

Determinarea paralelismului între operații elementare în cadrul structurilor numerice.

- La nivelul unității de comandă a structurii numerice, e necesar să se implementeze controlul concurent al op. elementare.
- Obiectivul problemei de determinare a microoperațiilor paralele constă în a stabili, pe baza:
 - grafului de dependențe de date și
 - a cererilor de resurse,seturile de microoperații ce pot controla simultan resurse diferite ale sistemului.
- Obs: Conceptele și algoritmi prezentati pot fi aplicati și la execuția sarcinilor sau a task-urilor.

Definitii

Def.1 Se numește microsubbloc μSB un subset maxim al unui set de microoperații ce constituie o secvență indivizibilă cu un singur punct de intrare și un singur punct de ieșire și conține maximum o microoperație de salt, ca ultimă operație din secvență.

➤ $\mu SB(\mu o) = (\mu B(\mu o), <) \text{ unde}$

➤ $\mu B(\mu o) = \{\mu o_1, \mu o_2, \mu o_3, \dots, \mu o_{|\mu B|}\}$

➤ $<$ relatie de ordine nereflexiva

➤ **Def.2 μOd** – microoperatie disponibila- acea microoperatie pentru care toate microoperatiile de care ea era dependenta de date au fost asigurate unor microinstrucțiuni.

$$\mu O_d = \{\mu O_j / \forall \mu O_i \angle \mu O_j \Rightarrow \mu O_i \in \mu I\}$$

➤ **Def. 3** Prin nivel într-un subbloc μON_k înțelegem toate microoperatiile independente situate la aceeași distanță (adâncime) față de nodul inițial.

Def.4 Setul de operatii disponibile la nivelul k , μOD_k , este construit din multimea de microoperatii disponibile din cadrul asociat microoperatiilor la nivelul k de generare de seturi paralele de microoperatii.

PROP.1. Microoperatiile din seturile de microoperatii disponibile sunt independente.

Presupunem ca nu ar fi independente. Daca avem

$$\mu O_i \in \mu OD_k \quad \mu O_j \in \mu OD_k$$

si nu sunt independente, atunci fie $\mu O_i < \mu O_j$ sau $\mu O_j < \mu O_i$.

Conform Def 2 $\forall \mu O_l < \mu O_j \quad \mu O_j \in \mu OD_k$

rezulta ca μO_l a fost asignata (atribuita) unei microinstructiuni.

Rezulta ca μO_l nu poate face parte din μOD_k si singura solutie este ca ele sa fie independente.

$|\mu ON|$ - numarul de niveluri dintr-un microsubloc egale cu timpul minim de executie pentru microsubblocul respectiv

Fiecare nivel va specifica toate microoperatiile paralele care se desfasoara la un anumit moment de timp.

nm μ l- numarul minim de microinstructiuni ce ar putea fi generate intr-un microsubloc

Evaluare estimativa:

➤ $nm_{\mu I} = \frac{|\mu OB(\mu O)|}{nR}$ - multimea de microoperatii din bloc

➤ unde:

nR – *numarul de resurse ale sistemului*

$\mu CR = \emptyset$ conflictul intre resurse

$nM_{\mu I}$ – *numarul maxim de microinstructiuni*

$nM_{\mu I} = |\mu OB(\mu O)|$ - toate microoperatiile se executa secvential

$nm_{\mu I}$, $nM_{\mu I}$ sunt cazuri extreme.

$nO_{\mu I}$ - *numarul optim de microinstructiuni*

$|\mu ON|$ = cardinalul nivelelor pe care le stabilim

$\forall \mu ON_j \in \{\mu ON\} \Rightarrow \mu CR(\mu ON_j) = \emptyset$

μPT_I – partitia cea mai timpurie de executie a unei μ operatii.

Aceasta este constituita din multimea de niveluri care contin toate μ operatiile independente situate la aceeaasi distanta fata de un nod initial in graful de dependente de date.

► $\mu PT_I = \{\mu ON | \forall \mu O_i, \mu O_j \in \mu ON \text{ avem } DI(\mu O_i) = DI(\mu O_j)\}$

unde prin DI am notat adancimea fata de noul initial in graful de dependente de date.

μPT_L – partitia cea mai tarzie de executie a unei μ operatii

► $\mu PT_L = \{\mu ON | \forall \mu O_i, \mu O_j \in \mu ON \text{ avem } DF(\mu O_i) = DF(\mu O_j)\}$

unde prin DF am notat adancimea fata de nivelul final (nodul final)

EX:

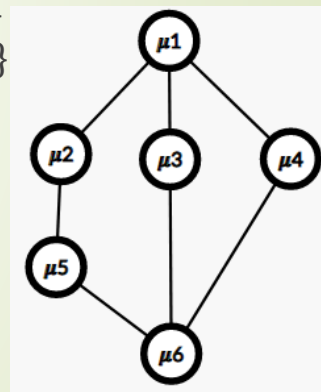
$$\begin{aligned}\mu PT_I &= \{< \mu O_1 >; < \mu O_2, \mu O_3, \mu O_4 >; < \mu O_5 >; < \mu O_6 >\} \\ \mu PT_L &= \{< \mu O_1 >; < \mu O_2 >; < \mu O_3, \mu O_4, \mu O_5 >; < \mu O_6 >\}\end{aligned}$$

Plecand de la cele doua definitii , se numeste

microoperatie critica o microoperatie care apartine aceluiasi subnivel in ambele partitii ($\mu PT_I, \mu PT_L$).

Ex: in cazul anterior avem μO_2 si μO_5

Orice modificare (mutare pe alt nivel) pentru o microoperatie critica conduce la un numar suplimentar de momente de timp pentru executie. Pentru celelalte (necritice), le putem pune oriunde intre nivelele date din partitia cea mai timpurie si cea mai tarzie, fara ca pentru executie sa avem necesar un timp suplimentar.



PARTITIA OPERATIILOR ELEMENTARE PE NIVELURI DE EXECUTIE

Partitiile stabilite numai pe baza dependentei de date nu sunt optime, intrucat intervine si conflict de resurse. Pentru a pune in evidenta acest conflict, in cadrul partiilor se face o grupare dupa tipul de resurse utilizate de fiecare microoperatie.

O micropartitie este egala cu multimea de niveluri.

$$\mu PT = \{\mu ON_1, \mu ON_2, \dots, \mu ON_{|\mu PT|}\}$$

care reprezinta multimea de niveluri

in care

$$\mu ON_j = \{\mu ON_{j1}, \dots, \mu ON_{j|nR|}\}$$

$$N_{ji} = \{\mu O_h \mid \forall \mu O_h \in \mu ON_j \rightarrow \mu O_h \text{ utilizeaza aceeasi } RES_i\}; 1 \leq i \leq NR$$

RES_i – resursa i

μON_j reprezinta nivelul j in cadrul partitiei, care este format din subniveluri care utilizeaza aceeasi resursa

Este bine ca un nivel sa-l impartim in subniveluri .

Pentru cele care folosesc aceeasi resursa, intre ele avem conflicte.

Trebuie executate secvential.

- **DEF** Cererea de resurse QRES este o aplicatie definita pe μOB si multimea de resurse din bloc: $\mu OB \times RES \rightarrow N$

unde $RES = RESS \cup RESD \cup RESB \cup RESULC \cup RESSH$.

RESS – resursa de tip sursa

RESD – resursa de tip destinatie

RESB – resursa de tip bus

ULC – unitati logice combinationale

SH – shuffler

DRES_i – resurse de tipul i disponibile

QRES_j – resurse de tipul j cerute

$DRES = \{DRES_1, DRES_2, \dots, DRES_{nR}\}$

$QRES_j = \{QRES_{j1}, QRES_{j2}, \dots, QRES_{jnR}\}$

– *vectorul de cereri de resurse necesare microoperatiilor*


care apartin nivelului j (μON_j)

$$QRES_{ji} = \sum_{\mu O_h \in \mu ON_j} \mu O_h \quad \text{sau}$$

$$QRES_{ji} = \sum QRES_{i, \mu O_h}$$

unde $\mu O_h \in \mu ON_j$ care utilizeaza resursa i

- Daca cererea de resurse este mai mare decat disponibilul de resurse, atunci pe nivelul j care foloseste resursa i apare conflict de resurse.
- $QRES_{ji} > DRES_i \Rightarrow \mu CR(\mu ON_{ji})=1$



Se numeste microinstructiune completa μIC acea microinstructiune la care nu se poate adauga nici o operatie din setul de microoperatii disponibile, fara a se crea conflict de resurse.

$$\mu IC: \forall \mu I (\forall \mu O_i \in \mu OD, \mu O_i \notin \mu I \Rightarrow \mu CR(\mu O_i, \mu I) = 1)$$

In cazul in care cererea de resurse:

$$QRES_{ji} > DRES_i \Rightarrow \mu IC \subset \mu ON_j; \quad \mu ON_j \setminus \mu IC \neq \emptyset$$

Acest lucru inseamna ca exista microoperatii din nivelul j care vor fi trecute pe nivelele urmatoare datorita conflictului de resurse.

DEFINITII

Intarzierea in ceea ce priveste numarul de niveluri cu care se va muta o microoperatie din nodul curent j spre nodul final din graful de precedenta datorita conflictului de resurse i :

$$nN_{ji} = \begin{cases} 0 & , \text{daca } QRES_{ji} \leq DRES_i \\ \left\lceil \frac{QRES_{ji} - DRES_i}{DRES_i} \right\rceil & , \text{daca } QRES_{ji} > DRES_i \end{cases}$$

Intarzierea maxima pe nivelul j :

$$nN_j = \max(nN_{ji}) , \text{ unde } i = 1, 2, \dots, nR$$

Stabilirea unei metodologii de determinare a partitionarilor unui microsubloc in microinstructiuni complete (care sunt actiunile pe care le fac la un moment dat).

Stabilirea efectului setului de microoperatii disponibile asupra generarii partiilor.

Pentru a genera o singura microinstructiune completa din μOD ,

ar trebui sa existe un criteriu care sa fie dependent numai de celelalte microoperatii din setul disponibil.

Aceasta alegere ar trebui sa fie independenta de microoperatiile ce vor deveni disponibile dupa asignarea microoperatiei respective.

PROPOZITIE: Pentru a alege o microoperatie care apartine setului de microoperatii disponibile k , pentru a o introduce intr-o instructiune μIC (completa, astfel incat numai o partitie, si anume μPT optima, sa fie generata), trebuie sa existe un criteriu de selectie dependent numai de microoperatiile care apartine setului disponibil dat, si independent de orice microoperatie din setul ce va deveni disponibil. *Aceasta implica existenta unui criteriu de ordonare a microoperatiilor din setul disponibil.*

JUSTIFICARE: Presupunem ca $\mu CR(\mu OD_k) = 1$, atunci avem conflict de resurse. Nu toate microoperatiile pot fi plasate in aceeaasi microinstructiune.

Fie μIC generata de μOD_k .

Faptul ca generam aceasta microinstructiune inseamna ca toate operatiile dependente de aceasta devin disponibile. Vom genera un nou set de microoperatii disponibile:

- $\mu OD_{k+1} = \{\mu OD_k \setminus \mu IC\} \cup \{\mu O_d\}_k$
 - microoperatii care devin disponibile datorita asignarii acelei instructiuni complete
- $\{\mu O_d\}_k = \{\mu O \mid \forall \mu O_i \in \mu IC \Rightarrow \mu O_i < \mu O\}$

Fara examinarea efectului unor microinstructiuni suplimentare asupra dinamicii lui μOD_{k+1} , alegerea unei microoperatii trebuie sa fie independenta de setul $\{\mu O_d\}_k$ rezultat. Trebuie sa existe acest criteriu de ordonare (care nici nu exista).

Orice metoda de ordonare a microoperatiilor din setul μOD_k , bazata pe resursele folosite de fiecare microoperatie si independenta de resursele folosite de celelalte microoperatii produce o ordonare arbitrara.

Masurarea efectului se poate face pe baza urmatoarelor informatii:

- resursele solicitate:
 - de microoperatia respectiva
 - de microoperatiile dependente de microoperatia analizata
- gradul de dependenta a celorlalte microoperatii fata de cea analizata

Microoperatiile din setul disponibil sunt independente, deci nu avem acces la cele 3 aspecte (influenta uneia asupra celeilalte).

Orice metoda de ordonare va conduce la o alegere arbitrara.

Fiecare microinstrucțiune generată produce un singur set de microoperații disponibile.

Dacă două microinstrucțiuni complete generează același set de microoperații disponibile, atunci cele două microinstrucțiuni complete sunt identice.

► Fie microinstrucțiunile μIC_i , μIC_j generate din setul disponibil la nivelul k (μOD_k).

Presupunem că $\mu IC_i \neq \mu IC_j$.

Fie μOD_{k+1}^i , μOD_{k+1}^j seturile de microoperații disponibile rezultate la momentul următor.

$$\mu OD_{k+1}^i = (\mu OD_k \setminus \mu IC_i) \cup \{\mu O_d\}_{ki}$$

$$\mu OD_{k+1}^j = (\mu OD_k \setminus \mu IC_j) \cup \{\mu O_d\}_{kj}$$

$$\mu IC_i \setminus (\mu IC_i \cap \mu IC_j) \subset \mu OD_{k+1}^i$$

$$\mu IC_j \setminus (\mu IC_i \cap \mu IC_j) \subset \mu OD_{k+1}^j$$

► Dacă $\mu OD_{k+1}^i \equiv \mu OD_{k+1}^j$ ar trebui ca:

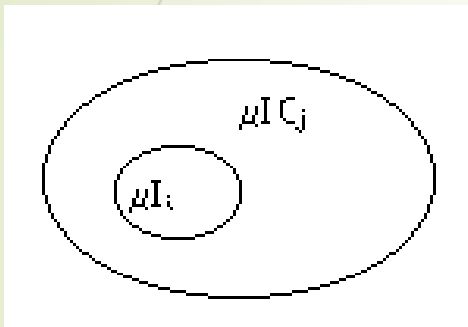
► $\mu IC_j \setminus (\mu IC_i \cap \mu IC_j) \equiv \mu IC_i \setminus (\mu IC_i \cap \mu IC_j) \Rightarrow \mu IC_i \equiv \mu IC_j$

PROPOZITIE: Partitia unui microsubloc se obtine numai prin generarea de μIC (nu pot lua combinatii mai reduse de microoperatii).

Presupunem ca din setul μOD_k putem genera μI_i si μIC_j .

Alegerea μI_i va produce un set de operatii disponibile

$$\mu OD_{k+1}^i$$




$$\mu OD_{k+1}^i = \mu OD_{k+1}^j \cup (\mu IC_j \setminus \mu I_i)$$
$$\mu I_i \subset \mu IC_j \text{ atunci } \mu IC_j \setminus \mu I_i \neq \emptyset.$$

$$\mu IC_j \setminus \mu I_i \neq \emptyset$$

$$\mu OD_{k+1}^i \subset \mu OD_{k+1}^j,$$

- ➡ avem eventual un numar mai mare de subniveluri (nu conduce la solutia optima).

Pe baza observatiei rezulta 2 categorii de algoritmi.



Partitie minima: - prin generarea din setul de microoperatii μOD_k a tuturor μIC posibile si pentru fiecare dintre ele sa se analizeze influenta ce o are asupra generarii setului de microoperatii disponibile urmator, μOD_{k+1} . **Acest algoritm are complexitate NP complete.**

Partitie euristica: - bazat pe ordonare a microoperatiilor din setul disponibil μOD_k in functie de succesorii in graful de dependente de date. Aceasta metoda nu va genera o partitie optima, insa va fi mult mai rapida in situatii acceptabile.

Calculul limitei inferioare pentru nivelurile de alocare

Presupunem că avem $\mu SB = (\mu B(\mu o), \angle)$ care are graful asociat G

$\mu B(\mu o) = \{ \mu o_1, \mu o_2, \dots, \mu o_{|\mu B|} \}$ microoperatiile din subblock

$P = \{P_1, P_2, \dots, P_{|P|}\}$ resursele, elemente de executie

În procesul de calcul al limitei inferioare apar 4 situații care trebuie analizate.

CAZUL 1 . Prima valoare a limitei inferioare a înălțimea h a grafului G , unde h = calea cea mai lungă de la un nod rădăcină la un nod terminal.

Valoarea h = valoarea minimă a limitei inferioare (nu s-a ținut seama de numărul de resurse și de conflictul de resurse).

CAZUL al 2-lea – estimare mai corectă – se analizează și distribuția microoperatiilor pe setul de resurse pe care le controleaza.

Fie ρ – această valoare limită.

Pentru calculul lui ρ , graful G va fi împărțit într-un subsistem de grafuri $G_i, 1 \leq i \leq |P|$

Resursele pot fi omogene sau nu.

Împărțirea se face în stabilirea – unor subseturi de sarcini alocate pe resurse, definite astfel:

$$G_i : \mu B(\mu o)_{|P_i|} = \{ \mu o \mid \mu o \text{ controleaza resursa } i \}$$

Subgraf maximal

Microoperatie terminală

Def. O componentă a unui graf orientat aciclic G_i este un subgraf maximal $SGM_{ij} \subset G_i$ aî pentru orice microoperatie μo_k care aparține lui SGM_{ij}

$(\mu o_k \in SGM_{ij})$ există $\mu o_m \in SGM_{ij}$ aî

$\mu o_k < \mu o_m$ sau $\mu o_m < \mu o_k$

(deci, trebuie să existe o legătură între cele două noduri)

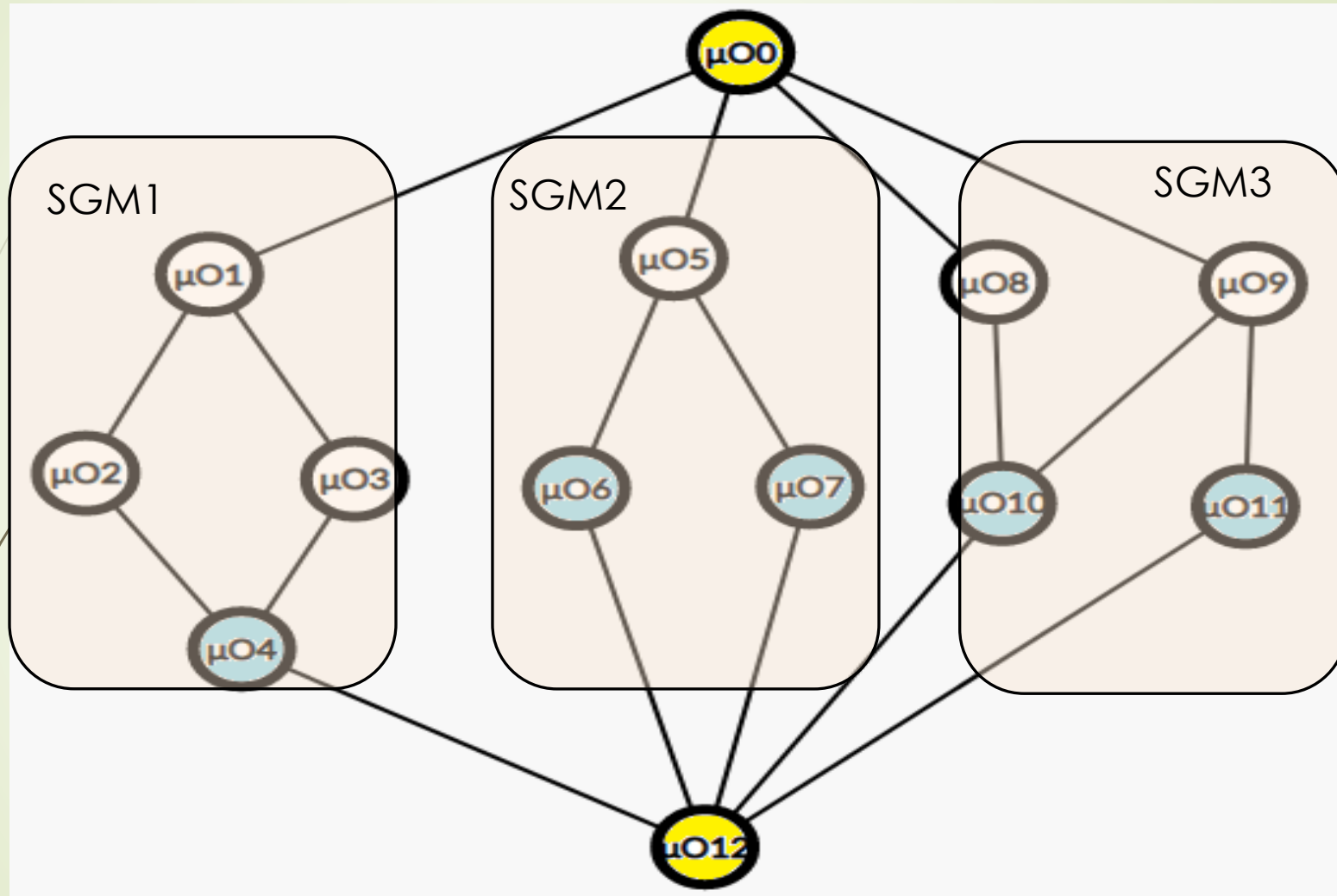
Obs. G_i e considerat ca o combinare paralelă a subgrafurilor maxime.,

$$G_i = || SGM_{ij}$$

Def. Microoperatie terminală. O microoperatie μo a unui graf orientat aciclic e o microoperatie terminală μo_T dacă nu mai există nici o sarcină μo_j aî $\mu o_T < \mu o_j$

Nivelul unei microoperatii μo_i într-un graf G_i orientat e dat de nr. de arce obținute într-o parcurgere pe drumul maxim între μo_i și μo_T

Exemplu



Considerand ca avem trei resurse $P = \{P1, P2, P3\}$

Sarcini terminale: $\mu O4$, $\mu O6$, $\mu O7$,

$\mu O10$, $\mu O11$

Nivelul $\mu O1 = 2$ (sarcinile terminale au nivelul 0)

Calcul ρ

Pentru fiecare microoperatie terminală din SGM_{ij} obținută prin împărțirea pe resurse a microoperatiilor, vom calcula nivelul din cadrul lui G și alegem microoperatia terminală cu nivel maxim.

Fie μo_{Tj} aceste microoperatii, iar cu $\|\mu o_{Tj}\|$ notăm nivelul unei astfel de micro-operatii în, graful G .

Pentru fiecare subgraf G_i , $i=1, \dots, |P|$ calculăm limita inferioară.

$$\rho_i = \max \{ \rho_{ij} \} \text{ unde } 1 \leq i \leq |P|, \quad 1 \leq j \leq |G_i|$$

$$\rho_{ij} = |SGM_{ij}| + \|\mu o_{Tj}\|$$

$$\|\mu o_{Tj}\| = \min_k \{ \|\mu o_{Tjk}\| \} \text{ cu } 1 \leq k \leq N_{\mu o_T}$$

$|SMG_{ij}|$ = nr. de microoperatii din cadrul componentei j a subgrafului G_i

$\|\mu o_{Tj}\|$ - nivelul minim al micro-operatiei terminale din cadrul componentei j a subgrafului G_i calculat față de poziția în G asociat sistemului de microoperatii asociat μSB .

$|G_i|$ = nr. de componente ale subgrafului G_i

$$\rho = \max_i \left\{ \max_j \left\{ |SGM_{ij}| + \min_k \{ \|\mu o_{Tjk}\| \} \right\} \right\} \text{ unde, } \begin{matrix} 1 \leq k \leq N_{\mu o_T} \\ 1 \leq i \leq |P| \\ 1 \leq j \leq |G_i| \end{matrix}$$

Exemplu

$\mu B(\mu_0) = (\mu_0_1, \mu_0_2, \mu_0_3, \mu_0_4, \mu_0_5, \mu_0_6, \mu_0_7, \mu_0_8, \mu_0_9, \mu_0_{10}, \mu_0_{11}, \mu_0_{12})$

$P = \{P_1, P_2, P_3\}$

Presupunem că avem următoarea repartitie pe resurse.

$\mu_{0_{P1}} = \{\mu_0_1, \mu_0_4, \mu_0_5, \mu_0_7, \mu_0_9\}$ G1

$\mu_{0_{P2}} = \{\mu_0_2, \mu_0_3, \mu_0_6, \mu_0_8, \mu_0_{10}\}$ G2

$\mu_{0_{P3}} = \{\mu_0_{11}, \mu_0_{12}\}$ G3

G1:

$SGM_{11} = \{\mu_0_1, \mu_0_4, \mu_0_5, \mu_0_7, \mu_0_9\}$

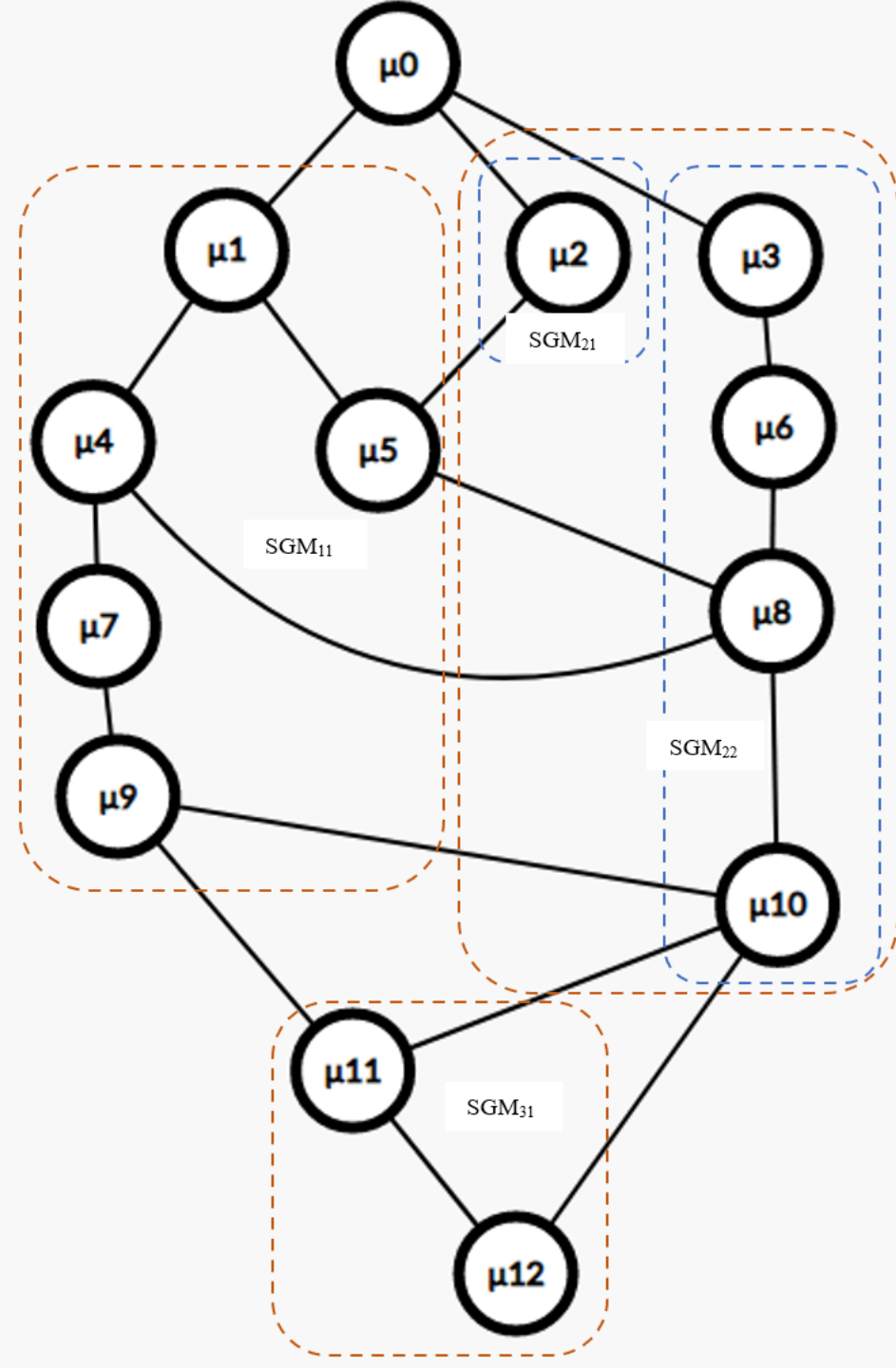
G2:

$SGM_{21} = \{\mu_0_2\}$;

$SGM_{22} = \{\mu_0_3, \mu_0_6, \mu_0_8, \mu_0_{10}\}$

G3:

$SGM_{31} = \{\mu_0_{11}, \mu_0_{12}\}$



G₁: 2 sarcini terminale μ_5, μ_9 , nivelul lui G_5 , calculat în G, pe calea cea mai lungă

$$\begin{aligned} \|\mu_5\| &= 4 \\ \|\mu_9\| &= 3 \\ |SGM_{11}| &= 5 \end{aligned}$$

$$\rho_1 = |SGM_{11}| + \min \{ \|\mu_5\|, \|\mu_9\| \} = 5 + 3 = 8$$

G₂: pentru SGM_{21} , o singură sarcină care este și terminală
 $\|\mu_2\| = 5$

$$\rho_{21} = \left\{ \left| \underbrace{SGM_{21}}_1 \right| + \min \|\mu_2\| \right\} = 1 + 5 = 6$$

pentru SGM_{22} ,

$$\rho_{22} = \left\{ \left| \underbrace{SGM_{22}}_4 \right| + \min \|\mu_{10}\| \right\} = 4 + 2 = 6$$

$$\Rightarrow \rho_2 = \max(\rho_{21}, \rho_{22}) = \max(6, 6) = 6$$

pentru SGM_{31} și $\mu_{12}=12$
 $|SGM_{31}| = 2$
 $\|\mu_{12}\| = 0$

$$\rho_3 = 2 + 0 = 2$$

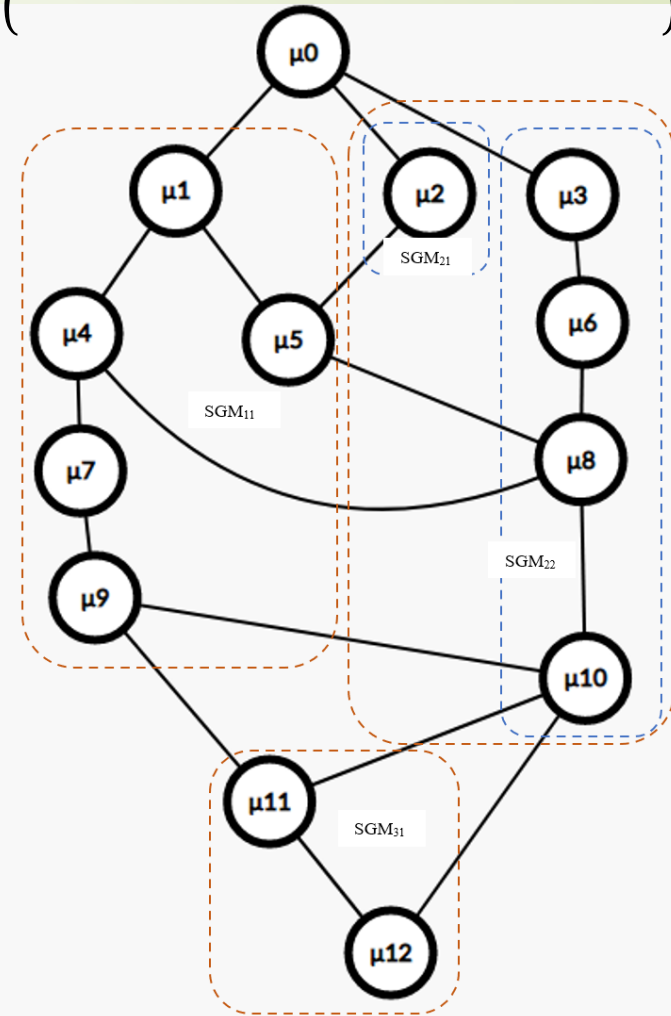
Am obținut 2, 6, 8 $\Rightarrow \rho = 8$

Pe același exemplu, $h=6$ (înălțimea maximă a grafului).

Deci, le luăm pe cele cu 8 niveluri sau mai mult.

Justificarea alegerii, în cadrul componentei SGM_{ij} , a sarcinii terminale cu nivel minim în G e aceea că orice parcurgere a unei componente trebuie să se termine într-un astfel de nod.

$$\rho = \max_i \left\{ \max_j \left\{ |SGM_{ij}| + \min_k \{ \|\mu_{Tjk}\| \} \right\} \right\}$$



Nu s-a ținut cont de conflictul de resurse.

În general, resursele nu sunt distincte, ρ va fi mai mare decât în realitate.

Vom defini pentru fiecare resursa un ρ_i

$$\rho_i = \max_j \left\{ \max \left[\left(\frac{|SGM_{ij}|}{|P_i|} \right), \left(h(SGM_{ij}) + \|\mu o_{T_j}\| \right) \right] \right\}$$

unde $|P_i|$ = număr de resurse de tip i

$|SGM_{ij}|$ = nr. de microoperații din cadrul componentei j ,

nr. minim de momente de timp.

Trebuie respectată dependența de date.

$h(SGM_{ij}) + \|\mu o_{T_j}\|$ dat de dependența de date

$\left\lceil \frac{|SGM_{ij}|}{|P_i|} \right\rceil$ - e dat de nr. de resurse pe care le avem

$h(SGM_{ij})$ - nr. de momente de timp necesare pentru execuția sarcinilor, având în vedere dependența de date

CAZUL 3

Limita inferioară = μ , limita inferioară pentru situația în care resursele sunt distincte.

$$\Rightarrow \mu = \max_i \{|\mu o P_i|\} \quad 1 \leq i \leq |P|$$

Fiind o singur resursa, se va fi controlata secvențial, deci avem $|\mu o P_i|$, fiind $|\mu o P_i|$ microoperatii.

Când resursele nu ar fi fost diferite, am putea considera că

$$\Rightarrow \mu = \max_i \{[|\mu o P_i|/|P_i|]\}$$

În calculul acestei limite, s-a presupus că nu există dependență de date între microoperatiile care controleaza diferite resurse.

CAZUL 4

Se consideră că o microoperatie nu se desfășoară numai într-o perioadă de timp ca în cazurile anterioare, ci pe durata a mai mulți cicli.

Notăm cu μM_c setul de microoperatii care se desfășoară pe n_c cicli.

\Rightarrow limita inferioară e influențată de nr. de cicli și o posibilitate de calcul ar fi

$z = |\mu M_c| \cdot n_c$ - e o variantă foarte simplistă, numai pentru punerea problemei.

În general, ajungem la $LI = \max\{h, \rho, \mu, z\}$

$C_{|AD|}^K \Rightarrow Q$; în funcție de LI, eliminăm f. mult.

Algoritmul de partitie va merge până la limita inferioară.

Nu se vor găsi soluții cu valori mai mici decât LI.

Dacă am găsit o soluție egală cu LI, rezultă că am găsit o soluție.

ALGORITHM CARE CONDUCE LA PARTITIE MINIMA:

Algoritm de partiționare pe niveluri a setului de micro-operații dintr-un micro-subbloc

$\mu SB = (\mu B(\mu o), <)$ - sistemul de microoperatii pe care trebuie sa-l partitionam

μPT – partiția ce se va genera

μPTA – partiția de microinstrucțiunea anterioară, considerată cea mai bună până în momentul respectiv

μoD – setul de microoperatii disponibile

μlC_i – microinstrucțiunea completă generată din setul de microoperatii disponibil curent

$nm_{\mu l}$ – numărul minim de microinstrucțiuni ce se poate obține prin codificarea microoperatiilor din subloc

$\{\mu o_d\}_i$ – mulțimea de microoperatii ce devin disponibile prin generarea microinstrucțiunii complete curente

Algoritmul generează o partiție optimă de μlC , ținând cont de conflictul de resurse și dependența de date impusă de relația impusă de ordine parțială.

Algoritm stabilire partitie microinstructiuni complete μPT

P1. Initializare

- Se initializeaza partitia $\mu PT = \Phi$, $\mu PTA = \mu B(\mu o)$ - tot microsublocul (deci, daca le luam si le facem secvential)
- $\mu oD = \{\mu o_j | \forall \mu o_j \in \mu B(\mu o) \text{ nu exista } \mu o_i < \mu o_j\}$ - deci pentru care nu exista predecesori
- $\mu oND = \{\mu o_i | \mu o_i \in \mu B(\mu o) \setminus \mu oD\}$ - operatii nedisponibile

P2. Daca $\mu oND = \Phi$ si $|\mu oD| \leq 3$ atunci salt la pas 5

P3. Se genereaza $\{\mu IC\}$ – multimea de microinstructiuni complete

- $\{\mu IC\} = \{\mu IC_j, \dots, \mu IC_k\}$ - din μoD curent
daca $j \neq k$ (s-au generat mai multe μIC)
atunci salvare $\mu PT_j = \mu PT$
 $\mu oD_j = \text{setul curent } \mu oD$
 $\{\mu IC\}_j = \{\mu IC\} \setminus \mu IC_j$

P4. $\mu PT = \mu PT \cup \mu IC_j$

$$\mu oD = \mu oD \setminus \mu IC_j \cup \{\mu o_d\}_j$$

$$\mu oND = \mu oND \setminus \{\mu o_d\}_j$$

- daca $|\mu oD| \neq 0$ si $|\mu PT| \leq |\mu PT_j| - 1$ atunci salt pas 2

P5. daca $\mu OD = \Phi$ atunci salt pas 7

P6. Se genereaza μIC din μoD curent.

$$\mu PT = \mu PT \cup \mu IC;$$

$$\mu oD = \mu oD \setminus \mu IC;$$

daca $\mu oD \neq \Phi$ si $|\mu PT| < |\mu PTA| - 1$ atunci pas 4

altfel pas 2

P7. daca $|\mu PT| < |\mu PTA|$ atunci $\mu PTA = \mu PT$

altfel daca $|\mu PTA| \leq nm\mu I$ atunci μPTA este optima

GATA

P8. daca $\{\mu IC\}_j \neq 0$ (cel care a fost salvat la pasul 3)

atunci reface $\mu PT = \mu PT_j$;

$$\mu oD = \mu oD_j;$$

$j \leftarrow j + 1$; (trec la urmatorul, selecteaza urmatoarea μIC din setul generat)

$$\mu IC = \mu IC_j$$

$$\mu oND = \{\mu o_j | \mu o_j \in \mu B(\mu o) \setminus (\mu PT \cup \mu oD)\}$$

salt pas 4

altfel μPTA este optima. GATA.

➡ Obs $nm\mu I$ – Limita inferioara

Algoritm euristic pentru partitionarea unui microsubbloc

Pentru a evita generarea tuturor partițiilor de microinstrucțiuni complete, folosim un criteriu de ordonare al operațiilor din setul de microoperații disponibile pe baza numărului de succesori. Nu duce la soluția optimă, dar duce la o soluție acceptabilă.

$$psucc(\mu O_i) = \{\mu O_j | \forall \mu O_i, \mu O_j \in \mu B(\mu O), \text{ cu } i \neq j, \text{ avem } \mu O_i < \mu O_j\}$$

$$psucc(\mu I_i) = \sum_{j=1}^n psucc(\mu O_{ij})$$

n = numărul de microoperații din μI_i

Etapele algoritmului

Pas 1

Se initializeaza partitia $\mu PT = \Phi$, $\mu PTA = \mu B(\mu o)$

$\mu oD = \{\mu o_j | \forall \mu o_j \in \mu B(\mu o) \text{ nu exista } \mu o_i < \mu o_j\}$ – deci pentru care nu exista predecesori

$\mu oND = \{\mu o_i | \mu o_i \in \mu B(\mu o) \setminus \mu oD\}$ – operatii nedisponibile

pas 2

facem o noua partitie, numai in anumite conditii

daca $\mu oND = \Phi$ (am avansat cu acest test pana la sfarsit) atunci **pas 4**

pas 3.

genereaza $\mu IC = \{\mu IC_j , ..., \mu IC_k\}$ microinstructiuni complete, ordonam acest set in functie de succesori; ordonam astfel incat

$$\mathbf{psucc}(\mu IC_j) \geq \mathbf{psucc}(\mu IC_{j+1}) \geq ... \geq \mathbf{psucc}(\mu IC_k)$$

Dintre doua microinstructiuni , pastram pe nivel pe cea cu cei mai multi succesori, intrucat mutarea ei pe nivelul urmator ar implica mutarea pe nivelul urmator a tuturor succesorilor ei. Deci, ea ar apartine micropartitiei curente.

$$\mu PT = \mu PT \cup \mu IC_j$$

Reactualizam operatiile disponibile, nedisponibile la momentul respectiv.

$$\mu oD = \mu oD \setminus \{\mu o \setminus \mu o \in \mu IC_j\} \cup \{\mu o_d\}_j$$

$$\mu oND = \mu oND \setminus \{\mu o_d\}$$

transfer **pas 2**

Pas 4

spatiul disponibil nu exista; gasesc ultima structura de microinstructiuni complete.

Genereaza $\mu IC = \{\mu IC_h, \dots, \mu IC_l\}$.

Ultima data: iau toate μIC care se genereaza aici (intrucat aici nu mai pot merge pe o cale sau alta).

$$\mu PT = \mu PT \cup \mu IC_h$$

$\mu oD = \mu oD \setminus \{\mu o \mid \mu o \in \mu IC_h\}$ - nu mai adaug nimic, nemaiavand succesorii (e ultima data).

$h = h+1$ – le iau pe toate, dar din cele generate din spatiul disponibil.

Continuam acest pas pana ce nu mai am nici o μIC .

daca $\mu oD \neq \Phi$ atunci **pas 4**

altfel μPT – partitia generata STOP

