální funkce, tedy náklady na sestavení výrobně-montážní linky, je nižší než u řešení získaného původním algoritmem. Této nižší hodnoty bylo dosaženo aplikací úpravy založené na přiřazování úloh stejného modelu (UPRAVA_3) 10.3.3 na příklad Berger 12.1.4.

Ke zlepšení došlo z toho důvodu, že celkový počet stanic byl o jednu menší než u původního algoritmu. Původní algoritmus v tomto případě vytvářel takové řešení, aby nedocházelo k duplikaci úloh, ale celkový počet stanic je tak o jednu větší, než u metody UPRAVA_3. Přiřazení úloh je

znázorněno v příloze v souboru 2011_DP_Dolezal_Zbynek_45281.ods, list BER_DOBA_ULOHY2.

V některých případech došlo ke zhoršení kriteriální funkce oproti hodnotě kriteriální funkce získané původním algoritmem. Je to z toho důvodu, že při mazání primárních kandidátů dochází ke ztrátě informace o některých úlohách. U každého příkladu ale vždy několik metod poskytovalo výsledek se stejnou hodnotou kriteriální funkce jako původní algoritmus.

12.4.3.2 Zhodnocení výsledků podle celkové doby výpočtu

Ve většině případů byla při použití jakékoliv úpravy celková doba doba výpočtu několikrát menší než u původního algoritmu. Největší zrychlení výpočtu při zachování stejné hodnoty kriteriální funkce bylo více než padesátinásobné: příklad Berger, doba trvání úloh podle direktivy preprocesoru BER_DOBA_ULOHY2, úprava mazání více primárních kandidátů podle nákladů na úlohu TC 10.3.1.6 v kombinaci s úpravou mazání primárních kandidátů podle UA 10.3.2.2. Doba výpočtu původního algoritmu byla 930 s, doba výpočtu při použití této úpravy byla 18 s.

Jak již bylo uvedeno v předchozí podkapitole 12.4.3.1, v některých případech došlo ke zhoršení kriteriální funkce oproti hodnotě kriteriální funkce získané původním algoritmem. U každého příkladu ale vždy alespoň jedna metoda poskytovala výsledek se stejnou hodnotou kriteriální funkce jako původní algoritmus a zároveň byla doba výpočtu této metody několikanásobně menší než doba výpočtu původního algoritmu.

12.4.4 Přehled vylepšení původního algoritmu

Původní algoritmus podle [3] byl v této práci vylepšen těmito způsoby:

1. lepší hodnota kriteriální funkce - úprava založená na přiřazování úloh stejného modelu (UPRAVA_3) 10.3.3, data Berger 12.1.4
2. kratší doba výpočtu - u každého testovaného příkladu vždy alespoň jedna metoda poskytovala výsledek se stejnou hodnotou kriteriální funkce jako původní algoritmus a zároveň byla doba výpočtu této metody několikanásobně menší než doba výpočtu původního algoritmu.

Protože se při každém spuštění programu použije vždy jen jedna metoda nebo původní algoritmus, pro nalezení optimálního řešení je vhodné vyzkoušet všechny možné kombinace a z nich vybrat výsledek s nejlepší hodnotou kriteriální funkce. Popřípadě je možné stávající program upravit tak, aby se automaticky spustily všechny možné kombinace úprav i původní algoritmus a z těchto výsledků se vybralo optimální řešení.

12.5 Ukázka zdrojového kódu

```
bool CastecneReseni::UlohaJePrirazena(int nUloha, int nModel) const
{
   for (int i = 0; i < (int) m_UA_NeprirazeneUlohy[nModel].size(); i++)
   {
      if (nUloha + 1 == m_UA_NeprirazeneUlohy[nModel][i])
      {
         return false;
      }
   }

   return true;
}

int BukchinData::Stanice(int nUloha, int nModel, const VECTOR_3D_INT & rx_PrirazeniUloh)
{
   for (int k = 0; k < (int) rx_PrirazeniUloh[nUloha][nModel].size(); k++)
   {
```

```
      if (rx_PrirazeniUloh[nUloha][nModel][k] == 1)
      {
         return k;
      }
   }

   return -1;
}

bool CastecneReseni::JeSplnenaPodminkaCasuModelu(int nUloha, int nModel, int nStanice) const
{
   if (nUloha >= 0 && nModel >= 0 && nStanice >= 0)
   {
       double dCasModelu = 0; // udava aktualni casovou vytizenost modelu. Musi byt mensi nez cas
modelu c(j)
      for (size_t i = 0; i < m_x_PrirazeniUloh.size(); i++)
         if (x_PrirazeniUloh(i, nModel, nStanice) == 1)
         {
            if ((int) i != nUloha)
            {
               dCasModelu += m_crData.t_DobaUlohy()[i][nModel];
            }
         }

      if ( dCasModelu + m_crData.t_DobaUlohy()[nUloha][nModel] <
         m_crData.c_PozadovanyCasModelu()[nModel] + dNula )
      {
         return true;
      }
      else
      {
         return false;
      }
   }
   return false;
}
```

13. ZÁVĚR

V této práci byla provedena analýza dosavadního stavu výzkumu v oblasti stanovování konfigurace vícemodelové výrobně-montážní linky a podobných problémů. Bylo provedeno porovnání kvality řešení získaných metodou Bukchin [3] a tradičními metodami: Largest-candidate rule, Kilbridge a Wester, Ranked positional weight. U příkladu s jedním modelem byl počet stanic stejný. Podle očekávání nejkvalitnější řešení u vícemodelového příkladu poskytuje metoda Bukchin, protože je navržena pro tento typ úloh.

Také bylo provedeno porovnání metody Bukchin s metodami, které jsou určené pro stanovování konfigurace vícemodelové výrobně-montážní linky: metoda Xhao [4] a Vilarinho [5]. Kritériem pro porovnání byl maximální počet úloh a modelů, které lze těmito metodami vyřešit v reálném čase. Podle tohoto kritéria je nejlepší metoda Xhao, následovaná metodou Vilarinho a Bukchin.

Dále bylo v této práci navrženo několik úprav původního algoritmu Bukchin za účelem zlepšení kvality řešení, které tento algoritmus poskytuje. Tyto úpravy byly testovány na různých příkladech. V rámci každého příkladu vždy alespoň jedna úprava poskytovala výsledek se stejnou hodnotou kriteriální funkce jako původní algoritmus a zároveň byla doba výpočtu při použití této úpravy několikanásobně menší než doba výpočtu původního algoritmu. Největší zrychlení výpočtu při zachování stejné hodnoty kriteriální funkce bylo více než padesátinásobné. Také byl zaznamenáno zlepšení hodnoty kriteriální funkce oproti původnímu algoritmu a tím snížení nákladů na sestavení výrobně montážní linky. Jedná se o úpravu založenou na přiřazování úloh stejného modelu 10.3.3, příklad Berger 12.1.4.

Protože se při každém spuštění programu použije vždy jen jedna úprava nebo původní algoritmus, pro nalezení optimálního řešení je vhodné vyzkoušet všechny možné kombinace a z nich vybrat výsledek s nejlepší hodnotou kriteriální funkce. Popřípadě je možné stávající program upravit tak, aby se automaticky spustily všechny možné kombinace úprav i původní algoritmus a z těchto výsledků se vybralo optimální řešení.

Dále by se tento program mohl vylepšit přidáním dalších úprav – např. mazání více kandidátů podle počtu následníků v grafu priority úloh a podle RPW. Dále by se mohly provést podrobnější testy úpravy založené na přiřazování úloh stejného modelu 10.3.3 - v kombinaci s mazáním jednoho primárního kandidáta a s mazáním primárních kandidátů podle modelu 10.3.2.1 a podle UA 10.3.2.2.

14. SEZNAM POUŽITÉ LITERATURY

[1] Poliščuk, R.: Titulní strana závěrečné práce, ,
[2] Poliščuk, R.: Instrukce pro autory závěrečných prací, 2006, http://autnt.fme.vutbr.cz/doc/SZZ2006_Instrukce.pdf
[3] Bukchin, Y., Rabinowitch, I.: A Branch-And-Bound Based Solution Approach For The Mixed-Model Assembly Line-Balancing Problem For Minimizing Stations And Task Duplication Costs. European Journal of Operational Research, Vol. 174, Issue 1, October 2006, pp 492-508.
[4] Xhao X., Ohno K., Lau H.S.: A Balancing Problem for Mixed Model Assembly Lines with a Paced Moving Conveyor. Naval Research Logistic 51 /2004/, pp.446-464.
[5] Vilarinho, P.M., Simaria, A.L.: A Two-Stage Heuristic Method for Balancing Mixed-Model Assembly lines with Parallel Workstations. International Journal of Production Research 40, No.6 / 2002, pp. 1405–1420.
[6] Van Zante-de Fokkert, De Kok, T.G., 1997. The mixed and multi model line-balancing problem: A comparison. European Journal of Operational Research 100, pp. 399-412.
[7] Karp, R.M., 1972. Reducibility among combinatorial problems. V: Miller, R.E., Thatcher, J.W. (Eds.), Complexity of Computer Computation. Plenum Press, New York, pp. 85-103.
[8] Johnson, J.R., 1988. Optimally balancing large assembly lines with FABLE. Management Science 34, pp. 240.
[9] Berger, I., Bourjolly, J.M., Laporte, G., 1992. Branch-and-bound algorithm for the multi-product assembly line balancing problem. European Journal of Operational Research 58, pp. 215-222.
[10] Scholl, A., Klein, R., 1997. SALOME: A bi-directional branch-and-bound procedure for assembly line balancing. INFORMS Journal on Computing 9 (4), pp. 319-334.
[11] Helgeson, W. P., Birnie, D. P., Assembly line balancing using the ranked positional weight technique, Journal of Industrial Engineerin, Vol. 12(6), pp. 384-398, 1961.
[12] Becker C., Scholl, A., A survey on problems and methods in generalized assembly line balancing. [online]. European Journal of Operational Research, 2004. [cit. 14.5.2011]. Dostupný z: <http://dx.doi.org/doi:10.1016/j.ejor.2004.07.023>.
[13] McConnell, S. Dokonalý kód. Umění programování a techniky tvorby software. 1. vyd. Brno: Computer Press, 2005. 274 s. ISBN 80-251-0849-X.
[14] [Jméno autora neuvedeno]. Assembly systems and line balancing. [online]. [cit. 16.5.2011]. Dostupný z: <http://wwwme.nchu.edu.tw/~CIM/courses/Flexible%20Manufacturing%20Systems/Microsoft%20Word%20-%20Chapter8F-ASSEMBLY%20SYSTEMS%20AND%20LINE%20BALANCING.pdf>.
[15] [Jméno autora neuvedeno]. Product Layout. [online]. [cit. 21.5.2011]. Dostupný z: <https://pantherfile.uwm.edu/uksaxena/www/course/IE370-Saxena/product.PDF>.

15. SEZNAM PŘÍLOH

Příloha 1. Rozdíl LB jednotlivých metod a původního algoritmu.
Příloha 2. Disk CD-ROM s obsahem:

- elektronická podoba práce
- zdrojový kód
- soubory s výsledky testů

16. PŘÍLOHA 1. ROZDÍL LB JEDNOTLIVÝCH METOD A PŮVODNÍHO ALGORITMU