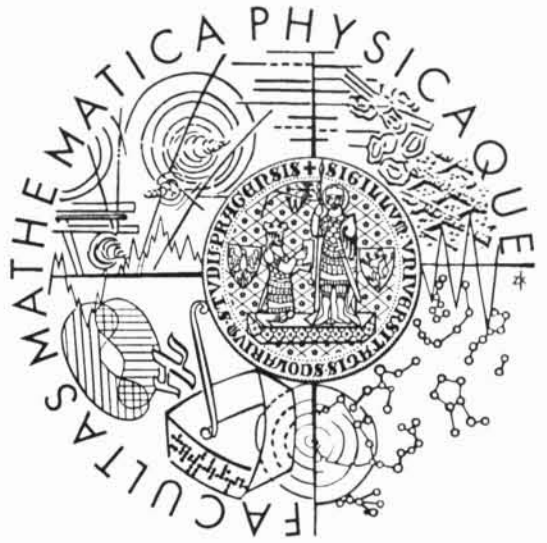
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Tomáš Dyntar

Rozvrhovací úlohy typu Flow-shop

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: RNDr. Miron Tegze, CSc.

Studijní program: matematika

Prosím, aby čtenář všechny případné nedostatky připsal mně jako autorovi práce, vedoucí práce neměl mnoho příležitostí se k ní vyjádřit. Děkuji přátelům Karlovi Mokrému a Petrovi Burianovi za vydatnou pomoc při pronikání do tajů TEXu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 16.12.2005

Tomáš Dyntar

Obsah

Abstrakt

Název práce: Rozvrhovací úlohy typu Flow-shop
Autor: Tomáš Dyntar
Katedra: Katedra pravděpodobnosti a matematické statistiky
Vedoucí diplomové práce: RNDr. Miron Tegze, CSc.
e-mail vedoucího: miron@tegze.cz
Abstrakt: Je podán popis rozvrhování flow shop a lot streaming včetně příkladu. Dále jsou ukázány metody a algoritmy pro řešení různých verzí problému určení velikosti skupin pro přepravu mezi stroji v prostředí flow shop se dvěma stroji. Začne se s výpočtem úlohy se dvěma skupinami (diskrétní případ), třemi skupinami (spojitý případ) a pak je představen Trietschův polynomiální algoritmus pro libovolný počet skupin. Tento algoritmus je poté využit při řešení modifikací, kdy se hledá příslušný počet skupin pro danou celkovou dobu trvání procesu, kdy se uvažuje konstantní či lineární doba na přepravu včetně variant s omezeným počtem přepravních zařízení či jejich omezenou kapacitou, kdy se uvažuje konstantní či lineární doba nastavení strojů. Dále je vytvořen model s přepravou i dobou nastavení a Trietschův algoritmus je příslušným způsobem upraven. Pro rozšíření výsledků případů s jedním strojem na případy s více stroji je navrženo užít větu R.G.Vicksona, která řeší problém s určením skupin v případě více jobů pomocí pojednání problému s určením velikosti skupin pro každý job samostatně a poté užitím Johnsonova algoritmu. Také je ukázán známý způsob k rozšíření výsledků na úlohu se třemi stroji prostřednictvím řešení vždy dvou sousedních strojů zvlášť nebo upravením Trietschova algoritmu pro tři stroje. Většina postupů je předvedena na příkladech.
Klíčová slova: rozvrhování, flow shop, lot streaming

Abstract

Title: Flow Shop Scheduling
Author: Tomáš Dyntar
Department: Department of Probability and Statistics
Supervisor: RNDr. Miron Tegze, CSc.
Supervisor's e-mail address: miron@tegze.cz
Abstract: A description of flow shop and lot streaming including an example is given. Methods and algorithms for solving various modifications of the single job two-machine flow shop transfer lot sizing problem are shown. It starts with the calculation of two lots (discrete), three lots (continuous) problem and then Trietsch's polynomial algorithm for an arbitrary number of sublots is introduced. This algorithm is then used for solving modifications with finding the proper number of sublots for given makespan, with constant or linear transfer times including variants with limited number of transfer machines or their limited capacity, with constant or linear setup times. A model with both transfer and setup times is then developed and Trietsch's algorithm modified. A theorem by R.G.Vickson for solving multiple job lot streaming problem by treating the lot streaming problem for each job separately and then using Johnson's algorithm is proposed to extend the results of the single job cases to those of multiple jobs. A known way to extend results to the three machine problem by treating each pair of successive machines separately or developing Trietsch's algorithm for three machines is shown. Most of the procedures are illustrated in examples.
Keywords: scheduling, flow shop, lot streaming

Kapitola 1

Úvod

Tato diplomová práce se zabývá rozvrhováním typu flow shop. Jedná se o problematiku, která se neustále rozvíjí od 50. let minulého století. Díky tomu jde dnes o již velmi rozsáhlou, rozpracovanou oblast s mnoha výsledky. Tomu odpovídá i rozsáhlá literatura.

Základní úloha určení optimálního pořadí zpracovávaných jobů na jednotlivých strojích při zachování stejného pořadí strojů pro všechny joby byla obohacena různými dodatečnými požadavky, nepočítaje v to ani počet strojů. Může tak jít např. o požadavek, aby každý stroj pracoval nepřetržitě či naopak aby každý job byl zpracován bez čekání mezi jednotlivými operacemi, požadavky mohou být kladeny na přepravu mezi stroji, na nastavení na jednotlivých strojích apod. Velice důležitý je rovněž celkový přístup k charakteru procesu. Konkrétně se jedná o to, zda jde o model deterministický, ve kterém je dopředu vše známo, či zda se uvažuje s prvkem náhody a vytváří se model stochastický.

Jedním ze zajímavých směrů, kudy se zájem ubírá, je optimalizace celého procesu, pokud se skládá z jednoho či více jobů, z nichž každý je složen ze stejných úkolů. V těchto úlohách pak jde primárně o to, jak nejlépe rozdělit job či joby na skupiny tak, aby byl proces optimalizován. Když si představíme převážení palet se stejnými výrobky od jednoho stroje k druhému, otázkou je, po kolika kusech se mají výrobky převážet. A opět si lze klást různé požadavky či omezení - počet přepravních zařízení, jejich kapacita či nosnost, počet skupin apod.

Právě touto oblastí se tato diplomová práce zabývá. Nechť tato práce napomůže k tomu, aby byl čtenář tématem zaujat, obeznámen a hlavně motivován k vlastnímu přemýšlení a tvůrčí činnosti.

Kapitola 2

Flow shop

2.1 Popis problematiky

Celá problematika, o kterou se v této práci jedná, je zasazena do prostředí flow shop. Jedná se o prostředí, kde je dané úkoly třeba zpracovat na jednotlivých strojích, a sice tak, že pořadí strojů, na kterých budou úkoly zpracovávány, je stejné pro všechny úkoly. Operace prováděné na jednotlivých strojích se mohou úkol od úkolu lišit či dokonce nemusí dojít k žádné operaci pro určitý výrobek na určitém stroji. Pokud se žádná operace neprovádí, jsou dvě základní možnosti, jak tuto situaci v modelu pojmout: buď je její doba stanoví jako 0, přičemž se s jejím provedením musí i tak počítat. To znamená, že na daném stroji nemůže probíhat operace na jiném úkolu, aby tato operace proběhla, ač je její trvání nulové; nebo se tako operace úplně vypustí, a není tak třeba nikterak brát v potaz, zda a kdy může být úkol na daném stroji zpracován. Doba provedení operace na strojích se může úkol od úkolu lišit. Ba naopak, kdyby se nelišily, byly by některé úlohy řešitelné triviálně.

Přirozenou úlohou v takovém prostředí je optimalizace celého procesu. Kritérií a omezení může přitom být celá řada. Jako první nasnadě je zkrátit celkovou dobu zpracování úkolů na minimum. K tomu pak přidat omezení na dopravu: například dobu přepravy od jednoho stroje ke druhému, dobu návratu převravního zařízení, dobu nakládky či vykládky, kapacitu přepravního zařízení apod. Další omezení mohou plynout od strojů - např. čas na přenastavení strojů na jednotlivé operace, požadavek na nepřerušovanou činnost stroje či naopak potřeba přerušit jeho činnost po určité době. Vlastní optimalizační kritéria mohou být rovněž různá - již zmíněná celková doba zpracování úkolů, minimalizace součtu dob od zahájení zpracování jednotlivých úkolů na prvním stroji až do ukončení jejich zpracování na posledním stroji apod.

Důležitým kritériem je znalost všech faktorů, které v modelu vystupují. Buď je známe ještě před zahájením celého procesu, nebo neznáme. Typicky se to týká samotných úkolů a jejich počtu. Může tak být již dopředu známo, kolik bude úkolů a jak dlouho budou trvat operace na jednotlivých strojích, příp. další potřebné informace. Nebo do procesu vstupuje prvek náhody a přesný počet úkolů či jejich

vlastnosti nejsou známy. Jde tak buď o deterministický, nebo stochastický model. Stochastický model však může vzniknout nejen díky úkolům, ale např. i díky zahrnutí poruch strojů či přepravních zařízení apod.

Zobecněním prostředí flow shop je open shop. Úkoly jsou v tomto prostředí rovněž zpracovávány na daných strojích, ovšem jejich pořadí není pevně dané. Jestliže se tedy nalezne řešení pro prostředí flow shop, je toto řešení jistě také řešením v prostředí open shop. Toto řešení pak lze již jen zlepšit, protože množina přípustných rozvrhů prostředí flow shop je podmnožinou řešení v prostředí open shop.

Je zřejmé, že celá problematika flow shop je velice rozsáhlá a lze ji nadále rozšiřovat. K tématu bylo rovněž publikováno mnoho literatury. Proto ji není možné v rámci této práce pojmout v její komplexnosti a zaměření na určitý okruh je nezbytností. Následující kapitoly se zabývají tematikou *lot streaming*.

Kapitola 3

Lot streaming

3.1 Popis problematiky

Výchozí situace je následující: Pohybujeme se v prostředí *flow shop*, tj. máme úkoly, z nichž každý musí být zpracován na daných strojích, a to tak, že pořadí strojů je pro všechny úkoly stejné. Nechť do této základní situace vstoupí prvek dopravy. Úkoly je třeba mezi jednotlivými stroji přemísťovat. Tím do hry vstupuje či může vstoupit kapacita přepravního zařízení, doba přepravy, doba návratu přepravního zařízení, doba nakládky či vykládky a jistě by dalo uvažovat i o dalších faktorech. Názorným příkladem může být přepravování výrobků po výrobní hale od stroje ke stroji na paletách pomocí vysokozdvižných vozíků.

Jsou dvě krajní situace, jak úkoly nechat proudit (stream) celým procesem. Buď se každý úkol přemísťuje samostatně, tzn. jakmile je hotov na jednom stroji, lze ho přepravit na další stroj. Druhou krajní situací je zpracovat všechny úkoly na jednom stroji a uvolnit je k přepravě či dalšímu zpracování až poté, kdy jsou všechny hotovy. Pokud bychom měli všechny úkoly stejné (ve smyslu, že by všechny byly zpracovávány stejně dlouho) a neomezené přepravní možnosti, optimální řešení pro kritéria s dobou zpracování všech úkolů by určitě bylo dosaženo, kdyby byly úkoly přemísťovány po jednom. Doba přepravy mezi stroji již nejde zkrátit, takže musí jít o optimální řešení. Zde by ani nezáleželo na pořadí jednotlivých úkolů na daných strojích - záleželo by jen na tom, zda je v okamžiku uvolnění stroje nějaký úkol k dispozici.

Protipólem je přemísťování všech úkolů mezi stroji najednou. Kdyby například přepravní náklady byly vysoké nebo třeba bylo k dispozici jen jedno přepravních zařízení s dlouhým časem návratu, mohlo by být výhodné zpracovat všechny úkoly na jednom stroji, přemístit je najednou na další stroj a tak podobně až do konce.

Základní otázkou tedy je, jak vhodně danou skupinu úkolů dělit po zpracování jednotlivých úkolů na daném stroji do skupin, které potom budou přemístěny po jedné ke zpracování na dalším stroji. Přičemž vhodnost je dána tím, aby bylo optimalizováno dané kritérium. Tato problematika se nazývá *lot streaming*.

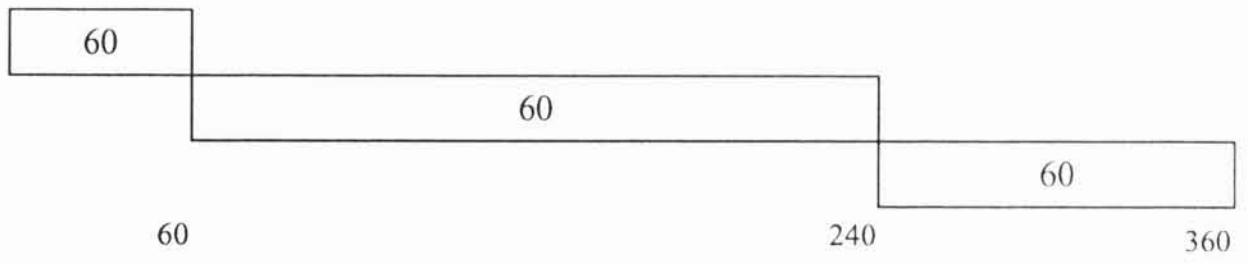
3.2 Ilustrační příklad

Na úvod uveďme ilustrační příklad, který ukazuje, jak se mění výsledky, pokud měníme vstupní požadavky na celý proces (další příklad viz [1]):

Příklad 3.2.1

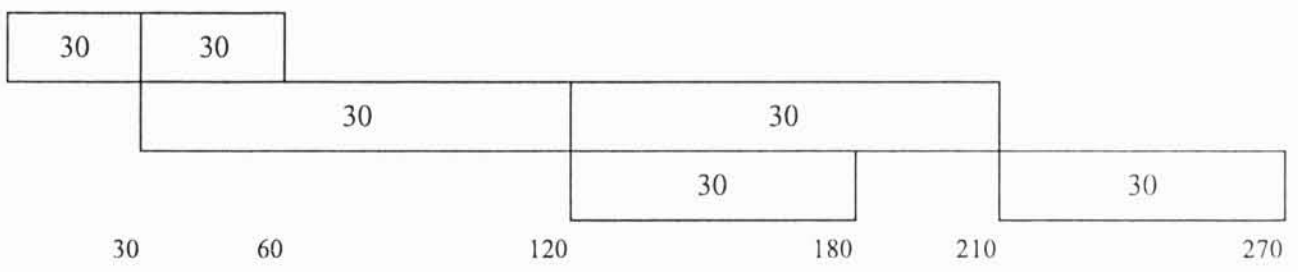
Za úkol je zpracovat 60 úkolů na třech strojích. Na prvním stroji se každý úkol zpracovává 1 minutu, na druhém 3 minuty a na třetím 2 minuty. Doba na přepravu je nulová.

1. Jestliže celou partii přemísťujeme najednou, je celková doba zpracování rovna $(1 + 3 + 2) * 60 = 360$ minut (obr. 3.1).



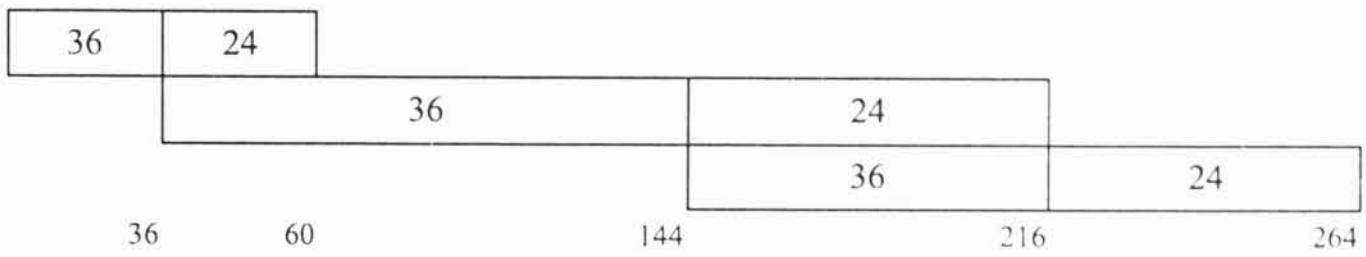
Obr. 3.1: Přemístění najednou

2. Další způsob, který je nasnadě, je rozdělit partii na dvě stejné poloviny a zpracovávat a přemísťovat každou polovinu samostatně. Díky tomu je celková doba zpracování partie 270 minut, což je zlepšení o 25% (obr. 3.2).



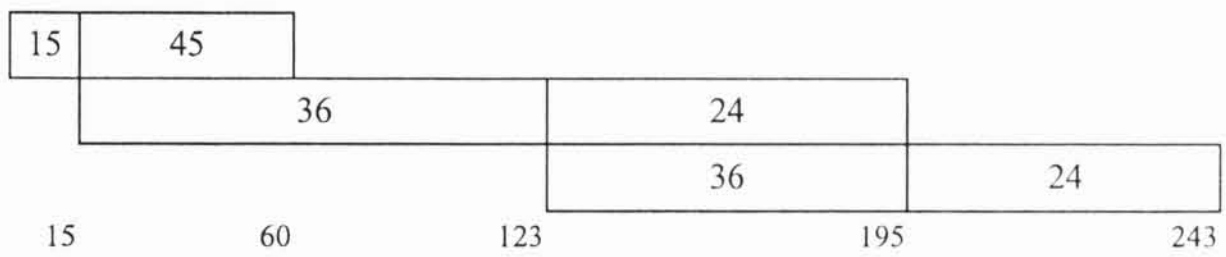
Obr. 3.2: Rozdělení na poloviny

3. Na obr. 3.2 je zřejmý prostoj na třetím stroji mezi první a druhou částí partie. Celkovou dobu zpracování partie by mohlo zkrátit, kdyby se partie rozdělila tak, aby nedošlo k tomuto prostoji. K němu nedojde, pokud doba zpracování druhé části partie na druhém stroji se rovná době zpracování první části na třetím stroji. Opět se každá z obou částí bude přemísťovat samostatně. Celková doba zpracování partie vyjde 264 minut, což je zlepšení o 26, 7% (obr. 3.3).
4. Jestliže navíc povolíme, že mezi prvním a druhým strojem mohou být skupiny jiné velikosti než mezi druhým a třetím strojem, dostáváme délku zpracování



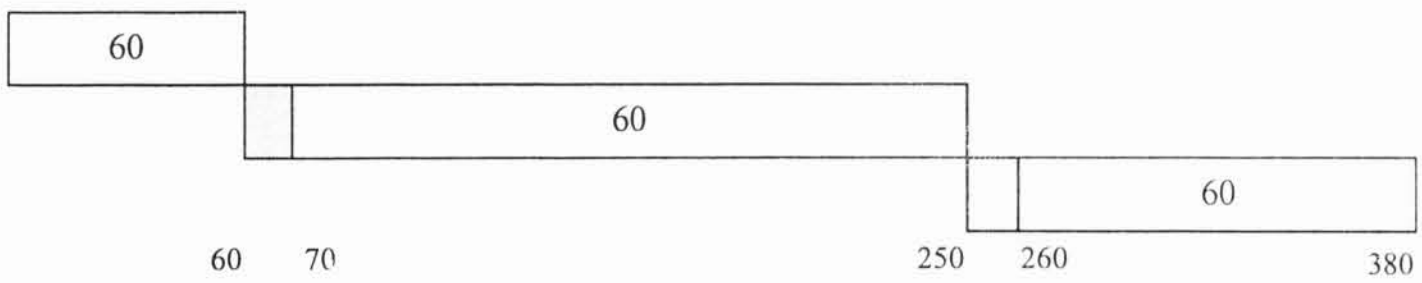
Obr. 3.3: Nestejné části

rovnou 243 minutám, což je 32, 5% zlepšení proti situaci v bodu 1. Postup z bodu 3 se využije i pro dělení mezi prvním a druhým strojem, tj. doba zpracování první skupiny na druhém stroji se musí rovnat době zpracování druhé skupiny na prvním stroji, tedy $3x = 60 - x$, z čehož plyne, že $x = 15$. Velikost první skupiny nechť je 15, na druhou zbývá 45. Tento rozvrh je přípustný vzhledem k rozdělení $36 - 24$ mezi druhým a třetím strojem, proto dostáváme další možný rozvrh a celkové zlepšení (viz obr. 3.4).



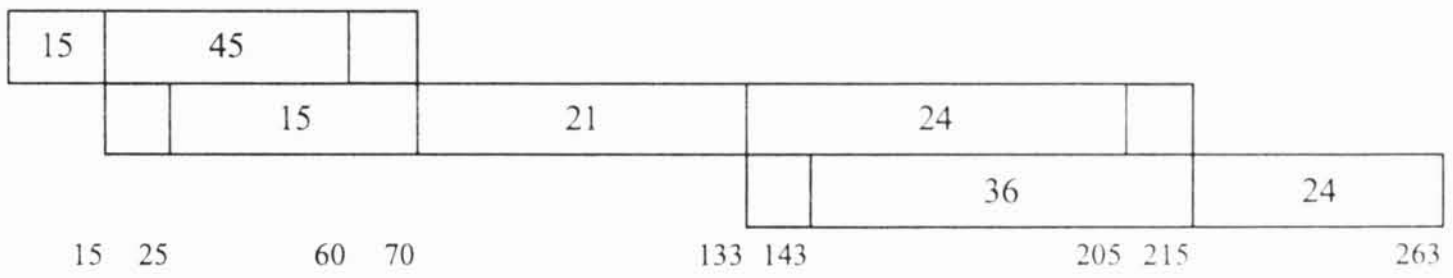
Obr. 3.4: Nestejné části na strojích

5. Faktorem, který může mít vliv na výsledek, je přepravní zařízení. V případě, že a) doba přepravy je konstatní a její délka je 10 minut, přičemž je k dispozici dostatečný počet přepravních zařízení, takže není třeba uvažovat dobu návratu přepravního zařízení, a b) přepravujeme celou partii najednou, je celková doba zpracování partie 380 minut (viz obr. 3.5).



Obr. 3.5: Zahrnuta doba na přepravu

6. V bodě 4 byla z uvedených možností doba zpracování nejkratší (243 minut). Při zahrnutí dopravy se stejně jako v bodě 5 doba prodlužuje o 20 minut na celkových 263 minut (viz obr. 3.6).



Obr. 3.6: Nestejné části na strojích s dopravou

△

Jak je vidět, závisí doba zpracování na omezeních, která na proces máme. Přirozeně platí, jak je naznačeno i v našem příkladě, že čím méně omezení, tím větší je naděje na zlepšení výsledků, neboť množina možností se tím zvětšuje, přičemž zahrnuje případy se striktnějšími předpoklady. Kdybychom použili více skupin, mohla by se doba zpracování ještě zlepšit. V žádném případě se však nedostaneme pod 183 minut - doba zpracování na 2. stroji je 180 minut a k tomu musíme připočíst 3 minuty, které zabere zpracování prvního úkolu na strojích 1 a 3.

3.3 Značení

Obecně uvažujeme prostředí flow shop. Úkol je považován za dokončený, jakmile je dokončen na posledním stroji.

U - počet identických úkolů

n - počet skupin, do kterých rozdělíme úkoly

m - počet strojů

k - počet přepravních zařízení

p_i - délka zpracovávání jednoho úkolu na i-tém stroji

TT (transfer time) - doba nakládky, přepravy a vykládky

CT (cycle time) - doba nakládky, přepravy, vykládky a návratu přepravního zařízení

RT (return time) - doba návratu přepravního zařízení ($RT = CT - TT$)

w - koeficient pro lineární TT

$s_i^0, i = 1, 2, ..., m$ (setup time) - doba nastavení na stroji i před první skupinou

s_{ij} (setup time) - doba nastavení na stroji i před skupinou j

$r_i, i = 1, 2, ..., m$ - koeficient pro lineární dobu nastavení

$L_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$ - velikost j-té skupiny, která opouští stroj i

$x_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$ - relativní velikost j-té skupiny, která opouští stroj i, ve spojitém případě

C_{max} - doba od zahájení zpracovávání úkolů na prvním stroji až do jejich dokončení na posledním stroji.

Platí tedy, že $x_{ij} = L_{ij}/U$, $\sum_{j=1}^{n} L_{ij} = U$, $i = 1, \ldots, m$, $\sum_{j=1}^{n} x_{ij} = 1$, $i = 1, \ldots, m$.

Uvedené značení odpovídá úlohám, ve kterých jsou jen jedny úkoly (jinak řečeno jeden job skládající se z identických úkolů). Těm je věnována většina této práce. V kapitole 4.4 do hry vstupují různé úkoly, a jobů je tedy více. V takovém případě se u příslušných položek navíc objevuje index l, který specifikuje jejich příslušnost k příslušnému jobu. Přibývá tak položka

$J_l, l = 1, \ldots, q$ - joby

Pokud jde o n, úlohy se mohou lišit v tom, zda je pevně dáno, zda je omezeno, či zda může být libovolné dle potřeb optimality rozvrhu. Pokud je velikost každé skupiny mezi všemi dvojicemi strojů stejná ($L_{kj} = L_{lj}$ pro všechny kombinace přípustných j, k, l), nazýváme je skupinami *konzistentními* (C - consistent). Tento požadavek je automaticky splněn, pokud chceme, aby skupiny byly *stejné* (E - equal), tzn. nejen aby každá skupina neměnila svou velikost, ale také aby všechny skupiny byly stejně velké ($L_{ij} = L_{kl}$ pro všechny kombinace i, j, k, l). [1] Na *variabilní* skupiny není kladen žádný požadavek. Skupiny tedy mohou mít různý počet úkolů jak mezi skupinami navzájem, tak se počet úkolů v jednotlivých skupinách může měnit i v průběhu procesu. Může se měnit i počet skupin.

Dalším požadavkem, tentokrát na stroje, bývá, aby některé či všechny stroje pracovaly *bez prostojů* (NI - no idling). Když jde o první nebo poslední stroj, nejde o požadavek, který by ovlivnil výsledný rozvrh. Na prvním stroji totiž vždy můžeme (pokud nepožadujeme např. *no-wait*, kdy musí každý úkol okamžitě po zpracování na jednom stroji začít být zpracováván na dalším stroji) posílat jeden úkol za druhým, stejně tak i na posledním stroji lze úkoly začít zpracovávat až v okamžiku, kdy stroj bude pracovat nepřetržitě - přitom se hodnota kriteriální funkce nezmění.

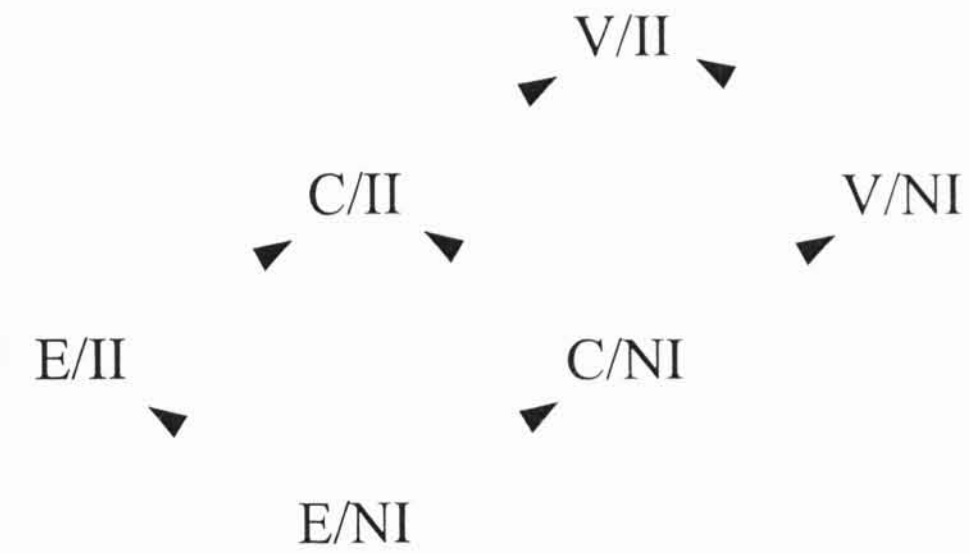
Speciálně pro *lot streaming* lze zavést toto značení pro klasifikaci situací:

$$m/\{V, C, E\}/\{II, NI\}/\{CV, DV\},$$

kde m tedy znamená počet strojů, druhá položka se týká skupin: V - proměnné, C - konzistentní, E - stejné. Na dalším místě je požadavek na prostoje strojů: II - povoleny na vnitřních strojích, NI - prostoje zakázány. Nakonec se dozvíme, zda se úloha má řešit pro spojitý (CV) či diskrétní (DV) případ.

Nejméně omezující, a tedy i případ, který hodnotu kriteriální funkce může jenom zlepšit, je $m/V/II/CV$ - povoleny jsou jak proměnné skupiny, tak prostoje strojů, a to společně s libovolným dělením souboru úkolů. Říkáme, že je tato situace *dominantní* vůči modelům s *konzistentními* či *stejnými* skupinami vzhledem k situaci *bez prostojů* či vzhledem k *diskrétnímu* případu. D.Trietsch a K.R.Baker ([1]) uvádějí obr. 3.7, který shrnuje dominance.

[1] Ještě bychom mohli uvažovat modifikaci obou právě zmíněných požadavků, a sice aby nejen počty úkolů ve skupinách byly stejné, ale aby v nich byly stále ty samé úkoly. Tento požadavek však nemá v situaci, kde jsou všechny úkoly stejné, vliv na optimální hodnotu zadaného kritéria ani na optimální rozvrh.



Obr. 3.7: Dominance

Kapitola 4

Dva stroje

Model s jedním strojem je za podmínek, kterými se zabývá tato práce, triviálně řešitelný, proto není z tohoto pohledu zajímavý. Model se dvěma stroji již přináší úlohy, které je třeba pojednat. Ani v případě dvou strojů však některá dělení a požadavky, které byly uvedeny výše, nemají význam. Jelikož je zde jen jeden přesun mezi stroji, je jen jedno rozdělení do skupin. Proto nemá požadavek konzistence skupin valný smysl, vždyť skupiny jsou v tomto případě tak či tak konzistentní. Stejně tak požadavek zpracování bez prostojů strojů. Zopakujme: První stroj totiž může pracovat neustále. Pokud zpracovávání úkolů na druhém stroji trvá stejně dlouho či déle, může se se zpracováváním začít hned, jakmile je první úkol k dispozici. Když je zpracovávání rychlejší, lze s ním počkat až do okamžiku, kdy vše půjde zpracovat najednou. Celkově je tedy jedno, jestli se jedná o model $2/V/NI$, či $2/C/NI$ včetně mezistupňů $2/C/II$ a $2/V/NI$.

4.1 Základní model

Tato kapitola se bude zabývat situací *lot streaming* v prostředí *flow shop*, a to tak, že z přepravy neplynou žádná omezení. Neuvažují se ani žádné jiné faktory, které by měly na proces vliv. Kritériem je celková doba zpracování úkolů C_{max}.

Dvě skupiny Jak bude situace vypadat, pokud mají být úkoly rozděleny jen do dvou skupin? Požadavek na *stejné* skupiny je triviálně řešitelný, takže uvažujme skupiny *variabilní*.

Lemma 4.1.1 *Máme situaci* $2/V/NI/DV, U \geq 2, n = 2, p_1 \geq 0, p_2 \geq 0$. *Nechť je* L_1 *takové, že je pro něj hodnota*

$$min(L_1 p_2, (U - L_1)p_1) \tag{4.1}$$

maximální.
Pak tento rozvrh minimalizuje C_{max}.

Důkaz Pro $U = 2$ nastává jediná možná situace. Proto i C_{max} musí být minimální. Uvažujme tedy $U > 2$. Vezmeme libovolné rozdělení do skupin $L_1, U - L_1$. 1) Ukážeme, že při takovém přemístění jednoho úkolu z jedné skupiny do druhé, aby se 4.1 nezmenšila, se ani C_{max} nezhorší. 2) Pak ukážeme, že při zmenšení výrazu se zvětší C_{max}. 3) Každý přesun se dá složit z jednotlivých přesunů, a proto tím bude důkaz hotov.

1) Máme skupiny $L_1, L_2 = U - L_1$ a vytvoříme nové skupiny $L_1 + 1, L_2 - 1$.
Nechť

$$min(L_1p_2, L_2p_1) \leq min(p_2(L_1 + 1), p_1(L_2 - 1).$$

Potom mohou nastat následující situace:

a) 4.1 se rovnala L_1p_2 a po přesunu se rovná $p_2(L_1 + 1)$. Pak platí

$$C'_{max} = Up_1 + (L_2 - 1)p_2 = Up_1 + L_2p_2 - p_2 < C_{max}$$

b) 4.1 se rovnala L_1p_2 a po přesunu se rovná $p_1(L_2 - 1)$. Pak platí

$$C'_{max} = (L_1 + 1)p_1 + Up_2 = L_1p_1 + p_1 + L_1p_2 +$$

$$+ L_2p_2 \leq Up_1 - p_1 + p_1 + L_2p_2 = Up_1 + L_2p_2 = C_{max},$$

neboť dle předpokladu je

$$L_1p_2 \leq (L_2 - 1)p_1.$$

Obdobně pro $L_1 - 1, L_2 + 1$.

2) Pro nové skupiny $L_1 + 1, L_2 - 1$ nastává jedna z následujících možností, aby

$$min(L_1p_2, L_2p_1) > min(p_2(L_1 + 1), p_1(L_2 - 1)) :$$

a) 4.1 se rovnala L_2p_1 a po přesunu se rovná $p_1(L_2 - 1)$. Pak platí

$$C'_{max} = (L_1 + 1)p_1 + Up_2 = L_1p_1 + p_1 + Up_2 > C_{max}.$$

b) 4.1 se rovnala L_2p_1 a po přesunu se rovná $p_2(L_1 + 1)$. Pak platí

$$C'_{max} = Up_1 + (L_2 + 1)p_2 = L_1p_1 + L_2p_1 +$$

$$+ L_2p_2 + p_2 > L_2p_2 + L_1p_1 + p_2 + p_2 + L_1p_2 > Up_2 + L_1p_1 = C_{max}.$$

neboť dle předpokladu je

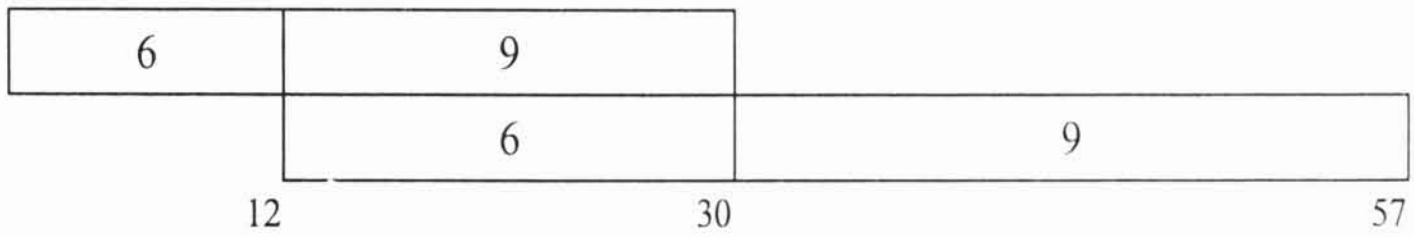
$$L_2p_1 > (L_1 + 1)p_2.$$

◊

Podívejme se nyní na tento příklad:

Příklad 4.1.1 $p_1 = 2, p_2 = 3, U = 15$.

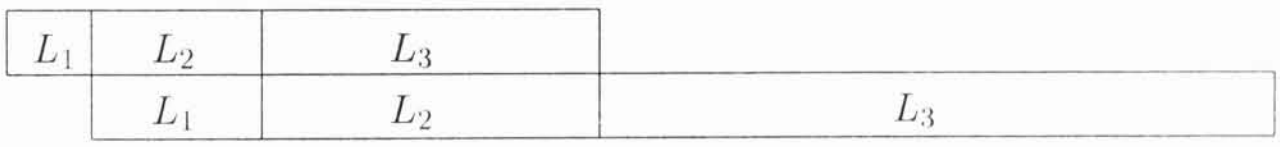
Dle lemma maximalizujeme $min(L_1 p_2, (U - L_1)p_1) = min(3L_1, 2(15 - L_1)$, což je $3L_1$ pro $L_1 \leq 6$ a $30 - 2L_1$ pro $L_1 \geq 6$. 4.1 je maximální pro $L_1 = 6, L_2 = 9$ (obr. 4.1). △



Obr. 4.1: Příklad 1

Tři skupiny Jak by celá záležitost vypadala, kdybychom měli 3 skupiny? Pomineme triviální případ *stejných* skupin a budeme uvažovat skupiny *variabilní*. Jak zde nalézt optimální rozvrh?

Protože máme pouze dva stroje, nemusíme uvažovat prostoje (2/V/NI). Jde nám vlastně o to, aby celkové překrytí práce na prvním a druhém stroji bylo co největší. Pokud by $p_2 = 2p_1$, situace by vypadala jako na obr. 4.2.



Obr. 4.2: Situace pro $p_2 = 2p_1$

To znamená, že rozvržení má být takové, aby

$$L_1 p_2 = L_2 p_1 \wedge L_2 p_2 = L_3 p_1 \wedge L_1 + L_2 + L_3 = U \tag{4.2}$$

pro $p_1 \leq p_2$. Pokud by např. $p_1 = 1, p_2 = 2, U = 7$, potom vycházejí skupiny $L_1 = 1, L_2 = 2, L_3 = 4$. Obecně po úpravách rovnic (4.2) dostáváme

$$L_1 = U\frac{p_1^2}{p_1 p_2 + p_1^2 + p_2^2}, L_2 = U\frac{p_1 p_2}{p_1 p_2 + p_1^2 + p_2^2}, L_3 = U\frac{p_2^2}{p_1 p_2 + p_1^2 + p_2^2}.$$

Pokud $p_1 = p_2$, pak $L_1 = L_2 = L_3$. Pro $p_1 > p_2$ máme situaci vlastně obrácenou, skupiny se musí zmenšovat v poměru $\frac{p_1}{p_2}$, takže zde máme

$$L_1 = U\frac{p_2^2}{p_1p_2 + p_1^2 + p_2^2}, L_2 = U\frac{p_1p_2}{p_1p_2 + p_1^2 + p_2^2}, L_3 = U\frac{p_1^2}{p_1p_2 + p_1^2 + p_2^2}.$$

Takto lze velikosti skupin počítat u velkých U, kde můžeme zaokrouhlovat a považovat případ za spojitý. Zvláště u menších sérií bychom však místo přibližného výsledku způsobeného zaokrouhlováním chtěli znát řešení přesné. Tímto problémem se zabýval D.Trietsch a navrhl následující algoritmus ([1]), který umí nalézt optimální rozdělení do libovolného počtu skupin a současně C_{max}. Algoritmus výsledek nalezne v polynomiálním čase.

Trietschův algoritmus Máme situaci $2/C/NI/DV$. S_j bude znamenat kumulativní počet úkolů v prvních j skupinách, tzn. $S_j = L_1 + L_2 + ... + L_j$. Nechť M je nějaký dolní odhad celkové doby zpracování všech úkolů. Určitě bude celý proces trvat nejméně tak dlouho, než se všechny úkoly zpracují na prvním stroji a jeden úkol na druhém či jeden na prvním a všechny na druhém. Proto za M můžeme zvolit

$$max(p_1U + p_2; p_2U + p_1). \tag{4.3}$$

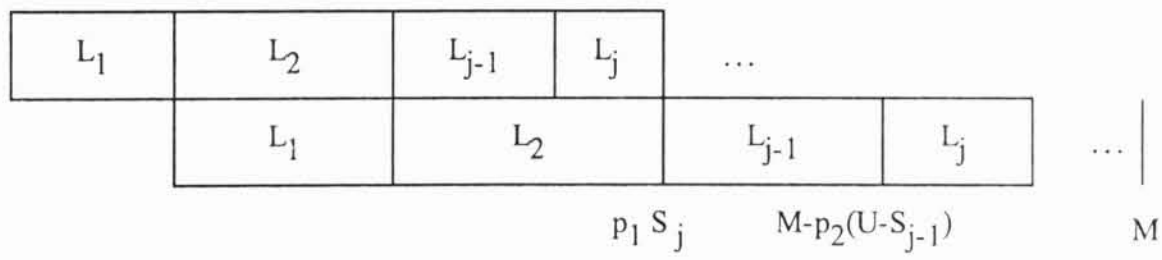
Protože jsme M stanovili jako dolní odhad, nemůže se stát, že by se dalo za tuto dobu zpracovat více úkolů či že by se doba pro dané úkoly dala zkrátit. LS_j nechť označuje okamžik, kdy nejpozději může j-tá skupina začít být zpracovávána na druhém stroji, aby se vše stihlo do M. Musí tedy platit

$$LS_j = M - p_2(U - S_{j-1}).$$

Na druhou stranu však zpracovávání j-té skupiny na druhém stroji nemůže začít dříve, než se dokončí její zpracovávání na prvním stroji. Proto musí platit (viz obr. 4.3)

$$p_1S_j \leq LS_j, \text{ tj.}$$

$$p_1S_j \leq M - p_2(U - S_{j-1}). \tag{4.4}$$



Obr. 4.3: Nerovnost pro Trietschův algoritmus

Tato nerovnice však nezaručuje, že všechna S_j budou menší než U, proto musí

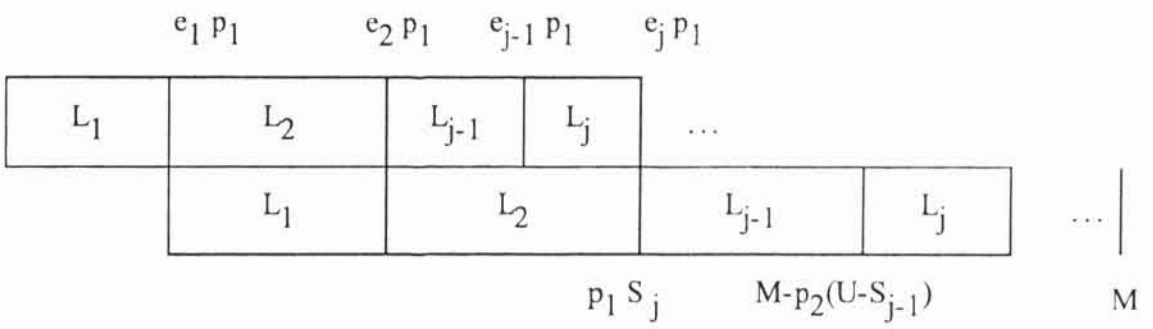
$$S_j \leq min\left(\frac{M - p_2(U - S_{j-1})}{p_1}; U\right). \tag{4.5}$$

Toto je rekurzivní nerovnice, kterou musí splňovat všechna S_j, aby byl rozvrh přípustný. S_j vždy volíme jako maximální celé číslo splňující (4.5), protože S_j je rostoucí v S_{j-1} - když se S_{j-1} zvětšuje, může být větší i S_j. Pomocí S_j můžeme spočítat L_j, jestliže známe $L_1, L_2, ...L_{j-1}$. Pro první skupinu L_1, pro níž $L_1 = S_1$, musí platit $p_1 L_1 \leq M - p_2 U$, takže rekurzivní výpočet můžeme začít pro $S_0 = 0$. Výpočet pro dané M končí, jakmile určíme velikost S_n. Pokud $S_n = U$, našli jsme optimální rozvrh a $M = C_{max}$. Jestliže však $S_n < U$, nedalo se za M stihnout U úkolů a M musí být zvětšeno. To provedeme následujícím způsobem:

Zavedeme veličinu f_j tak, že

$$f_j = min\left(\frac{M - p_2(U - S_{j-1})}{p_1}; U\right) - S_j \tag{4.6}$$

a e_j nechť je $1 - f_j$. Hodnoty $e_j p_1$ říkají, o kolik by se musel posunout počátek zpracovávání L_j na druhém stroji, aby se L_j mohla zvětšit o jeden prvek (viz obr. 4.4). Protože volíme $min\{e_j; j = 1, ..., n\}$, nemůže se stát, že bychom optimální rozvrh přeskočili.



Obr. 4.4: Kroky pro zvětšování M

S_j jsme vždy volili maximální celé číslo splňující nerovnici (4.5). $f_j p_1$ je rozdíl mezi časem, kdy nejpozději musíme s L_j začít na druhém stroji, a časem, kdy L_j skončí na prvním stroji. f_j je tedy rozpracovaná část úkolu, který se již nevešel do L_j, v okamžiku, kdy L_j začíná na druhém stroji ($p_1 f_j = LS_j - p_1 S_j$). e_j je část úkolu, kterou zbývá dokončit. Zvětšením M se do skupiny s nejvíce rozpracovaným dalším prvkem tento prvek dostane, a tím se počet jejích prvků zvětší o 1. Při libovolné hodnotě M mezi původním M a nově zvoleným by se situace nezměnila. Velikosti následujících skupin se mohou také změnit (zvětšit i zmenšit). Tímto postupem docílíme, že na konci bude $M \geq C_{max}$ a že se algoritmus nezacyklí. Ovšem začínali jsme dolním odhadem a postupovali po nejmenších krocích, při kterých se něco stalo, takže současně $M \leq C_{max}$.

Algoritmus 4.1.1 *(dále někdy jako Trietschův)*

a) $S_0 := 0$
b) *For* $j := 1$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}))/p_1, U\} \rfloor$:
$f_j = \min\{(M - p_2(U - S_{j-1}))/p_1, U\} - S_j$:
$e_j = 1 - f_j$
c) *Je-li* $S_n < U$, *pak*
$M := M + p_1 \min\{e_j\}$ *a jdi do a);*
Pokud $S_n = U$, *STOP.*

Vyzkoušejme tento algoritmus na následujícím příkladu:

Příklad 4.1.2 Obchodní společnost organizuje telefonický průzkum trhu. Počítá, že každý hovor bude trvat průměrně 10 minut a vykonávat ho bude jeden speciálně zaškolený pracovník. Bylo vytipováno 20 respondentů. Třikrát denně budou tyto záznamy odeslány dalšímu pracovníkovi, který je bude dále zpracovávat, přičemž zpracování každé odpovědi by mělo trvat zhruba 25 minut. Oba pracovníci jsou najati speciálně na tuto práci od 9:00 a placeni hodinovou mzdou. Úkolem je co nejlépe práci rozvrhnout tak, aby se náklady na mzdy těchto pracovníků minimalizovaly.

Pohybujeme se v prostředí *flow shop*, neboť nejprve musíme odpovědi získat a až poté je můžeme zpracovávat. Počet úkolů (U) je 20, $p_1 = 10$, $p_2 = 25$. Odpovědi rozdělíme do tří skupin ($n = 3$). Pro řešení použijeme algoritmus 4.1.1:

Dle 4.3 určíme výchozí

$$M = max(10 * 20 + 25; 25 * 20 + 10) = 510$$

Spočítáme příslušná S_j a z nich L_j a dostáváme $L_1 = 1, L_2 = 2, L_3 = 5$, takže $S_3 = 8 \leq U = 20$. Příslušná f_j jsou 0; 0,5; 0,5, a e_j tedy jsou 1; 0,5; 0.5.

$$M := M + p_1\ min\{e_j, j = 1, 2, 3\} = 515.$$

Tentokrát dostáváme $L_1 = 1, L_2 = 3, L_3 = 7$, $S_3 = 11 \leq U = 20$. f_j jsou 0,5; 0; 0.5 a e_j se rovnají 0,5; 1; 0,5.

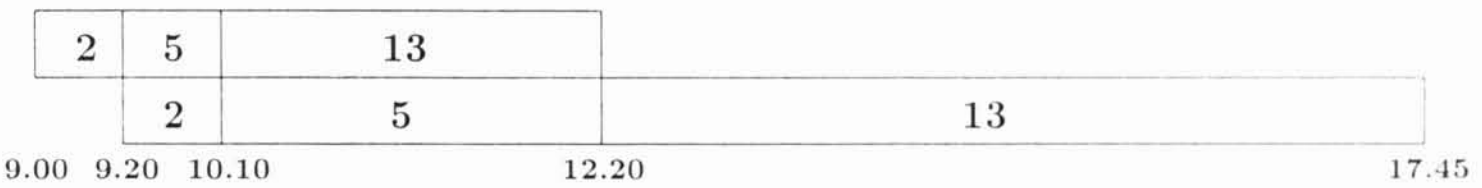
$$M := M + p_1\ min\{e_j, j = 1, 2, 3\} = 520.$$

Pro toto M máme $L_1 = 2, L_2 = 5, L_3 = 12$, $S_3 = 19 \leq U = 20$. Příslušná f_j jsou 0; 0; 0,5, a e_j tedy jsou 1; 1; 0,5.

$$M := M + p_1\ min\{e_j, j = 1, 2, 3\} = 525.$$

L_j mají hodnoty 2, 5 a 13 a $S_3 = 20 = U$.

Optimální rozvrh je tedy 2-5-13 odpovědí (viz obr. 4.5).



Obr. 4.5: Optimální přemísťování dotazníků

Při daných podmínkách je nejlepší odeslat hned první dva dotazníky společně, poté další skupinu po sedmém dotazníku (druhá skupina má pět dotazníků) a na poslední skupinu zbyde 13 dotazníků a je třeba ji odeslat ihned po dokončení posledního hovoru. △

Určení n pro dané M Mohlo by nás nejen z ekonomického hlediska zajímat, jak by situace vypadala, kdybychom povolili 4 přesuny dat. Spočtěme celkovou dobu zpracování pro tuto úlohu:

Příklad 4.1.3 (pokračování příkladu 4.1.2) V tomto případě hned napoprvé dostáváme tyto výsledky: Výchozí M je stejné jako v předchozím příkladu ($M = 510$). Stejně tak zvětšení n neovlivní výpočet prvních tolika skupin, kolik bylo původní n. Takže $L_1 = 1$ ($e_1 = 1$), $L_2 = 2$ ($e_2 = 0,5$), $L_3 = 5$ ($e_3 = 0,5$),

$$S_4 \leq min\left(\frac{510 - 25(20 - 8)}{10}; 20\right).$$

Z této nerovnice získáváme $S_4 = 20$ a $L_4 = 12$. Při čtyřech skupinách jsme dosáhli nejnižší možné celkové doby zpracování ankety 510 minut, což je zlepšení přibližně o 3 %. △

Lemma 4.1.2 *Vždy existuje takový rozvrh, že* $C_{max} = max(p_1U + p_2; p_2U + p_1)$, *pokud není n omezeno.*

Lemma 4.1.2 je zřejmé. Na předchozím příkladu jsme však viděli, že tohoto výsledku lze někdy dosáhnout i s menším počtem skupin než je U. Lze nějak zjistit, kdy to lze? Vraťme se nyní k Trietschovu algoritmu 4.1.1. Pokud takový rozvrh existuje, musí ho tento algoritmus nalézt již při prvním cyklu, protože jinak by se M zvětšilo. Doposud jsme měli pevné n a přizpůsobovali jsme mu M. Nyní to uděláme opačně a k danému M nalezneme n. Tím zjistíme, kolik je třeba skupin, než dosáhneme U.

Příklad 4.1.4 Nechť $p_1 = 8$, $p_2 = 3$, $U = 100$. Úkolem je spočítat, na kolik nejméně skupin je nutné úkoly rozdělit, aby mohly být zpracovány v minimálním čase. Dle (4.3) je dolní hranice pro zpracování úkolů rovna 803. Postupným počítáním dle

Trietschova algoritmu 4.1.1 dostáváme tato L:

$$L_1 = 62, L_2 = 24, L_3 = 9, L_4 = 3, L_5 = 1\ (S_5 = 99), L_6 = 1\ (S_6 = 100).$$

Získali jsme 6 skupin o těchto počtech úkolů: 62, 24, 9, 3, 1, 1. △

V tomto příkladu jsme získali 6 skupin a C_{max} rovno 803. C_{max} již určitě nelze zlepšit. Pokud jde o rozvrh, počet skupin (n) již také pro toto C_{max} nelze zmenšit. Kdyby šlo, Trietschův algoritmus by ho musel odhalit. Kdybychom stanovili libovolné $n < 6$, S_n by vždy bylo menší než U, protože výpočet je stejný pro všechna L_i, $i \leq n$.

Vraťme se nyní k příkladu 4.1.1, který jsme řešili pomocí lemma 4.1.1, a vyřešíme ho pomocí Trietschova algoritmu.

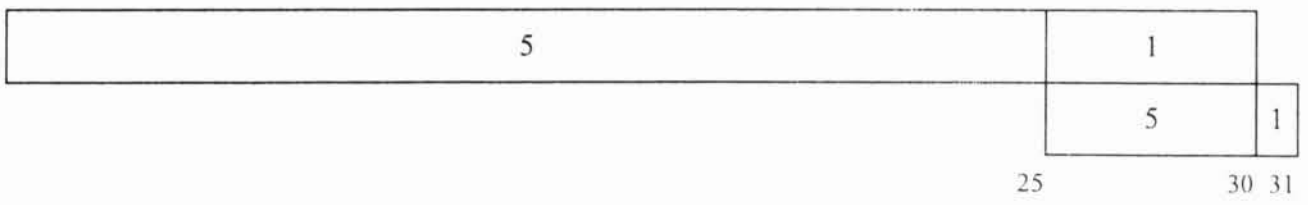
Příklad 4.1.5 Máme stejnou situaci jako v příkladu 4.1.1 ($p_1 = 2, p_2 = 3, U = 15, n = 2, M = 47$).

Pro M dostáváme $L_1 = 1, L_2 = 1$. Poté M musíme zvětšit o jedničku a příslušná L jsou 1 a 2. Dále výpočet postupuje takto: Pro $M = 49 : L_1 = 2, L_2 = 3$, pro $M_3 = 51 : L_1 = 3, L_2 = 4$, pro $M = 52 : L_1 = 3, L_2 = 5$, pro $M = 53 : L_1 = 4, L_2 = 6$, pro $M = 55 : L_1 = 5, L_2 = 7$, pro $M = 56 : L_1 = 5, L_2 = 8$ a konečně pro $M = 57 : L_1 = 6, L_2 = 9$ ($S_2 = 15 = U$). △

Jednoznačnost řešení Na tomto jednoduchém příkladu je vidět, že výpočet je trochu zdlouhavější, ale samozřejmě jsme dospěli ke stejnému C_{max}. Stejně tak optimální rozvrh je stejný (6-9). Počty v jednotlivých skupinách však nemusejí být vždy totožné, rozvrh nalezený Trietschovým algoritmem nemusí být jediný optimální, což ilustruje následující příklad:

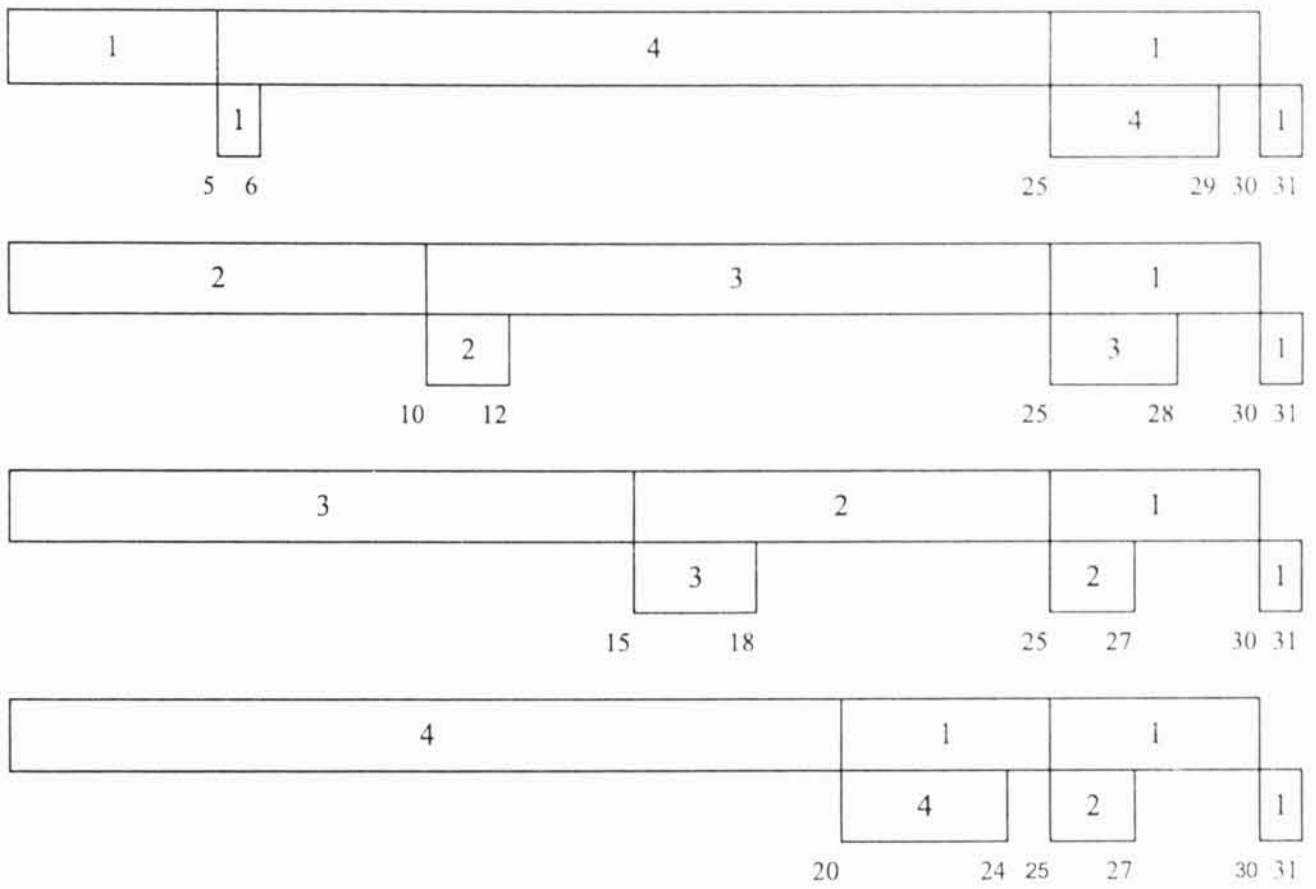
Příklad 4.1.6 $p_1 = 5, p_2 = 1, U = 6, n = 3$

Trietschovým algoritmem získáváme rozvrh $L_1 = 5, L_2 = 1, L_3 = 0$ a C_{max}= 31. Rozvrh je na obrázku 4.6.



Obr. 4.6: Výsledek Trietschova algoritmu

Optimálních rozvrhů však může být víc. Obr. 4.7 ukazuje další rozvrhy, které mají také C_{max}= 31. △



Obr. 4.7: Další optimální rozvrhy

Poznámka: V příkladu 4.1.6 Trietschův algoritmus nalezl optimální rozvrh s menším n, než bylo zadané. Protože stanovuje vždy maximální možné S_j, najde vždy rozvrh s minimálním n.

Volba M Podívejme se, zda bychom nemohli celou proceduru urychlit a M určit přesněji, aniž bychom přeskočili optimum. Jedním ze způsobů, který by proces urychlil, je provést nejprve výpočet pro spojitý případ. Pokud tvoříme jen 2 skupiny, musí pro optimální spojitý rozvrh platit $p_1 L_2 = p_2 L_1$, aby překrytí procesů na prvním a druhém stroji bylo maximální. Z této rovnice jednoduše spočítáme L_1 a L_2. To je stav s ideálními podmínkami, a tedy C_{max} spočtené pro tento rozvrh již nemůže být zlepšeno. Pokud tedy položíme $M = C_{max}$, dostáváme dolní odhad. Pak již můžeme použít Trietschův algoritmus. Celý postup příkladu 5 by vypadal takto:

Příklad 4.1.7 $p_1 = 2, p_2 = 3, U = 15$

$L_1 = \frac{2}{3}(15 - L_1) \Rightarrow L_1 = 6, L_2 = 9, M = 57$. V tomto příkladu jsme přímo získali optimální rozvrh. $\triangle$

4.2 Model s dopravou

Zatím jsme uvažovali pouze takové situace, kdy do hry nevstupovala žádná omezení plynoucí z přepravy. Nanejvýš jsme omezení kladli na počet skupin (pevné n). Počí-

tali jsme s tím, že přepravní doba je 0, kapacita přepravního zařízení je neomezená, přepravní zařízení je neustále k dispozici. Pokud jsme v takovém případě našli optimální rozvrh vzhledem k C_{max}, může nám sloužit jako dolní odhad. Pokud přidáme nějaké omezení, pak se C_{max} může jedině zvýšit. Je dobré rozlišovat dvě možnosti, které mohou být ovlivněny změnou podmínek: může se změnit C_{max} či optimální rozvrh (ev. obojí).

Přepravní čas od jednoho stroje k následujícímu se skládá z doby nakládky, přepravy a vykládky a budeme ho souhrnně označovat TT (*transfer time*).

Tento čas tvoří nutný interval, který musí být mezi ukončením zpracovávání dané skupiny úkolů na jednom stroji a okamžikem, kdy mohou být zpracovávány na druhém stroji. Kromě toho si lze představit situaci, kdy se dané zařízení musí vrátit zpět k původnímu stroji, aby se na něj mohly naložit další úkoly. Jestliže celý cyklus označíme CT (*cycle time*), na návrat zbývá čas RT (*return time*). $RT = CT - TT$.

nakládka	převoz	vykládka	návrat

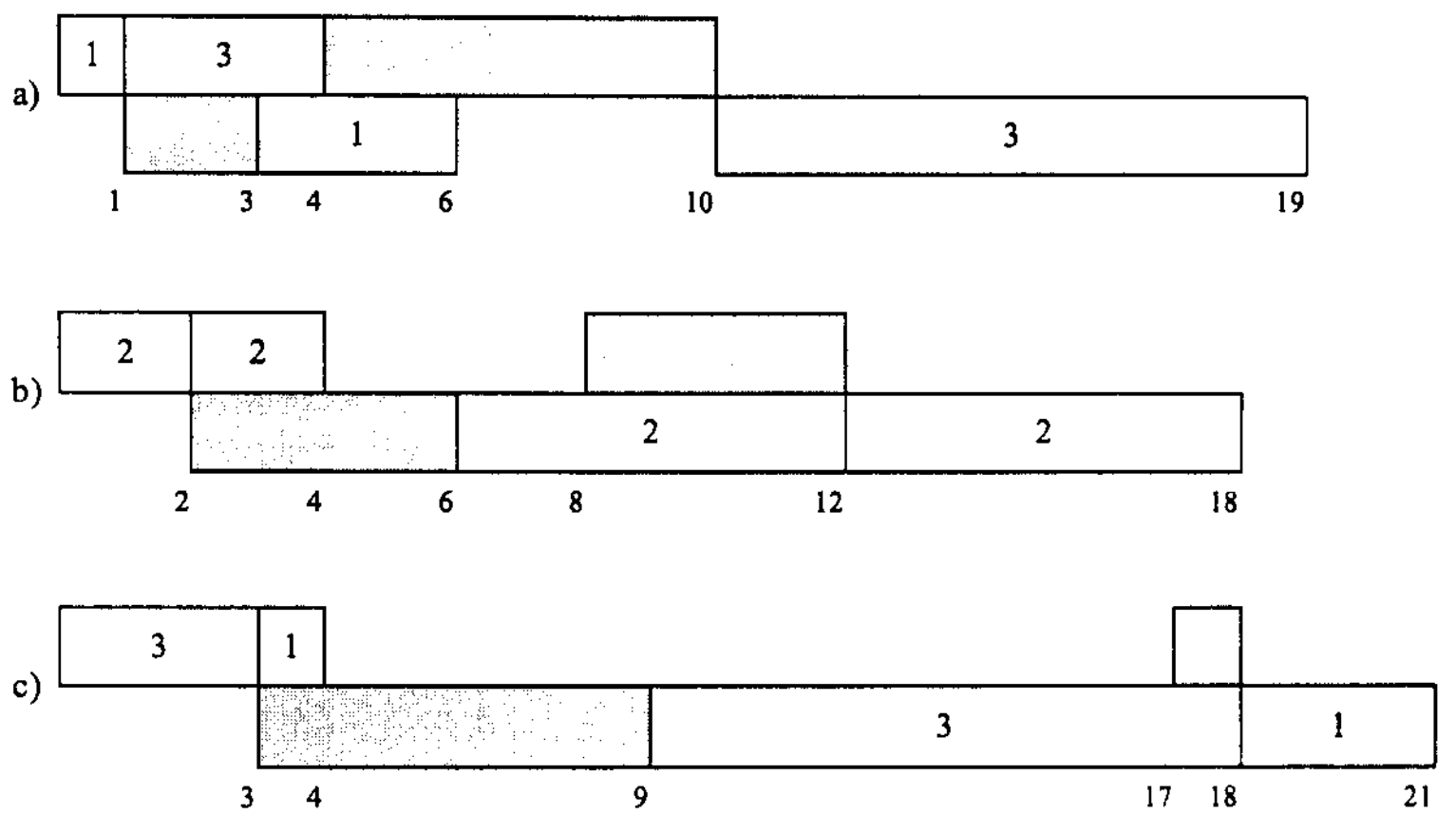
Obr. 4.8: CT - cycle time

Vliv CT/TT Základní otázkou je, zda a jak CT, resp. TT ovlivňuje výsledek kritéria optimality. Kromě toho může mít CT, resp. TT vliv na optimální rozvrh ve smyslu, že nalezený optimální rozvrh bez omezení na přepravu se může změnit, pokud dodáme do modelu CT či TT. Obecně dopravní omezení vliv na C_{max} i rozvrh má. Názorně je to vidět na následujícím příkladu.

Příklad 4.2.1 Mějme situaci, kdy jsou třeba zpracovat 4 úkoly na dvou strojích. Na prvním stroji se každý úkol zpracovává 1 minutu, na druhém 3 minuty. Úkoly lze rozdělit na dvě skupiny a každou k druhému stroji přepravit samostatně. K dispozici je přepravní zařízení, u něhož doba přepravy skupiny úkolů v minutách odpovídá dvojnásobku počtu úkolů ve skupině. Takovéto zařízení je k dispozici jedno a doba návratu od druhého stroje k prvnímu je 1 minuta. Jaká je minimální doba, kterou je potřeba ke zpracování těchto úkolů, a jak je nejlépe úkoly přepravovat? Jak se situace změní, pokud $CT = 0$?

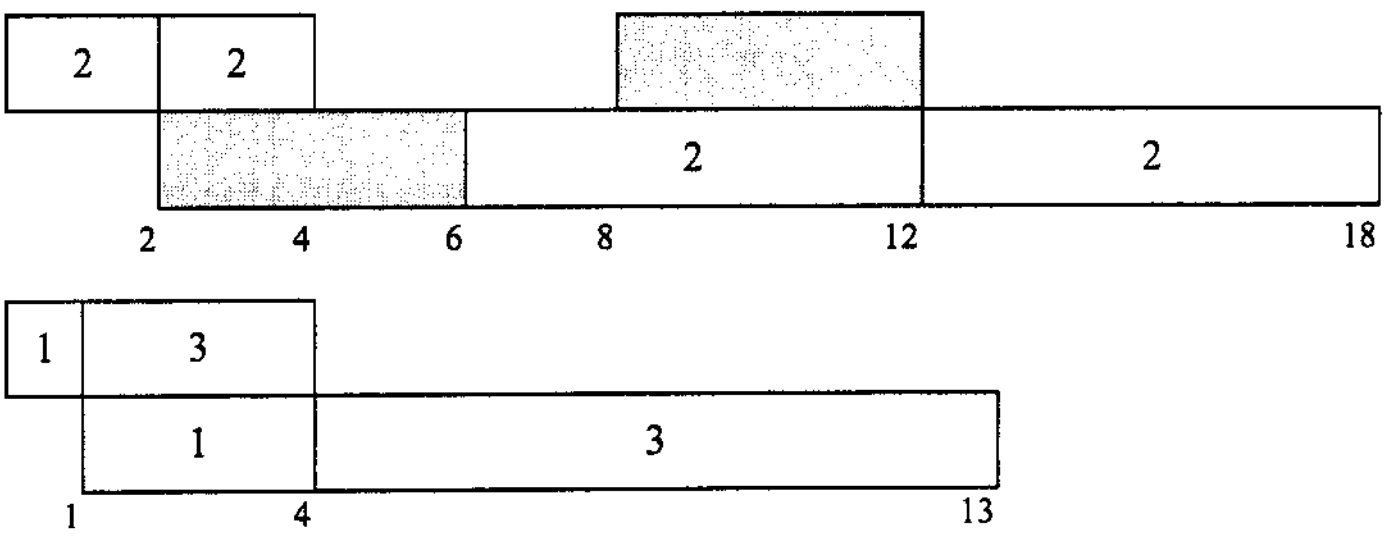
V symbolech zadání je: $p_1 = 1$ min, $p_2 = 3$ min, $U = 4, n \leq 2. TT = 2L_{ij}$ min, $CT = (TT + 1)$ min, s přepravou L_2 ke druhému stroji lze začít nejdříve v $(L_1 p_1 + 2L_1 + 1)$ min. Úkolem je nalézt optimální rozvrh a příslušné C_{max} a výsledek porovnat s případem $CT = 0$.

ŘEŠENÍ: Tři možné rozvrhy jsou zobrazeny na obr. 4.9. Každý z nich splňuje podmínku, že přepravní zařízení musí mít 1 min na vrácení. Jak je vidět, rozvrh b) (2-2) je optimální a minimální $C_{max} = 18$ min.



Obr. 4.9: Možné rozvrhy

Ovšem v případě $CT = 0$ je optimální rozvrh 1-3 a $C_{max} = 13$ min (pro výpočet lze použít lemma 4.1.1 či Trietschův algoritmus 4.1.1). Srovnání obou rozvrhů je na na obr. 4.10. △



Obr. 4.10: Srovnání rozvrhů

Příklad 4.2.1 ukazuje, že vlivem přepravních omezení se změnilo jak minimální C_{max}, tak i optimální rozvrh. Obecně tedy může dojít ke změně obojího. Pokud jde o způsob hledání řešení, nelze obecně situaci řešit tak, že by se určil optimální rozvrh bez přepravních omezení a do tohoto rozvrhu by se jen zakomponovaly příslušné požadavky na dopravu. Ve speciálních případech tomu tak může být. Nenulové CT

nijak neovlivní optimální rozvrh ani C_{max} případu bez přepravních omezení např. tehdy, pokud je doba zpracovávání každé skupiny na prvním stroji větší či rovna CT. To jsou však speciální případy, které jsou spíše výjimkou.

Podívejme se nyní na některé případy situací s přepravou a zda by bylo možné nějakým způsobem využít výsledky modelů bez přepravy.

Konstantní doba V této části budeme uvažovat pouze takové případy, kdy TT ani CT nezávisí na velikosti přepravované skupiny, a protože máme jen dva stroje, tak ani na dvojici strojů, mezi kterými přeprava probíhá. TT i CT tedy budou konstanty, a to stejné pro všechny $L_j, 1 \leq j \leq n$.

V případě, v němž se do modelu nezahrnoval vliv přepravy, lze užít Trietschův algoritmus (str. 16). Ten je založen na principu stanovení dolní meze a jejího zvětšování po takových nejmenších krocích, které znamenají zvětšení aspoň jedné skupiny o jeden úkol. Dá se očekávat, že algoritmus půjde využít i v modelu s konstantními dobami TT i CT. Je toto očekávání správné?

Dolní mez (4.3) lze upravit na

$$M = max(p_1U + p_2 + TT; p_2U + p_1 + TT) \tag{4.7}$$

analogicky dle nerovnice (4.4) musí platit

$$p_1S_j + TT \leq M - p_2(U - S_{j-1}). \tag{4.8}$$

Dle (4.5)

$$S_j \leq min\left(\frac{M - p_2(U - S_{j-1}) - TT}{p_1}; U\right). \tag{4.9}$$

f_j je analogicky dle (4.6)

$$f_j = min\left(\frac{M - p_2(U - S_{j-1}) - TT}{p_1}; U\right) - S_j \tag{4.10}$$

Modifikovaný algoritmus 4.1.1 pak vypadá následovně:

Algoritmus 4.2.1

a) $S_0 := 0$

b) *For* $j := 1$ *to* n *do*

$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}) - TT)/p_1, U\} \rfloor$;

$f_j = \min\{(M - p_2(U - S_{j-1}) - TT)/p_1, U\} - S_j$;

$e_j = 1 - f_j$

c) *Je-li* $S_n < U$, *pak*

$M := M + p_1 \min\{e_j\}$ *a jdi do a);*

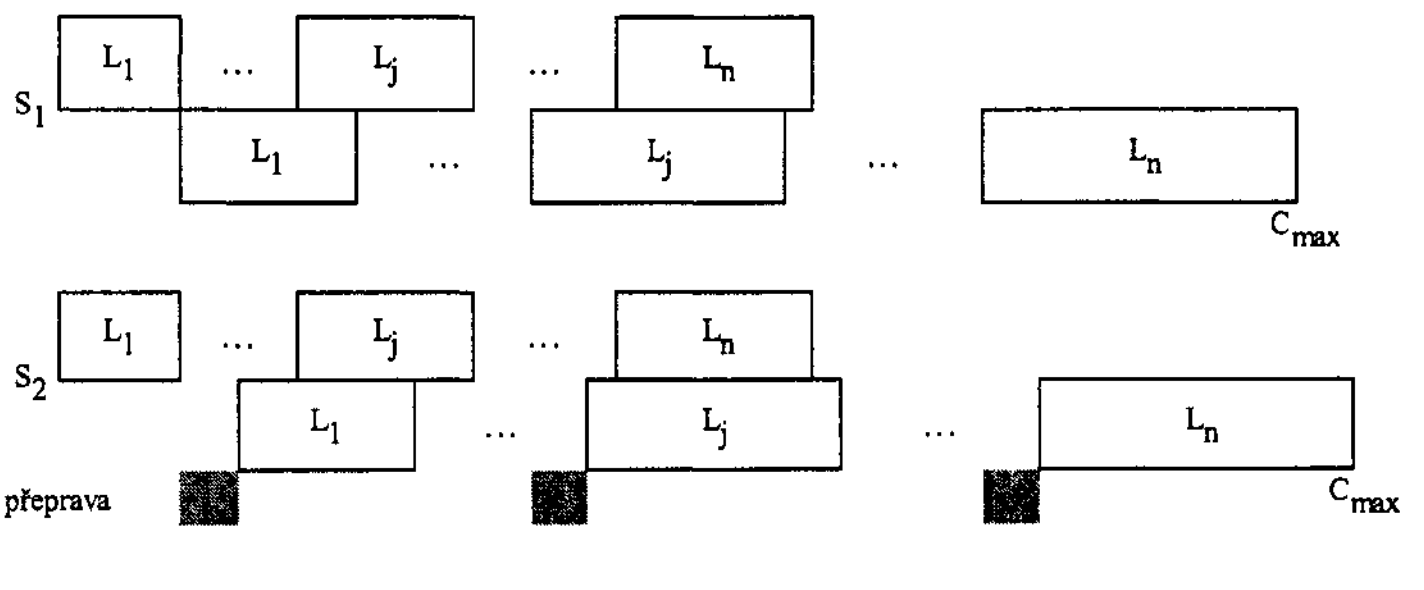
Pokud $S_n = U$, *STOP.*

Vrátíme se k příkladu 4.1.2 a doplníme ho o dobu potřebnou na přepravu dotazníků.

Příklad 4.2.2 (pokračování příkladu 4.1.2) Zadání je stejné jako v příkladu 4.1.2: $U = 20$, $p_1 = 10$, $p_2 = 25$, $n = 3$. Navíc $TT = 15$.
ŘEŠENÍ Aplikací algoritmu 4.2.1 získáváme optimální rozvrh $L_1 = 2$, $L_2 = 5$, $L_3 = 13$ a $C_{max} = 540$. $\triangle$

Je zajímavé výsledky porovnat s výsledky zadání bez přepravy. Tehdy byl optimální rozvrh 2-5-13, takže se shodoval s tím, který jsme získali nyní. C_{max} bylo 525 a to je doba přesně o 15 minut kratší než doba z modelu s přepravou, neboť 15 minut byla právě doba na přepravu. V tomto případě jsme tedy dostali stejný optimální rozvrh a doba zpracování se lišila právě o dobu na přepravu.

Následující lemma ukazuje, že tomu tak je ve všech takovýchto modelech, shrnuje vliv přepravního času TT (obr. 4.11):



Obr. 4.11: Změna při přidání TT

Lemma 4.2.1 *Nechť Q_1 a Q_2 jsou dva modely, které se liší jedině tím, že $0 = TT_1 < TT_2$, přičemž TT jsou konstanty a k dispozici je dostatečný počet přepravních zařízení na to, aby skupiny nemusely čekat kvůli dopravě.*

Pak pro optimální C_{max} platí $C^1_{max} < C^2_{max}$ a existují takové optimální rozvrhy S_1 a S_2, že $S_1 = S_2$. Konkrétně $C^2_{max} = C^1_{max} + TT_2$.

Důkaz K ověření tohoto lemma použijeme úvahu, na které je založen Trietschův algoritmus. Nejprve upravíme výraz (4.3) pro počáteční dolní odhad M. Dolní odhad pro Q_2 označíme M' a M bude představovat dolní odhad pro Q_1. M se v tomto případě zvětší o TT_2, jak je vidět z následující rovnice:

$$M' = max(p_1U + p_2 + TT_2; p_2U + p_1 + TT_2) = M + TT_2.$$

Pro všechny skupiny musí platit modifikované podmínky přípustnosti (4.4)

$$p_1S_j + TT_2 \leq M' - p_2(U - S_{j-1}),$$

takže

$$p_1S_j + TT_2 \leq M + TT_2 - p_2(U - S_{j-1}),$$

což znamená, že dostáváme původní podmínky (4.4). Pro Q_1 a Q_2 jsou podmínky přípustnosti ekvivalentní. Stejným postupem získáme v obou případech stejný rozvrh.

$M' = M + TT_2$ a přitom minimální zvětšení M', při kterém dojde ke změně rozvrhu, a tím i změně C_{max}, je $\min\{e'_j, j = 1, ..., n\}$, kde $e'_j = 1 - f'_j$ a

$$f'_j = \min\{(M' - p_2(U - S_{j-1}) - TT_2)/p_1, U\} - S_j =$$
$$= \min\{(M + TT_2 - p_2(U - S_{j-1}) - TT_2)/p_1, U\} - S_j = f_j.$$

M i M' se tedy zvětšují po stejných krocích. Vzhledem k tomu, že všechny podmínky a kroky Trietschova algoritmu pro Q_1 a Q_2 jsou ekvivalentní, musí se optimální řešení nalézt po stejném počtu iterací. Při každém kroku je $M' = M + TT_2$, takže C_{max} se zvětší pouze o TT. $\Diamond$

Z lemma 4.2.1 plyne, že přidáním TT nevzniká nová úloha. Stačí vyřešit původní úlohu bez TT pomocí Trietschova algoritmu 4.1.1 a její optimální rozvrh je optimální i pro úlohu rozšířenou o TT. Pokud však do modelu zahrneme i $CT > TT$, může se změnit nejen C_{max}, ale i optimální rozvrh. Protože je přidáno více omezení a množina řešení se tak zužuje, může se C_{max} jedině zhoršit. Nemá význam uvažovat modely, kde sice bude čas na návrat přepravního zařízení nenulový, ale kde těchto zařízení bude takový počet, že vždy bude nějaké k dispozici u stroje 1, protože v takovém případě se vliv $CT > TT$ neprojeví.

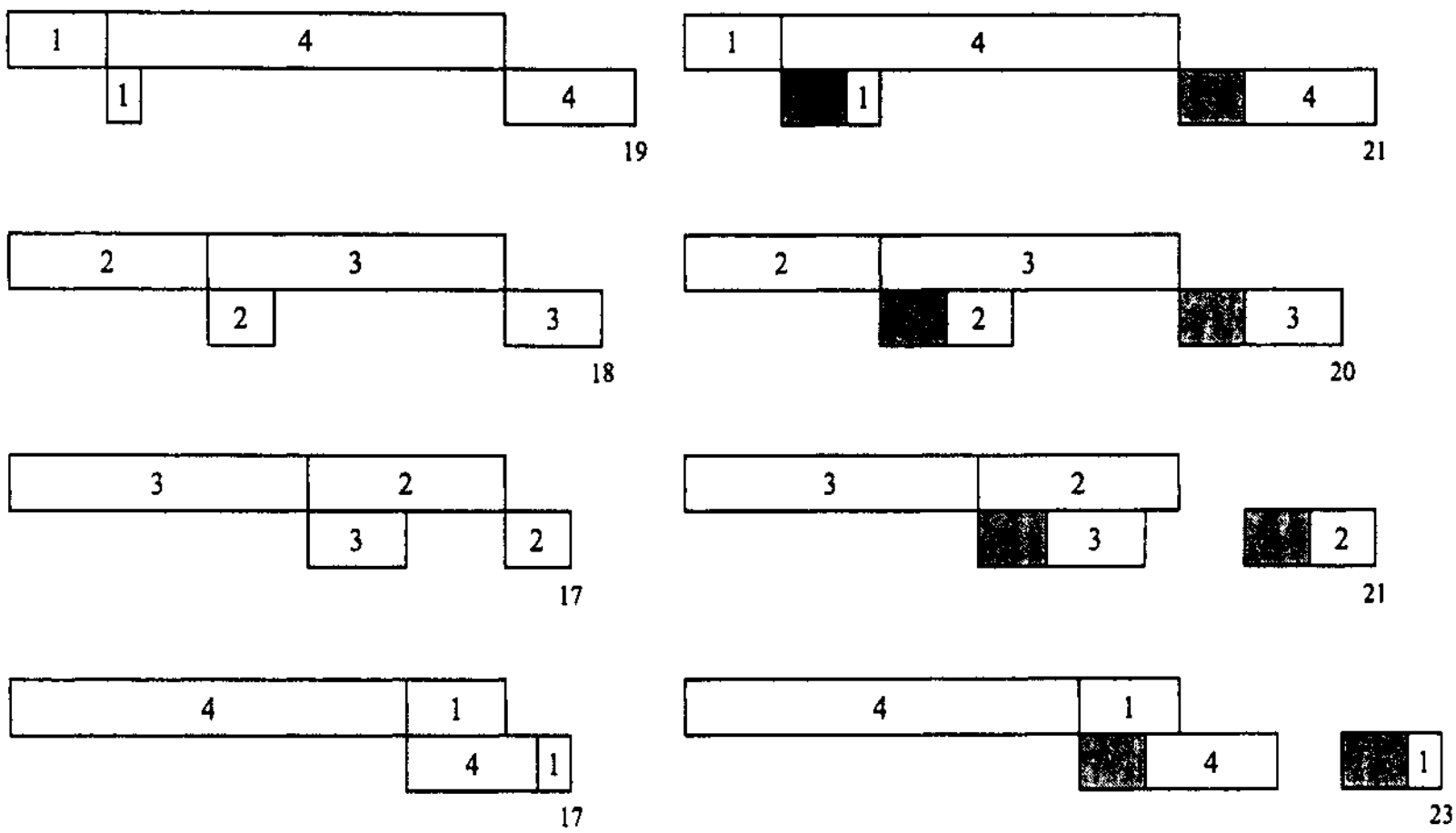
Příklad 4.2.3 $p_1 = 3, p_2 = 1, U = 5, n = 2, TT = 2, CT = 8$, jedno přepravní zařízení. Podívejme se, jak se na jednotlivých rozvrzích projevuje CT. V prvním sloupci obr. 4.12 jsou rozvrhy bez přepravy, v druhém s přepravou.

V případě rozvrhů bez přepravy jsou optimálními rozvrhy možnosti 3-2 a 4-1 s $C_{max} = 17$. Dle lemma 4.2.1 musí pro případ, kdy je $CT = TT$, platit $C^2_{max} = C^1_{max} + TT = 19$. V našem zadání je však $TT < CT = 8$. Pro tento případ je optimálním rozvrh 2-3 a $C_{max} = 20$. Δ

Příklad 4.2.3 je příkladem toho, že doba návratu má vliv jak na minimální C_{max}, tak na optimální rozvrh.

Omezený počet přepravních zařízení Omezený počet přepravních zařízení může být určujícím faktorem, kdy a po jakých skupinách budou úkoly přepravovány. V takových případech tedy nemusí být požadavek na n žádoucí. Pro tyto modely uvádějí algoritmus D.Trietsch, K.R.Baker ([1]). Výpočet je založen na již citovaném algoritmu (algoritmus 4.1.1), který nepočítal s přepravními omezeními. Tentokrát je upraven tak, aby popisoval situaci s přepravou. t_j je doba, kdy je L_j k dispozici na druhém stroji. Opět stanovíme M jako dolní odhad C_{max}, k znamená počet přepravních zařízení.

Pro S_j, f_j, e_j platí stejné vztahy jako v modelu s TT a $CT = TT$ (viz 4.9, 4.10). Nová je zde veličina t_j, která sleduje, zda je rozvrh uskutečnitelný vzhledem



Obr. 4.12: Vliv CT

k přepravním omezením. Pomocí ní se při každé iteraci otestuje, zda lze ještě vůbec do M stihnout zpracovat všechny zbylé úkoly vzhledem k možnostem přepravních zařízení. t_j jako doba, kdy je skupina L_j k dispozici na druhém stroji, je buď přímo doba zpracování prvních j skupin na prvním stroji plus TT, pokud je v okamžiku dokončení j-té skupiny na prvním stroji k dispozici přepravní zařízení, nebo okamžik, kdy se vrátí zařízení, které je nejdéle pryč, plus TT.

Algoritmus 4.2.2

a) $S_0 := 0, j := 0$

b) *While not* $S_j = U$

j:=j+1;

$S_j = \lfloor \min\{(M - TT - p_2(U - S_{j-1}))/p_1, U\} \rfloor$;

$f_j = \min\{(M - TT - p_2(U - S_{j-1}))/p_1, U\} - S_j$;

$e_j = 1 - f_j$;

$$t_j = \begin{cases} p_1 S_j + TT, j = 1, 2, ..., k \\ \max\{t_{j-k} + CT, p_1 S_j + TT\} \textit{ jinak.} \end{cases}$$

Je-li $j > k$, *pak*

pro $t_j > M - p_2(U - S_{j-1}) \rightarrow M := M + p_1 \min\{e_j\}$ *a jdi do a);*

pro $t_j \leq M - p_2(U - S_{j-1})$ *a pro* $S_j < U \rightarrow j := j + 1$ *a jdi do b);*

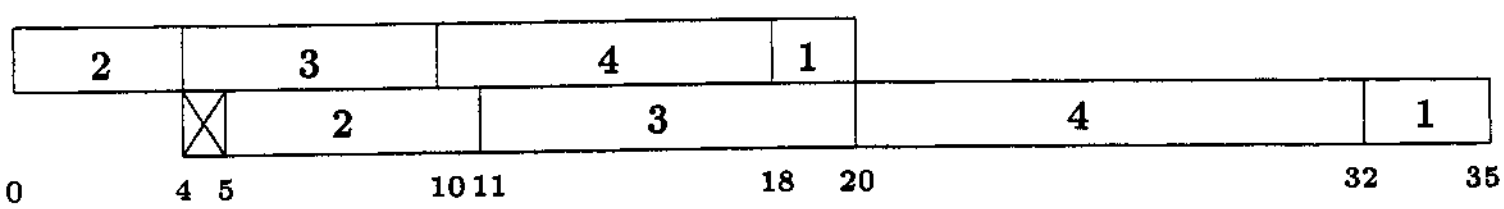
pro $t_j \leq M - p_2(U - S_{j-1})$ *a pro* $S_j = U \rightarrow STOP$.

Tento algoritmus nyní vyzkoušíme na následujícím příkladu:

Příklad 4.2.4 Mějme tyto vstupní hodnoty:
$p_1 = 2; p_2 = 3; U = 10; TT = 1; CT = 7, k = 2.$
Nejdříve spočítáme dolní odhad $M = 3 * 10 + 2 + 1 = 33$.
Nyní již aplikujeme vlastní algoritmus:

a) $S_0 = 0, j = 0,$
b) $S_1 = \lfloor \min\{(33 - 1 - 30 + 0)/2, 10\} \rfloor = 1 \Rightarrow L_1 = 1$
$f_1 = 0, e_1 = 1, t_1 = 2 * 1 + 1 = 3$
$S_2 = \lfloor \min\{(33 - 1 - 30 + 3)/2, 10\} \rfloor = 2 \Rightarrow L_2 = 1$
$f_2 = 0,5, e_2 = 0,5, t_2 = 2 * 2 + 1 = 5$
$S_3 = \lfloor \min\{(33 - 1 - 30 + 6)/2, 10\} \rfloor = 4 \Rightarrow L_3 = 2$
$f_3 = 0, e_3 = 1, t_3 = \max\{3 + 7, 2 * 4 + 1\} = 10$
c) kontrola: platí, že $10 > 33 - 3(10 - 2) \Leftrightarrow 10 > 9 \rightarrow M := 33 + 2 \min\{1; 0,5; 1\} = 34$
a) $S_0 = 0, j = 0,$
b) $S_1 = 1 \Rightarrow L_1 = 1$
$f_1 = 0,5, e_1 = 0,5, t_1 = 2 * 1 + 1 = 3$
$S_2 = 3 \Rightarrow L_2 = 2$
$f_2 = 0, e_2 = 1, t_2 = 7$
$S_3 = 6 \Rightarrow L_3 = 3$
$f_3 = 0, e_3 = 1, t_3 = \max\{7 + 7, 2 * 6 + 1\} = 14$
c) kontrola: platí, že $14 > 34 - 3(10 - 3) \Leftrightarrow 14 > 13 \rightarrow M := 34 + 2 \min\{0,5; 1; 1\} = 35$
a) $S_0 = 0, j = 0,$
b) $S_1 = 2 \Rightarrow L_1 = 2$
$f_1 = 0, e_1 = 1, t_1 = 5$
$S_2 = 5 \Rightarrow L_2 = 3$
$f_2 = 0, e_2 = 1, t_2 = 11$
$S_3 = 9 \Rightarrow L_3 = 4$
$f_3 = 0,5, e_3 = 0,5, t_3 = \max\{5 + 7, 2 * 9 + 1\} = 19$
c) kontrola: platí, že $19 \leq 35 - 3(10 - 5) \Leftrightarrow 19 \leq 20 \wedge 9 < 10 \rightarrow j := 4$
b) $S_4 = 10 \Rightarrow L_4 = 1$
$f_4 = 0,5, e_4 = 0,5, t_4 = \max\{11 + 7, 20 + 1\} = 21$
c) kontrola: platí, že $21 \leq 35 - 3(10 - 9) \Leftrightarrow 21 \leq 32 \wedge S_4 = 10 \rightarrow STOP.$

Výsledný optimální rozvrh je tedy $2 - 3 - 4 - 1$ (viz obr. 4.13). $\triangle$



Obr. 4.13: Optimální rozvrh příkladu 4.2.4

Omezený počet přepravních zařízení a dané n Posledním algoritmem jsme řešili případ s omezeným počtem přepravních zařízení s neomezenou kapacitou bez požadavku na počet skupin, na které se úkoly rozdělí. V posledním příkladu bylo 10 úkolů rozděleno na 4 části. Kdyby bylo CT např. 2, měli bychom 5 částí (1-1-2-3-3). Nyní se podívejme, jak by šlo upravit Trietschův algoritmus na situaci, kdy máme omezený počet přepravních zařízení, ale i omezený počet skupin, na které se mají úkoly rozdělit. Opodstatněním takového modelu by byla situace, kdy jsou přepravní náklady vysoké a vyplatí se vlastní přepravování snížit i za cenu zvýšení celkové doby zpracovávání zakázky.

Algoritmus 4.2.3

a) $S_0 := 0$;

b) *For* $j = 1$ *to* n *do*

$S_j = \lfloor \min\{(M - TT - p_2U + p_2S_{j-1})/p_1, U\} \rfloor$;

$f_j = \min\{(M - TT - p_2U + p_2S_{j-1})/p_1, U\} - S_j$;

$e_j = 1 - f_j$;

$$t_j = \begin{cases} p_1S_j + TT, j = 1, 2, ..., k \\ \max\{t_{j-k} + CT, p_1S_j + TT\} \textit{ jinak.} \end{cases}$$

c) Je-li $j > k$, *pak*

pro $t_j > M - p_2(U - S_{j-1})$ *nebo pro* $S_j < U \rightarrow M := M + p_1 \min\{e_j\}$ *a jdi do a);*

pro $t_j \leq M - p_2(U - S_{j-1})$ *a pro* $S_j = U \rightarrow STOP$.

Příklad 4.2.5 Předchozí příklad 4.2.4 upravíme tak, že všechny úlohy můžeme rozdělit pouze na 3 skupiny. Zadání tedy zní:

$p_1 = 2, p_2 = 3, U = 10, n = 3, TT = 1, CT = 7, k = 2.$

Pro $M = 33$ ani $M = 34$ není k dispozici přepravní zařízení na dopravu 3. skupiny ke stroji 2, pro $M = 35$ je $S_3 = 9 < U$, a tak $M := 35 + p_1 \min\{e_j\} = 36$.

a) $S_0 = 0$

b) $S_1 = \lfloor \min\{(36 - 1 - 30 + 0)/2, 10\} \rfloor = 2 \Rightarrow L_1 = 2$

$f_1 = 0,5, e_1 = 0,5, t_1 = 2 * 2 + 1 = 5$

$S_2 = \lfloor \min\{(36 - 1 - 30 + 3 * 2)/2, 10\} \rfloor = 5 \Rightarrow L_2 = 3$

$f_2 = 0,5, e_2 = 0,5, t_2 = 2 * 5 + 1 = 11$

$S_3 = \lfloor \min\{(36 - 1 - 30 + 3 * 5)/2, 10\} \rfloor = 10 \Rightarrow L_3 = 5$

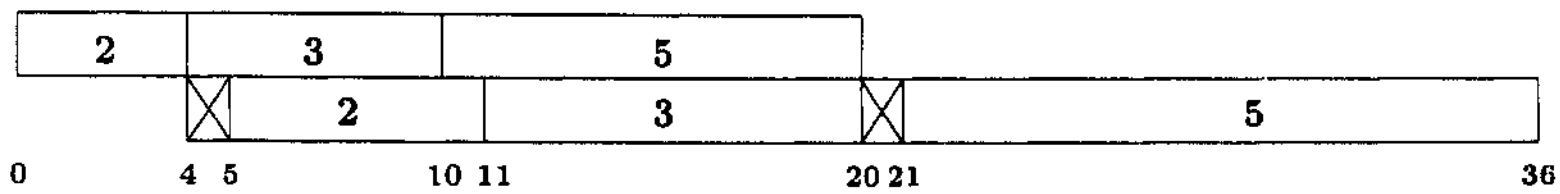
$f_3 = 0,5, e_3 = 0,5, t_3 = \max\{5 + 7, 2 * 10 + 1\} = 21$

c) kontrola: platí, že $21 \leq 36 - 3(10 - 5) \Leftrightarrow 21 \leq 21 \wedge S_3 = U \rightarrow$ STOP.

Výsledný optimální rozvrh je $2 - 3 - 5$, C_{max} se zvýšilo o jednu jednotku na 36 (viz obr.4.14).

△

Opět vidíme, že přidáním dalšího omezení se C_{max} může jedině zhoršit, takže lze za počáteční dolní odhad vzít C_{max} ze situace s méně omezeními, v našem případě pouze s omezeným počtem přepravních zařízení.



Obr. 4.14: Optimální rozvrh příkladu 4.2.5

Lineární doba Po případech, kdy TT ani CT nezávisí na velikosti přepravované skupiny (TT i CT jsou konstanty), se podívejme na případ, kdy jsou tyto veličiny závislé na velikosti přepravované skupiny. Jedná se hlavně o TT, které zahrnuje nakládku, přepravu a vykládku. Zvláště nakládka a vykládka mohou lineárně záviset na počtu nakládaných a vykládaných úkolů. Nebudeme rozlišovat mezi jednotlivými fázemi TT, a budeme tak celé TT považovat za lineárně závislé. Takže $TT_j = wL_j, 1 \leq j \leq n$, kde w je daná konstanta. Pokud není řečeno jinak, $CT = TT$.

Trietschův algoritmus 4.1.1 by šlo upravit následujícím způsobem:

Dolní mez (4.3) vypadá

$$M = max(p_1U + p_2 + w; p_2U + p_1 + w), \tag{4.11}$$

základní podmínka (4.4) pak

$$p_1S_j + w(S_j - S_{j-1}) \leq M - p_2(U - S_{j-1}). \tag{4.12}$$

S_j, f_j se upraví dle (4.5), resp. (4.6). Při zvětšování M je třeba do úvahy zahrnout i čas na přemísťování, protože je závislý na velikosti přepravované skupiny. Minimální e_j se tedy vynásobí součtem $p_1 + w$, čímž dojde k dostatečnému zvětšení M tak, aby se skupina s největší šancí, že by mohla být o jedničku větší, o jeden úkol zvětšila. Algoritmus 4.1.1 tak vypadá následujícím způsobem:

Algoritmus 4.2.4

a) $S_0 := 0$

b) *For* $j := 1$ *to* n *do*

$S_j = \lfloor \min\{(M - p_2U + S_{j-1}(w + p_2))/(p_1 + w), U\} \rfloor$;

$f_j = \min\{(M - p_2U + S_{j-1}(w + p_2))/(p_1 + w), U\} - S_j$;

$e_j = 1 - f_j$

c) *Je-li* $S_n < U$, *pak*

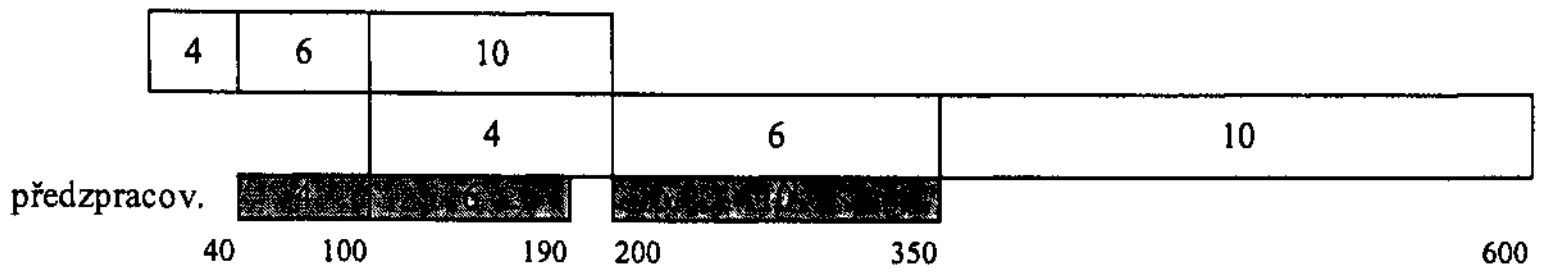
$M := M + (p_1 + w) \min\{e_j\}$ *a jdi do a);*

Pokud $S_n = U$, *STOP.*

V příkladu 4.1.2 šlo o zpracovávání dotazníků. Předpokládejme nyní navíc, že jejich předzpracování předtím, než budou zpracovávány na druhém stanovišti, trvá u každého dotazníku 15 minut.

Příklad 4.2.6 (pokračování příkladu 4.1.2) Doplněné zadání příkladu 4.1.2 tedy je: $U = 20$, $p_1 = 10$, $p_2 = 25$, $n = 3$. Navíc $w = 15$.

ŘEŠENÍ Aplikací algoritmu 4.2.4 získáváme optimální rozvrh $L_1 = 4$, $L_2 = 6$, $L_3 = 10$ a $C_{max} = 600$. Viz obr. 4.15. $\triangle$



Obr. 4.15: Optimální rozvrh příkladu 4.2.6

V případě bez přepravy (příklad 4.1.2) byl optimální rozvrh 2-5-13 a C_{max} bylo 525. Jestliže bylo zadání rozšířeno o přepravu (příklad 4.2.2, na každou skupinu úkolů bylo třeba 15 min), optimální rozvrh zůstal stejný (2-5-13) a C_{max} se zvýšilo na 540. V případě předzpracování (v tomto případě má stejné vlastnosti a požadavky jako lineárně závislá přeprava) každého dotazníku v délce 15 minut je optimální rozvrh 4-6-10 a $C_{max} = 600$ (příklad 4.2.6).

Omezený počet přepravních zařízení Pro případ, kdy je k dispozici jen určitý počet přepravních zařízení, se dá s výhodou využít algoritmus 4.2.3. V tomto algoritmu jen stačí upravit položky, ve kterých se projeví lineární závislost TT, a může být využit i v modelu s TT lineárně závislým na počtu úkolů ve skupinách. Výsledný algoritmus by tedy vypadal následujícím způsobem:

Algoritmus 4.2.5

a) $S_0 := 0, j := 0$

b) *While not* $S_j = U$

$j{:=}j{+}1;$

$S_j = \lfloor \min\{(M - p_2U + S_{j-1}(w + p_2))/(p_1 + w), U\} \rfloor;$

$f_j = \min\{(M - p_2U + S_{j-1}(w + p_2))/(p_1 + w), U\} - S_j;$

$e_j = 1 - f_j;$

$$t_j = \begin{cases} p_1S_j + w(S_j - S_{j-1}), j = 1, 2, ..., k \\ \max\{t_{j-k} + RT + w(S_j - S_{j-1}), p_1S_j + w(S_j - S_{j-1})\}, \textit{ jinak.} \end{cases}$$

Je-li $j > k$, *pak*

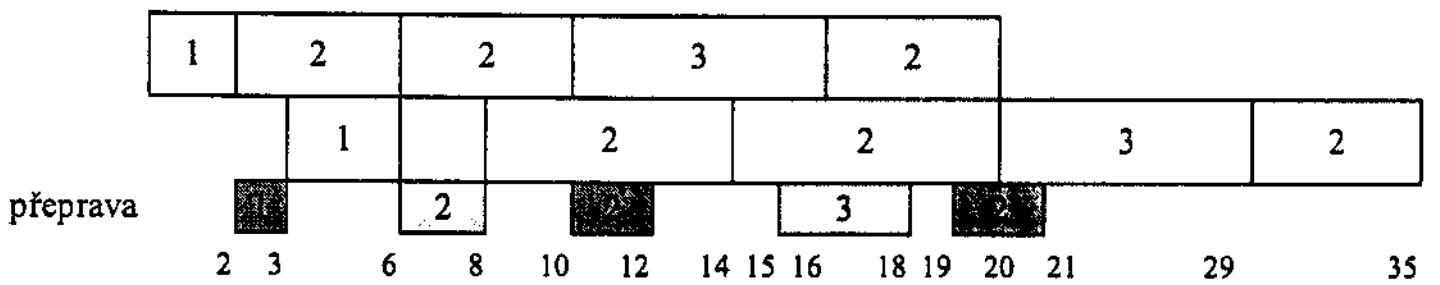
pro $t_j > M - p_2(U - S_{j-1}) \rightarrow M := M + (p_1 + w)\min\{e_j\}$ *a jdi do a);*

pro $t_j \leq M - p_2(U - S_{j-1})$ *a pro* $S_j < U \rightarrow j := j + 1$ *a jdi do b);*

pro $t_j \leq M - p_2(U - S_{j-1})$ *a pro* $S_j = U \rightarrow STOP.$

Příklad 4.2.4 upravíme o proměnlivé TT a aplikujeme tento algoritmus:

Příklad 4.2.7 Vstupní hodnoty:
$p_1 = 2; p_2 = 3; U = 10; w = 1; RT = 7, k = 2.$
ŘEŠENÍ: Výsledný optimální rozvrh je $1-2-2-3-2$ a C_{max} je 35 (viz obr. 4.16). △



Obr. 4.16: Optimální rozvrh příkladu 4.2.7

Omezený počet přepravních zařízení a dané n Algoritmus 4.2.3 se upraví pro případ s proměnlivým TT analogicky, proto zde modifikaci s pevně daným k i n nebudeme uvádět.

Omezená kapacita Omezení plynoucí z přepravních zařízení může být i takové, že na jedno zařízení lze naložit pouze dané množství úkolů. Lze si v praxi představit přemísťování výrobků v kontejnerech, které jsou omezeny objemem či nosností, omezení dané nosností jeřábů, nákladních automobilů apod.

Jak uvádí D. Trietsch ([1]), stačí v tomto případě upravit podmínku na S_j, a sice $S_j = \lfloor \min\{(M - TT - p_2U + p_2S_{j-1})/p_1, S_{j-1} + UB, U\} \rfloor$. Nově se v ní objevuje výraz $S_{j-1} + UB$, kde označení UB znamená horní mez (*upper bound*). j-tá skupina L_j tedy nemůže být větší než UB.

Nyní bychom tuto podmínku mohli dodat do všech algoritmů odvozených od Trietschova algoritmu a získat tak nové modifikace. Někde by se mohlo stát, že by řešení za daných podmínek vůbec neexistovalo. Konkrétně by se to stalo tehdy, pokud by povolený počet skupin vynásobený horní hranicí byl menší než celkový počet úkolů, tj. pro $nUB < U$. Kdyby $nUB = U$, pak bychom zákonitě dostali rozvrh se *stejnými* skupinami.

Velikost UB může mít vliv jak na C_{max}, tak na optimální rozvrh. Někdy se C_{max} nemusí změnit, ale změní se optimální rozvrh, jindy se může změnit obojí. Jak uvádí D. Trietsch ([1]), když

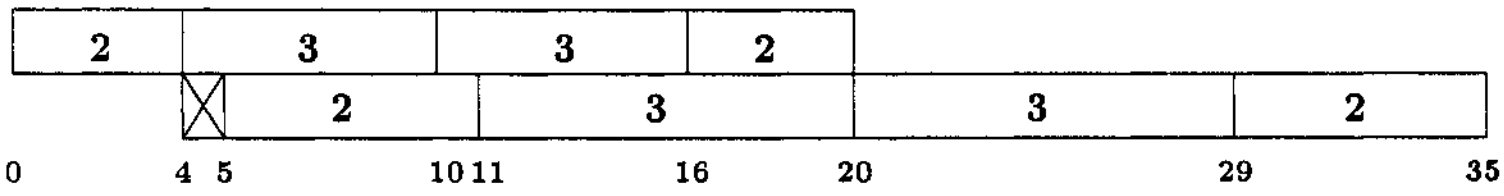
$$UB < CT/max\{p_1(1 + (1/q) + ... + (1/q)^{k-1}), p_2(1 + q + ... + q^{k-1})\},$$

pak doprava omezuje kapacitu systému.

Příklad 4.2.8 K zadání příkladu 4.2.4 dodejme podmínku, že každé zařízení je schopné přepravit jen 3 úkoly ($UB = 3$). Výraz pro určování S_j má tedy tvar: $S_j = \lfloor \min\{(M - 1 - 3 * 10 + 3S_{j-1})/2, S_{j-1} + 3, 10\} \rfloor$. Původní optimální rozvrh byl 2−3−4−1. Při výpočtu tedy narazíme na novou podmínku až při určování L_3. Ta vypadá následovně: $S_3 = \lfloor \min\{(35 - 1 - 30 + 3 * 5)/2, 5 + 3, 10\} \rfloor = 8 \Rightarrow L_3 = 3$ $f_3 = 0, e_3 = 1, t_3 = \max\{5 + 7, 2 * 8 + 1\} = 17$

kontrola: platí, že $17 \leq 35 - 3(10 - 5) \Leftrightarrow 17 \leq 20 \wedge 8 < 10 \rightarrow j := 4$. $S_4 = 10 \Rightarrow L_4 = 2$ kontrola: platí, že $21 \leq 32 \wedge S_4 = 10 \rightarrow STOP$.

Výsledný optimální rozvrh je 2 − 3 − 3 − 2, přičemž C_{max} zůstává stejné (35 jednotek). Vlivem omezení kapacity přepravního zařízení se přesunul jeden úkol z třetí skupiny do poslední (obr.4.17). △



Obr. 4.17: Optimální rozvrh příkladu 4.2.8

4.3 Model s nastavením strojů

V kapitole 4.1 jsme pracovali s modelem, ve kterém šlo o nalezení optimální velikosti skupin, které se budou přepravovat mezi stroji, tak, aby doba celkového zpracování byla co nejmenší. Tehdy jsme neuvažovali s časem na přepravu ani s dalšími dodatečnými požadavky. V kapitole 4.2 jsme již do modelu zahrnuli přepravu - šlo o celkovou délku cyklu CT skládající se z nakládky, přepravy, vykládky a návratu. V této kapitole do modelu místo přepravy připojíme čas potřebný k nastavení stroje pro zpracovávání příslušného úkolu.

Tento požadavek má své reálné opodstatnění. Linku na výrobu produktu je třeba před jeho výrobou buď seřídit, aby byla s to pracovat v mezích povolené tolerance, nebo přenastavit, pokud se s výrobou začíná či pokud se na lince vyrábí více různých výrobků a každý vyžaduje specifické nastavení.

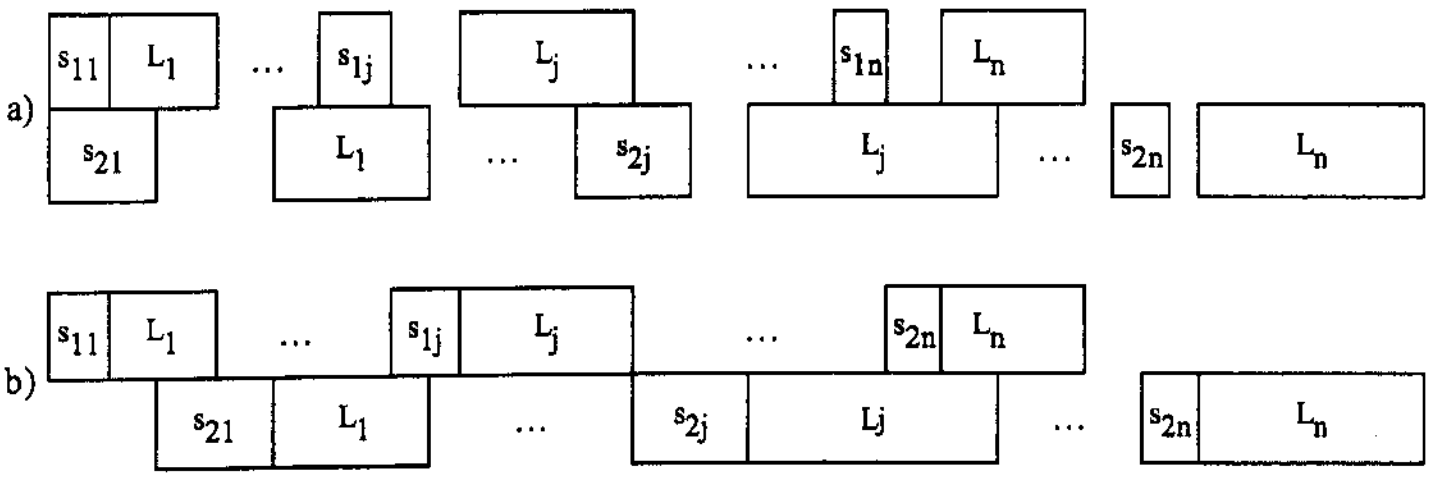
V literatuře se rozlišují dva druhy nastavení:

- *nezávislé* (detached) - lze ho provést, jakmile byl na stroji dokončen předchozí úkol a má přijít na řadu úkol, kvůli kterému se nastavení provádí. Toto nastavení lze provést, aniž by ke stroji dorazil daný úkol. Je tedy *nezávislé*, neboť ho lze nezávisle provést předem. Toto nastavení lze provést tam, kde je předem známo, jak má nastavení vypadat, a je eventuálně již z předchozí výroby odzkoušeno.

- *závislé* (attached) - lze ho provést, jakmile byl na stroji dokončen předchozí úkol a když už na něj dorazil úkol, který se bude zpracovávat. Ten je tedy k jeho provedení potřeba. Jedná se o nastavení *závislé* na přítomnosti příslušného úkolu. Tento typ nastavení se provádí např. tam, kde při zpracovávání daného úkolu sledujeme žádoucí parametry a průběžně zařízení nastavujeme, aby splňovalo stanovené požadavky.

Z hlediska modelování těchto situací má uvedený rozdíl svůj význam, který se může promítnout do celkového výsledného kritéria optimality. Pokud oba druhy nastavení porovnáme, lépe na tom mohou (za určitých podmínek není mezi oběma rozdíl) být modely s *nezávislým* nastavením, neboť je lze provést s předstihem, pokud byl předchozí úkol dokončen dříve. Jakmile daný úkol dorazí ke stroji, je zařízení nastaveno a může se rovnou začít se zpracováváním. Při *závislém* nastavení je třeba vždy čekat, až úkol dorazí na stroj, i kdyby na něm byl předtím dostatek času na přenastavení. Tak je automaticky zpracovávání zpožděno o čas potřebný k nastavení. Množina *závislých* nastavení je podmnožinou *nezávislých*, proto je *závislé* nastavení jen stejné či horší, nikdy nemůže být lepší.

Čas k nastavení může být kostanta, a tedy jeho velikost nezávislá na konkrétních úkolech či jejich počtu, může být násobkem počtu úkolů či jinak odvozená od jiných veličin. Doba nastavení se rovněž může lišit dle stroje. Toto nastavení (*závislé* i *nezávislé*) se v našich modelech vždy bude provádět jen pro celou skupinu úkolů, které společně dorazí k danému stroji. Nebude se tedy provádět v průběhu zpracovávání skupiny úkolů. Dle konkrétního modelu nemusí být v některých fázích nastavení provedeno vůbec. Jak to bude konkrétně vypadat, bude zřejmé v textu. V praxi se provádí nastavení (či spíše seřízení) i v průběhu výroby - buď deterministicky po zpracování každého x-tého výrobku (např. nabroušení soustružnických nožů), či na základě monitoringu kvality - jakmile se objeví odchylka od povolené tolerance, dochází k seřízení stroje. Druhá možnost však zahrnuje prvek pravděpodobnosti, který by vedl k nedeterministickým modelům - těmi se zde zabývat nebudeme. Zde budeme uvažovat pouze nastavování před první skupinou na daném stroji či před každou skupinou (viz obr. 4.18).

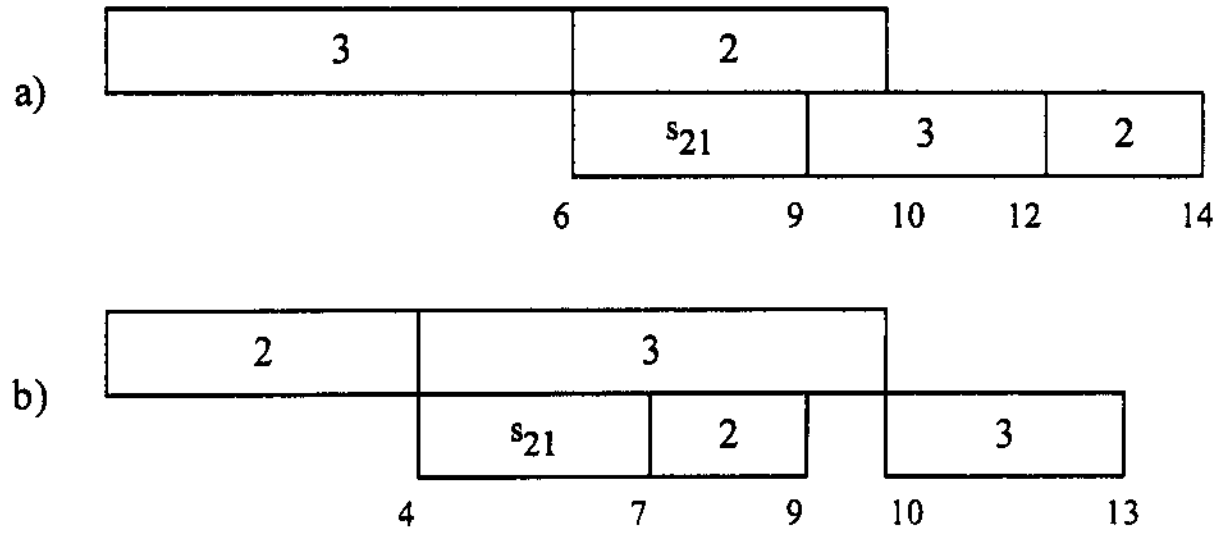


Obr. 4.18: Doba nastavení - a)*nezávislé*, b)*závislé*

Pokud do modelu zahrneme dobu potřebnou na nastavení, ptáme se, jak ovlivní C_{max} a optimální rozvrh v porovnání s modelem, kde se doba nastavení neuvažuje. Podobně jsme zkoumali situaci s dobou na převoz - v ní se v případě, kdy $RT = 0$ a TT byla konstanta, optimální rozvrh nezměnil a C_{max} se zvětšilo o TT. Určitě bychom ke stejnému závěru došli v případě, kdyby nastavení proběhlo jen na prvním stroji a jen jednou, a to před zahájením zpracovávání úkolů. To je zřejmá situace. Nastavení na tomto stroji jen posune počátek zpracovávání o konstantu, ať máme *nezávislé*, či *závislé* nastavení. Optimální rozvrh se nezmění a C_{max} se zvětší o dobu nastavení s_{11} (setup time).

Na druhém stroji toto obecně neplatí, což ukazuje následující příklad:

Příklad 4.3.1 $U = 5, p_1 = 2, p_2 = 1, s_{21} = 3, s_{11} = 0, s_{ij} = 0$ pro $j > 1, i = 1, 2, s_{ij}$ závislé.
ŘEŠENÍ Pro úlohu bez doby nastavení máme dle lemma 4.1.1 optimální rozvrh 3-2 s $C_{max} = 12$. Jakmile přidáme dobu nastavení, dostali bychom při stejném rozvrhu $C_{max} = 14$, zatímco při rozvrhu 2-3 jen 13. Srovnání je vidět na obr. 4.19. △



Obr. 4.19: a) Optimální rozvrh pro případ bez s, b) lepší rozvrh

Tento příklad potvrzuje, že se může změnit jak rozvrh, tak C_{max}. Souhrnem lze tedy říci, že se přidáním pouze s_{11} vždy C_{max} zvýší o s_{11}, ale optimální rozvrh zůstává, přidáním pouze s_{21} se může změnit jak C_{max}, tak optimální rozvrh.

Tento zřejmý závěr je obsahem následujícího lemma:

Lemma 4.3.1 *Mějme dva modely Q_1, Q_2, oba 2/V/II/DV, všechny úkoly jsou stejné v obou modelech. Modely se liší jedině tím, že v Q_1 je $s_{ij} = 0$ pro všechny kombinace i a j, v Q_2 taktéž s tím, že může být $s_{11} > 0$. Pak existuje optimální rozvrh S_1 pro Q_1 a S_2 pro Q_2 takové, že $S_1 = S_2$, a platí $C^2_{max} = C^1_{max} + s_{11}$.*

Využijeme výsledků z modelů zahrnujících přepravu a podíváme se, zda by jich nebylo možné využít i v případě modelů s nastavením strojů. Začneme nejprve

konstantní dobou a pak se podíváme na dobu lineární. Navíc bude ku prospěchu každou část, pokud to bude smysluplné, rozdělit na případ nastavení na prvním stroji a na druhém stroji s variantou nastavení *závislého* a *nezávislého*.

Úloha zůstává stále stejná - jak nejlépe nechat proudit úkoly systémem, aby se minimalizovalo C_{max}.

Konstantní doba Doba na nastavení s bude konstanta, která se může různit nanejvýš dle stroje (nemusí $s_1 = s_2$). Proto je tedy zbytečné užívat druhý index ze zavedeného značení s_{ij}. Kde je navíc z kontextu zřejmé, o jaké nastavení se jedná, budeme užívat jednoduše jen symbol s.

i. Pokud by se nastavení provádělo jen před první skupinou na prvním stroji, pak stačí určit optimální rozvrh, který nastavení nezahrnuje. Tento rozvrh pak je optimální i pro tento případ (viz lemma 4.3.1).
Jestliže se nastavení provádí před každou skupinou, a to jen na prvním stroji, lze k určení optimálního rozvrhu a C_{max} modifikovat algoritmus 4.1.1.
Dolní mez M je $M = max(ns + p_1U + p_2; p_2U + p_1 + s)$. Do všech vztahů 4.4, 4.5, 4.6 stačí přidat příslušný násobek doby nastavení (js), takže modifikovaný algoritmus 4.1.1 pak vypadá následovně:

Algoritmus 4.3.1

a) $S_0 := 0$

b) For $j := 1$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}) - js)/p_1, U\} \rfloor$;
$f_j = \min\{(M - p_2(U - S_{j-1}) - js)/p_1, U\} - S_j$;
$e_j = 1 - f_j$

c) Je-li $S_n < U$, *pak*
$M := M + p_1 \min\{e_j\}$ *a jdi do a);*
Pokud $S_n = U$, *STOP.*

Nezáleží přitom na tom, zda je nastavení *závislé* či *nezávislé.*

ii. Pokud se nastavování provede jen na druhém stroji, lze algoritmus 4.1.1 modifikovat následujícím způsobem:
1.1 *závislé* nastavení jen před první skupinou: Pro S_1 musí platit

$$S_1p_1 + s \leq M - p_2U,$$

pro ostatní pak

$$S_jp_1 \leq M - p_2(U - S_{j-1}),$$

takže algoritmus by vypadal

Algoritmus 4.3.2

a) $S_0 := 0$

b) $S_1 = \lfloor \min\{(M - p_2U - s)/p_1, U\} \rfloor$;
$f_1 = \min\{(M - p_2(U - S_{j-1}) - s)/p_1, U\} - S_j$;
$e_1 = 1 - f_1$

c) For $j := 2$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}))/p_1, U\} \rfloor$;
$f_j = \min\{(M - p_2(U - S_{j-1}))/p_1, U\} - S_j$;
$e_j = 1 - f_j$

d) Je-li $S_n < U$, *pak*
$M := M + p_1 \min\{e_j\}$ *a jdi do a);*
Pokud $S_n = U$, *STOP.*

1.2 *závislé* nastavení před každou skupinou: Pro každou skupinu musí platit

$$S_j p_1 \leq M - p_2(U - S_{j-1}) - s(n - j + 1),$$

z čehož se jednoduše vytvoří odpovídající algoritmus.

2.1 *nezávislé* nastavení jen před první skupinou: Jeden z optimálních rozvrhů je optimální rozvrh určený pro model bez nastavování z následujícího důvodu: Na druhém stroji lze úkoly v každém rozvrhu posunout tak, aby stroj pracoval od začátku zpracovávání prvního úkolu bez přestání, což platí i o optimálním rozvrhu vytvořeném pro model bez nastavování. Pokud doba nastavování neskončí později, než začne v takto upraveném optimálním rozvrhu zpracovávání L_1 (z optimálního rozvrhu bez nastavování) na stroji 2, nezasáhne nikterak do C_{max}. Pokud je větší, posune jen začátek zpracovávání první skupiny na stroji 2 a spolu s ní i zpracovávání všech skupin. V takovém případě však $C_{max} = s + p_2U$ a to již nelze nijak zlepšit. Optimální rozvrh je tedy optimální rozvrh z modelu bez nastavování a

$$C_{max} = \begin{cases} C^1_{max}, pro : s \leq C^1_{max} - p_2U \\ s + p_2U, jinak, \end{cases}$$

kde C^1_{max} je C_{max} z modelu bez nastavování.

2.2 *nezávislé* nastavení před každou skupinou: Řešení se skládá ze dvou kroků: Nejprve se určí optimální rozvrh pro model, u kterého se neuvažuje nastavení před první skupinou ($s_{21} = 0$). Tato situace je částečně analogická situaci ii-1.2, kdy bylo nastavení *závislé*. Tentokrát však nastavování pro skupinu j na druhém stroji nemusí čekat, až bude tato skupina ukončena na prvním stroji, takže skupina může být na stroji dokončena až v okamžiku, kdy se na druhém stroji ještě musí stihnout zpracovat skupina j a všechny další skupiny a pro tyto další skupiny se musí ještě nastavit druhý

stroj. Takže tedy musí platit:

$$S_j p_1 \leq M - p_2(U - S_{j-1}) - s(n-j)$$

Tímto způsobem určíme optimální rozvrh pro model bez nastavení před první skupinou a pak už jen pro zakomponování nastavení před první skupinou užijeme postup z ii-2.1.

iii. Pokud by se nastavování provádělo na obou strojích, byla by situace následující:

1.1 *závislé* nastavení jen před prvními skupinami: Nastavení na stroji 1 před první skupinou na optimální rozvrh vliv nemá, a tak ho lze nalézt bez tohoto požadavku. Použijeme tedy algoritmus 4.3.2, který nalezne optimální rozvrh pro model s nastavením pouze na druhém stroji, a to jen před první skupinou. Tento rozvrh bude optimální pro celkovou situaci, přičemž C_{max} se zvětší o s_{11}.

1.2 *závislé* nastavení před každou skupinou:
Základní podmínka je

$$S_j p_1 + j s_1 \leq M - p_2(U - S_{j-1}) - s_2(n - j + 1),$$

z čehož se již snadno určí příslušný algoritmus.

2.1 *nezávislé* nastavení jen před prvními skupinami: Využitím bodů i a ii-2.1 získáme optimální rozvrh a

$$C_{max} = \begin{cases} C^1_{max} + s_1, pro : s_2 \leq C^1_{max} + s_1 - p_2 U \\ s_2 + p_2 U, jinak, \end{cases}$$

kde C^1_{max} je jako předtím C_{max} z modelu bez nastavování.

2.2 *nezávislé* nastavení před každou skupinou: Zde opět využijeme výsledků z předchozích modelů a vytvoříme společné řešení. Postup bude opět probíhat ve dvou krocích, neboť a) nejprve určíme optimální rozvrh pro model bez nastavení na druhém stroji před první skupinou a poté b) toto nastavení zahrneme případným posunutím úkolů a nastavení, pokud bude třeba.
a) Tvorba optimálního rozvrhu:
Dolní mez 4.3 může být

$$M = max(p_1 U + p_2; p_2 U + p_1 + s_1) \tag{4.13}$$

s_2 do 4.13 není zahrnuto záměrně, neboť jeho vliv není v této fázi a) jasný. Základní podmínka 4.4 zní

$$p_1 S_j + s_1 j \leq M - p_2(U - S_{j-1}) - s_2(n-j), \tag{4.14}$$

přičemž S_j, f_j se analogicky upraví dle 4.5, resp. 4.6. Modifikovaný algoritmus 4.1.1 vypadá následovně:

Algoritmus 4.3.3

a) $S_0 := 0$

b) For $j := 1$ *to* n *do*

$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}) - s_2(n-j) - s_1 j)/p_1, U\} \rfloor$;

$f_j = \min\{(M - p_2(U - S_{j-1}) - s_2(n-j) - s_1 j)/p_1, U\} - S_j$;

$e_j = 1 - f_j$

c) Je-li $S_n < U$, *pak*

$M := M + p_1 \min\{e_j\}$ *a jdi do a);*

Pokud $S_n = U$, *STOP.*

b) Určení C_{max} :

$$C_{max} = \begin{cases} M, pro : s_2 \leq M - p_2 U \\ s_2 + p_2 U, jinak, \end{cases}$$

kde M je taková dolní mez, při které byl nalezen optimální rozvrh.

Podívejme se nyní na souhrnný příklad, na kterém bude vidět vliv nastavení.

Příklad 4.3.2 $U = 10$, $p_1 = 1$, $p_2 = 2$, $n = 3$. $s_1 = 4, s_2 = 3$.

ŘEŠENÍ (viz obr. 4.20)

a) Bez nastavení: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 22$

b) *Závislé/nezávislé* nastavení jen na prvním stroji a jen před první skupinou: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 22 + 4 = 26$

c) *Závislé/nezávislé* nastavení jen na prvním stroji před každou skupinou: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 26$

d) *Závislé* nastavení jen na druhém stroji a jen před první skupinou: $L_1 = 1, L_2 = 5, L_3 = 4, C_{max} = 24$

e) *Závislé* nastavení jen na druhém stroji před každou skupinou: $L_1 = 0, L_2 = 3, L_3 = 7, C_{max} = 23$

f) *Nezávislé* nastavení jen na druhém stroji a jen před první skupinou: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 3 + 10 * 2 = 23$

g) *Nezávislé* nastavení jen na druhém stroji před každou skupinou: $L_1 = 0, L_2 = 3, L_3 = 7, C_{max} = 26$

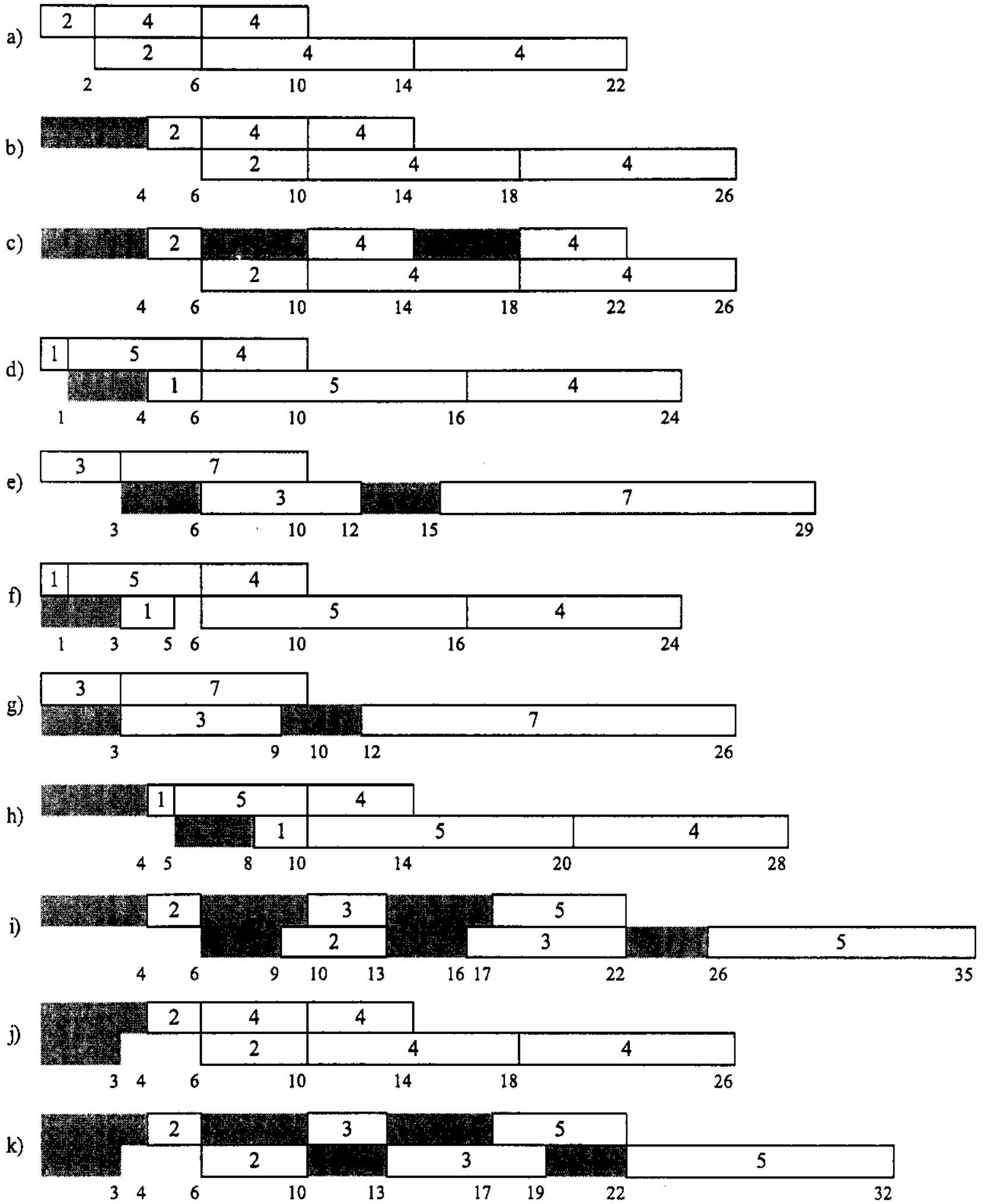
h) *Závislé* nastavení na obou strojích jen před prvními skupinami: $L_1 = 1, L_2 = 5, L_3 = 4, C_{max} = 24 + 4 = 28$

i) *Závislé* nastavení na obou strojích před všemi skupinami: $L_1 = 2, L_2 = 3, L_3 = 5, C_{max} = 35$

j) *Nezávislé* nastavení na obou strojích a jen před prvními skupinami: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 22 + 4 = 26$

k) *Nezávislé* nastavení na obou strojích před všemi skupinami: $L_1 = 2, L_2 = 3, L_3 = 5, C_{max} = 32$.

△

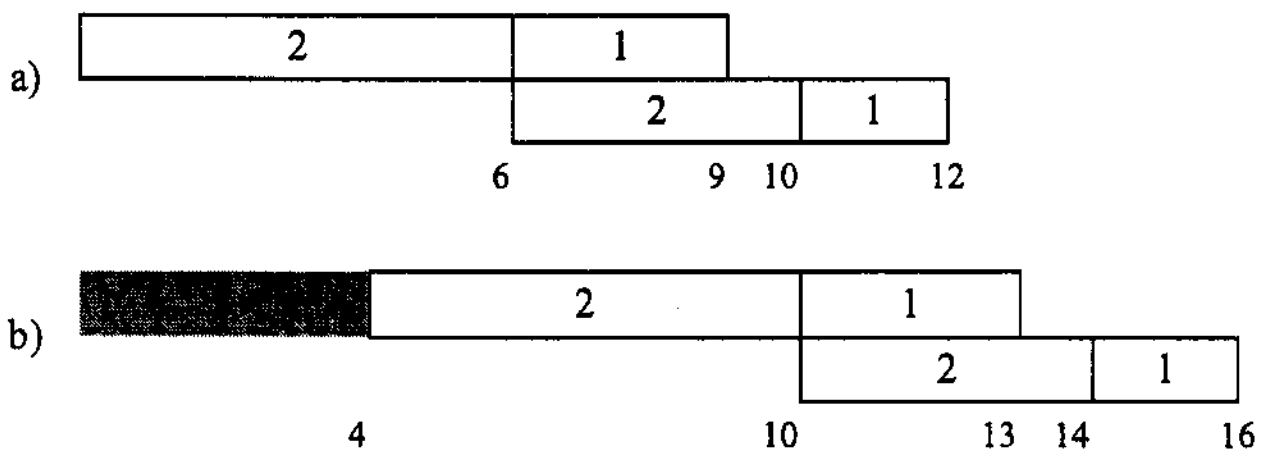


Obr. 4.20: Rozvrhy příkladu 4.3.2

Lineární doba Kromě pevně dané doby na nastavení, která nezávisí na velikosti skupiny, před kterou probíhá, ani se jinak na daném stroji nemění, může být tato doba i přímo úměrná velikosti skupiny. Nechť tedy $s_{1j} = r_1 L_j$ a $s_{2j} = r_2 L_j$, kde r_1 a r_2 jsou kladné konstanty.

Projdeme nyní jednotlivé případy uvedené v paragrafu věnovanému konstatní době:

i. Nastavení jen před první skupinou na prvním stroji: Zde nelze vytvořit optimální rozvrh pro situaci bez nastavení a poté dodatečně nastavení dodat, neboť nastavení závisí na L_1 a může výrazně do optimálního rozvrhu promluvit (viz obr. 4.21). Proto je třeba již od začátku s touto dobou počítat. Pro hledání



Obr. 4.21: a) Optimální rozvrh S_1 bez nastavení, b) S_1 s nastavením, c) Optimální rozvrh (S_2) s nastavením ($S_1 \neq S_2$)

optimálního rozvrhu a C_{max} lze modifikovat základní algoritmus 4.1.1 tak, že musí platit

$$S_j p_1 + S_1 r_1 \leq M - p_2(U - S_{j-1}).$$

Do zvyšování M se musí zahrnout i r_1, proto formule zní

$$M := M + \min\{e_1(p_1 + r_1); p_1 \min\{e_j\}\}, j > 1.$$

Nastavení před všemi skupinami na prvním stroji: V tomto případě stačí v algoritmu 4.1.1 upravit výraz $p_1 S_j$ na $(p_1 + r_1)S_j$.
Opět nezáleží na tom, zda je nastavení *závislé* či *nezávislé*.

ii. Nastavování jen na druhém stroji:

1.1 *závislé* nastavení jen před první skupinou: Pro S_1 musí platit

$$S_1(p_1 + r_2) \leq M - p_2U,$$

pro ostatní zůstává

$$S_j p_1 \leq M - p_2(U - S_{j-1})$$

a algoritmus 4.3.2 se dle toho upraví, přičemž M se zvětšuje o

$$\min\{(p_1 + r_2)e_1; p_1 e_j, j = 2, \ldots, n\}.$$

1.2 *závislé* nastavení před každou skupinou: Zde pro každou skupinu musí platit

$$S_j p_1 \leq M - (p_2 + r_2)(U - S_{j-1}).$$

2.1 *nezávislé* nastavení jen před první skupinou: V tomto případě nelze z důvodu uvedeného v odstavci (i) použít techniku využití optimálního rozvrhu bez nastavování a je třeba s dobou nastavení počítat již při hledání optimálního rozvrhu. Každá skupina musí splňovat základní podmínku

$$S_j p_1 \leq M - p_2(U - S_{j-1}),$$

která speciálně pro první skupinu vypadá

$$L_1 p_1 \leq M - p_2 U.$$

Navíc ještě musí platit

$$L_1 r_2 \leq M - p_2 U.$$

Protože chceme stanovit L_1 co největší, vychází

$$L_1 = \lfloor \min\{(M - p_2U)/p_1, (M - p_2U)/r_2\} \rfloor .$$

Je patrné, že o velikosti první skupiny rozhoduje ve výsledném rozvrhu p_1, či r_2. Pokud je $r_2 \leq p_1$, nastavení proběhne v době, kdy se L_1 zpracovává na prvním stroji a nijak do optimálního rozvrhu nezasáhne. Optimální rozvrh se určí pro situaci bez nastavování a tento rozvrh je pak optimální i pro tuto situaci. Se zvětšujícím se r_2 však roste tlak na zmenšování L_1.

Výsledný algoritmus tedy zní:

Algoritmus 4.3.4

a) $S_0 := 0$

b) $S_1 = \lfloor \min\{(M - p_2U)/p_1, (M - p_2U)/r_2\} \rfloor$;
$f_1 = \min\{(M - p_2U)/p_1, (M - p_2U)/r_2, U\} - S_1$;
$e_1 = 1 - f_1$

c) *For* $j := 2$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}))/p_1, U\} \rfloor$;
$f_j = \min\{(M - p_2(U - S_{j-1}))/p_1, U\} - S_j$;
$e_j = 1 - f_j$

d) *Je-li* $S_n < U$, *pak*

$$M := M + \begin{cases} p_1 \min\{e_j, j = 1, \ldots, n\} \text{ pro } p_1 > r_2 \\ \min\{r_2 e_1; p_1 e_j, j = 2, \ldots, n\} \text{ jinak} \end{cases}$$

a jdi do a).
Pokud $S_n = U$, *STOP.*

2.2 *nezávislé* nastavení před každou skupinou: V tomto případě musí pro každou skupinu platit

$$S_j p_1 \leq M - p_2(U - S_{j-1}) - r_2(U - S_j),$$

což po úpravě vypadá

$$S_j(p_1 - r_2) \leq M - p_2(U - S_{j-1}) - r_2 U. \tag{4.15}$$

Vztah 4.15 však třeba ještě nemusí zaručovat, že se do M vše stihne včetně nastavení před první skupinou. Proto ještě musí platit

$$U(p_2 + r_2) \leq M.$$

Jestliže je však $r_2 \leq p_1$, je na nastavení před první skupinou dostatek času a tato podmínka je automaticky splněna, což plyne i z rozpisu
$U(p_2 + r_2) \leq U(p_2 + r_2) + S_1(p_1 - r_2) \leq M$, neboť 4.15 pro S_1 vypadá $S_1(p_1 - r_2) + U(p_2 + r_2) \leq M$.
Na nerovnici 4.15 je vidět, že výjimečná situace nastává pro $p_1 = r_2$. Pokud tento případ nastane, mohou být S_j libovolná a každý přípustný rozvrh je optimální. Vezme-li se totiž průběh libovolného rozvrhu na druhém stroji, aniž by šel zkrátit ($M = U(r_2 + p_2)$), vždy je to rozvrh přípustný, protože intervaly potřebné pro nastavení před konkrétní skupinou na druhém stroji a její zpracování na prvním stroji jsou stejné, takže na druhém stroji nemůže dojít k žádnému prostoji. Toto M se tedy rovná C_{max}. Stejná logika platí i pro $r_2 > p_1$, protože intervaly potřebné pro nastavení jsou dokonce ještě větší. Souhrnně tedy pro $r_2 \geq p_1$ a $M = U(r_2 + p_2)$ nerovnice 4.15 vypadá

$$S_j(p_1 - r_2) \leq S_{j-1}p_2.$$

Protože dle předpokladu je levá strana menší nebo rovná 0 a pravá větší nebo rovná 0, platí tento vztah pro libovolný rozvrh. Zbývá tedy určit optimální rozvrh pro $r_2 < p_1$.
Výchozí M je

$$M = \max\{p_1U + p_2, p_2U + p_1, (p_2 + r_2)U\}$$

a optimální rozvrh a C_{max} lze určit následujícím algoritmem:

Algoritmus 4.3.5

a) $S_0 := 0$
b) Pokud $r_2 \geq p_1$,
pak $S_j := \min\{S_{j-1} + 1, U\}, j = 1, \ldots, n-1; S_n := U$
a jdi do d).
c) For $j := 1$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}) - r_2U)/(p_1 - r_2), U\} \rfloor$;
$f_j = \min\{(M - p_2(U - S_{j-1}) - r_2U)/(p_1 - r_2), U\} - S_j$;
$e_j = 1 - f_j$
d) Je-li $S_n < U$, *pak*
$M := M + (p_1 - r_2)\min\{e_j\}, j = 1, \ldots, n$
a jdi do a).
Pokud $S_n = U$, *STOP.*

iii. Pokud by se nastavování provádělo na obou strojích, byla by situace následující:
1.1 *závislé* nastavení jen před prvními skupinami: Zde se zkombinují podmínky ze *závislých* nastavení jen před prvními skupinami z případů každého stroje zvlášť (i, ii-1.1) a pro S_1 musí platit

$$S_1(p_1 + r_1 + r_2) \leq M - p_2U,$$

pro ostatní pak

$$S_jp_1 \leq M - p_2(U - S_{j-1}) - r_1S_1$$

a algoritmus 4.3.2 se dle toho upraví.
1.2 *závislé* nastavení před každou skupinou:
Základní podmínka je

$$S_j(p_1 + r_1) \leq M - (p_2 + r_2)(U - S_{j-1}),$$

z čehož se již snadno určí příslušný algoritmus.

2.1 *nezávislé* nastavení jen před prvními skupinami:
Analogicky dle ii.-2.1 musí platit

$$S_1 r_1 + S_j p_1 \leq M - p_2(U - S_{j-1})$$

a

$$L_1 r_2 \leq M - p_2 U.$$

Na základě těchto vztahů L_1 vychází

$$L_1 = \lfloor \min\{(M - p_2U)/(r_1 + p_1), (M - p_2U)/r_2\} \rfloor$$

Algoritmus 4.3.4 se příslušným způsobem upraví s tím, že

$$M := M + \begin{cases} \min\{(r_1 + p_1)e_1; p_1 e_j, j = 2, \ldots, n\} \text{ pro } (p_1 + r_1) > r_2 \\ \min\{r_2 e_1; p_1 e_j, j = 2, \ldots, n\} \text{ jinak} \end{cases}$$

2.2 *nezávislé* nastavení před každou skupinou:
Tento případ je obdobný případu ii.-2.2. Pro každou skupinu musí být splněno

$$S_j(p_1 + r_1 - r_2) \leq M - p_2(U - S_{j-1}) - r_2 U \tag{4.16}$$

a celkově také

$$U(p_2 + r_2) \leq M$$

Tuto podmínku jednoduše splníme, když bude jedním z kritétií při volbě počátečního M. Pokud položíme $U(p_2 + r_2) = M$, vypadá nerovnice 4.16

$$S_j(p_1 + r_1 - r_2) \leq S_{j-1} p_2$$

Opět je zřejmé, že pro $(p_1 + r_1 - r_2) \leq 0$ lze S_j volit libovolně, a tak každý přípustný rozvrh je při splnění této podmínky optimální.
Výchozí M dejme

$$M = \max\{(p_1 + r_1)U + p_2, (p_2 + r_2)U\}$$

a optimální rozvrh a C_{max} vyplyne z následujícího algoritmu:

Algoritmus 4.3.6

a) $S_0 := 0$

b) Pokud $r_2 \geq p_1 + r_1$,
pak $S_j := \min\{S_{j-1} + 1, U\}, j = 1, \ldots, n-1; S_n := U$
a jdi do d).

c) For $j := 1$ *to* n *do*
$S_j = \lfloor \min\{(M - p_2(U - S_{j-1}) - r_2 U)/(p_1 + r_1 - r_2), U\} \rfloor$;
$f_j = \min\{(M - p_2(U - S_{j-1}) - r_2 U)/(p_1 + r_1 - r_2), U\} - S_j$;
$e_j = 1 - f_j$

d) Je-li $S_n < U$, *pak*
$M := M + (p_1 + r_1 - r_2) \min\{e_j\}, j = 1, \ldots, n$
a jdi do a).
Pokud $S_n = U$, *STOP.*

Různé varianty, které jsme nyní zvažovali, znázorní následující souhrnný příklad.

Příklad 4.3.3 $U = 10$, $p_1 = 1$, $p_2 = 2$, $n = 3$. $r_1 = 2, r_2 = 0,5$.
ŘEŠENÍ (viz obr. 4.22)

a) Bez nastavení: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 22$

b) *Závislé/nezávislé* nastavení jen na prvním stroji a jen před první skupinou: $L_1 = 1, L_2 = 3, L_3 = 6, C_{max} = 24$

c) *Závislé/nezávislé* nastavení jen na prvním stroji před každou skupinou: $L_1 = 5, L_2 = 3, L_3 = 2, C_{max} = 35$

d) *Závislé* nastavení jen na druhém stroji a jen před první skupinou: $L_1 = 1, L_2 = 3, L_3 = 6, C_{max} = 22$

e) *Závislé* nastavení jen na druhém stroji před každou skupinou: $L_1 = 1, L_2 = 3, L_3 = 6, C_{max} = 26,5$

f) *Nezávislé* nastavení jen na druhém stroji a jen před první skupinou: $L_1 = 2, L_2 = 4, L_3 = 4, C_{max} = 22$

g) *Nezávislé* nastavení jen na druhém stroji před každou skupinou: $L_1 = 1, L_2 = 4, L_3 = 5, C_{max} = 25,5$

h) *Závislé* nastavení na obou strojích jen před prvními skupinami: $L_1 = 1, L_2 = 3, L_3 = 6, C_{max} = 24$

i) *Závislé* nastavení na obou strojích před všemi skupinami: $L_1 = 4, L_2 = 3, L_3 = 3, C_{max} = 37,5$

j) *Nezávislé* nastavení na obou strojích a jen před prvními skupinami: $L_1 = 1, L_2 = 3, L_3 = 6, C_{max} = 24$

k) *Nezávislé* nastavení na obou strojích před všemi skupinami: $L_1 = 4, L_2 = 3, L_3 = 3, C_{max} = 36$.

△

Obr. 4.22: Rozvrhy příkladu 4.3.3

4.4 Model s přepravou a nastavením strojů

Viděli jsme v předchozích případech, že Trietschův algoritmus lze upravit na nejrůznější případy. Pokusíme se ho nyní upravit na obecný případ, který bude zahrnovat jak nastavení na jednotlivých strojích, tak i dopravu.

Doba nastavení se může skládat z nastavení jen před první skupinou s_i^0, pevné složky s_i před každou skupinou stejné pro každou skupinu a lineárně závislé složky na velikosti skupiny, jejíž koeficient je r_i. Nastavení před konkrétní skupinou je tedy $s_i + r_i L_j, i = 1, ..., m, j = 1, ..., n$. Doba přepravy může mít taktéž dvě složky a celkově se může rovnat $TT + wL_j, j = 1, ..., n$ (TT zde bude označovat konstantu).

Pro přepravu a *závislé* nastavení musí platit:

Pro S_1:

$$s_1^0 + s_1 + r_1 S_1 + p_1 S_1 + TT + wS_1 + s_2^0 \leq M - (p_2 + r_2)U - ns_2,$$

tedy

$$(r_1 + p_1 + w)S_1 \leq M - s_1^0 - s_2^0 - s_1 - TT - (p_2 + r_2)U - ns_2.$$

Pro ostatní:

$$s_1^0 + s_1 j + r_1 S_j + p_1 S_j + TT + w(S_j - S_{j-1}) \leq M - (p_2 + r_2)(U - S_{j-1}) - (n - j + 1)s_2,$$

tedy

$$(r_1 + p_1 + w)S_j \leq M + wS_{j-1} - s_1^0 - s_1 j - TT - (p_2 + r_2)(U - S_{j-1}) - (n - j + 1)s_2.$$

Algoritmus 4.4.1

a) $S_0 := 0$

b) $S_1 = \lfloor \min\{(M - s_1^0 - s_2^0 - s_1 - TT - (p_2 + r_2)U - ns_2)/(r_1 + p_1 + w), U\} \rfloor$;
$f_1 = \min\{(M - s_1^0 - s_2^0 - s_1 - TT - (p_2 + r_2)U - ns_2)/(r_1 + p_1 + w), U\} - S_1$;
$e_1 = 1 - f_1$

c) *For* $j := 2$ *to* n *do*
$S_j = \lfloor \min\{(M + wS_{j-1} - s_1^0 - s_1 j - TT - (p_2 + r_2)(U - S_{j-1}) +$
$- (n - j + 1)s_2)/(r_1 + p_1 + w), U\} \rfloor$;
$f_j = \min\{(M + wS_{j-1} - s_1^0 - s_1 j - TT - (p_2 + r_2)(U - S_{j-1}) +$
$- (n - j + 1)s_2)/(r_1 + p_1 + w), U\} - S_j$;
$e_j = 1 - f_j$

d) *Je-li* $S_n < U$, *pak*
$M := M + (r_1 + p_1 + w) \min\{e_j\}$ *a jdi do a);*
Pokud $S_n = U$, *STOP.*

Pro přepravu a *nezávislé* nastavení musí platit (pro každou skupinu):

$$s_1^0+s_1j+r_1S_j+p_1S_j+TT+w(S_j-S_{j-1}) \leq M-p_2(U-S_{j-1})-r_2(U-S_j)-s_2(n-j),$$

tedy

$$(r_1-r_2+p_1+w)S_j \leq M+wS_{j-1}-s_1^0-s_1j-TT-p_2(U-S_{j-1})-r_2U-s_2(n-j).$$

Současně musí platit

$$s_2^0+U(p_2+r_2)+ns_2 \leq M$$

Z toho plyne následující algoritmus:

Algoritmus 4.4.2

a) $S_0 := 0, M := s_2^0+U(p_2+r_2)+ns_2$

b) *For* $j := 1$ *to* n *do*

$S_j = \lfloor \min\{(M+wS_{j-1}-s_1^0-s_1j-TT-p_2(U-S_{j-1})+$
$-r_2U-s_2(n-j))/(r_1-r_2+p_1+w), U\}\rfloor;$

$f_j = \min\{(M+wS_{j-1}-s_0-s_1j-TT-(p_2+r_2)(U-S_{j-1})+$
$-(n-j+1)s_2)/(r_1+p_1+w), U\}-S_j;$

$e_j = 1-f_j$

d) *Je-li* $S_n < U$, *pak*

$M := M+(r_1-r_2+p_1+w)\min\{e_j\}$ *a jdi do a);*

Pokud $S_n = U$, *STOP.*

Různé úkoly Doposud jsme uvažovali, že všechny úkoly jsou stejné, jednalo se tedy o jeden job skládající se ze stejných úkolů. Obecněji však lze uvažovat výrobu s různými joby, přičemž každý job se skládá ze stejných úkolů. Naším úkolem je stejně jako v případě jednoho jobu nalézt optimální rozdělení jednotlivých úkolů do skupin, které se pak přepravují ke stroji 2. Budeme přitom předpokládat, že všechny skupiny jednoho jobu budou zpracovávány za sebou, což znamená, že nastavování prováděné jen před prvním úkolem daného jobu se provede pouze tolikrát, kolik je úkolů.

Definice 4.4.1 *Pokud existuje stroj a dvě skupiny téhož jobu, mezi nimiž je skupina jiného jobu, hovoříme o* pre-emption.

Jedním ze základních výsledků týkajících se tohoto problému je následující věta. Ta opravňuje postup, kdy se hledají optimální skupiny v modelu s ve větě specifikovanými omezeními a s více joby pro každý job zvlášť (viz [2]):

Věta 4.4.1 *Nechť je dán proces skládající se z q jobů $J_l, l = 1, \ldots, q$, každý se skládá z U_l stejných úkolů, nechť je dáno n_l pro každý job, kritériem optimality celého procesu je C_{max}, neuvažujeme pre-emption, doba nastavení s^0_{1l}, resp. s^0_{2l} může být jak nezávislá, tak závislá, doba přepravy se skládá z TT_l a koeficientu w_l.*

Pak je optimální rozvrh pro každý job J_l vzhledem k optimalitě celého procesu určen minimalizací přes S_{kl} výrazu

$$z_l = \max(0, s^0_{1l} + TT_l - s^0_{2l} + \max_{1 \le k \le n_l} \{(p_{1l} + w_l)S_{kl} - (p_{2l} + w_l)S_{(k-1)l}\}) \qquad (4.17)$$

pro každý job J_l zvlášť bez ohledu na pořadí jobů nebo na rozvrh ostatních jobů.

Poznámky:

- optimalizačním kritériem může být libovolné regulární kritérium, tj. takové, které je neklesající vzhledem k dobám dokončování úkolů
- lze uvažovat i skupiny velikosti 0, takže n je horní hranicí.

Důkaz lze nalézt v [2], str.559-561.

K minimalizaci výrazu (4.17) však docházelo vždy při použití Trietschova algoritmu či algoritmů, které jsme od něj odvozovali pro uvedené předpoklady. Základní podmínka vždy byla (ve své základní formě)

$$p_1 S_j \le M - p_2(U - S_{j-1}),$$

což po úpravě vypadá

$$p_1 S_j - p_2 S_{j-1} \le M - p_2 U.$$

$p_2 U$ je konstanta a M bylo minimální takové, aby mohl být úkol zpracován. Takže levá strana je opravdu minimalizována přes všechna S_j. Dá se tedy očekávat, že se bude moci využít Trietschův algoritmus pro hledání optimálního rozvrhu pro jeden job i pro případ s více joby. Úlohu optimálního rozdělení úkolů jednotlivých jobů do skupin vzhledem k času potřebnému na nastavení a dopravu řeší algoritmy z úvodu této kapitoly, zbývá tedy nalezení optimálního seřazení jobů na jednotlivých stojích.

Jedním z předpokladů věty 4.4.1 bylo *no pre-emption*, takže se všechny úkoly každého jobu zpracovávají po sobě (ne nutně bez přestávky). Tím se úloha redukuje na určení pořadí jobů na stroji 1 a 2. Tento problém pro jednotlivé joby řeší Johnsonův algoritmus a ukazuje se, že ho lze využít i v tomto případě následujícím způsobem.

Definujme pro *nezávislé* nastavení

$$z^*_l := M - s^0_{2l} + s^0_{1l} + TT_l$$

a

$$z^{*'}_l := z_l - (s^0_{1l} - s^0_{2l} + p_{1l}U_l - p_{2l}U_l)$$

a pro *závislé* nastavení

$$Z_l^* := M - s_{2l}^0 - p_{2l} + s_{1l}^0 + TT_l + p_{1l}$$

a

$$Z_l^{*'} := Z_l - (s_{1l}^0 - s_{2l}^0 + p_{1l}U_l - p_{2l}U_l).$$

Na hodnoty z_l^* a $z_l^{*'}$, resp. Z_l^* a $Z_l^{*'}$ pak lze jednoduše využít Johnsonův algoritmus (viz [2], str. 562).

Kapitola 5

Tři stroje

Bez prostojů V této kapitole se nejprve zaměříme na situaci $3/V/NI/DV$, to znamená situaci s proměnnými skupinami mezi jednotlivými stroji (V) a bez prostojů (NI), omezení plynoucí z přepravy ani žádné jiné neuvažujeme.

Příklad 5.0.1

Podívejme se na jednoduchý příklad $p_1 = 2, p_2 = 1, p_3 = 3, U = 3, n = 2$. Naším úkolem je opět stanovit optimální rozvrh a příslušné minimální C_{max}.

Máme 4 možnosti dělení do skupin: $1-2\&1-2, 1-2\&2-1, 2-1\&1-2, 2-1\&2-1$. Z těchto možností vybíráme možnost 3, která má minimální C_{max} ze všech (14). $\triangle$

Nyní jde o to, jak tento rozvrh určit obecně. Jak již bylo řečeno ([1]), tento typ úloh se dá řešit rozdělením na dvě podúlohy - vyřeší se zvlášť optimální rozvrh mezi prvními dvěma stroji a pak mezi druhým a třetím.

Náš úvodní příklad by se řešil takto:

Příklad 5.0.2 (pokračování příkladu 5.0.1)

i. $p_1 = 2, p_2 = 1, U = 3, n = 2$
   Dle lemmatu 4.1.1 hledáme L_1 takové, aby $min(L_1 p_2, (U - L_1)p_1)$ bylo maximální:
   $L_1 = 1 : min(1, 6 - 2) = 1 \; L_2 = 2 : min(2, 6 - 4) = 2.$
   Optimální rozvrh je $2 - 1$.

ii. $p_2 = 1, p_3 = 3, U = 3, n = 2$
   $L_1 = 1 : min(3, 2) = 2 \; L_1 = 2 : min(6, 1) = 1.$
   Optimální rozvrh je $1 - 2$.

Pokud obé spojíme dohromady, získáváme rozvrh $2 - 1\&1 - 2$. Ten jsme nalezli už při vyhodnocení všech možností příkladu 5.0.1. $\triangle$

Důvod pro tento postup je asi zřejmý a plyne z možnosti mít *variabilní* skupiny a z požadavku nepřerušené práce na jednotlivých strojích (*no-idling*). Díky tomu je kritickým kritériem požadavek, aby se se zpracováváním na stroji 2 začalo co

nejdříve. Kdyby se totiž na druhém stroji započalo se zpracováváním úkolů později, než by muselo při optimálním rozvrhu řešeném jen pro první dva stroje při organizaci skupin dle požadavku *no-idling*, dá se doba zpracování celého procesu zlepšit právě optimálním rozvrhem řešeným jen pro první dva stroje mezi prvními dvěma stroji. Rozvrh pro druhou dvojici strojů lze řešit zcela nezávisle, neboť skupiny jsou *variabilní*. Již jsme viděli, že v případě dvou strojů je vždy možné mít automaticky splněn požadavek *no-idling*.

S možnými prostoji Jak již víme, v případě dvou strojů lze úkoly zpracovávat tak, aby nedocházelo k prostojům ani na jednom stroji za předpokladu, že není omezující čas návratu přepravního zařízení. Pokud by omezující byl, a muselo by tedy díky němu docházet k prostojům, pak už by výsledný model nebyl *NI*, ale s prostoji. Na tento případ se podívejme nyní:

Zabýváme se modelem $3/V/II/DV$, opět v základní verzi bez jakýchkoli dodatečných omezení. V článku [1] nalezneme tento rozbor situace:

Definice 5.0.1 *Množinu strojů, které úkoly zpracovávají nepřetržitě, tj. doba, kdy je úkol i zpracován na stroji, se rovná době, kdy na něm začíná být zpracováván úkol* $i+1$, *se nazývá* rozkladová *(partition) množina.*

Definice 5.0.2 *Rozvrh s vlastností* bez čekání *(no-wait) je takový, že každá skupina je okamžitě zpracovávána, jakmile je dokončena na předchozím stroji.*

Rozbor diskrétního případu $3/V/II/DV$ vychází z případu spojitého. Předpokládejme, že rozkladová množina je {1,3}. Vychází se ze skutečnosti, že v případě *variabilních* skupin musí existovat optimální rozvrh *bez čekání*. Jedna z podmínek existence takového rozvrhu je

$$x_{j+1}(p_1 + p_2) = x_j(p_2 + p_3) \tag{5.1}$$

Nechť $q = (p_2 + p_3)/(p_1 + p_2)$. Z podmínky (5.1) sestrojíme geometrickou řadu

$$x_j = qx_{j-1} = q^{j-1}l_1, j \geq 2.$$

Protože $\sum x_i = U$, tak

$$x_1 = U(1-q)/(1-q^n), q \neq 1$$

$$x_1 = U/n, q = 1.$$

Na stroji 2 musí být zajištěn rozvrh *bez čekání*, tedy

$$p_1(x_1 + x_2 + \ldots + x_{j+1}) \geq p_1x_1 + p_2(x_1 + x_2 + \ldots + x_j).$$

Po úpravách dostaneme

$$(p_2)^2 - p_1 p_3 \leq 0. \tag{5.2}$$

Podmínka (5.2) je klíčová i pro diskrétní případ. Pokud není splněna, pak nebyl splněn předpoklad a *rozkladovou* množinou není {1,3}, ale {1,2,3}. Tím se vlastně dostáváme do situace $3/V/NI/DV$, protože na stroji 2 zpracovávání probíhá *bez prostojů*. Jak jsme viděli v paragrafu **Bez prostojů**, tento případ se dá jednoduše řešit rozdělením na podúlohy nalezení optimálního rozvrhu mezi první dvojicí strojů a druhou dvojicí.

Pokud (5.2) splněna je, pak je *rozkladovou* množinou {1,3}. V tomto případě lze upravit algoritmus 4.1.1 pro dva stroje na případ tří strojů:

Musí platit

$$p_1 S_j + p_2 L_j \leq M - p_3(U - S_{j-1}) \tag{5.3}$$

Protože $L_j = S_j - S_{j-1}$, musí platit (i vzhledem k tomu, aby $S_j \leq U$)

$$S_j \leq \min\left(\frac{M + p_2 S_{j-1} - p_3(U - S_{j-1})}{p_1 + p_2}; U\right). \tag{5.4}$$

Podobně jako v případě dvou strojů je f_j definována jako

$$f_j = \min\left(\frac{M + p_2 S_{j-1} - p_3(U - S_{j-1})}{p_1 + p_2}; U\right) - S_j \tag{5.5}$$

a e_j nechť je $1 - f_j$. M se zvyšuje dle potřeby o $(p_1 + p_2)\min(e_j)$.

Trietschův algoritmus pro tři stroje pak vypadá

Algoritmus 5.0.1

a) $S_0 := 0$

b) *For* $j := 1$ *to* n *do*
$S_j = \lfloor \min\{(M + p_2 S_{j-1} - p_3(U - S_{j-1}))/(p_1 + p_2), U\} \rfloor$;
$f_j = \min\{(M + p_2 S_{j-1} - p_3(U - S_{j-1}))/(p_1 + p_2), U\} - S_j$;
$e_j = 1 - f_j$

c) *Je-li* $S_n < U$, *pak*
$M := M + (p_1 + p_2)\min\{e_j\}$ *a jdi do a);*
Pokud $S_n = U$, *STOP.*

Kapitola 6

Závěr

Tato práce nejprve popisuje problematiku flow shop (kapitola 2), pak speciálně lot streaming (kapitola 3), a to pro případ dvou strojů (kapitola 4) a tří strojů (kapitola 5). Největší pozornost je věnována případu dvou strojů, a to v různých modifikacích. Výchozím je Trietschův algoritmus, který je upraven pro různá omezení. Nejvíce se jedná o případ jednoho jobu. Kapitola 4.4 uvádí důležitý výsledek z literatury, který umožňuje mnohé z případu jednoho jobu rozšířit na jobů více. Podobně v případě tří strojů byly uvedeny známé výsledky, které opět umožňují rozšíření pro tři stroje. Jak případ více jobů, tak případ více strojů je uveden stručně a je již na čtenáři, aby domyslel závěry z případu jednoho jobu a dvou strojů pro tato rozšíření.

Práce kromě vlastního textu obsahuje rovněž příklady a pokud možno názorné obrázky.

Celkově tato práce podává hlavně přehled o popisované problematice. Hlavním zdrojem je článek (1), který sleduje a doplňuje či rozvádí tam, kde je to vhodné. Přínos práce spočívá především v autorově příležitosti zabývat se zajímavým tématem, které pro něj bylo v takto úzké specifikaci zcela nové, a přistoupit k němu nesamozřejmým způsobem. Snahou bylo látku pojmout, rozvést ji či aplikovat a vnést do ní vlastní úsilí. Významnou součástí práce na této diplomové práci bylo rovněž navržení konkrétní problematiky z oblasti flow shop, čemuž předcházel průzkum literatury a získání přehledu o směrech, kudy se bádání v oblasti flow shop ubírá.

Literatura

[1] Trietsch D., Baker K.R.: Basic Techniques for Lot Streaming, *Operations Research*, Vol.41, No.6, str. 1065-1076, 1993.

[2] Vickson R.G.: Optimal lot streaming for multiple products in a two-machine flow shop, *European Journal of Operational Research*, 85, str. 556-575, 1995.

[3] Glass C.A., Gupta J.N.D., Potts C.N.: Lot streaming in three-stage production processes, *European Journal of Operational Research*, 75, str. 378-394, 1994.

[4] Şen A., Topaloğlu E., Benli Ö.S.: Optimal streaming of a single job in a two-stage flow shop, *European Journal of Operational Research*, 110, str. 42-62, 1998.

[5] Hall N.G., Sriskandarajah C.: A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research*, Vol. 44, No.3, str. 510-525, 1996.

[6] Schaller J.E., Gupta J.N.D., Vakharia A.J.: Scheduling a flowline manufacturing cell with sequence dependent family setup times, *European Journal of Operational Research*, 125, str. 324-339, 2000.

[7] Sung C.S., Kim Y.H., Yoon S.H.: A problem reduction and decomposition approach for scheduling for a flowshop of batch processing machines, *European Journal of Operational Research*, 121, str. 179-192, 2000.

[8] Dudek R.A., Panwalkar S.S., Smith M.L.: The lessons of flowshop scheduling research, *Operations Research*, 40, str. 7-13, 1992.

[9] Dessouky M.M., Dessouky M.I., Verma S.K.: Flowshop scheduling with identical jobs and uniform parallel machines, *European Journal of Operational Research*, 109, str. 620-631, 1998.

[10] Chen J., Steiner G.: Lot streaming with detached setups in three machine flow shops, *European Journal of Operational Research*, 96, str. 591-611, 1996.

[11] Potts C.N., Kovalyov M.Y.: Scheduling with batching: A review, *European Journal of Operational Research*, 120, str. 228-249, 2000.

[12] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.: Sequencing and Scheduling: Algorithms and Complexity, *Handbooks in OR & MS*, 4, str.445-522, 1993.

[13] Čáp P.: Řešení speciálních rozvrhovacích úloh, diplomová práce, MFF UK, Praha, 2000.