Contents

1 Úvod **2**

2 Chyba v důkazu Lemma 4.3 **5**

3 Příčiny neúspěchu konstrukce MDCM stroje **9**

4 Neúspěšný pokus o opravu důkazu **17**

5 Oprava důkazu **19**
5.1 Simulace PRAMu na M-MDCM stroji 21
5.2 Dolní odhad doby výpočtu M-MDCM stroje 23

6 Varianta důkazu **27**

7 Závěr **30**

A Subadivita entropie **33**

1 Úvod

Jednou z hlavních oblastí informatiky, kterým je věnována v posledních letech zvýšená pozornost, je oblast týkající se paralelních výpočtů. Tento zájem je veden přirozenou snahou o získání silnějších (tj. zejména rychlejších) výpočetních prostředků, které by umožnily řešení těch problémů, jejichž velikost to zatím na běžných prostředcích v rozumném čase neumožňuje. Dochází ke konstrukci různých paralelních počítačů a zároveň k teoretickému studiu různých více či méně reálných modelů paralelních počítačů. Nejrozšířenějším a nejvíce zkoumaným takovým modelem je PRAM (Parallel Random Access Machine), který je přirozeným rozšířením jednoprocesorového modelu RAM.

PRAM počítač je tvořen n synchroními procesory P_1,...,P_n, které spolu komunikují přez nekonečnou sdílenou paměť (globální paměť). Kapacita buněk paměti je neomezená a jejich obsah je na počátku výpočtu inicializován nulou. Jeden krok výpočtu má tři fáze:

1. Každý procesor P_i provádí lokální výpočet neomezeného rozsahu.

2. Každý procesor P_i zapisuje do některé buňky globální paměti, jejíž adresu určil v předchozím kroku.

3. Každý procesor P_i čte z některé buňky globální paměti, jejíž adresu určil v prvním kroku.

Vstupem pro PRAM počítač je n-tice $(x_1, ..., x_n)$, kde hodnota proměnné x_i je známa procesoru P_i (toto je ekvivalentní tomu, kdy na počátku výpočtu je obsah i-té buňky globální paměti roven x_i, $i \leq n$).

Síla PRAM počítače může být omezena zákazem současných zápisů více procesorů do stejné buňky (tzv. exlusive write), či navíc i zákazem současného čtení ze stejné buňky (tzv. exlusive read). Tyto varianty uvažovaného modelu se nazývají CREW PRAM, resp. EREW PRAM. Jsou-li násobné čtení i zápis povoleny (tzv. concurrent read, concurrent write, CRCW PRAM), je otázkou, jak řešit konflikty při zápisu (násobné čtení se vyřeší přirozeným způsobem). V literatuře je možno nalézt několik různých způsobů řešení, podle kterých se pak i jednotlivé varianty PRAM počítače nazývají:

PRIORITY : zapisuje-li více procesorů najednou do stejné buňky, uspěje z nich ten s největším identifikačním číslem (tj. hodnota zapsaná v buňce bude rovna hodnotě, kterou zapisoval tento procesor),

ARBITRARY : při násobném zápisu uspěje některý, ne blíže specifikovaný, procesor (náhodně vybraný); algoritmus pro tento model musí správně pracovat, ať při konfliktech uspějě kterýko-li z procesorů [SV 82],

COMMON : násobné zápisy do stejné buňky jsou povoleny pouze tehdy, zapisují-li všechny procesory stejnou hodnotu [K 82],

MINIMUM : při konfliktu uspěje procesor zapisující nejmenší hodnotu [CV 86],

TOLERANT : pokouší-li se dva či více procesorů zapsat do stejné buňky, zůstane její obsah nezměněn [GR 87],

OR : hodnota uložená do buňky při násobném zápisu je rovna bitovému OR všech zapisovaných hodnot [BDH 92].

Pro úplnost připomeňme některé další varianty PRAM počítačů, lišící se ještě jinými způsoby. Někdy používaným zeslabením PRAM počítače je omezení velikosti jeho globální paměti různou měrou - lineárně, $O(n \log n)$, polynomiálně, exponenciálně, či ještě jinak (tzv. bounded memory). Toto omezení bývá občas doplněno omezením na velikost buněk globální paměti, typicky se pro vstup velikosti n uvažuje počet bitů jedné buňky pouze $O(\log n)$. Podobně jako u sekvenčních modelů se někdy závádí Randomizovaný PRAM – každý procesor má k dispozici vlastní generátor náhodných čísel.

Vraťme se však zpět k různým způsobům řešení konfliktu při zápisu. Nabízí se otázka, jaký je vztah mezi uvedenými PRAM modely? Zřejmě se jejich síla bude lišit, ale jak hodně? V tomto textu se budeme zabývat vztahem mezi PRIORITY a COMMON modely (některé výsledky platí i pro vztah ARBITRARY a COMMON). Připomeňme si nejprve některé známé způsoby simulace PRIORITY na COMMON. [K 82] popisuje simulaci jednoho kroku PRIORITY s n procesory na COMMON s n^2 procesory v konstantním čase. Počet procesorů potřebný k simulaci jednoho kroku byl později snížen na $n^{1+\epsilon}$ pro libovolné $\epsilon > 0$ [FRW 88] a ještě později až na $O(n \log n)$ [CDHR 88]. Při stejném počtu procesorů obou modelů byla popsána simulace kroku PRIORITY na COMMON v čase $O(\frac{\log n}{\log \log n})$ [FRW 88-2]. V [B 89] je ukázáno, že jeden krok PRIORITY s n procesory lze simulovat na COMMON s p procesory v čase $O(\frac{n \log n}{p \log(\frac{n}{p} \log n + 1)})$. Tento výsledek dává do vztahu počet procesorů a dobu simulace. Je zobecněním dvou předchozích vztahů a pro $p = n$, resp. $p = n \log n$ se s nimi shoduje.

Všechny dosud uvedené simulace říkají, že "PRIORITY není silnější než COMMON o více než *něco*". Přirozeně nás zajímá i opačný odhad – je PRIORITY opravdu alespoň o *něco* silnější? K tomuto oddělení dvou uvažovaných modelů se ukázal být velice užitečným problém různosti prvků. Jedná se o rozhodovací problém. Vstupem je n-tice čísel $x_1, ..., x_n$. Pokud $\exists i \neq j$, tž. $x_i = x_j$, je odpověď "ANO", v opačném případě je odpověď na tento problém "NE".

Uvedený problém lze jednoduše vyřešit v konstantním čase na PRIORITY modelu s n procesory:

1. P_i zapíše hodnotu i na adresu x_i,

2. P_i čte z adresy x_i, přečte-li číslo různé od i, ohlásí "ANO, ve vstupu existují dvě stejná čísla."

V roce 1986 Fich, Meyer Auf Der Heide a Wigderson ukázali, že na COMMON s n procesory vyžaduje tento problém $\Omega(\log \log n)$ kroků, z čehož již vyplývá rozdíl v silách uvažovaných modelů [FMW 86]. Tento odhad později Ragde, Steiger, Szemeredi a Wigderson vylepšili na $\Omega(\sqrt{\log n}\,)$ [RSSW 88]. Boppana uvádí v [B 89] dolní odhad $\Omega(\frac{n \log n}{p \log(\frac{n}{p} \log n + 1)})$, který se již až na konstantu shoduje s jeho horním odhadem. Všechny uvedené dolní odhady předpokládají nekonečnou globální paměť a neomezenou velikost vstupních čísel.

Původním záměrem této diplomové práce bylo zkoumat, jak se uvedené vztahy mezi modely COMMON a PRIORITY změní (či nezmění), omezíme-li nějakým způsobem velikost globální paměti, nebo rozsah vstupních čísel, nebo obé. Výchozím bodem pro toto zkoumání měl být již zmíněný Boppanův článek [B 89]. Při jeho podrobném čtení však byla zjištěna závažná chyba právě v důkazu uvedeného dolního odhadu. Tato skutečnost pro nás opět otevřela otázku, jaký je vztah mezi modely COMMON a PRIORITY a zásadním způsobem ovlivnila obsah této diplomové práce. Kapitoly 2 a 3 jsou věnovány podrobnému popisu chyby, kapitoly 4-6 snaze o její odstranění. Žel tato snaha nevede zcela ke kýženému cíli. Kapitola 5 uvádí částečnou opravu důkazu – chyba v důkazu je odstraněna, ale nepodařilo se zachovat celé znění dokazovaného tvrzení. Je ukázán dolní odhad

$$\Omega\left(\sqrt{\frac{n\log n}{p\log(\frac{n}{p}\log n+1)}}\right),$$

který je alespoň zobecněním dříve dokázaného odhadu $\Omega\left(\sqrt{\log n}\right)$ pro případ, kdy COMMON a PRIORITY nemají stejný počet procesorů. Kapitola 6 obsahuje částečně odlišný důkaz téhož tvrzení. V příloze A je uveden důkaz pomocného tvrzení, na které se v textu odvoláváme. Příloha B obsahuje kopii Boppanova článku [B 89], kterého se práce značně dotýká. K tomuto řešení nás vedla špatná dostupnost tohoto článku.

Na závěr dodejme, že několik málo týdnů před koncem dubna se nám po upozornění pana F. Meyera Auf Der Heide podařilo získat další článek týkající se relativní síly COMMON a PRIORITY modelů [E 91]. Edmonds v něm pro složitost problému různosti n prvků na COMMON modelu s p procesory dokazuje stejný dolní odhad, o který se snažil Boppana, a to dokonce pro případ, kdy vstupní čísla problému nejsou z nekonečné množiny, ale pouze z množiny velikosti $2^{2^{\Omega(n)}}$. Tím dává kromě jinéno odpověď i na námi nezodpovězenou otázku po skutečném vztahu COMMON a PRIORITY modelů – jeden krok PRIORITY s n procesory vyžaduje při simulaci na COMMON s p procesory $\Theta(\frac{n\log n}{p\log(\frac{n}{p}\log n+1)})$ kroků.

2 Chyba v důkazu Lemma 4.3

Jak již bylo řečeno v úvodu, v této a následující kapitole se budeme věnovat popisu chybu v článku [B 89]. Chybu lze nalézt v důkazu Lemma 4.3. Připomeňme si nejprve vlastní tvrzení tohoto lemma, jak je v článku uvedeno:

> *Existuje-li COMMON PRAM s p procesory řešící problém různosti n prvků*[1] *v čase T, pak existuje MDCM stroj*[2] *s p procesory řešící tento problém v čase 2T.*

[1] Element Distinctness

[2] Merging and Distinctness-checking machine, viz. def. na str. 323 [B 89]

V následujícím textu bude popsán jednoduchý COMMON PRAM počítač, označovaný dále CP-1, se dvěma procesory řešící problém různosti dvou prvků ve třech krocích. Dle konstrukce popsané v důkazu Lemma 4.3 sestrojíme odpovídající MDCM stroj. Záměrem této konstrukce v uvažovaném důkazu bylo ukázat existenci MDCM stroje, o němž lemma hovoří. Avšak analýzou MDCM stroje sestrojeného v našem konkrétním případě zjistíme, že vzniklý stroj problém neřeší. Tím bude ukázána nesprávnost popsané konstrukce a tedy i celého důkazu. Uvědomme si však, že v otázce pravdivosti tvrzení daného lemma nám tento příklad nikterak nepomůže. Příklad toto lemma nevyvrací, pouze ukazuje nesprávnost důkazu.

Z formálního hlediska má důkaz kromě zmíněné zásadní chyby jednu drobnou a snadno odstranitelnou. Konstrukce MDCM stroje k danému COMMON PRAMU je popsána indukcí, ale je opomenut počáteční krok. Předpokládejme tedy, že vypadá takto (velice přirozeně): Procesor P_i, pro $1 \leq i \leq n$, sjednotí svou množinu proměnných (v tuto chvíli ještě prázdnou) s množinou $\{x_i\}$, procesor P_i, pro $n < i \leq p$, slévací krok přeskočí.

Popišme nyní slibovaný COMMON PRAM CP-1. Je tvořen dvěma procesory a globální paměti, jejíž buňky jsou indexovány od -2 do ∞. Na počátku výpočtu zná procesor P_i proměnou x_i, $1 \leq i \leq 2$. V prvním kroku čte procesor P_i, $1 \leq i \leq 2$, obsah buňky na adrese x_i, po té do ní zapíše hodnotu 19. Ve druhém kroku čte procesor P_1 z adresy x_1 a zapíše na tutéž adresu hodnotu 1, procesor P_2 čte z adresy -2 a zapíše tamtéž hodnotu 7. V posledním, třetím kroku čte P_1 z adresy -1, po té do ní zapíše číslo 9, P_2 čte z adresy x_2, zapíše na ni číslo 17. Přečte-li P_2 v tomto kroku hodnotu 1, ohlásí *"V zadaných číslech existují dvě stejná."*, v opačném případě, tj. přečte-li 19, ohlásí *"V zadaných číslech neexistují dvě stejná."* Uvádění čtecí funkce v prvním kroku a zapisové v poslením je pouze formální záležitostí.

COMMON PRAM CP-1: globální paměť indexována od -2 do ∞

		P_1	P_2
Krok 1:			
	čte z adresy	x_1	x_2
	zápis na adr: co zapisuje	$x_1 : 19$	$x_2 : 19$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$
Krok 2:			
	čte z adresy	x_1	-2
	zápis na adr: co zapisuje	$x_1 : 1$	$-2 : 7$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$
Krok 3:			
	čte z adresy	-1	x_2
	zápis na adr: co zapisuje	$-1 : 9$	$x_2 : 17$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$

Uvědomme si, že program je zabezpečen proti zápisu různých hodnot na stejnou adresu. Dále je zřejmé, že popsaný PRAM problém různosti dvou čísel řeší.

Rozeberme podrobně simulaci tohoto COMMON PRAM počítače na MDCM stroji dle konstrukce popsané v důkazu Lemma 4.3. Uvažme nejprve zapisové a čtecí funkce jednotlivých procesorů:

$$R_{1,1}(x_1, x_2) = x_1, R_{2,1}(x_1, x_2) = x_2$$

$$W_{1,1}(x_1, x_2) = x_1, W_{2,1}(x_1, x_2) = x_2$$

$$R_{1,2}(x_1, x_2) = x_1, R_{2,2}(x_1, x_2) = -2$$

$$W_{1,2}(x_1, x_2) = x_1, W_{2,2}(x_1, x_2) = -2$$

$$R_{1,3}(x_1, x_2) = -1, R_{2,3}(x_1, x_2) = x_2$$

$$W_{1,3}(x_1, x_2) = -1, W_{2,3}(x_1, x_2) = x_2$$

Všimněme si, že popsaný PRAM splňuje bez dalších úprav požadavek na totožnost čtecí a zápisové funkce každého procesoru v každém kroku. Množina funkcí $F = \{W_{i,j} \mid 1 \leq i \leq 2, 1 \leq j \leq 3\}$ je na celém jejich definičním oboru redukovatelná[3] na množinu $G = \{g_1(y) = y, g_2() = -2, g_3() = -1\}$. Vezměme $C = D$, pak množina $F|_C$ je kanonická[4] (mimochodem $F|_C = F$).

[3] reducible

[4] canonical

Postupujme nyní po jednotlivých krocích výpočtu a dle uvedené konstrukce popíšeme akce všech procesorů vytvářeného MDCM stroje v každém kroku. Množiny proměnných těchto procesorů budeme pro vetší přehlednost indexovat dole i nahoře tak, že V_i^j označuje množinu proměnných, jejichž hodnoty jsou známy i-tému procesoru MDCM stroje na konci kroku j.

Konstrukce MDCM stroje M-1:

Krok 0:

P_1: $V_1^0 = \{x_1\}$	P_2: $V_2^0 = \{x_2\}$

$\Gamma^0 = \emptyset$.

Krok 1: Zvolíme $\hat{x} = (1, 2)$, pak $O = \{(a, b) \mid a < b,\ a, b \in C\}$

P_1:	P_2:
$g = g_1$	$g = g_1$
$\rho(y) = x_1$	$\rho(y) = x_2$
$g^\rho(y_1, y_2) = y_1$	$g^\rho(y_1, y_2) = y_2$
$\vec{k} = [x_1]$	$\vec{k} = [x_2]$
dvojice $(g_1, [x_1])$ dosud nepoužita pro vstup $\hat{x}$	dvojice $(g_1, [x_2])$ dosud nepoužita pro vstup $\hat{x}$
DC-případ, P_1 zvolí $(g_1, [x_1])$	DC-případ, P_2 zvolí $(g_1, [x_2])$
$V_1^1 = \{x_1\}$	$V_2^1 = \{x_2\}$

$\Gamma^1 = \emptyset$.

Krok 2: Zvolíme $\hat{x} = (1, 2)$, pak $O = \{(a, b) \mid a < b,\ a, b \in C\}$

P_1:	P_2:
$g = g_1$	$g = g_2$
$\rho(y) = x_1$	$\rho = \emptyset$
$g^\rho(y_1, y_2) = y_1$	$g^\rho(y_1, y_2) = -2$
$\vec{k} = [x_1]$	$\vec{k} = [\,]$
dvojice $(g_1, [x_1])$ již použita pro vstup $\hat{x}$, naposledy v kroku 1	dvojice $(g_2, [\,])$ dosud nepoužita pro vstup $\hat{x}$
M-případ, $P_l = P_1$	DC-případ, P_2 zvolí $(g_2, [\,])$
$V_1^2 = \{x_1\}$	$V_2^2 = \{x_2\}$

$\Gamma^2 = \emptyset$.

Krok 3: Zvolíme $\hat{x} = (1, 2)$, pak $O = \{(a, b) \mid a < b,\ a, b \in C\}$

P_1:	P_2:
$g = g_3$	$g = g_1$
$\rho = \emptyset$	$\rho(y) = x_2$
$g^\rho(y_1, y_2) = -1$	$g^\rho(y_1, y_2) = y_2$
$\vec{k} = [\,]$	$\vec{k} = [x_2]$

dvojice $(g_3, [\,])$ dosud nepoužita pro vstup $\hat{x}$	dvojice $(g_1, [x_2])$ již použita pro vstup $\hat{x}$, naposledy v kroku 1
DC-případ, P_1 zvolí $(g_3, [\,])$	M-případ, $P_l = P_2$
$V_1^3 = \{x_1\}$	$V_2^3 = \{x_2\}$

$\Gamma^3 = \emptyset$.

Protože neexistují dvě přístupové dvojice $(r, \vec{a})$, $(r, \vec{b})$ zvolené v různých krocích výpočtu MDCM stroje takové, že $r \in G$ a $(\vec{a}, \vec{b})$ kryje[5] (x_1, x_2), bude DG-graf[6] odpovídající právě zkonstruovanému MDCM stroji prázdný. Jenžto $V_1^3 = \{x_1\}$, $V_2^3 = \{x_2\}$, bude i částečné uspořádání Γ^3 na konci výpočtu prázdné. MDCM stroj problém neřeší.

3 Příčiny neúspěchu konstrukce MDCM stroje

Uvažujme dále o výpočtech COMMON PRAM počítačů i o jejich simulaci na MDCM strojích. Naší hlavní snahou bude ujasnit si příčinu selhání konstrukce popsané v důkazu Lemma 4.3. Uveďme si nejprve další dva jednoduché příklady COMMON PRAM počítačů, pro které zmíněnou konstrukcí získáme fungující MDCM stroje. Popis obou těchto PRAM počítačů bude opět uveden v podobě, kde čtecí i zápisové funkce každého procesoru jsou v každém kroku stejné. Uvádění čtecí funkce v prvním kroku a zapisové v poslením je i zde pouze formální záležitostí.

COMMON PRAM CP-2: globální paměť indexována od 0 do ∞

		P_1	P_2
Krok 1:			
	čte z adresy	x_1	0
	zápis na adr: co zapisuje	$x_1 : 1$	$0 : 10$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$
Krok 2:			
	čte z adresy	x_1	x_2
	zápis na adr: co zapisuje	$x_1 : 103$	$x_2 : 103$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$

[5] covers

[6] distinctness graph, str. 323 [B 89]

Přečte-li P_2 v druhém kroku 0, ohlásí *"V zadaných číslech neexistují dvě stejná."* Přečte-li 1, ohlásí *"V zadaných číslech existují dvě stejná."*

Podobně jako u CP-1 je i u tohoto počítače množina zápisových funkcí $F = \{W_{i,j} \mid 1 \leq i \leq 2,\ 1 \leq j \leq 2\}$ redukovatelná na celém jejich definičním oboru, tentokrát na množinu $G = \{g_1(y) = y, g_2() = 0\}$. Pro $C = D$ je $F|_C$ kanonická.

Konstrukce MDCM stroje M-2:

Krok 0:

P_1:	$V_1^0 = \{x_1\}$	P_2:	$V_2^0 = \{x_2\}$

$\Gamma^0 = \emptyset$.

Krok 1: Zvolíme $\hat{x} = (3, 11)$, pak $O = \{(a, b) \mid a < b,\ a, b \in C\}$

P_1:	$g = g_1$	P_2:	$g = g_2$
	$\rho(y) = x_1$		$\rho = \emptyset$
	$g^\rho(y_1, y_2) = y_1$		$g^\rho(y_1, y_2) = 0$
	$\vec{k} = [x_1]$		$\vec{k} = [\ ]$
	dvojice $(g_1, [x_1])$ dosud nepoužita pro vstup $\hat{x}$,		dvojice $(g_2, [\ ])$ dosud nepoužita pro vstup $\hat{x}$
	DC-případ, P_l zvolí $(g_1, [x_1])$		DC-případ, P_2 zvolí $(g_2, [\ ])$
	$V_1^1 = \{x_1\}$		$V_2^1 = \{x_2\}$

$\Gamma^1 = \emptyset$.

Krok 2: Zvolíme $\hat{x} = (15, 2)$, pak $O = \{(a, b) \mid a > b,\ a, b \in C\}$

P_1:	$g = g_2$	P_2:	$g = g_1$
	$\rho = \emptyset$		$\rho(y) = x_2$
	$g^\rho(y_1, y_2) = 0$		$g^\rho(y_1, y_2) = y_2$
	$\vec{k} = [\ ]$		$\vec{k} = [x_2]$
	dvojice $(g_1, [x_1])$ již použita pro vstup $\hat{x}$ naposledy v kroku 1		dvojice $(g_1, [x_2])$ dosud nepoužita pro vstup $\hat{x}$,
	M-případ, $P_l = P_1$		DC-případ, P_2 zvolí $(g_1, [x_2])$
	$V_1^2 = \{x_1\}$		$V_2^2 = \{x_2\}$

$\Gamma^2 = \emptyset$.

Protože existují přístupové dvojice $(g_1, [x_1])$, $(g_1, [x_2])$ zvolené v různých (v prvním a ve druhém) krocích výpočtu MDCM stroje takové, že $([x_1], [x_2])$ kryje (x_1, x_2), odsahuje DG-graf odpovídající zkonstruovanému MDCM stroji hranu mezi vrcholy x_1, x_2. DG-graf je tedy úplným grafem na množině všech proměnných V a MDCM stroj problém řeší.

COMMON PRAM CP-3: globální paměť indexována od -1 do ∞

		P_1	P_2	P_3
Krok 1:				
	čte z adresy	-1	0	0
	zápis na adr: co zapisuje	$-1 : x_1$	$0 : 105$	$0 : 105$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$
Krok 2:				
	čte z adresy	-1	-1	-1
	zápis na adr: co zapisuje	$-1 : 7$	$-1 : 7$	$-1 : 7$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_1, x_3\}$
Krok 3:				
	čte z adresy	0	x_1	0
	zápis na adr: co zapisuje	$0 : 7$	$x_1 : x_2$	$0 : 7$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_1, x_3\}$
Krok 4:				
	čte z adresy	0	0	x_1
	zápis na adr: co zapisuje	$0 : 7$	$0 : 7$	$x_1 : 7$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_1, x_2\}$	$\{x_1, x_2, x_3\}$

Procesor P_3 zná pro každý vstup na konci výpočtu hodnoty všech tří proměnných, může tedy snadno rozhodnout, nabývají-li některé z nich stejných hodnot.

U tohoto počítače je množina zápisových funkcí $F = \{W_{i,j} \mid 1 \leq i \leq 3,\ 1 \leq j \leq 4\}$ redukovatelná na množinu $G = \{g_1(y) = y, g_2() = 0, g_3() = -1\}$, opět na celém definičním oboru funkcí, tedy $F|_C$ je kanonická pro $C = D$.

Konstrukce MDCM stroje M-3

Krok 0:

P_1: $V_1^0 = \{x_1\}$ P_2: $V_2^0 = \{x_2\}$ P_3 $V_3^0 = \{x_3\}$

$\Gamma^0 = \emptyset$.

Krok 1: Zvolíme $\hat{x} = (1,2,3)$, pak $O = \{(a,b,c) \mid a < b < c,\ a,b,c \in C\}$

P_1:	P_2:	P_3:
$g = g_3$	$g = g_2$	$g = g_2$
$\rho = \emptyset$	$\rho = \emptyset$	$\rho = \emptyset$
$g^\rho(y_1, y_2) = -1$	$g^\rho(y_1, y_2) = 0$	$g^\rho(y_1, y_2) = 0$
$\vec{k} = [\ ]$	$\vec{k} = [\ ]$	$\vec{k} = [\ ]$
dvojice $(g_3, [\ ])$ nepoužita	dvojice $(g_2, [\ ])$ nepoužita	dvojice $(g_2, [\ ])$ nepoužita
DC-případ,	DC-případ,	DC-případ,
P_1 zvolí $(g_3, [\ ])$	P_2 zvolí $(g_2, [\ ])$	P_3 zvolí $(g_2, [\ ])$
$V_1^1 = \{x_1\}$	$V_2^1 = \{x_2\}$	$V_3^1 = \{x_3\}$

$\Gamma^1 = \emptyset$.

Krok 2: Zvolíme $\hat{x} = (2,3,4)$, pak $O = \{(a,b,c) \mid a < b < c,\ a,b,c \in C\}$

P_1:	P_2:	P_3:
$g = g_3$	$g = g_3$	$g = g_3$
$\rho = \emptyset$	$\rho = \emptyset$	$\rho = \emptyset$
$g^\rho(y_1, y_2) = -1$	$g^\rho(y_1, y_2) = -1$	$g^\rho(y_1, y_2) = -1$
$\vec{k} = [\ ]$	$\vec{k} = [\ ]$	$\vec{k} = [\ ]$
dvojice $(g_3, [\ ])$ použita	dvojice $(g_3, [\ ])$ použita	dvojice $(g_3, [\ ])$ použita
M-případ,	M-případ,	M-případ,
$P_l = P_1$	$P_l = P_1$	$P_l = P_1$
$V_1^1 = \{x_1\}$	$V_2^1 = \{x_1, x_2\}$	$V_3^1 = \{x_1, x_3\}$

$\Gamma^2 = \{x_1 - x_2, x_1 - x_3\}$.

V tuto chvíli dochází ke větvení konstrukce MDCM stroje v závislosti na stavu částečného uspořádání Γ.

a) $\Gamma_2 = \{x_1 < x_2, x_1 < x_3\}$

Krok 3: Zvolíme $\hat{x} = (1,2,3)$, pak $O = \{(a,b,c) \mid a < b < c,\ a,b,c \in C\}$

P_1:	P_2:	P_3:
$g = g_2$	$g = g_1$	$g = g_2$
$\rho = \emptyset$	$\rho(y) = x_1$	$\rho = \emptyset$
$g^\rho(y_1, y_2) = 0$	$g^\rho(y_1, y_2) = x_1$	$g^\rho(y_1, y_2) =$
$\vec{k} = [\ ]$	$\vec{k} = [x_1]$	$\vec{k} = [\ ]$
dvojice $(g_2, [\ ])$ použita	dvojice $(g_1, [x_1])$ nepoužita	dvojice $(g_2, [\ ])$ použita
M-případ,	DC-případ,	M-případ,
$P_l = P_2$	P_2 zvolí $(g_1, [x_1])$	$P_l = P_2$
$V_1^1 = \{x_1\}$	$V_2^1 = \{x_1, x_2\}$	$V_3^1 = \{x_1, x_3\}$

$\Gamma^3 = \{x_1 < x_2, x_1 < x_3\}$.

Krok 4: Zvolíme $\hat{x} = (1, 2, 5)$, pak $O = \{(a, b, c) \mid a < b < c,\ a, b, c \in C\}$

P_1:	$g = g_2$	P_2:	$g = g_2$	P_3:	$g = g_1$
	$\rho = \emptyset$		$\rho = \emptyset$		$\rho(y) = x_1$
	$g^\rho(y_1, y_2) = 0$		$g^\rho(y_1, y_2) = x_1$		$g^\rho(y_1, y_2) =$
	$\vec{k} = [\ ]$		$\vec{k} = [\ ]$		$\vec{k} = [x_1]$
	dvojice $(g_2, [\ ])$ použita		dvojice $(g_2, [\ ])$ použita		dvojice $(g_1, [x_1])$ použita
	M-případ,		M-případ,		M-případ,
	$P_l = P_1$		$P_l = P_1$		$P_l = P_2$
	$V_1^1 = \{x_1\}$		$V_2^1 = \{x_1, x_2\}$		$V_3^1 = \{x_1, x_2, x_3\}$

$\Gamma^4 = \{x_1 < x_2, x_1 < x_3, x_2 - x_3\}$.

V tomto okamžiku je Γ^4 úplným uspořádáním na množině všech proměnných, není pro nás důležité, jak skutečně vypadá, tj. zda $x_2 < x_3$, $x_2 = x_3$ či $x_2 > x_3$.

Obdobným způsobem by se pokračovalo i v případě b) pro $\Gamma^2 = \{x_2 < x_1, x_3 < x_1\}$. Došli bychom též ke zjištění, že na konci 4. kroku bude Γ^4 úplným uspořádáním. Dokonce díky jednoduchosti funkcí z množiny F bude výpočet v této větvi probíhat stejně jako ve větvi pro případ a). Pro $\Gamma^2 = \{x_1 < x_2 < x_3\}$ a pro $\Gamma^2 = \{x_3 < x_2 < x_1\}$ je úloha již vyřešena, výpočet končí po 2. kroku. Výpočet končí i v případě, že $x_1 = x_3$ nebo $x_2 = x_3$

Ve všech větvích výpočtu sestrojeného MDCM stroje je buď částečné uspořádání Γ nejpozději ve čtvrtém kroku výpočtu úplným uspořádáním, nebo některý procesor do konce 2. kroku zjistí porovnáním hodnot dvou proměnných jejich totožnost. MDCM stroj úlohu řeší.

Všimněme si, že u dvou předchozích MDCM strojů bylo Γ stále prázdným uspořádáním, proto nedocházelo ke větvení výpočtu.

Přemýšlejme nyní obecněji o různých způsobech předávání informace při výpočtu COMMON PRAM počítačů při řešení problému různosti prvků. Pro každý počítač budeme bez dalšího zmiňování předpokládat totožnost čtecí a zápisové funkce každého procesoru v každém kroku. Splnění tohoto předpokladu je možno u každého COMMON PRAMu dosáhnout za cenu zdvojnásobení počtu kroků výpočtu. Dále budeme uvažovat pouze vstupy z množiny C, na které je množina F zápisových funkcí počítače kanonická. Existenci takové množiny zajišťuje Lemma 4.2 uvažovaného článku. Předpokládejme ještě další požadavek na vstup - shodnost maximálně dvou zadaných čísel. Poslední dva uvedené předpoklady znamenají pouze omezení se na jistou podmnožinu všech možných vstupů. Vzhledem k tomu, že se zabýváme dolním odhadem doby výpočtu, je možno oba tyto předpoklady provést.

Písmenem G budeme vždy označovat množinu prostých funkcí s disjunktními obory hodnot, na kterou je redukovatelná množina $F|_C$. Nechť vektor $\vec{x} = (x_1, ...x_n)$ značí zadání úlohy, O třídu uspořádání odpovídající vektoru $\vec{x}$, tj. $O = \{\vec{y}|\ \vec{x} \equiv \vec{y}\}$ (pokud pro $\vec{x} = (x_1, ..., x_n)$ a $\vec{y} = (y_1, ..., y_n)$ platí: $\forall\ i, j\ x_i < x_j$ právě když $y_i < y_j$, budeme psát $\vec{x} \equiv \vec{y}$ a hovořit o konzistentních vektorech).

Bude-li v některých místech vhodné z důvodu větší srozumitelnosti rozlišit mezi *proměnnou* x_i a *hodnotou* proměnné x_i, budeme pro hodnotu proměnné x_i užívat označení $\hat{x}_i$. Pro funkci $f(x_1, ..., x_n)$ budeme množinu $\{x_1, ..., x_n$ označovat $var(f)$ a její velikost $dim(f)$. Pro částečné uspořádání Γ množiny proměnných $\{x_1, ..., x_n\}$ a vektor $\hat{x} = (\hat{x}_1, ..., \hat{x}_n)$ řekneme, že $\hat{x}$ je konzistentní s Γ, pokud $\forall\ i, j\ \hat{x}_i < \hat{x}_j$ právě když $x_i <_\Gamma x_j$.

Popišme si první možný způsob. Nechť v kroku t_1 užije procesor P_i k zápisu do paměti funkci W_{i,t_1}, nechť d značí adresu buňky, do které bylo zapsáno. Dle provedených předpokladů a definice redukovatelnosti existuje funkce $g \in G$ a prosté zobrazení $\rho_1 : var(g) \rightarrow var(W_{i,t_1})$ tak, že $W_{i,t_1} = g^{\rho_1}$ na třídě uspořádání O. Nechť t_2 je první krok po kroku t_1, ve kterém některý procesor, označme ho P_j, užil ke čtení funkci W_{j,t_2} takovou, že $W_{j,t_2} = W_{i,t_1}$. Dále předpokládejme, že v krocích $t_1 + 1, \ldots, t_2 - 1$ nepoužil žádný procesor k zápisu funkci $W_{k,t}$ takovou, že: $W = g^\rho$ a zároveň $\exists! y \in var(g) : \rho(y) \neq \rho_1(y)$. Pak v kroku t_2 přečte procesor P_j v buňce d to, co do ní v kroku t_1 procesor P_i zapsal.

Tímto způsobem může procesor P_i předat procesoru P_j informace o hodnotách všech proměnných, které v kroku t_1 zná. U třetího zmíněného PRAM počítače probíhá tímto způsobem komunikace mezi procesory $P_1 - P_2$ a $P_1 - P_3$ v prvním (zápis) a druhém (čtení) kroku (jedná se o přenos hodnoty proměnné x_1) a dále mezi procesory $P_2 - P_3$ ve třetím (zápis) a čtvrtém (čtení) kroku (přenos hodnoty proměnné x_2). U počítačů v prvním a druhém příkladu se tento způsob komunikace nevyskytuje.

Druhý způsob přenosu informace napomáhající k vyřešení problému různosti probíhá takto: Nechť v kroku t_2 užije procesor P_j ke čtení funkci W_{j,t_2} takovou, že na třídě O $W_{j,t_2} = g^{\rho_2}$ pro nějaké $g \in G$ a $\rho_2 : var(g) \rightarrow$

$var(W_{j,t_2})$. Nechť t_2 je první krok výpočtu, ve kterém je tato funkce užita. Přečte-li procesor P_j tomto v kroku jakouko-li hodnotu různou od 0, nabývají jistě některé dvě proměnné stejné hodnoty (uvědomme si, že funkce z množiny G jsou prosté a vzájemně disjunktní). V opačném případě lze obdobným způsobem usoudit o různosti hodnot některých proměnných.

Rozpoznání, zda některé proměnné nabývají stejných hodnot, se v tomto případě děje na základě toho, zda daný procesor v jistém kroku přečte v buňce globální paměti číslo 0, či nějaké jiné. Rozhodující není přečtená *hodnota*, ale pouze skutečnost, zda buňka globální paměti již byla či dosud nebyla *použita*. Tento přenos informace užívá druhý uvedený PRAM počítač CP-2. Ve druhém kroku procesor P_2 na základě přečtené hodnoty rozpozná, zda $\hat{x}_1 = \hat{x}_2$. Uvědomme si, že toto rozeznání je děláno, aniž by procesoru P_2 byla známa hodnota proměnné x_1.

Oba doposud zmíněné způsoby jsou uvažovány v důkazu Lemma 4.3 při simulaci COMMON PRAM počítače. První z nich odpovídá tzv. *Merging case* při konstrukci MDCM stroje. Pro daný vstup proměnné známé na konci výpočtu jednomu procesoru PRAM počítače jsou na konci výpočtu odpovídajícího MDCM stroje porovnatelné v částečném uspořádání Γ. Druhý popsaný způsob odpovídá při konstrukci tzv. *Distinctness-checking case*. Pro daný vstup dvojice proměnných, totožnost či různost jejichž hodnot je PRAM počítač schopen rozeznat zmíněným způsobem, budou na konci odpovídajícího výpočtu MDCM stroje spojeny hranou v DG-grafu.

Věnujme se nyní třetímu, poslednímu způsobu. V kroku t_1 užije procesor P_k k zápisu funkci $W_{k,t_1} = g^{\rho_1}$, a procesor P_l, užije k zápisu funkci $W_{l,t_1} = g^{\rho_2}$, přičemž $\exists! y \in var(g) : \rho_1(y) \neq \rho_2(y)$. V kroku t_2, $t_2 > t_1$, zapíše procesor P_i, do paměti funkcí $W_{i,t_2} = W_{k,t_1}$. Ještě později, v kroku t_3, $t_3 > t_2$, čte procesor P_j, $j \neq i$, pomocí funkce $W_{j,t_3} = W_{l,t_1}$. Předpokládejme, že kromě procesorů P_k a P_l v kroku t_1 a procesoru P_i v kroku t_2, nepoužil žádný procesor od počátku výpočtu do kroku $t_3 - 1$ funkci W takovou, že $W = W_{k,t_1}$ nebo $W = W_{l,t_2}$ a že v krocích $t_1 + 1, ...t_3 - 1$ nepoužil žádný procesor funkci W takovou, že $W = g^\rho$ a $\exists! y \in var(g) : \rho(y) \neq \rho_1(y)$. Dále předpokládejme $\rho_2(y) = x_p$, $\rho_2(y) = x_q$. Pokud $\hat{x}_p = \hat{x}_q$, obsahuje buňka čtená procesorem P_j v kroku t_3 hodontu zapsanou do ní procesorem P_i v kroku t_2. V opačném případě obsahuje hodnotu zapsanou do ní procesorem P_l v kroku t_1. Na základě této úvahy vidíme, že procesor P_j je schopen rozeznat (při vhodně zvolených zapisovaných hodnotách procesoru P_l v kroku t_1 a procesoru P_2 v kroku t_2), zda $\hat{x}_p = \hat{x}_q$ či zda $\hat{x}_p \neq \hat{x}_q$.

Vzhledem k tomu, že obě funkce W_{k,t_1} i W_{l,t_1} byly obě poprvé použity ve stejném kroku, budou v simulujícím MDCM stroji zvoleny přístupové dvojice $(g, \vec{k})$, $(g, \vec{l})$ zvoleny ve *stejném* kroku, kde $\vec{k}$, resp. $\vec{l}$ je vektor proměnných z V odpovídající[7] zobrazení ρ_1, resp. ρ_2 (tj. $\vec{k} = (\rho(y_1), \ldots, \rho(y_l))$, je-li g funkcí proměnných $y_1, \ldots, y_l$ v tomto pořadí).

Jsou-li funkce W_{k,t_1}, W_{l,t_1} jedinou dvojicí funkcí použitých během výpočtu, pro které platí

1. $W_{k,t_1} = f^{\rho}$, $W_{l,t_1} = f^{\rho'}$ pro nějaké $f \in G$,
2. $\exists! y \in var(f) : \rho(y) \neq \rho'(y)$,
3. $\rho(y) = x_p$, $\rho'(y) = x_q$,

pak na konci žádné větve výpočtu MDCM stroje odpovídajícího popisovanému PRAMu nebudou v DG-grafu vrcholy x_p, x_q spojeny hranou.

Pokud navíc existuje výpočet PRAM počítače, na jehož konci žádný procesor nezná hodnoty obou proměnných x_p, x_q, pak v MDCM stroji sestrojenému k tomuto PRAMu existuje výpočet, na jehož konci nejsou proměnné x_p, x_q porovnatelné v částečném uspořádání Γ (plyne ze způsobu konstrukce MDCM stroje). Vidíme slabost tohoto MDCM stroje - nedokáže porovnat hodnoty proměnných x_p, x_q, zatímco PRAM počítač dokáže.

U počítače CP-1 a odpovídajícího M-1 stroje nastává tato situace pro $P_k = P_1$, $P_l = P_2$, $P_i = P_1$, $P_j = P_2$, $t_1 = 1$, $t_2 = 2$, $t_3 = 3$.

Druhý a třetí zmíněný způsob mají mnoho společného. Každý z nich je jedním ze dvou speciálních případů obecnějšího způsobu, který oba zahrnuje. Zde jsou uvedeny samostatně zejména z toho důvodu, že při diskutované simulaci je uvažován pouze první z nich.[8]

Některé předpoklady v předcházejících úvahách by bylo možno oslabit, naopak místy by bylo možno vyvodit ještě silnější závěry. Snažili jsme se o zachování co možno největší srozumitelnosti, úsilí o úplnou obecnost by vedlo zřejmě ještě k většímu zatemnění. Proto je necháváme milému čtenáři.

---

[7] identify the mapping ρ_1 with a vector $\vec{k}$, str. 324[B 89]

[8] V [RSSW 88], §2 jsou uvedeny dva algoritmy na řešení uvažovaného problému. Oba pracují v čase $O(\log n)$. Jeden z nich užívá pouze první uvedený způsob přenosu informace, druhý pouze zobecnění druhého a třetího způsobu.

4 Neúspěšný pokus o opravu důkazu

První možnost opravy důkazu člověka možná napadne už u prvního zmíněného příkladu (počítač CP-1 a M-1 stroj). Jedná se o drobnou modifikaci MDCM stroje, která nemá vliv na dolní odhad doby výpočtu tohoto stroje. Spočívá v tom, že procesory MDCM stroje nevolí dvojice $(g, \vec{k})$ nutně při *prvním* jejich použití, ale při libovolném jejich použití, ovšem stále pouze při jediném. Např. kdyby procesor P_1 M-1 stroje zvolil dvojici $(g_1, [x_1])$ ve druhém kroku a procesor P_2 dvojici $(g_1, [x_2])$ ve třetím, bude DG-graf na konci výpočtu úplným grafem na množině všech proměnných a M-1 stroj by problém řešil.

Otázkou tedy je, stačí-li uvažovat pouze jedno vhodně zvolené použití každé dvojice. Jednoduchým způsobem ukážeme, že nestačí. Uvažujme COMMON PRAM počítač CP-4:

COMMON PRAM CP-4: globální paměť indexována od 0 do ∞

		P_1	P_2	P_3	P_4
Krok 1:					
	čte z adresy	x_1	0	x_3	0
	zápis na adr: co zapisuje	$x_1 : 9$	$0 : 1$	$x_3 : 9$	$0 : 1$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$
Krok 2:					
	čte z adresy	0	x_2	x_3	0
	zápis na adr: co zapisuje	$0 : 2$	$x_2 : 8$	$x_3 : 8$	$0 : 1$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$
Krok 3:					
	čte z adresy	x_1	0	0	x_4
	zápis na adr: co zapisuje	$x_1 : 7$	$0 : 3$	$0 : 3$	$x_4 : 7$
	zná hodnoty proměnných	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$

Tento počítač dokáže porovnat rovnost všech dvojic proměnných. Jeho množina zápisových funkcí je redukovatelná na množinu $G = \{g_1(y) = y, g_2() = 0\}$. Uvažujme jeho simulaci na standartním MDCM stroji. Pro tvorbu DG-grafu je důležitou pouze funkce g_1. Jenžto Γ bude při simulaci na konci výpočtu prázdným uspořádáním, potřebujeme, aby DG-graf byl úplným grafem na množině všech proměnných. Protože však výpočet trvá pouze tři kroky, není možné zvolit každou dvojici $(g_1, [x_i])$ v jiném kroku, a tedy není možné docílit toho, aby DG-graf byl na konci výpočtu úplným grafem. Jinými slovy, potřebujeme obarvit vrcholy grafu K_4 třemi barvami (časy), což nelze.

Vidíme, že uvažujeme-li při tvorbě DG-grafu u každé dvojice $(g, \vec{k})$ pouze jeden, byť chytře zvolený, krok použití, dochází při simulaci PRAM počítače na MDCM stroji k podstatné ztrátě informace.

Naskýtají se další otázky. Když nestačí uvažovat jedno použití každé dvojice, stačilo by uvažovat dvě použití či nějaký jiný (konstantní) počet? A je vůbec možné v obecném případě nějaké použití neuvažovat? Opět uvidíme, že odpověď na obě uvedené otázky je záporná. Pomůže nám k tomu již dříve zmíněný (kap. **??**) algoritmus z [RSSW 88] řešící problém různosti n prvků v čase $O(\log n)$. Stručně si ho popišme (předpokládáme n procesorů):

Krok i:

- zápisová fáze: je-li i-tý bit binárního zápisu čísla j roven jedné, zapíše procesor P_j na adresu x_j číslo i.
- čtecí fáze: je-li i-tý bit binárního zápisu čísla j roven nule, čte procesor P_j z adresy x_j. Přečte-li číslo i, ohlásí *"Existují dvě stejná."* a algoritmus skončí.

Pokud do konce kroku $\log n$ žádný procesor nezjistil totožnost dvou čísel, pak v zadaných číslech dvě stejná neexistují. Všimneme si, že na konci výpočtu zná každý procesor stále pouze hodnotu své proměnné.

Zajímá nás, do jaké míry je třeba pozměnit zbůsob výpočtu MDCM stroje, aby na něm bylo možné uvedený algoritmus (či lépe PRAM vykonávající tento algorimus) simulovat. Nyní uvažovaná změna je takováto:

- každá dvojice $(g, \vec{k})$ může být zvolena vícekrát,
- x_i, x_j jsou spojeny hranou v DG-grafu, existují-li dvojice $(g, \vec{k})$, $(g, \vec{h})$ $g \in G$, zvolené v různých krocích tž. vektory $\vec{k}, \vec{h}$ se liší právě na jediné souřadnici (ozn. l) a $k_l = x_i$, $h_l = x_j$.

Při simulaci na takovémto MDCM stroji (po zajištění totožnosti čtecích a zápisových funkcí procesorů) je množina zápisových funkcí procesorů redukovatelná na množinu $G = \{g_1(y) = y, g_2() = 0\}$. Každá dvojice $(g_1, [x_i])$ je použita v polovině kroků, tj. $\log n$ krát. Lze snadno nahlédnout, že neuvažujeme-li při tvorbě DG-grafu byť jediné použití některé dvojice $(g_1, [x_i])$, nebude DG-graf úplným grafem. Tím dostáváme odpověď na naše otázky.

V obecném případě je třeba při tvorbě DG-grafu uvažovat každé použití každé dvojice, resp. je třeba uvažovat u každé dvojice buď alespoň $\log n$ použití, nebo všechna (nutná podmínka vyplývající z uvedeného příkladu). V následující kapitole si ukážeme, že uvažování všech použití každé dvojice naznačeným způsobem je i postačující podmínkou pro úspěšnou simulaci PRAMu na pozměněném MDCM stroji. Žel pro takto modifikovaný MDCM stroj již nejsme schopni dokázat dolní odhad doby výpočtu tak hezký, jak bychom si přáli.

Snaha, uvedená na začátku této kapitoly, o jednoduché opravení simulace PRAM počítače na MDCM stroji při zachování dokazovaného tvrzení se ukázala být neúspěšnou.

5 Oprava důkazu

V minulé kapitole jsme se neúspěšně snažili o opravu důkazu při zachování celého tvrzení. Nyní si ukážeme, kterak "důkaz" pozměnit tak, aby byl opravdu důkazem. Žel tato oprava povede k oslabení dokazovaného tvrzení. Jako dolní odhad časové složitosti problému různosti n prvků na COMMON PRAM počítači s p procesory dokážeme pouze $\Omega\left(\sqrt{\frac{n\log n}{p}}\,\right)$ místo původně dokazovaného dolního odhadu $\Omega(A(n,p))$, kde

$$A(n,p) = \frac{n\log n}{p\log(\frac{n}{p}\log n)}.$$

Pro případ $p = n$ byl jiný důkaz pro dolní odhad $\Omega\left(\sqrt{\log n}\right)$ publikován v roce 1988 v [RSSW 88].

Výpočet PRAMu budeme simulovat na stroji podobném MDCM stroji. Budeme mu říkat Modifikovamý MDCM stroj a označovat ho M-MDCM.

Definice 5.1 *M-MDCM stroj je tvořen p procesory $P_1,\ldots,P_p$, množinou $V = \{x_1,\ldots,x_n\}$ proměnných, se kterými stroj pracuje, a systémem množin C_r, kde indexy množin jsou z nějaké konečné množiny I. Každá množina C_r má svůj rozměr $c(r) \in N$ a platí $C_r \subseteq V^{c(r)}$. Označme $V_t^i \subseteq V$, $1 \le i \le p$ množinu proměnných, jejichž hodnoty jsou známy procesoru P_i v t-tém kroku výpočtu. Na počátku výpočtu platí: $V_0^i = \emptyset$ pro $1 \le i \le p$, $C_r = \emptyset$ pro $r \in I$. V kroku t znají všechny procesory částečné uspořádání Γ množiny V,* *které je sjednocením úplných uspořádání množin V_t^i, $1 \le i \le p$. Na základě Γ zvolí procesor P_i nějakou množinu C_r a nějaký vektor $\vec{a}$ různých proměnných, $\vec{a} \in (V_i^t)^{c(r)}$ (dvojici $(r, \vec{a})$ budeme nazývat přístupový pár). Není-li $\vec{a} \in C_r$, bude do této množiny přidán. Každý procesor P_i, $1 \le i \le p$, dále buď zvolí nějaké j_i, $1 \le j_i \le p$, a do množiny svých proměnných přibere všechny proměnné z $V_t^{j_i}$ (tj. $V_{t+1}^i = V_t^i \cup V_t^{j_i}$), nebo zvolí j_i, $1 \le j_i \le n$ a do množiny svých proměnných přibere pouze proměnnou x_{j_i}, (tj. $V_{t+1}^i = V_t^i \cup \{x_{j_i}\}$). Na konci výpočtu je zkonstruován následujícím způsobem graf DG na množině V. Dvojice $x, y \in V$ je spojena hranou právě tehdy, jestliže existují dvě přístupové dvojice $(r, \vec{a})$, $(r, \vec{b})$, $r \in I$, $\vec{a}, \vec{b} \in C_r$ a kroky výpočtu t_1 a t_2, $t_1 < t_2$ takové, že $(r, \vec{a})$ bylo použito v kroku t_1, $(r, \vec{b})$ bylo použito v kroku t_2 a nebylo použito v kroku t_1 a $\{\vec{a}, \vec{b}\}$ kryje $\{x, y\}$ (tj. vektory $\vec{a}, \vec{b}$ se shodují na všech souřadnicích kromě jedné a tato je pro $\vec{a}$ rovna x a pro $\vec{b}$ rovna y).*

Řekneme, že M-MDCM stroj řeší problém různosti prvků, pokud na konci výpočtu každá dvojice $x, y \in V$ je buď porovnatelná v částečném uspořádání Γ, nebo je spojena hranou v grafu DG.

Uvědomme si, že způsob výpočtu modifikovaného MDCM stroje se příliš neliší od výpočtu běžného MDCM stroje. Rozdíl je ve tvorbě DG-grafu.

Přemýšlejme chvíli o způsobu výpočtu M-MDCM stroje. Všimněme si, že pro libovolné dvě stejně uspořádané vstupní n-tice budou množiny C_r vypadat v každém kroku stejně (jde hezky indukcí).

Uvažujme strom všech možných výpočtů M-MDCM stoje. Každý vrchol je označen množinami V_t^i, $1 \le i \le p$ a množinou právě těch částečných uspořádání, která můžeme dostat sjednocením všech možných úplných uspořádání množin V_t^i. Uvědomme si, že libovolná dvě částečná částečná uspořádání Γ^k, Γ^l z této množiny jsou vzájemně neslučitelná, tj. existuje pro ně dvojice proměnných x, y tž. jejich vztah v obou uspořádáních existuje a je různý. Hrany vycházející z vrcholu jsou označeny vždy jedním Γ^k z dané množiny a popisem, se kterou množinou V_t^j sleje každý procesor P_i svou množinu V_t^i pro toto částečné uspořádání v daném M-MDCM stroji. Kořen stromu je označen prázdným uspořádáním a množinami V_0^i dle definice počátečního stavu M-MDCM stroje. Množiny C_t způsob výpočtu M-MDCM stroje neovlivňují, proto je ve stromu uvažovat nemusíme uvažovat.

5.1 Simulace PRAMu na M-MDCM stroji

Lemma 5.1 *Existuje-li COMMON PRAM s p procesory řešící problém různosti n prvků v čase T, pak existuje M-MDCM stroj s p procesory řešící tento problém v čase 2T.*

Důkaz: Uvažujme PRAM splňující předpoklady věty, pracující na množině proměnných V s oborem hodnot D, $D = \mathcal{N}$. Zdvojnásobením počtu kroků výpočtu můžeme předpokládat totožnost čtecí a zápisové funkce každého procesoru v každém kroku, tj. $W_{i,j} = R_{i,j}$ pro všechna i, j. Uvažujeme PRAM, který v každém kroku nejprve čte a který zapisovanou hodnotu zvolí již na počátku kroku. Položme

$$F = \{W_{i,j} | 1 \leq i \leq p, 1 \leq j \leq 2T\}$$

Dle *lemma4.2*[B 89] existuje množina $C, C \subseteq D$ taková, že $|C| \geq n$ a $F|_C$ je redukovatelná nějakou na množinu G prostých funkcí s disjuntními obory hodnot. M-MDCM stroj sestrojíme následujícím způsobem. Množiny C_r budeme indexovat funkcemi z množiny G, položme $c(g) = |var(g)|$ pro $g \in G$. Konstrukci stroje popíšeme indukcí. Počáteční stav je dán definicí M-MDCM stroje, tj. $V_0^i = \emptyset$, $1 \leq i \leq p$. Inicializační krok provedeme stejně jako při snaze o simulaci PRAMu na MDCM stroji, tj. procesor P_i, pro $1 \leq i \leq n$, sjednotí svou množinu proměnných (v tuto chvíli ještě prázdnou) s množinou $\{x_i\}$, procesor P_i, pro $n < i \leq p$, slévací krok přeskočí.

Předpokládejme dále, že máme činnost stroje popsanou až do kroku t výpočtu, popíšeme krok $t + 1$. Jde o to určit, co má procesor P_i udělat v případě částečného uspořádání Γ. Vezměme nějaký vektor $\hat{x} \in C^V$ s různými souřadnicemi, konzistentní s Γ (pro všechny procesory stejný) a nechme M-MDCM stroj provést pro tento vstup prvých t kroků. Položme $O = \{\hat{y} | \hat{y} \in C^V, \hat{y} \equiv \hat{x}\}$. Uvažujme libovolný procesor P_i, $1 \leq i \leq p$. Dle definice redukovatelnosti existují funkce $g \in G$ a $\rho : var(g) \to var(W_{i,t})$ takové, že $W_{i,t} = g^\rho$. Nechť $\vec{k}$ označuje vektor proměnných z V odpovídajících zobrazení ρ (viz. str. 16, poznámka ..). Procesor P_i zvolí přístupový pár $(g, \vec{k})$. Je-li $\vec{k} \in C_g$, sjednotí procesor P_i svou množinu proměnných V_t^i s množinou V_t^j, kde P_j je poslední procesor, který v některém z dosud provedených kroků zvolil přístupový pár $(g, \vec{k})$. Pokud je takových procesorů více, vezmeme ten s nejmenším id. číslem. Není-li $\vec{k} \in C_g$, bude vektor $\vec{k}$ do množiny

C_g přidán a $V^i_{t+1} = V^i_t$ (tj. P_i sjednotí svou množinu proměnných se svou množinou proměnných). Tímto je ukončen popis simulace, zbývá dokázat, že sestrojený M-MDCM stroj opravdu daný problém řeší. Pomůže nám k tomu následující *lemma*.

Lemma 5.2 *Pro každé $\hat{x} \in C^V$ s různými souřadnicemi, $t \leq 2T$ platí: $U^i_t \subseteq V^i_t$, kde U^i_t, resp. V^i_t, je množina proměnných známých procesoru P_i PRAMu, resp. odpovídajícího M-MDCM stroje, pro vstup $\hat{x}$ na konci kroku t.*

Důkaz: Užijeme indukci. Případ $t = 0$ je triviální. Postoupíme od t k $t+1$. Pro $\hat{x} \in C^V$ uvažujme vrchol ve stromu možných výpočtů odpovídající kroku t M-MDCM stroje (uvědomme si, že tento vrchol je jednoznačně určen) a uvažujme Γ to částečné uspořádání z množiny částečných uspořádání daného vrcholu, se kterým je $\hat{x}$ konzistentní. Všimněme si, že funkce $W_{i,t}$, $1 \leq i \leq p$, pro vstupy z C^V s různými souřadnicemi, konzistentní s Γ, závisí pouze na proměnných z množiny V^i_t (ne nutně na všech) a tato je v Γ úplně uspořádaná. Označme $(g, \vec{k})$ přístupovou dvojici zvolenou procesorem P_i M-MDCM stroje v tomto kroku pro vstup konzistentní s Γ, P_j ten procesor, s jehož V^j_t slévá P_i svou množinu V^i_t a $t' \leq t$ krok, ve kterém P_j přístupový pár $(g, \vec{k})$ naposledy použil. Pro t' užijeme indukční předpoklad a dále užijeme znalostí o způsobu výpočtu M-MDCM stroje. Platí: $U^j_{t'} \subseteq V^j_{t'} \subseteq V^j_t$. Nyní uvažujme výpočet PRAMu v kroku t pro vstup $\hat{x}$. Procesor P_i přečte obsah buňky, do které v kroku $t'' \leq t$ zapsal nějaký procesor P_k funkcí $W_{k,t''} = W_{i,t}$. Označme U množinu proměnných, jejichž hodnoty P_k do buňky zapsal. Uvědomme si, že $t' = t''$. Z $U \subseteq U^j_{t'}$ a $U^j_{t'} \subseteq V^j_t$ dostáváme $U \subseteq V^j_t$ a tedy $U^i_{t+1} \subseteq V^i_{t+1}$, čímž je důkaz indukčního kroku i celého tvrzení dokončen.

Předpokládejme sporem, že M-MDCM stroj zkonstruovaný popsaným způsobem k danému PRAMu problém různosti prvků v čase $2T$ nevyřeší. Pak ve stromu všech možných výpočtů existuje list s takovým částečným uspořádáním Γ a grafem DG, pro které platí: existují $x_i, x_j \in V$ tž. x_i, x_j nejsou porovnatelné v Γ a zároveň x_i, x_j nejsou spojeny hranou v grafu DG. Uvažme vektor $\hat{x} \in C^V$ s různými souřadnicemi konzistentní s Γ takový, že pro $k \neq i, j$ platí, že $\hat{x}_k$ neleží mezi hodnotami $\hat{x}_i, \hat{x}_j$, a dále vektor $\bar{x}$ tž. $\bar{x}_k = \hat{x}_k$ pro $k \neq j$, $\bar{x}_j = \hat{x}_i$. S pomocí předcházejícího *lemma* dostáváme, že PRAM tyto dva vstupy nedokáže rozlišit, což je spor. Tedy zkonstruovaný M-MDCM stroj úlohu vskutku vyřeší.

5.2 Dolní odhad doby výpočtu M-MDCM stroje

Zaveďme si nejprve některé pojmy, které budeme dále používat. Většina z nich se vyskytuje v článku [B 89], přesto je pro snazší čtení následujícího textu zopakujume.

Nechť X je náhodná veličina, nabývající konečného počtu hodnot $x_1, ..., x_m$ s pravděpodobnostmi $p_1, ..., p_m$, $\sum_{j=1}^{m} p_j = 1$. Podobně nechť Y je též náhodná veličina nabývající konečného počtu hodnot $y_1, ..., y_n$ s pravděpodobnostmi $r_1, ..., r_n$, $\sum_{j=1}^{n} r_j = 1$. Označme $q_{j,k} = P(Y = y_k | X = x_j)$. Dále zaveďme funkci $L : (0, \infty) \to R$,

$$L(x) = (x + 1) \log_2(x + 1) - x \log_2 x.$$

Definice 5.2 *Entropie*[9] $H(X)$ *náhodné veličiny* X *je definována vztahem*

$$H(X) = \sum_{j=1}^{m} -p_j \log_2 p_j.$$

Jsou-li X, Y náhodné veličiny, pak uspořádaná dvojice (X, Y) je též náhodná veličina. Její entropii $H((X, Y))$ budeme stručně označovat $H(X, Y)$. Námi nejvíce užívanou vlastností entropie je její subadivita

$$H(X, Y) \leq H(X) + H(Y).$$

Pro důkaz tohoto tvrzení viz. Příloha B.

Entropií funkce f s konečným oborem hodnot rozumíme $H(f) = H(f(v))$, kde v je náhodná veličina s rovnoměrným rozdělením hodnot na definičním oboru funkce f. Pro každou takovou funkci platí tento jednoduchý odhad

$$0 \leq H(f) \leq \log |(dom(f)|.$$

Definice 5.3 *: Vnořením* [10] *částečného uspořádání označujeme zobrazení částečně uspořádané množiny do nějaké úplně uspořádané množiny při zachování uspořádání.*

[9] entrophy
[10] layering

Nechť l je vnořením částečně uspořádané množiny V, nechť V_i, V_j jsou úplně uspořádané podmnožiny V.

Definice 5.4 *Srovnávací graf*[11] $C_l(V_i, V_j)$ *je definován množinou vrcholů* V *a množinou hran* E, $E = \{(a,b)|\ a \in V_i, b \in V_j, a \neq b, l(a) = l(b)\}$.

Protože V_i, V_j jsou úplně uspořádané podmnožiny V, odpovídá graf nějakému párování na množině V.

Definice 5.5 *Pro funkci* $f : A \to B$, *kde* $A \subseteq V^{dim(f)}$, B *je libovolná množina, definujme graf* $K(f)$ *funkce* f, $K(f) = (V, E)$, *kde*

$$E = \{(a,b)|\ \exists \vec{a}, \vec{b} \in A \ \ f(\vec{a}) \neq f(\vec{(b)}\ \&\ (\vec{a}, \vec{b})\ kryje\ (a,b)\}.$$

Je-li $dim(f) = 1$, *budeme tento někdy graf označovat* $K_u(f)$. *Pro množinu funkcí* $F = \{f|\ f : A_f \to B_f, kdeA_f \subseteq V^{dim(f)}\}$ *definujeme graf* $K(F)$ *jako sjednocení grafů* $K(f)$, $f \in F$. *Je-li pro každou* $f \in F$ $dim(f) = 1$, *budeme graf někdy pro zdůraznění této skutečnosti označovat* $K_u(F)$.

Definice 5.6 *Pro funkci* f *a množinu funkcí* F *uvedených vlastností definujme dále*

$$cost(f) = \frac{|dom(f)|}{|V|} H(f),$$

$$profit(f) = \frac{|dom(f)|^{\frac{3}{2}}}{|V|} \sqrt{2dim(f)},$$

$$cost(F) = \sum_{f \in F} cost(f),$$

$$profit(F) = \sum_{f \in F} profit(f).$$

Definice 5.7 *Pro graf* G *na* V *definujme*

$$content(G) = \min_{K_u(F) \supseteq G} cost(F).$$

[11] comparison graph

Lemma 5.3

$$content(F) \leq cost(F) + profit(F).$$

Důkaz: bez, viz. Lemma 4.5 [B 89].

Lemma 5.4 *Každý M-MDCM stroj s p procesory potřebuje k řešení problému různosti n prvků, $n \leq p$, čas T, kde*

$$T = \Omega\left(\sqrt{\frac{n \log n}{p}}\right).$$

Důkaz: Předpokládejme $p \leq n \log n$, v opačném případě je dolní odhad triviální. Pro daný M-MDCM stroj popíšeme jednu z možných cest jeho výpočtu délky alespoň $\sqrt{\frac{n}{p}\log n}$. Popíšeme ji induktivně, začneme v kořenu stromu všech možných výpočtů a dále pro každý vrchol, do kterého se dostaneme a který není listem, zvolíme vhodné pokračování cesty. Pro každé částečné uspořádání Γ odpovídající hraně cesty vycházející z t-tého vrcholu cesty popíšeme jeho vnoření l_t následujících vlastností ($m = 9 \log p$):

(A) $H(l_t) \leq (L(\frac{pm}{n}) + 3))t$
(B) $\forall\ 1 \leq i < j \leq p : |C_{l_t}(V_t^i, V_t^j)| \leq m$

Vezměme l_0 triviální vnoření, platnost (A), (B) je pro l_0 zřejmá. Nyní máme sestrojený počáteční úsek cesty délky t, chceme ho prodloužit. Nechť $V_t^{j_i}, 1 \leq i \leq p$ označuje množinu, kterou v kroku t zvolil procesor P_i ke slití se svou množinou. Nechť G označuje graf vzniklý sjednocením grafů $C_{l_t}(V_t^i, V_t^{j_i})$. Díky platnosti (B) v čase t má graf G nejvýše pm hran. Užitím *lemma 3.4.[**?**]* získáme obarvení χ grafu G , tž. $H(\chi) \leq L(\frac{pm}{n})$. Barvy χ libovolně úplně uspořádáme. Definujeme $l' = (l_t, \chi)$, přičemž obor hodnot l' lexikálně utřídíme. l' je vnořením částečného uspořádání Γ v kroku $t+1$, ale neplatí pro něj vlastnosti (A),(B). Je třeba l' ještě upravit.

Graf $C_{l'}(V_{t+1}^i, V_{t+1}^j)$ má díky (B) nejvýšše $4m$ hran pro každé $1 \leq i < j \leq p$. Ukážeme existenci funkce $f : V \to \{1, 2, ..., 8\}$ tž. pro $l'' = (l', f)$ s lexikálně uspořádaným oborem hodnot má graf $C_{l''}(V_{t+1}^i, V_{t+1}^j)$ maximálně m hran pro každé $1 \leq i < j \leq p$. Zvolme náhodnou funkci f, $f : V \to \{1, 2, ..., 8\}$ (všechny 8^n možnosti jsou stejně pravděpodobné) a uvažujme funkci $l'' = (l', f)$ s lexikálně uspořádaným oborem hodnot. Jenžto $C_{l'}(V_{t+1}^i, V_{t+1}^j)$

je bipartitní graf, jsou události "hrana je jednobarevná" pro všechny hrany vzájemně nezávislé. Pravděpodobnost jednobarevnosti libovolné hrany je $\frac{1}{8}$. S použitím Stirlingových vzorců dostáváme

$$Pr(|C_{l''}(V^i_{t+1}, V^j_{t+1})| \geq m) \leq \binom{4m}{m}\left(\frac{1}{8}\right)^m \left(\frac{7}{8}\right)^{3m} \leq$$

$$\leq 4^m \left(\frac{4}{3}\right)^{3m} \left(\frac{1}{8}\right)^m \left(\frac{7}{8}\right)^{3m} = \left(\frac{343}{432}\right)^m \leq \frac{1}{p^2}.$$

Proto

$$Pr(\exists i < j, |C_{l''}(V^i_{t+1}, V^j_{t+1})| \geq m) \leq \binom{p}{2}\frac{1}{p^2} < 1,$$

tedy slibovaná funkce f existuje. Všimneme si, že $H(f) = 3$. Zvolíme-li $l_{t+1} = l''$, zbývá dokázat platnost (A). Díky subadivitě entropie platí požadované

$$H(l_{t+1}) \leq H(l_t) + H(\chi) + H(f) \leq (t+1)\left(L\left(\frac{pm}{n}\right) + 3\right).$$

Jako pokračování cesty zvolíme hranu označenou takovým částečným uspořádáním, že l_{t+1} je jeho vnořením, uvědomme si, že existuje právě jedna taková hrana. Tímto je dokončen popis indukčního kroku. Zbývá ukázat, že tímto způsobem najdeme cestu požadované délky.

Označme CG graf odpovídající částečnému uspořádání na konci naší cesty. Jenžto l_T je obarvením grafu CG, dostáváme

$$content(CG) \leq H(l_T) \leq \left(L\left(\frac{9p\log p}{n}\right) + 3\right) T.$$

Uvažujme množiny C_r na konci výpočtu a vytvořme množinu funkcí B těchto vlastností: Ke každé množině C_r existuje funkce $h \in B$ definovaná na prvcích množiny C_r, pro $\vec{a} \in C_r$

$$h(\vec{a}) = \{t | \; \exists \; procesor, \; který \; v \; kroku \; t \; použil \; dvojici \; (r, \vec{a})\}.$$

Uvědomme si, že pro B platí $K(B) \supseteq DG$. Protože každý procesor v každém kroku zapříčiní vložení maximálně jednoho prvku do některé z množin C_r, můžeme odhadnout $\sum_{h\in B} |dom(h)| \leq pT$. S užitím $H(h) \leq T$, $h \in B$, dostáváme

$$cost(B) = \sum_{h\in B} cost(h) = \sum_{h\in B} \frac{|dom(h)|}{|V|} H(h) \leq \frac{pT^2}{n}.$$

Z $|V_t^i| \leq 2^t$ plyne $dim(h) \leq 2^T$ pro každé $h \in B$. Podobně jako $cost(B)$ odhadneme i $profit(B)$

$$profit(B) = \sum_{h\in B} profit(h) = \sum_{h\in B} \frac{|dom(h)|^{\frac{3}{2}}}{|V|} \sqrt{2|c(h)|} \leq$$
$$\leq \frac{\sqrt{2.2^T}}{n} \sum_{h\in B} |dom(h)|^{\frac{3}{2}} \leq \frac{(pT)^{\frac{3}{2}}}{n} \sqrt{2^{T+1}}.$$

Užitím *lemma 5.3* dostáváme

$$content(DG) \leq \frac{pT^2}{n} + L\left(\frac{(pT)^{\frac{3}{2}}}{n}\sqrt{2^{T+1}}\right)$$

Protože dle předpokladu M-MDCM stroj problém různosti prvků řeší, platí $DG \cup CG = K_V$, kde K_V je úplný graf na množině V. V [?] Fredman a Komlós ukázali, že $content(K_n) = \log_2 n$. Tedy

$$\log_2 n \leq \frac{pT^2}{n} + L\left(\frac{(pT)^{\frac{3}{2}}}{n}\sqrt{2^{T+1}}\right) + T\left(L\left(\frac{9p\log p}{n}\right) + 3\right).$$

Protože první člen na pravé straně asymptoticky překračuje ostatní dva, dostáváme požadované

$$T = \Omega\left(\sqrt{\frac{n\log n}{p}}\right).$$

Všimněme si ještě, že pro tento výsledek by nám stačil i hrubší odhad velikosti $content(CG)$.

6 Varianta důkazu

Uveďme si ještě jiný důkaz pro stejný dolní odhad. Tento důkaz užívá některé techniky použité v [RSSW 88] a [?], záverečná část je v podstatě shodná s předchozím důkazem. Základní myšlenkou je opět omezení se pouze na jisté podmnožiny všech možných vstupů, pro které bude snazší popsat výpočet PRAM počítače. K popisu těchto podmnožin nám pomůže definice z [RSSW 88].

Definice 6.1 *Nechť A je množina n-tic délky l se složkami z množiny X, nechť B je jiná taková množina. Pokud $A \cap B = \emptyset$, nazýváme $\{A, B\}$ tuple-systémem délky l*[12]*. Kolekce tuple-systémů C je množina $C = \{\{A_1, B_1\}, ...\{A_k, B_k\}\}$, kde $\{A_i, B_i\}$ je tuple-systém délky l_i.*

Lemma 6.1 *Každý COMMON PRAM s p procesory potřebuje k vyřešení problému různosti n prvků čas T, kde*

$$T = \Omega\left(\sqrt{\frac{n \log n}{p}}\right).$$

Důkaz: Pro daný PRAM popíšeme induktivně až do indexu T množiny vstupů I_t takové, že pro libovolný vstup z množiny I_t trvá výpočet slespoň t kroků. Množina I_t je určena množinou S_t, zobrazením l_t množiny V do nějaké úplně uspořádané množiny a tuple-systémem C_t tak, že $I_t = \{\hat{x} \mid \hat{x} \in S_t^n \;\&\hat{x}$ je konzistentní s l_t & max. dvě souřadnice $\hat{x}$ jsou stejné $\&\hat{x}_i = \hat{x}_j \Rightarrow (x_i, x_j)$ není kryta žádným tuple-systémem z $C_t\}$.

Předpokládejme, že $W_t^i = R_t^i,\ \forall\ i, t$ a že zapisovanou hodnotu i adresu zápisu zvolí každý procesor již na počátku kroku.

Označme V_t^i množinu proměnných, jejichž hodnoty mohou ovlivňovat stav procesoru P_i na konci kroku t. Popisované objekty budou splňovat následující vlastnosti, pro $m = 9 \log p$

1. $S_t \subseteq S_{t-1}$
2. l_t je zjemněním l_{t-1} a platí:
   (A) $H(l) \leq \left(L(\frac{pm}{n})\right)$
   (B) $\forall\ 1 \leq i < j \leq p : |C_{l_t}(V_t^i, V_t^j)| \leq m$
3. $C_t \supseteq C_{t-1}$
4. $\forall\ t, i\ \forall\ x, y\ \in V_t^i :\ l_t^i(x) \neq l_t^i(y)$

[12] tuple system of lenght l

5. Činnost procesoru P_i v kroku $t+1$ závisí pro všechny vstupy z I_t pouze na proměnných z V_t^i

Díky těmto vlastnostem je zajištěno $I_t \subseteq I_{t-1}$.

Položme $S_0 = \mathcal{N}$, $C_0 = \emptyset$, za l_0 vezměme triviální zobrazení , tj. $\forall\, x, y \in V : l_0(x) = l_0(y)$. Dle definice počátečního stavu PRAM počítače $V_0^i = \{x_i\}$. Splnění všech podmínek je tedy pro $t = 0$ zajištěno. Postupme k indukčnímu kroku. Popíšeme všechny potřebné objekty pro index $t+1$.

Vezměme

$$F = \{W_i^{t'} \mid 1 \leq t' \leq t+1,\ 1 \leq i \leq p\}.$$

Dle lemma4.2[B 89] existují nekonečná množina $S' \subseteq S_t$ a množina G prostých funkcí s disjunktními obory hodnot takové, že $F|_{S'}$ je redukovatelná na G. Pro každou třídu uspořádání O konzistntní s l_t a každou funkci W_{t+1}^i existují $g \in G$, $\rho : var(g) \to var(W_{t+1}^i)$ tak, že $W_{t+1}^i|_O = g^\rho$. Díky skutečnosti, že $W_{t+1}^i|_{I_t}$ záleží pouze na proměnných z V_t^i, které jsou navíc v l_t úplně uspořádané, budou funkce g a ρ k $W_{t+1}^i|_{I_t}$ stejné pro všechny třídy uspořádání konzistentní s l_t. Pro $g \in G$ můžeme tedy smysluplně definovat

$$\begin{aligned}
R &= \{P_i \mid \exists \rho :\ W_{t+1}^i|_{I_t} = g^\rho\},\\
W &= \{P_i \mid \exists \rho,\ t' \leq t :\ W_{t'}^i|_{I_t} = g^\rho\},\\
A &= \{\vec{k} \mid \exists i,\ t' \leq t :\ W_{t'}^i|_{I_t} = g^\rho \textit{ a } \vec{k} \textit{ odpovídá } \rho\},\\
B &= \{\vec{k} \mid \exists i :\ W_{t+1}^i|_{I_t} = g^\rho \textit{ a } \vec{k} \textit{ odpovídá } \rho\}.
\end{aligned}$$

Položme $S_{t+1} = S'$, $l_{t+1} = l_t$, $C_{t+1} = C_t$. Probírejme postupňe všechny funkce $g \in G$ a pro každou z nich definujme výše uvedeným způsobem množiny A, B, R, W. Tuple-systém $\{A, B\}$ přidáme do C_{t+1}.

Pro každé $P_j \in R$ definujme dále $W_j = \{P_i \mid P_i \in W\ \&\exists t' \leq t :\ W_{t'}^i|_{I_t} = W_{t+1}^j|_{I_t}\ \&\forall t'', t' < t'' \leq t \forall k, 1 \leq k \leq p :\ W_{t''}^k|_{I_t} \neq W_{t+1}^j|_{I_t}\}$. Nechť t' označuje poslední krok před krokem $t+1$, kdy procesory z množiny W_j použili k zápisu funkci $W_{t+1}^j|_{I_t}$.Položme

$$U_{t+1}^j = \bigcap_{P_i \in W_j} V_{t'}^i.$$

Na základě předchozích úvah lze usoudit

$$V_{t+1}^j = V_t^j \cup U_{t+1}^j.$$

V tuto chvíli máme dostatečným způsobem popsáno S_{t+1} a C_{t+1}, zbývá upravit l_{t+1} tak, aby zachovávalo platnost (A),(B). Vzhledem k tomu, že konstrukce l_{t+1} lze provést stejným způsobem, jako v důkazu Lemma ??9 ıMdolní nebudeme ji zde opakovat. Tímo je ukončen popis indukčního kroku.

Položme $CG_t = K(l_t)$. Pro kolekci $C_t = \{\{A_1, B_1\}, ...\{A_m, B_m\}\}$ definujme graf $DG_t = (V, E_t)$, kde

$$E_t = \{(x_i, x_j) | \; \exists \, \vec{a}, \vec{b}, k : \; \vec{a} \in A_k, \vec{b} \in B_k, \; (A_k, B_k) \in C_t \; \& \; (\vec{a}, \vec{b}) \; kryje \; (x_i, x_j)\}.$$

Nutnou podmínkou k tomu, aby byl problém daným PRAMem vyřešen v čase T, je $DG_T \cup CG_T = K_V$ (jinak najdeme nerozlišitelné vstupy). K odhadu T nám pomůže množina B níže popsaná. Pro každé $g \in G$ definujeme funkci h_g těchto vlastností (uvažujeme G použité v kroku T):

$$\begin{aligned} dom(h_g) &= \{\vec{k} | \; \exists \, i, t : \; W_t^i|_{I_T} = g^\rho \; a \; \vec{k} \; odpov\acute{\imath}d\acute{a} \; \rho\}, \\ h_g(\vec{k}) &= \{t | \; \exists \, i : \; W_t^i|_{I_T} = g^\rho \; a \; \vec{k} \; odpov\acute{\imath}d\acute{a} \; \rho\}. \end{aligned}$$

Pak pro $B = \{h_g | \; g \in G\}$ platí $DG \subseteq K(B)$.

Zbytek důkazu už je vpodstatě shodný s postupem v předcházející variantě důkazu. Opět užijeme rovnosti $content(K_n) = \log_2 n$.

7 Závěr

Na závěr dodejme, že několik málo týdnů před ukon/vcením práce dubna se nám po upozornění pana F. Meyera Auf Der Heide podařilo získat další článek týkající se relativní síly COMMON a PRIORITY modelů [E 91]. Edmonds v něm pro složitost problému různosti n prvků na COMMON modelu s p procesory dokazuje stejný dolní odhad, o který se snažil Boppana, a to dokonce pro případ, kdy vstupní čísla problému nejsou z nekonečné množiny, ale pouze z množiny velikosti $2^{2^{\Omega(n)}}$. Tím dává kromě jinéno odpověď i na námi nezodpovězenou otázku po skutečném vztahu COMMON a PRIORITY modelů – jeden krok PRIORITY s n procesory vyžaduje při simulaci na COMMON s p procesory $\Theta(\frac{n \log n}{p \log(\frac{n}{p} \log n + 1)})$ kroků.

References

[SV 82] Shiloach Y. and Vishkin U.: An $O(\log n)$ Parallel Connectivity Algorithm, J. Alg. 3, 1982, str. 57-67

[K 82] Kučera L.: Parallel Computation and Conflicts in Memory Access, Inform. Process. Lett., 14, 1982, str. 93-96

[CV 86] Cole R. and Vishkin U.: Deterministing Coin Tossing and Accelerating Cascades: Micro and Macro Techniques for Designing Parallel Algorithms, Proc. 18th STOC, 1986, str. 206-219

[GR 87] Grolmusz V. and Ragde P. L.: Incomparability in Parallel Computation, Proc. 28th FOCS, 1987, str. 89-98

[BDH 92] Bast H., Dietzfelbinger M., Hagerup T.: A Perfect Parallel Dictionary, Proc. 17th Inter. Symp. on Mathamatical Foundations of Computer Science, 1992, Springer Lecture Notes in Computer Science, Vol.629, str. 133-141

[FRW 88] Fich F. E., Ragde P. L., Wigderson A.: Relations Between Concurrent-Write Models of Parallel Computation, SIAM J. Comp. 17, 1988, str. 606-627

[CDHR 88] Chlebus B. S., Diks K., Hagerup T., Radzik T.: Efficient simulations between Concurrent-Read Concurrent-Write PRAM models, Proc 13th Inter. Symp. on Mathamatical Foundations of Computer Science, 1988, Springer Lecture Notes in Computer Science, Vol.324, str. 231-239

[FRW 88-2] Fich F. E., Ragde P. L., Wigderson A.: Simulations among Concurrent-Write PRAMs, Algorithmica 3, 1988, str. 43-52

[RSSW 88] Radge P. L., Steiger W., Szemeredi E., Wigderson A.: The Parallel Complexity of Element Distinctness is $\Omega(\sqrt{\log n})$, SIAM J. Disc. Math. 1, 1988, str. 399-410

[B 89] Boppana R.V.: Optimal Separations Between Concurrent-Write Parallel Machines, Proc. 21th ACM Symp. on Theory of Computing, 1989, str. 320-326

[FMW 86] Fich F. E., Meyer Auf Der Heide F., Wigderson A.: Lower Bounds for Parallel Random-Access Machines with Unbounded Shared Memory, Advances in Computing Research 4, 1986, str. 1-15

[FK 84] Fredman M., Komlós J.: On the size of separating systems and families of perfect hash functions, SIAM J. Alg. Disc. Meth. 5, 1984, str. 61-68

[MW 85] Meyer Auf Der Heide F., Wigderson A.: The Comlexity of Parallel sorting, 26th FOCS, 1985, str. 532-540

[CK 82] Csiszár I. and Körner J.: Information Theory: Coding Theorems for Discrete Memoryless Systems, Academic Press, 1982

[AD 75] Aczél J., Daróczy Z.: On Measures of Information and Their Characterization, Academic Press, New York, 1975

[E 91] Edmonds J.: Removing Ramsey Theory: Lower Bounds with Smaller Domain Size, Structures in Complexity Theory, 1991

[ER 50]

A Subadivita entropie

Zaveďme nejprve označení, které budeme dále používat. Nechť X je náhodná veličina, nabývající konečného počtu hodnot $x_1, ..., x_m$ s pravděpodobnostmi $p_1, ..., p_m$ tž. $\sum_{j=1}^{m} p_j = 1$. Podobně nechť Y je též náhodná veličina nabývající konečného počtu hodnot $y_1, ..., y_n$ s pravděpodobnostmi $r_1, ..., r_n$ tž. $\sum_{j=1}^{n} r_j = 1$. Označme $q_{j,k} = P(Y = y_k | X = x_j)$. Dále zaveďme funkci $l(x) = -x \log x$ pro $x \in (0, 1\rangle, \quad l(0) = 0$.

Entropie $H(X)$ náhodné veličiny X je definována vztahem

$$H(X) = \sum_{j=1}^{m} -p_j \log_2 p_j.$$

Jsou-li X, Y náhodné veličiny, pak uspořádaná dvojice (X, Y) je též náhodná veličina. Její entropii $H((X, Y))$ budeme stručně označovat $H(X, Y)$.

Tvrzení: Pro náhodné veličiny X, Y nabývajícího konečného počtu hodnot platí:

$$H(X, Y) \leq H(X) + H(Y).$$

Důkaz:

$$H(X, Y) = \sum_{j=1}^{m} \sum_{k=1}^{n} -p_j q_{j,k} \log(p_j q_{j,k}) =$$

$$= \sum_{j=1}^{m} -p_j \log p_j \sum_{k=1}^{n} q_{j,k} + \sum_{j=1}^{m} p_j \sum_{k=1}^{n} -q_{j,k} \log q_{j,k}$$

Protože $\sum_{k=1}^{n} q_{j,k} = 1$, je prvý výraz roven $H(X)$. K odhadu velikosti druhého výrazu nám poslouží následující lemma:

Lemma: Pro $s_k \in (0, 1)$, $k = 1, .., n$ a pro $t_j \geq 0$, $j = 1, .., n$ tž. $\sum_{j=1}^{n} t_j = 1$ platí:

$$\sum_{k=1}^{n} t_k l(s_k) \leq l(\sum_{k=1}^{n} t_k s_k).$$

Důkaz: Položme $\hat{s} = \sum_{k=1}^{n} t_k s_k$, pak $\hat{s} \in (0, 1)$. Všimněme si, že funkce l má všude na $(0, 1)$ druhou derivaci a platí $l''(x) \leq 0$ pro $x \in (0, 1)$. Užitím Taylorova rozvoje dostaneme

$$l(s_k) = l(\hat{s}) + l'(\hat{s})(s_k - \hat{s}) + \frac{1}{2} l''(\zeta_k)(s_k - \zeta_k)^2$$

pro $k = 1, ..., n$. Vynásobíme k-tou rovnici číslem t_k a všechny rovnice sečteme:

$$\sum_{k=1}^{n} t_k l(s_k) = l(\hat{s}) + l'(\hat{s}) \sum_{k=1}^{n} t_k (s_k - \hat{s}) + \frac{1}{2} \sum_{k=1}^{n} t_k l''(\zeta_k)(s_k - \hat{s})^2$$

Všimněme si, že druhý člen je roven nule a třetí není větší než nula, tedy

$$\sum_{k=1}^{n} t_k l(s_k) \leq l(\sum_{k=1}^{n} t_k s_k)$$

Aplikací lemma získáme:

$$\sum_{k=1}^{m} p_j l(q_{j,k}) \leq l(\sum_{k=1}^{m} p_j q_{j,k})$$

pro $k = 1, ..., n$. Dostáváme:

$$\sum_{j=1}^{m} p_j \sum_{k=1}^{n} -q_{j,k} \log q_{j,k} = \sum_{j=1}^{m} p_j \sum_{k=1}^{n} l(q_{j,k}) \leq$$

$$\leq \sum_{k=1}^{n} l(\sum_{j=1}^{m} p_j q_{j,k}) = \sum_{k=1}^{n} l(r_k) = H(Y).$$

Další vlastnosti entropie lze nalézt např. v [FK 84] nebo [AD 75].