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 şi
  - a cererilor de resurse,

  seturile de microoperații ce pot controla resurse diferite ale sistemului simultan.
- Obs: Conceptele si algoritmii prezentati pot fi aplicati şi la execuția task-urilor.

Definitii

- **Def.1** Se numeşte microsubbloc **μSB** un subset maxim al unui set de microoperații ce constituie o secvență indivizibilă cu un singur punct de intrare şi un singur punct de ieşire şi conține maximum o microoperație de salt, ca ultimă operație din secvență.
- μSB(μo)=(μB(μo), <) unde
  - $\mu B(\mu o) = \{\mu o_1, \mu o_2, \mu o_3, \ldots, \mu o_{|\mu B|}\}$
  - < relatie de ordine nereflexiva
- **Def.2 μOd** – microoperatie disponibila- acea microopertaie pentru care toate microoperatiile de care ea era dependenta de date au fost asignate 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 aceeaşi 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

$$\mu O_i \in \mu OD_k \qquad \mu O_j \in \mu OD_k$$

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

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

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 microsublocul respectiv

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

nmµI- numarul minim de microinstructiuni ce ar putea fi generate intr-un microsubloc

Evaluare estimativa:

- $\boldsymbol{nm\mu I} = \frac{|\mu oB(\mu O)|}{nR}$ - multimea de microoperatii din bloc
- unde:

$$nR - numarul\ de\ resurse\ ale\ sistemului$$

$\mu CR = \emptyset$ conflictul intre resurse

nMμI – numarul maxim de microinstructiuni

nMμI = |MB(μO)|

- toate microoperatiile se executa secvential

nmμI, nMμI sunt cazuri extreme.

nOμI- numarul optim de microinstructiuni

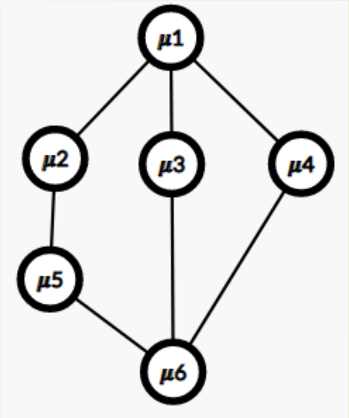
μON= |μON|= cardinalul nivelelor pe care le stabilim

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

- **μPT_l – partitia cea mai timpurie de executie a unei μoperatii.**

  Aceasta este constituita din multimea de niveluri care contin toate μoperatiile independente situate la aceeasi distanta fata de un nod initial in graful de dependente de date.
- $\boldsymbol{\mu PT_l = \{\mu ON | \forall \mu O_i, \mu O_J \in \mu ON\ avem\ DI(\mu O_i) = DI(\mu O_j)\}}$
- 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 μoperatii**
- $\boldsymbol{\mu PT_L = \{\mu ON | \forall \mu O_i, \mu O_J \in \mu CM\ avem\ DF(\mu O_i) = DF(\mu O_j)\}}$
- unde prin DF am notat adancimea fata de nivelul final (nodul final)

$$\mu PT_l = \{< \mu O_1 >; < \mu O_2, \mu O_3, \mu O_4 >; < \mu O_5 >; < \mu O_6 >\}$$
$$\mu PT_L = \{< \mu O_1 >; < \mu O_2 >; < \mu O_3, \mu O_4, \mu O_5 >; < \mu O_6 >\}$$



Plecand de la cele doua definitii , se numeste microoperatie critica o microoperatie care apartine aceluiasi subnivel in ambele partitii (μPT_l, μPT_L).

Ex: in cazul anterior avem 2 si 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 partitiilor se face o grupare dupa tipul de resurse utilizate de fiecare microoperatie.

O micropartitie este egala cu multimea de niveluri.

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

$$care\ reprez\,int\,a\ \ multimea\ de\ niveluri\ in\ care$$

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

$$N_{ji} = \{\mu O_h \,\middle|\, \forall \mu O_h \in \mu ON_j \to \mu O_h\ utilizeaza\ aceeasi\ RES_i\};\ \ 1 \le i \le NR$$

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 μB (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 destinati

RESB – resursa de tip bus

ULC – unitati logice combinationale

SH – shuffler

$DRES_i$ – resurse de tipul i disponibile

$QRES_i$ – resurse de tipul i cerute

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

$$QRESj \quad \{QRES_{j1}, QRES_2, ..., QRES_{j,nR}\}$$

$$- vectorul\ de\ cereri\ de\ resurse\ necesare\ microoperatiilor$$

$$care\ apartin\ nivelului\ j\ (\mu ON_j)$$

$QRES_{ji} = \sum \mu O_h \;\; sau \;\; QRES_{ji} = \sum QRES_{\mu O_h},$

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.

- $\boldsymbol{QRE_{S_{ji}} > DRE_{S_i} \Rightarrow \mu CR(\mu O_{N_{ji}}) = 1}$

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 interese.

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

In cazul in care cererea de resurse:

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

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

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

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:

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

Intarzierea maxima pe nivelul j:

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

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 partitiilor.

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 μI_C (completa, astfel incat numai o partitie, si anume μPI 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 aceeasi 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:

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

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 microoperatiil 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 microoinstructiune generata produce un singur set de microoperatii disponibile. **Daca doua microinstructiuni complete genereaza acelasi set de microoperatii disponibile, atunci cele doua microinstruinctiuni complete sunt identice.**

- Fie microinstructiunile μIC_i, μIC_j generate din setul disponibil la nivelul k (μOD_k). Presupunem ca $\mu IC_i \neq \mu IC_j$. Fie μOD_{k+1}^{i} , μOD_{k+1}^{j} seturile de microoperatii disponibile rezultate la momentul urmator.

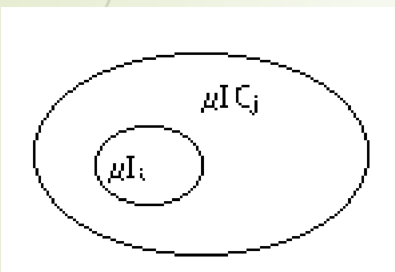
$$\mu OD_{k+1}^{i} = (\mu OD_k \backslash \mu IC_i) \cup \{\mu O_d\}_{ki}$$
$$\mu OD_{k+1}^{j} = (\mu OD_k \backslash \mu IC_j) \cup \{\mu O_d\}_{kj}$$
$$\mu IC_i \backslash (\mu IC_i \cap \mu IC_j) \subset \mu OD_{k+1}^{i}$$
$$\mu IC_j \backslash (\mu IC_i \cap \mu IC_j) \subset \mu OD_{k+1}^{j}$$

- Daca $\mu OD_{k+1}^{i} \equiv \mu OD_{k+1}^{j}$ ar trebui ca:
- $\mu IC_j \backslash (\mu IC_i \cap \mu IC_j) \equiv \mu IC_i \backslash (\mu IC_i \cap \mu IC_j) \Rightarrow \mu IC_i \equiv \mu IC_j$

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 μOD_{k+1}^{i}



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

$$\mu OD_{k+1}^{i} = \mu OD_{k+1}^{j} \cup (\mu IC_j \setminus \mu I_i)$$

Deoarece $\mu I_i \subset \mu IC_j$ atunci $\mu IC_j \setminus \mu I_i \neq \emptyset$.

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

- 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 MP completa.

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 μSB =(μB(μo) , ∠) care are graful asociat G

$\mu B(\mu o)=\{ \mu o_1, \mu o_{2,\dots,} \mu o_{|\mu B|} \}$ microoperatiile din subblock

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

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

CAZUL 1 . Prima valoare a limitei inferioare a înălţimea 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 ţinut seama de numarul de resurse si de conflictul de resurse).

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

Fie ρ – această valoare limită.
Pentru calculul lui ρ, graful G va fi împărţit într-un subsistem de grafuri $G_i, 1 \le i \le |P|$
Resursele pot fi omogene sau nu.
Împărţirea se face în stabilirea – unor subseturi de sarcini alocate pe resurse, definite astfel:

Gi : $\mu B(\mu o)_{|Pi|} = \{\mu o \mid \mu o$ controleaza resursa i}

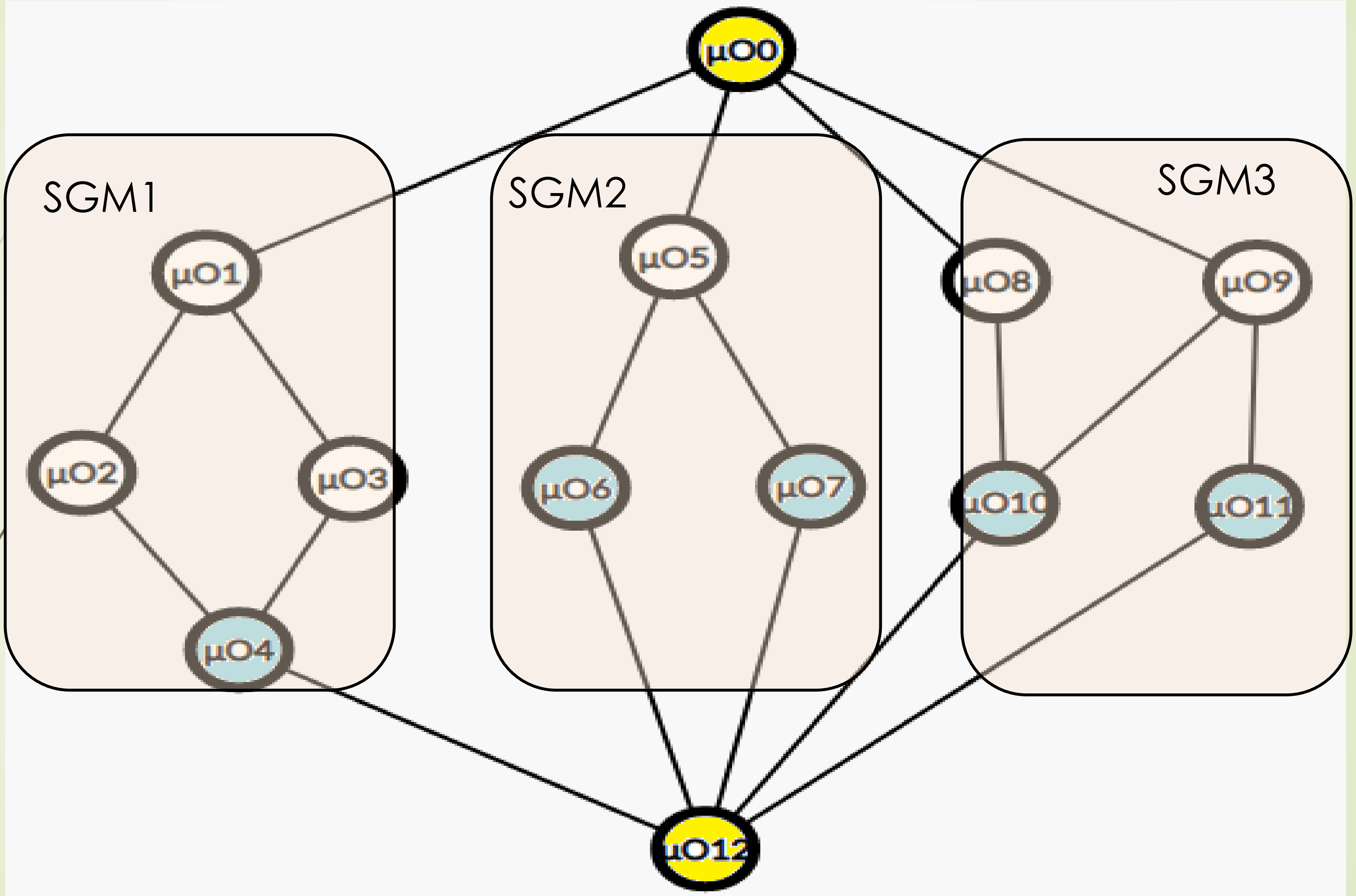
Subgraf maximal
Microoperatie terminală

Def. O componentă a unui graf orientat aciclic G_i este un subgraf maximal $SGM_{ij} \subset G_i$ aî pentru orice microperatie μo_k care aparţine lui SGM_{ij}
($\mu o_k \in SGM_{ij}$) există $\mu o_m \in SGM_{ij}$ aî
$$\mu o_k \angle \mu o_m \text{ sau } \mu o_m \angle \mu o_k$$
(deci, trebuie să existe o legătură între cele două noduri)

Obs. G_i e considerat ca o combinare paralelă a subgrafurilor maximale.,
$$G_i = || SGM_{ij}$$

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 obţinute într-o parcurgere pe drumul maxim între μo_i şi μo_T

Exemplu



Cosiderand ca avem trei resurse P= {P1, P2, P3 }
Sarcini terminale: µo 4, µo 6 , µo 7 , µo 10, µo 11
Nivelul µo1= 2 (sarcinile terminale au nivelul 0)

Calcul ρ

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

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

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

ρi= max (ρij} unde 1<= i<=|P|, 1<= j <= |Gi|

ρij= |SGMij| + $\| \mu o_{Tj} \|$

$\|\mu o_{Tj}\| = \min_{k}\{\|\mu o_{Tjk}\|\}$ cu 1<=k<=$N\mu o_T$

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

$\left\|S_{Tj}\right\|$- nivelul minim al sarcinilor terminale din cadrul componentei j a subgrafului G_i calculat faţă de poziţia în G asociat sistemului de microoperatii asociat µSB.

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

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

$\mu B(\mu o) = (\mu o_1, \mu o_2, \mu o_3, \mu o_4, \mu o_5, \mu o_6, \mu o_7, \mu o_8, \mu o_9, \mu o_{10}, \mu o_{11}, \mu o_{12}\}$

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

Presupunem că avem următoarea repartiţie pe resurse.

$\mu o_{P1}=\{ \mu o_1, \mu o_4, \mu o_5, \mu o_7, \mu o_9\}$ G1

$\mu o_{P2}= \{\mu o_2, \mu o_3, \mu o_6, \mu o_8, \mu o_{10}\}$ G2

$\mu o_{P3}=\{ \mu o_{11}, \mu o_{12}\}$ G3

G1:

$SGM_{11} = \{\mu o1, \mu o4, \mu o5, \mu o7, \mu o9\}$

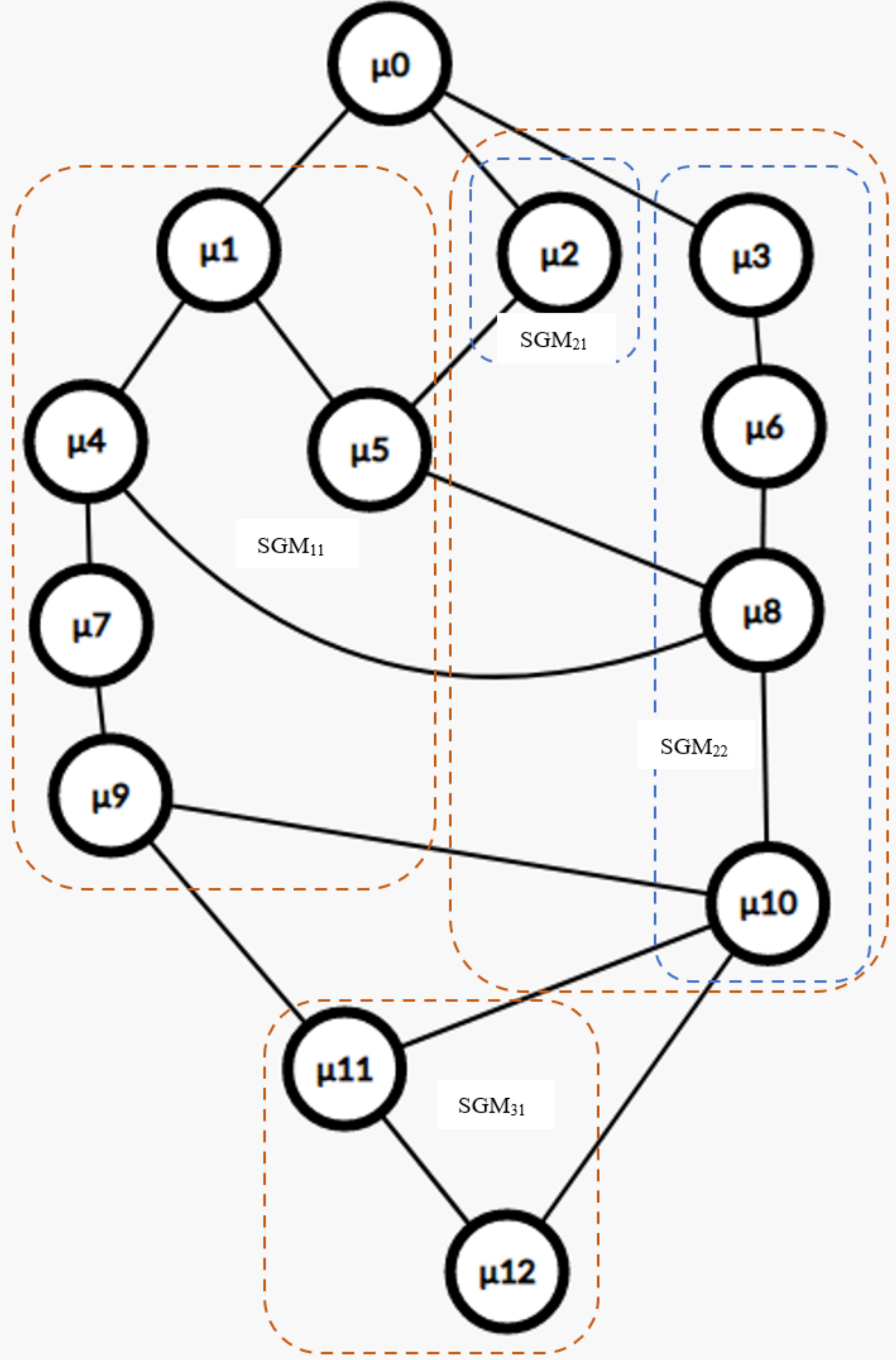
G2:

$SGM_{21} = \{ \mu o2\}$;

$SGM_{22} = \{ \mu o3, \mu o6, \mu o8, \mu o10\}$

G3:

$SGM_{31} = \{ \mu o11, \mu o12\}$



G_1: 2 sarcini terminale μo5, μo9, nivelul lui G_5, calculat în G, pe calea cea mai lungă
|| μo5||=4
|| μo9||=3
$|SGM_{11}|=5$

$\rho 1 = |SGM_{11}| + \min\{\|\mu o_5\|, \|\mu o_9\|\} = 5 + 3 = 8$

G_2: pentru SGM_{21}, o singură sarcină care este şi terminală

$$\|\mu o_2\| = 5$$

$$\rho_{21} = \left\{ \left| \underbrace{SGM_{21}}_{1} \right| + min\|\mu o_2\| = 1 + 5 = 6 \right\}$$

pentru SGM_{22},

$$\rho_{22} = \left\{ \left| \underbrace{SGM_{22}}_{4} \right| + min\|\mu o_{10}\| = 4 + 2 = 6 \right\}$$

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

G_3
SGM_{31} si $\mu o_T = 12$
$|SGM_{31}| = 2$
$|\mu o_{12}| = 0$

$\rho 3 = 2 + 0 = 2$

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

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

Pe acelaşi exemplu, h=6 (înălţimea 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.

Nu s-a ţinut 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 ρi

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

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

|SGMij|=nr. de microoeratii din cadrul componentei j,

nr. minim de momente de timp.

Trebuie respectată dependenţa de date.

$h(SGMij)+||\mu oTj||$ dat de dependenţa de date

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

$\mathrm{h}(SGM_{ij})$ - nr. de momente de timp necesare pentru execuţia sarcinilor, având în vedere dependenţa de date

CAZUL 3

Limita inferioară $= \mu$, limita inferioară pentru situaţia în care resursele sunt distincte.

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

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

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

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

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

CAZUL 4

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

Notăm cu μMc setul de microoperatii care se desfăşoară pe n_c cicli.

$\Rightarrow$ limita inferioară e influenţată de nr. de cicli şi 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 funcţie de LI, eliminăm f. mult.

Algoritmul de partitie va merge până la limita inferioară.

Nu se vor gasi soluţii cu valori mai mici decat LI.

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

ALGORITM 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 – partitia ce se va genera

μPTA – partitia de microinstructiunea anterioara, cosiderata cea mai buna pana in momentul respectiv

μOD – setul de microoperatii disponibile

μIC_i – microinstructiunea completa generata din setul de microoperatii disponibil curent

nmμI – numarul minim de microinstructiuni ce se poate obtine prin codificarea microoperatiilor din subloc

$\{\mu O_d\}_i$ – multimea de microoperatii ce devin disponibile prin generarea microinstructiunii complete curente

Algoritmul genereaza o partitie optima de μIC, tinand cont de conflictul de resurse si dependenta de date impusa de relatia impusa de ordine partiala.

Algoritm

P1. Initializare

- Se initializeaza partitia $\mu PT = \Phi$, $\mu PTA = \mu B(\mu O)$ - tot microsublocul (deci, daca le luam si le facem sevcential)
- $\mu OD = \{\mu O_j | \forall \mu O_j \in \mu B(\mu O)\ nu\ exista\ \mu O_i < \mu O_j\}$ - deci pentru care nu exista predecesori
- $\mu OND = \{\mu O_i | \mu O_i \in \mu B(\mu O) \backslash \mu OD\}$ - operatii nedisponibile

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

P3. Se genereaza {μIC} – multimea de microinstructiuni complete

- $\{\mu IC\} = \{\mu IC_j, ..., \mu IC_k\}$ - din μOD curent

  daca j ≠ k (s-au generat mai multe μIC)

  atunci salvare $\mu PT_j = \mu PT$

  μOD_j = setul curent μOD

  $\{\mu IC\}_j = \{\mu IC\} \backslash \mu IC_j$

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

$$\mu OD = \mu OD \backslash \mu IC_j \cup \{\mu O_d\}_j$$

$$\mu OND = \mu OND \backslash \{\mu O_d\}_j$$

- 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.

μPT = μPT U μIC;

μOD = μOD \ μIC;

daca μOD ≠ Φ si |μPT|<|μPTA|-1 atunci pas 4

altfel pas 2

P7. daca | μPT| < | μPTA| atunci μPTA = μPT

altfel daca |μPTA| ≤ nmμI atunci μPTA este optima

STOP.

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)

$\mu IC = \mu IC_j$

$\mu OND = \{\mu O_j | \mu O_j \in \mu B(\mu O) \backslash (\mu PT \cup \mu OD)\}$

salt pas 4

altfel μPTA este optima.

GATA.

- Obs nmμI – Limita inferioara

Algoritm euristic pentru partitionarea unui microsubloc

Pentru a evita generarea tuturor partitiilor de microinstructiuni complete, folosim un criteriu de ordonare al operatiilor din setul de microoperatii disponibile pa baza numarului de succesori.

Nu duce la solutia optima, dar duce la o solutie acceptabila.

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

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

n = numarul de microoperatii din μI_i

Etapele algoritmului

Pas 1

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

$\mu OD = \{\mu O_j | \forall \mu O_j \in \mu B(\mu O)\ nu\ exista\ \mu O_i < \mu O_j\}$ – deci pentru care nu exista predecesori

$\mu OND = \{\mu O_i | \mu O_i \in \mu B(\mu O) \backslash \mu OD\}$ – 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 $\mu IC = \{\mu IC_j , \ldots, \mu IC_k \}$ micoinstructiuni complete, ordonam acest set in functie de succesori; ordonam astfel incat

psucc(μIC_j) ≥ psucc (μIC_{j+1}) ≥ … ≥ psucc(μIC_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.

$\mu OD = \mu OD \backslash \{\mu O \backslash \mu O \in \mu IC_j\} \cup \{\mu O_d\}_j$

$\mu OND = \mu OND \backslash \{\mu\ O_d\}$

transfer **pas 2**

Pas 4

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

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

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