Codificarea minimă a unui set de microinstrucțiuni

În proiectarea unităților de comandă microprogramate, după ce s-a determinat setul de microinstrucțiuni complete ce realizează execuția unui microsubbloc în minimum de paşi prin controlul tuturor operațiilor paralele ce se pot executa în sistem, se pune problema codificării minime a acestui set de microinstrucțiuni.

Problema optimizării numărului de biți constă în a stabili lungimea cuvântului memoriei de control ce specifică microinstrucțiunile, astfel încât aceasta să fie minimă, având în vedere asigurarea controlului tuturor microoperațiilor paralele specificate de secvența de microinstrucțiuni ce descrie unitatea de comandă.

Practic se va stabili organizarea logică a microinstrucțiunilor prin specificarea câmpurilor şi a microoperațiilor din fiecare câmp, astfel încât orice microinstrucțiune completă să poată fi specificată. Stabilirea structurii logice a microinstrucțiunii va fi realizată având în vedere următoarele aspecte:

- microoperațiile specificate de fiecare microinstrucțiune completă ;
- natura componentelor fizice cu care se va face implementarea memoriei de control.

Schwartz [SCHW] a fost primul care a formulat această problemă prin introducerea unui model de w cuvinte a N biți fiecare, în care fiecare cuvânt putea specifica una sau mai multe microoperații. Algoritmul propus de el, bazat pe o enumerare exhaustivă, generează o codificare a setului de microoperații distincte într-un număr minim de câmpuri.

Grasselli şi Montanari [GRA] au arătat că soluția cu un număr minim de câmpuri nu asigură număr minim de biți. Ei au reformulat problema încadrând-o în teoria comutației, reducând problema optimizării numărului de biți la problema acoperirii mintermenilor cu implicanți primi.

Problema este reluată de Das şi Barrejee [DAS] plecând de la clasele maximale de compatibilitate, generând solutii mai bune decât cele obținute de Grasselli.

Toate aceste metode sunt bazate pe presupunerea că toate microoperațiile specificate de microinstrucțiunile complete ale partiției microblocului, care se execută în paralel, se desfăşoară în cadrul aceleiaşi faze a ciclului memoriei de control. Sunt aplicabile numai pentru microinstrucțiunile monofazice.

Problema minimizării în cadrul microinstrucțiunilor polifazice a fost tratată de Dasgupta. În prezentarea pe care o face, a considerat ca microoperațiile din acelasi câmp sunt executate în aceeaşi fază a ciclului microinstrucțiunii.

Problema generală a optimizării numărului de biți este o problemă din clasa "NP complete". Apartenență la clasa "NP complete" a fost demonstrată de Robertson [ROBE79]. Având în vedere faptul că memoriile PROM sunt configurate în general pe 4 sau 8 biți şi că problema generală a optimizării face parte din clasa "NP complete" este bine ca soluțiile adoptate pentru codificarea minimă a unui set de microinstrucțiuni să aibă şi un caracter ingineresc. în cele ce urmează o să se prezinte o modalitate de codificare aproape minimă a unui set de microinstrucțiuni bazată pe un algoritm cu caracter ingineresc.

Def. 1

Fie $mPt=\{mIC_1,mIC_2,...,mIC_{[mPt]}\}$ partiția unui microsubbloc în microinstrucțiuni complete şi $MB(mO)=\{mO_1,mO_2,...,\ mO_{[MB]}\}$ setul de microoperații distincte din cadrul microsubblocului.

Două microoperații mO_i şi mO_j sunt **compatibile** dacă pentru orice k, $1 \le k \le [mPT]$, dacă $mO_i \in mIC_k$ atunci $mO_j \notin mIC_k$.

Compatibilitatea între două microoperații trebuie privită în sensul că cele două microoperații nu sunt specificate (nu sunt active) niciodată împreună în cadrul unei microinstrucțiuni din microbloc. Controlul resurselor sistemului microprogramat nu necesită niciodată efectuarea în paralel a celor două microoperații.

Def. 2

Două microoperații $mO_i \in MB(mO)$, $mO_j \in MB(mO)$ sunt **incompatibile** dacă există cel puțin o microinstrucțiune completă mIC astfel încât $mO_i \in IC_k$ şi $mO_j \in mIC_k$.

Def. 3

O clasă de compatibilitate CC(mO) este un set (subset) al mulțimii MB(mO) în care oricare două microoperații sunt compatibile între ele.

$CC(mO)=\{mO \mid$ pt orice $mO_i, mO_j \in CC(mO)$ avem mO_i compatibilă cu $mO_j\}$

Def. 4

O clasă de compatibilitate maximă MCC(mO) este acea clasa de compatibilitate la care nu mai poate fi adăugată nici o microinstrucțiune fără a se pierde compatibilitatea.

$MCC(mO)=\{mO_j \mid$ pt orice $mO_i \notin MCC(mO)$, există $mO_j \in MCC(mO)$
astfel încât $mO_i \in mIC_k$ şi $mO_j \in mIC_k\}$.

În mod analog se defineşte clasa de incompatibilitate maximă MIC(mO).

Def. 5

Costul de implementare a unei clase de compatibilitate (măsurat în numărul de biți necesari pentru codificare) este dat de implementarea codificării verticale a microoperațiilor ce compun clasa.

Cost $CC_i=[\log_2(|CC_i|+1)]$

iar costul total de implementare al cuvântului de control

$$\text{Cost } \sum_{i=1}^{k} CC = S\ [\log_2(|CC_i|+1)]$$

unde k este numărul de clase de compatibilitate.

Obs

O clasă de compatibilitate corespunde unui câmp din cadrul microinstrucțiunii.

Implementarea microinstrucțiunilor complete se face prin control rezidual pentru a putea specifica toate microoperațiile paralele necesare.

Setul de clase compatibile $CC=\{CC_1,CC_2,...,CC_k\}$ poate specifica orice microoperație din multimea MB(mO) şi orice microinstrucțiune completă din cadrul partiției microsubblocului.

Noțiunea de compatibilitate poate fi extinsă în cadrul structurilor microprogramate cu ciclu polifazic, în sensul ca în cadrul aceleiaşi clase pot fi specificate microoperații ce se desfăşoară paralel, dacă ele se execută în faze diferite.

Se va descrie o metodă de codificare minimă a setului de microinstrucțiuni complete obținut prin partiționarea unui microsubbloc, plecând de la un subset de clase de compatibilitate maxime. Mai întâi vom face câteva precizări privind estimarea costului minim.

Estimarea costului minim

Problema codificării minime presupune două faze distincte:

- enumerarea tuturor claselor de compatibilitate maximă MCC;
- determinarea unui subset de MCC care să implementeze costul minim pentru codificare.

Costul implementării depinde de numărul de MCC necesar pentru acoperirea întregului set de microoperații şi de numărul de microoperații din fiecare clasă de compatibilitate maximă. De notat faptul că un cuvânt din memoria de control, adică o microinstrucțiune completă, este un set de clase incompatibile între microoperațiile componente.

O clasă de compatibilitate specifică cel mult o microoperație din cadrul unui cuvânt al memoriei de control.

Def. 6

Pentru orice clasă de incompatibilitate IC, clasele de compatibilitate maximă ce conțin un element din IC se numesc clase de compatibilitate maximă asociate (AMCC) clasei de incompatibilitate.

Prop. 1

Pentru orice clasă de incompatibilitate maximă MIC reuniunea claselor de compatibilitate maximă asociate acoperă întreg setul de microoperații.

Dem.

Fie $MIC(mO)=\{mO_1,...,mO_{|MIC|}\}$ formată din |MIC| microoperații, unde $|MIC| \leq |MB|$. Presupunem contrariul şi anume că reuniunea claselor de compatibilitate maximă asociate nu acoperă întreg setul de microoperații. Fie mO_j una din aceste microoperații presupuse neacoperite cu proprietatea că:

$mO_j \in MB(mO)$ dar $mO_j \notin MIC(mO)$ şi $mO_j \notin \bigcup_{MIC} AMCC$.

Datorită faptului că nu aparţine reuniunii claselor de compatibilitate maximă asociate, rezultă faptul ca mO_j este incompatibilă cu toate microoperaţiile din MIC. Înseamnă că poate fi adăugată acestei clase, ceea ce contrazice definiţia clasei de incompatibilitate maximă. Ajungându-se la contradicţie rezultă că presupunerea este falsă.

Prop. 2

Pentru o clasă de incompatibilitate maximă MIC, reuniunea oricăror k clase de compatibilitate, k < |MIC|, nu poate acoperi setul de microoperaţii MB(mO).

Dem.

Considerăm $MIC(mO)=\{mO_1,mO_2,...,mO_{|MIC|}\}$. |MIC| este limita inferioară pentru numărul de clase de compatibilitate ce satisface acoperirea, deoarece o clasă de compatibilitate poate acoperi un singur element din MIC. Rezultă că orice reuniune de k clase de compatibilitate k < |MIC| nu poate acoperi setul de microoperaţii.

Exemplul 1

Fie $MB(mO)=\{mO_1, mO_2, mO_3, mO_4, mO_5, mO_6, mO_7\}$ şi clasele de incompatibilitate maximale definite de structura microprogramată:

$MIC_1(mO)=\{mO_1,mO_2,mO_3,mO_4\}$
$MIC_2(mO)=\{mO_3,mO_4,mO_6,mO_7\}$
$MIC_3(mO)=\{mO_5,mO_6,mO_7\}$
$MIC_4(mO)=\{mO_1,mO_5\}$

Clasele de compatibilitate maximale sunt :

$MCC_1=\{mO_1,mO_6\}$ $MCC_2=\{mO_1,mO_7\}$ $MCC_3=\{mO_2,mO_5\}$
$MCC_4=\{mO_2,mO_6\}$ $MCC_5=\{mO_2,mO_7\}$ $MCC_6=\{mO_3,mO_5\}$
$MCC_7=\{mO_4,mO_5\}$ $MCC_8=\{mO_4,mO_6\}$ $MCC_9=\{mO_4,mO_7\}$

Clasele de incompatibilitate maximale sunt determinate pe baza grafului de dependenţă şi a conflictului de resurse şi reprezintă microoperaţiile specificate de microinstrucţiunile complete ce descriu microblocul.

Clasele MCC sunt determinate din matricea de microoperaţii ce indică incompatibilitatea. Aplicarea operatorului SAU între liniile matricii va specifica microoperaţiile ce fac parte din clasa de compatibilitate maximă.

Calcularea MCC este o binecunoscută problemă în teoria automatelor finite, existând numeroase metode pentru aceasta soluţie. AHO arată că problema calculării MCC este o problema "NP complete". Conform propoziţiei 1, reuniunea AMCC asociată clasei incompatibile MIC_4 este :

$$\bigcup_{MIC_4} AMCC = MCC_1 \cup MCC_3 \cup MCC_2 \cup MCC_6 \cup MCC_7$$

$$\bigcup_{MIC_4} = \{mO_1,mO_2,mO_3,mO_4,mO_5,mO_6,mO_7\}$$

Prop. 3

Dacă un set de microoperaţii MB(mO) este partiţionat în q câmpuri ale unei microinstrucţiuni, costul minim se va realiza atunci când (q-1) câmpuri specifică câte o

microoperație (au un singur bit), iar cel de al q-lea câmp codifică restul de |MB(mO)|<-q+1 microoperații.

Dem.

Vom descrie o demonstrație echivalentă. Vom arăta că pentru o lungime de microinstrucțiune dată LMI şi pentru un număr de câmpuri q dat se pot codifica maximum de microoperații, iar ultimul câmp are LMI-q+1 biți. Considerăm o partiție arbitrară a LMI biți în q câmpuri. Fie câmpul cu b_{max} biți, câmpul cu lungimea cea mai mare şi fie un oricare alt câmp care are b_i biți. Numărul de microoperații ce poate fi codificat de cele două câmpuri este:

$$NMO = (2^{bmax} -1)+(2^{bi} -1)$$

Facem o modificare în organizarea logică a microinstrucțiunii şi mutăm un bit din câmpul cu b_i biți în câmpul cu b_{max} biți. în acest caz numărul de microoperații ce se pot codifica este:

$$NMO'= (2^{(bmax+1)} -1)+(2^{(bi-1)} -1)$$

$$NMO'-NMO= 2^{bmax} -2^{bi} \geq 0$$ deoarece $b_{max} \geq b_i$.

Deci se pot codifica mai multe operații (microoperații) după modificarea structurii microinstrucțiunii.

Repetând procesul de mutare a unui bit dintr-un câmp oarecare în câmpul de lungime maximă, numărul de microoperații ce poate fi codificat creşte. Numărul maxim de microoperații ce poate fi codificat rezultă a fi egal cu $(q+2^{(LMI-q+1)}-2)$. Transformând totul în cost se obtine costul minim:

Cost = $q-1+[\log_2(|MB(mO)|-q+2)]$ când (q-1) câmpuri specifică fiecare câte o microoperație, iar ultimul câmp (|MB(mO)|-q+1) microoperații.

Prop. 4

Nu este posibilă o soluție de partiționare a setului MB(mO) microoperații în (q+h) câmpuri astfel încât costul să fie mai mic decât partiția în q câmpuri.

Dem.

Fie

C =$q-1+[\log_2(|MB(mO)|-q+2)]$ costul minim obținut prin partiția în q câmpuri şi

C = $q+h-1+[\log_2(|MB(mO)|-q-h+2)]$ costul minim obținut prin partiția în q+h câmpuri.

Deosebim 2 cazuri şi anume:

1) h=1 şi $|MB(mO)|-q-1=2^k$

2) $h \neq 1$ sau $|MB(mO)|-q+1 \neq 2^k$

în primul caz:

$C_q=q-1+k+1=q+k$

$C_{(q+h)}=q+k$

deci $C_q=C_{(q+h)}$

în al doilea caz, deoarece :

$[\log_2(|MB(mO)|-q+2)]-[\log_2(|MB(mO)|-q-h+2)] < h$

rezultă $C_q < C_{(q+h)}$

Observația 6

Costul minim pentru codificarea setului de microoperații MB(mO) poate să fie mai mare decât cel dat de propoziția 3 adică:

$C \geq q-1+[\log_2(|MB(mO)|-q+2)]$ când $|MCC|_{max} \leq |MB(mO)|-q$.

într-adevăr costul minim corespunde când:

(q-1) câmpuri specifică fiecare câte o microoperație;

cel de-al q-lea câmp codifică |MB(mO)|-q+1 microoperații.

Dacă $|MCC|_{max} \leq |MB(mO)|-q$, rezultă că $|MCC|_{max}<|MB(mO)|-q+1$, ceea ce ar face ca în ultimul câmp să nu fie toate microoperațiile compatibile.

În acest caz trebuie renunțat la organizarea microinstrucțiunii în (q-1) câmpuri de 1 bit, dar prin aceasta se măreşte şi costul de implementare.

În continuare vom prezenta o metodă de minimizare a numărului de biți necesari pentru codificarea microinstrucțiunilor complete generate prin partiția unui microbloc.

Etapele metodei

1. Se alege clasa de incompatibilitate maximă care are cardinalitatea maximă MIC. Fie

$MIC_m \in MIC$ astfel încât $|MIC_m| \geq |MIC_j<$ pentru $j \neq m$,

$1 \leq j \leq |MIC|$

Se generează clasele de compatibilitate maximă asociate AMCC clasei MIC.

$AMCC_m=\{MCC \geq$ pt orice $mO \in MIC_m \exists$ MCC astfel încât $mO \in MCC\}$

2. Se formează tabela de acoperire modificată prin considerarea numai a claselor de compatibilitate maximă ce aparțin AMCC.

TAM : $AMCC_m$ x MB(mO) --> B

Se caută multimea de clase de compatibilitate maximă esențiale $\{MCC_e\}$ inclus în $AMCC_m$ astfel încât există MCC_e unic pentru care : $mO_i \in MCC_e$ avem TAM $(MCC_e,mOi)=1$.

Se elimină coloanele corespunzătoare microoperațiilor ce sunt acoperite de clasele esențiale şi cele corespunzătoare microoperațiilor componente ale clasei MIC_m, obținându-se o tabelă de acoperire redusă :

TAR : $AMCC_m$ x $(MB(mO) \setminus MIC_m) \setminus \{MCC_e\}$ --> B

3. Se generează setul soluțiilor de acoperire a microoperațiilor $\{MCC_{ap}\}$: $\{MB(mO) \setminus MIC_m) \setminus \{MCC_e\}$

Fie SOL={SOL_1,SOL_2,..SOL_p} soluțiile de acoperire în care

$SO_j = \wedge / (\{MCC_e\} \cup \{MCC_{ap}\})$

Se consideră soluția parțială cu cardinalitatea minimă SOL_j.

Se generează soluția de acoperire a microoperațiilor ce aparțin clasei MIC_m, încă neacoperite: $\{MCC_{am}\}$.

4. Se generează setul soluțiilor de acoperire completă:

$SOLC_j=SOL_j \wedge \{MCC_{am}\}$.

Se calculează costul soluției de acoperire completă cu cardinalitatea minimă.

Dacă costul nu este minim (sau aproape minim, având în vedere şi considerațiile de ordin fizic ale componentelor memoriei de control), se alege o altă solutie de acoperire completă.

În cazul în care, toate soluțiile complete asociate unei soluții parțiale nu au cost minim, se alege o nouă soluție parțială pentru care se repetă etapa a 4-a.

Metoda propusă pentru codificarea microinstrucțiunilor complete nu va genera o soluție optimă, însă va genera o soluție într-un timp rezonabil.

Pentru multe probleme practice, soluția asigură o partiție optimă. De notat că, dacă numărul de biți obținut pentru codificarea microinstrucțiunilor este egal cu (4n+1) sau (8n+1) acesta se poate optimiza astfel încât să se obtină 4n sau 8n. Această optimizare este necesară datorită faptului că, în general, componentele fizice sunt organizate pe 4 sau 8 biți. Optimizarea propusă se poate face pe seama paralelismului, introducând relații de ordine parțială suplimentare în graful de dependență de date asociat microsubblocului.

Exemplul 2

Considerăm setul de microinstrucțiuni din cadrul microsubblocului :

MB={mO_1,mO_2,mO_3,mO_4,mO_5,mO_6,mO_7,mO_8,mO_9,mO_{10}}

Presupunem că pe baza dependenței de date şi a conflictului de resurse între microoperații a rezultat următoarea partiție a microsubblocului :

mIC_1={mO_1,mO_2,mO_4}
mIC_2={mO_1,mO_3,mO_5}
mIC_3={mO_2,mO_6,mO_8,mO_9}
mIC_4={mO_4,mO_5,mO_7,mO_8}
mIC_5={mO_3,mO_4,mO_5,mO_6,mO_7}
mIC_6={mO_6,mO_9,mO_{10}}
mIC_7={mO_7,mO_{10}}

Tabela de incompatibilitate este :

mO\ mO	mO_1	mO_2	mO_3	mO_4	mO_5	mO_6	mO_7	mO_8	mO_9	mO_{10}
m_1	x	x	x	x	x					
m_2	x	x		x		x		x	x	
m_3	x		x	x	x	x	x			
m_4	x	x	x	x	x	x	x	x		
m_5	x		x	x	x	x	x	x		

m_6		x	x	x	x	x	x	x	x	x
m_7			x	x	x	x	x	x		x
m_8		x		x	x	x	x	x	x	
m_9		x				x		x	x	x
m_{10}						x	x		x	x

Clasele maximale de compatibilitate sunt :

$MIC_1=\{mO_1,mO_2,mO_4\}$
$MIC_2=\{mO_1,mO_3,mO_4,mO_5\}$
$MIC_3=\{mO_2,mO_4,mO_6,mO_8\}$
$MIC_4=\{mO_2,mO_6,mO_8,mO_9\}$
$MIC_5=\{mO_3,mO_4,mO_5,mO_6,mO_7\}$
$MIC_6=\{mO_4,mO_5,mO_6,mO_7,mO_8\}$
$MIC_7=\{mO_6,mO_7,mO_{10}\}$
$MIC_8=\{mO_6,mO_9,mO_{10}\}$

Clasele maximale de compatibilitate sunt :

$MCC_1=\{mO_1,mO_6\}$
$MCC_2=\{mO_1,mO_7,mO_9\}$
$MCC_3=\{mO_1,mO_8,mO_{10}\}$
$MCC_4=\{mO_2,mO_3,mO_{10}\}$
$MCC_5=\{mO_2,mO_5,mO_{10}\}$
$MCC_6=\{mO_2,mO_7\}$
$MCC_7=\{mO_3,mO_8,mO_{10}\}$
$MCC_8=\{mO_3,mO_9\}$
$MCC_9=\{mO_4,mO_9\}$
$CC_{10}=\{mO_4,mO_10\}$
$CC_{11}=\{mO_5,mO_9\}$

Tabela de acoperire a microoperațiilor de către clasele maximale de compatibilitate este :

MCC\mO	mO_1	mO_2	mO_3	mO_4	mO_5	mO_6	mO_7	mO_8	mO_9	mO_{10}
MCC_1	x					x				
MCC_2	x						x		x	
MCC_3	x							x		x
MCC_4		x	x							x
MCC_5		x			x					x
MCC_6		x					x			
MCC_7			x					x		x
MCC_8			x						x	
MCC_9				x					x	
MCC_{10}				x						x
MCC_{11}					x				x	

Se observă că există două clase maximale de incompatibilitate cu cardinalitate 5 (MIC_5 şi MIC_6). Costul minim absolut, conform proprietății 4 este **C = 7**.

Să considerăm $AMCC_6$ - clasele maximale de compatibilitate asociate clasei maximale de incompatibilitate MIC_6 :

$AMCC_6=\{MCC_1,MCC_2,MCC_3,MCC_5,MCC_6,MCC_7,MCC_9,MCC_{10},MCC_{11}\}$

Conform propoziției 1 clasele maximale de compatibilitate din $AMCC_6$ acoperă toate microoperațiile microsubblocului.

Tabela de acoperire modificată este :

	mO_1	mO_2	mO_3	mO_4	mO_5	mO_6	mO_7	mO_8	mO_9	mO_{10}
MCC_1	x					**x**				
MCC_2	x						x		x	
MCC_3	x							x		x
MCC_5		x			x					x
MCC_6		x					x			
MCC_7			**x**					x		
MCC_9				x					x	
MCC_{10}				x						x
MCC_{11}					x				x	

Se observă că MCC_1 şi MCC_7 sunt esențiale şi vor face parte din soluția finală. Considerăm, conform algoritmului, acoperite microoperațiile incluse în clasele de compatibilitate maximă MCC_1 şi MCC_7 esentiale şi cele componente ale clasei de incompatibilitate maximă MIC_6. Astfel, din tabela de acoperire se elimină microoperațiile $mO_1,mO_6,mO_3,mO_8,mO_{10}$, respectiv mO_4,mO_5,mO_7.

Tabela de acoperire redusă este :

MCC \mO	mO_2	mO_9
MCC_2		x
MCC_5	x	
MCC_6	x	
MCC_9		x
MCC_{11}		x

Se observă că tabela de acoperire s-a redus substanțial. Acoperirea microoperațiilor mO_2 şi mO_9 se poate face cu ajutorul următoarelor clase maximale de compatibilitate :

MCC_2,MCC_5 MCC_2,MCC_6 MCC_5,MCC_9 MCC_5,MCC_{11} MCC_6,MCC_9 MCC_6,MCC_{11}

Se poate forma setul soluțiilor :

$SOL = \{SOL_1,SOL_2,SOL_3,SOL_4,SOL_5,SOL_6\}$ astfel :

SOL = {MCC_1 MCC_7 MCC_2 MCC_5 ; MCC_1 MCC_7 MCC_2 MCC_6 ;
MCC_1 MCC_7 MCC_5 MCC_9 ; MCC_1 MCC_7 MCC_5 MCC_{11} ;
MCC_1 MCC_7 MCC_6 MCC_9 ; MCC_1 MCC_7 MCC_6 MCC_{11}}

Considerând soluția : MCC_1 MCC_7 MCC_5 MCC_9 $\in$ SOL, se acoperă toate microoperațiile cu excepția $mO_7 \in MIC_6$. Pentru acoperirea lui mO_7 se poate lua una din clasele maximale de compatibilitate MCC_2 sau MCC_6.

Astfel setul soluțiilor complete este :

$SOLC_3$ = {$SOLC_{31}$,SOC_{32}}

unde

$SOLC_{31}$={MCC_1,MCC_7,MCC_5,MCC_9,MCC_2}
$SOLC_{32}$={MCC_1,MCC_7,MCC_5,MCC_9,MCC_6}

Ambele soluții complete au aceeaşi cardinalitate ce specifică numărul minim de câmpuri necesar pentru codificarea microinstrucțiunilor ce descriu microsubblocul.

Alegând $SOLC_{31}$, rezultă următoarea grupare a microoperațiilor:

{ mO_1 mO_7 mO_9 ; mO_2 mO_5 ; mO_3 mO_8 mO_{10} ; mO_4 ; mO_6 }

ce necesită 8 biți pentru codificare.

Soluția este acceptabilă deoarece are un cost apropiat de costul minim absolut.

Costul minim al acestei probleme, calculat prin metoda prezentată de Das şi Banerjee este tot 8, însă numărul de operații desfăşurat de algoritmul descris de ei este substanțial mai mare. Prin procedura prezentată se pot calcula soluții pentru multe probleme, însă în general, se stabilesc soluții aproape minim