Maxim paralelism in sisteme colaborative

ION IVAN, PhD
ACADEMY OF ECONOMIC STUDIES BUCHAREST
Email: ionivan@ase.ro

Cristian CIUREA
ACADEMY OF ECONOMIC STUDIES BUCHAREST
Email: cristian.ciurea@ie.ase.ro

Abstract. Se prezintă principiile din programarea producţiei în cazul sistemelor colaborative. Se determină activităţile secvenţiale şi cele paralele într-un sistem colaborativ şi se transformă activităţile secvenţiale în activităţi paralele prin utilizarea algoritmilor genetici, în scopul creşterii gradului de paralelism.

Cuvinte cheie: sisteme colaborative, activităţi secvenţiale, matrice de precedenţă, activităţi paralele, grad de paralelism, algoritmi genetici.

1. SISTEME COLABORATIVE

Sistemele colaborative reprezintă un domeniu interdisciplinar aflat la intersecţia economiei, informaticii, managementului şi sociologiei. Colaborarea implică organizaţii care au o misiune comună şi care se unesc pentru a forma o noua structură. Un sistem colaborativ informatic este asemenea unei firme de distribuţie al cărei obiectiv este de a vinde cantităţi din ce în ce mai mari din produsele sale.

Sistemele sunt formate din componente şi interacţiuni între acestea. Când sistemele colaborative sunt folosite în mod voluntar, unul dintre cei mai importanţi conducători către succes este modul în care utilizatorii simt experienţa lor cu sistemul, şi anume: dacă le place, dacă sistemul le oferă ceea ce aşteaptă de la el, dacă sunt capabili să comunice natural şi liber cu ceilalţi participanţi şi dacă doresc să-l recomande celorlalţi.

În cazul sistemelor colaborative de producţie, un proces decizional constă în adoptarea unor serii de acţiuni/comenzi privind funcţionarea sau modificarea structurii procesului de producţie condus în scopul creşterii performanţei acestuia. Obiectivele derivate, care precizează scopul general de mai sus sunt:

i) utilizarea eficientă a resurselor;
ii) creşterea calităţii produselor;
iii) micşorarea timpului de realizare a produselor;
iv) îmbunătăţirea securităţii muncii şi diminuarea efectelor negative asupra mediului, sau altfel spus, asigurarea desfăşurării de procese industriale "benigne" pentru operator şi mediu.

2. ACTIVITĂȚI SECVENȚIALE

În cadrul unui sistem colaborativ se desfăşoară o serie de activități A_1, A_2, ..., A_k în scopul atingerii anumitor obiective. Aceste activități se desfăşoară secvențial sau paralel, în funcție de duratele acestora, de importanța lor, de ordinea în care trebuie să se deruleze, dar şi în funcție de resursele disponibile pentru a fi alocate. Un sistem colaborativ presupune desfăşurarea acestor activități într-un timp cât mai scurt, având de rezolvat o problemă de alocare a resurselor.

Managementul de timp presupune următoarele etape:

- definirea activităților;
- secvențierea activităților;
- estimarea duratelor activităților;
- dezvoltarea agendei de lucru;
- controlul agendei de lucru.

Definirea activităților presupune identificarea şi documentarea îndeletnicirilor specifice ce trebuiesc realizate în vedera finalizării unui anumit proiect. Astfel, un proiect constă într-o înşiruire de activități cu un început şi sfârşit stabilite, activități ce sunt limitate de timp, resurse şi buget, în scopul obținerii unui obiectiv definit. În DEX online, *activitatea* este definită ca ansamblu de acte fizice, intelectuale şi morale făcute în scopul obținerii unui anumit rezultat. Etapa de definire a activităților necesită stabilirea:

- *intrărilor*: materiale de referință, istoricitatea activităților trecute şi viitoare, constrângerile şi ipotezele de lucru;
- *ieşirilor*: lista activităților, adică totalitatea activităților ce urmează a fi desfăşurate în cadrul proiectului şi materialele suport, adică întreaga documentație aferentă listei activităților;
- *tehnicilor*: descompunerea sau subdivizarea elementelor proiectului în componente mai mici şi mai uşor de gestionat.

Secvențierea activităților se referă la identificarea şi documentarea dependențelor dintre diferitele activități aferente unui proiect sau faze a acestuia. *Intrările* acestei etape sunt: lista activităților, descrierea produsului, dependențele impuse între diferitele activități, dependențele arbitrare, dependențele externe, constrângerile şi ipotezele de lucru. *Ieşirile* etapei sunt: diagrama proiectului şi lista actualizată a activităților. Tehnicile de secvențiere a activităților sunt:

- *metoda diagramei de precedență (Precedence Diagramming Method)*: se reprezintă prin diagrama proceselor, în care nodurile sunt activitățile, iar săgețile reprezintă dependențele dintre acestea. Tehnica se mai numeşte şi *activity on node* şi este utilizată în majoritatea produselor software pentru managementul proiectelor;
- *metoda diagramei cu săgeți (Arrow Diagramming Method)*: se reprezintă tot prin diagrama proceselor, săgețile se asociază activităților, iar nodurile reprezintă dependențele. Tehnica se mai numeşte şi *activity on arrow*;
- *metoda diagramelor condiționale*: introduce posibilitatea activităților non-secvențiale, cicluri sau condiționări, iar tipurile de diagrame sunt *Graphical Evaluation and Review Technique* şi *Sisteme Dinamice*.

Reprezentarea grafică a *n* activități secvențiale, A_1, A_2, ..., A_{i-1}, A_i, A_{i+1}, ..., A_n, este prezentată în figura 1:

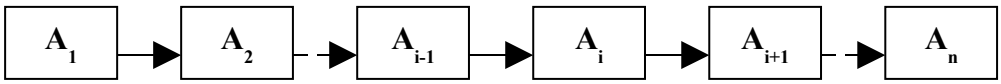

Fig. 1. Activități secvențiale

În cazul acestor *n* activităţi paralele care au loc într-un sistem colaborativ, ordinea de execuţie este următoarea: pentru *i=2..n-1*, se execută mai întâi activitatea A_{i-1}, apoi A_i şi, în final, activitatea A_{i+1}. Într-un sistem colaborativ reprezentat de o fabrică de pâine, procesul de panificaţie presupune execuţia secvenţială a activităţilor de *cântărire*, *amestec* şi *coacere*.

Diagrama logică a unui proiect care cuprinde şase activităţi este redată în figura 2:

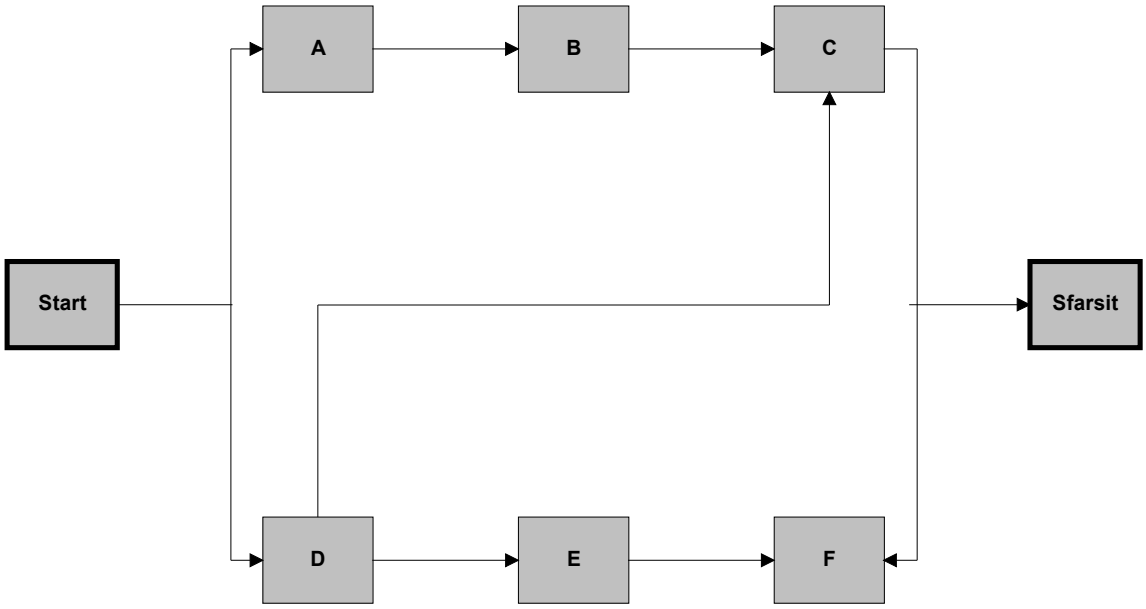

Fig. 2. Diagrama logică a unui proiect

Cele şase activităţi reprezentate în figura 2 se clasifică astfel:

- activităţile A, B, C sunt secvenţiale, deoarece trebuiesc executate pe aceleaşi echipamente şi în această ordine;
- activităţile D, E, F sunt tot secvenţiale;
- activităţile A, B, C şi D, E, F sunt paralele, deoarece se execută separat pe echipamente diferite şi în mod independent unele de altele.

Diagrama de tip reţea/graf reprezintă o metodă de a organiza şi aşeza secvenţial activităţile unui proiect. Aceasta arată drumurile/căile urmate de înlănţuirea activităţilor unui proiect, datele de începere/terminare, persoanele responsabile pentru fiecare activitate. Diagrama explică dintr-o privire cum se desfăşoară lucrul/munca la un proiect.

3. MATRICE DE PRECEDENŢĂ

Matricea de precedenţă este o reuniune a relaţiilor „<”, „=”, „>” prin care se marchează în ce fel de relaţie se află două elemente sau activităţi, respectiv se stabilesc care sunt activităţile ce preced sau succed alte activităţi. Secvenţa dintr-un program in C++ pentru calculul matricei de precedenţă în cazul a *n* activităţi este următoarea:

```
for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
            m[i][j]=' ';
bool prec=true;
for (i=0; i<n; i++)
{
```

```
        for (j=0; j<n; j++)
        {
                if (i<j)
                        if (m[i][j]==' ') m[i][j]='<';
                        else prec=false;
                if (i==j)
                        if (m[i][j]==' ') m[i][j]='=';
                        else prec=false;
                if (i>j)
                        if (m[i][j]==' ') m[i][j]='>';
                        else prec=false;
        }
}
```

Un program P este format din instructiunile I_1, I_2, … , I_n. Metrica McCabe ia în considerare graful G(V, A), unde V este mulțimea nodurilor şi A este mulțimea arcelor.

Se consideră programul următor:

```
void main ()
{                       I1
int a, b, c, d, e, x;
a= 3;                   I2
b = 5;                  I3
c =8;                   I4
b = 7;                  I5
scanf("%d", &x);        I6
if (x>0)                I7
e = a + b;              I8
else
e = c + d;              I9
printf("%d",e);         I10
}                       I11
```

Matricea P, a precedențelor, asociată grafului G al acestui program, este următoarea:

$$P = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \\ I_8 \\ I_9 \\ I_{10} \\ I_{11} \end{array} \begin{array}{c} \begin{array}{ccccccccccc} I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 & I_8 & I_9 & I_{10} & I_{11} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Precedența este rezultatul dispunerii instrucțiunilor în program în aşa fel încât acestea să conducă la obținerea de rezultate corecte. O problemă se obține prin scrierea a N programe diferite, care însă implementează acelaşi algoritm. Diferențele care apar sunt determinate de: limbajul utilizat, structurile de date definite si referite, structurile de control folosite, gradul de generalitate, dimensiunea problemelor care se rezolvă, precizia rezultatelor, opțiunile privind datele de intrare si structura rezultatelor obținute, nivelul de reutilizabilitate a procedurilor, a tipurilor de date, a claselor.

Matricele de precedență diferă de la program la program, chiar dacă este vorba de rezolvarea aceleeaşi probleme. Pentru că instrucțiunile, succesiunea lor determină grafuri diferite.

4. ACTIVITĂȚI PARALELE

Împărțirea unui proiect în activități/sarcini este necesară din următoarele motive:

a) proiectul apare ca o înşiruire logică de lucrări secvențiale care ajută la determinarea graficului optim de timp. Totodată, având proiectul divizat în astfel de elemente unitare este mai uşor de văzut cum se îmbină acestea, cum se suprapun sau interferă unele cu altele dacă nu sunt efectuate la timp;

b) activitățile sunt o modalitate de a diviza proiectul în segmente uşor de manevrat. Prin urmare, urmărirea progresului la nivelul activităților reprezintă o modalitate de evaluare şi control al muncii/lucrării desfăşurate în cadrul proiectului.

În cadrul unui sistem colaborativ există proiecte cu un număr mai mic sau mai mare de activități care se suprapun. Eficientizarea derulării proiectului presupune căutarea tuturor activităților care se execută în paralel, chiar şi transformarea unor activități secvențiale în activități paralele, în funcție de resursele şi echipamentele disponibile.

Planificarea graficului de timp este cel mai dificil şi riscant aspect al planificării activităților din cadrul unui sistem colaborativ. Timpul este strâns legat de lista de activități, de diagrama de tip rețea şi de buget. Dacă una din acestea suferă o schimbare, le va afecta pe celelalte. Planificarea graficului de timp înseamnă convertirea activităților aşezate secvențial în diagrama de tip rețea şi a listei de resurse şi furnizori într-un program zilnic/săptămânal cu date de începere, terminare şi responsabilități atribuite fiecărei activități.

Reprezentarea grafică a *n* activități paralele, A_1, A_2, ..., A_n, şi B_1, B_2, ..., B_n, este prezentată în figura 3:

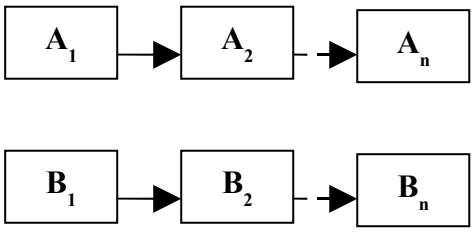

Fig. 3. Activități paralele

Un exemplu de desfăşurare a unor activități paralele este dat de problema procesoarelor, respectiv de execuția unor lucrări de diferite durate pe mai multe procesoare. Obiectivul este ordonarea execuției lucrărilor pe respectivele procesoare astfel încât timpul total de execuție să fie minim. Codul sursă pentru acest program realizat în C++ este următorul:

```
#include<stdio.h>
#include<conio.h>

int min(int *x,int n,int *poz)
{
int i,m=1000;
*poz=0;
for(i=0;i<n;i++)
if (m>x[i])
{
m=x[i];
```

```
*poz=i;
}
return m;
}

int max(int *x, int n)
{
int i,m=0;
for(i=0;i<n;i++)
if (m<x[i]) m=x[i];
return m;
}

void sortare(int *x, int n)
{
int i,j,aux;
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
if(x[i]<x[j])
{
aux=x[i];
x[i]=x[j];
x[j]=aux;
}
}

void main()
{
int n,m,i,t_min,poz;
int lucr[15],timp[15];
printf("Numarul procesoarelor, n=");  scanf("%d",&n);
printf("Numarul lucrarilor, m=");  scanf("%d",&m);
printf("Durata lucrarilor: \n");
for(i=0;i<m;i++)
{
printf("lucrarea %d:",i+1);
scanf("%d",&lucr[i]);
}
for(i=0;i<n;i++) timp[i]=0;
sortare(lucr,m);
if(m<=n)
t_min=max(lucr,m);
else
{
for(i=0;i<n;i++) timp[i]=lucr[i];
while(i<m)
{
min(timp,n,&poz);
timp[poz]+=lucr[i++];
}
t_min=max(timp,n);
}
printf("Timpul minim: %d\n",t_min);
getch();
}
```

Un exemplu de test pentru acest algoritm este prezentat în figura 4:

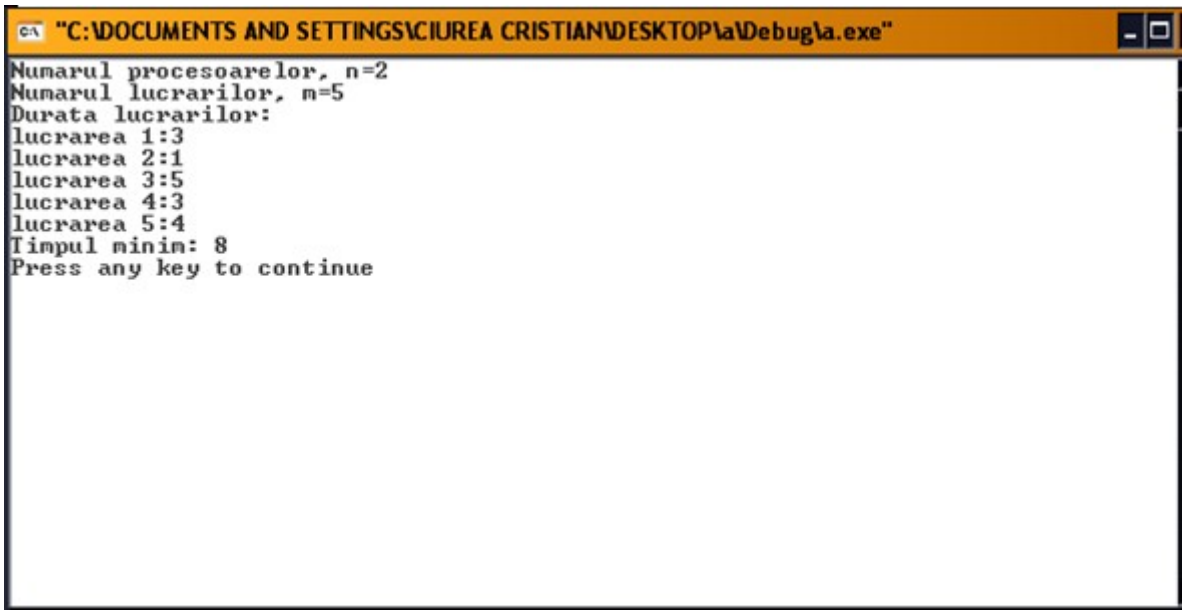

Fig. 4. Exemplu de test pentru problema procesoarelor

Din figura 4 se constată că pentru un număr de două procesoare şi cinci lucrări, având duratele 3, 1, 5, 3, 4, timpul minim de execuţie al acestor lucrări este de 8 unităţi.

Referitor la activităţile paralele, s-au dezvoltat algoritmi pentru procesarea paralelă a matricilor, prin care matricile sunt prelucrate la nivel de bloc, spre deosebire de abordarea clasică, în care matricile sunt prelucrate la nivel de element.

5. GRADUL DE PARALELISM

Toate activităţile şi fenomenele din cadrul unui sistem colaborativ trebuie să fie exprimate în unităţi de timp calendaristice. În acest fel, aceste fenomene diferite sunt aduse la acelaşi numitor comun având aceeaşi unitate de exprimare (timpul). Timpul necesar executării unei sarcini de producţie la o operaţie tehnologică, exprimat în zile lucrătoare, se calculează astfel:

$$T_{ij} = \frac{Q_j * NT_{ij}}{nu_i * ns * ds}$$, unde:

T_{ij} este timpul necesar executării operaţiei i la sarcina de producţie j;
Q_j este cantitatea realizată la sarcina de producţie j;
NT_{ij} reprezintă norma de timp;
nu_i reprezintă numărul de utilaje necesare;
ns este numărul de schimburi;
ds este durata unui schimb.

Începerea lucrărilor se face având la bază termenul de livrare contractuală a produselor. Pe lângă acestea se stabilesc termene intermediare care vor constitui şi criterii de apreciere a continuităţii procesului de producţie.

Programarea conducerii operative a producţiei trebuie să asigure paralelismul în execuţie a diferitelor sarcini în vederea reducerii perioadei de imobilizare a capitalului aflat sub forma producţiei neterminate.

Determinarea duratei de execuție a unui program operativ trebuie să fie precedată de analiza tehnologiilor specifice fiecărei sarcini. În cazul unor sarcini independente din punct de vedere tehnologic durata de execuție a programelor operative este dată de sarcina care are durata cea mai mare.

În cazul în care tehnologiile sarcinilor de producție ce compun programul operativ sau a unora dintre acestea sunt comune sau identice, urmează a se stabili succesiunea de execuție. Durata de execuție a întregului program operativ va corespunde cu momentul terminării ultimei sarcini programate la ultima operație.

Prin programarea producției trebuie să se asigure concordanța între sarcinile stabilite repartizate şi potențialul tehnico-productiv a fiecărei subunități sau grupe de maşini. Gradul de participare a unei verigi de producție la realizarea unei sarcini de producție se stabileşte prin volumul de muncă necesar executării ei:

$$Ttn_{ij} = Q_j * NT_{ij}$$

Pe baza volumului de muncă şi cunoscând durata impusă pentru executarea acestor sarcini se determină numărul de muncitori necesari executării ei:

$$Nm_{ij} = \frac{Ttn_{ij}}{D_{ij} * 8}$$

Compararea volumului de muncă necesar cu potențialul tehnico-productiv al perioadei de execuție se face cu ajutorul balanței operative de corelare capacitate-încărcare:

$\Delta Tdi = Tdi - Ttni = D_i * nu_i * ns * ds - \sum Q_j * NT_{ij}$, unde:

Tdi este timpul disponibil;
Ttni este timpul total necesar.

Conducerea operativă a producției trebuie să aibă la bază criteriul optimalității care presupune elaborarea mai multor variante şi alegerea variantei optime.

Gradul de paralelism din cadrul unui sistem colaborativ se determină ca raport între numărul de activități paralele şi numărul total de activități:

$$GP = \frac{AP}{AP + AS}$$, unde:

GP este gradul de paralelism;
AP este numărul activităților paralele;
AS este numărul activităților secvențiale.

Gradul de paralelism este un indicator care ia valori în intervalul [0; 1], valoarea 0 însemnând că toate activitățile din sistem sunt secvențiale, iar valorea 1 că toate activitățile se desfăşoară în paralel. Cu cât numărul activităților paralele este mai mare, cu atât gradul de paralelism se apropie de valoarea 1, iar sistemul colaborativ este mai eficient.

6. UTILIZAREA ALGORITMILOR GENETICI ÎN CREŞTEREA GRADULUI DE PARALELISM

În prezent se constată o mai mare prudență în aprecierea primordialității Inteligenței Artificiale (IA) asupra modelelor numerice. În plus, metodele IA nu conduc în mod necesar la soluții optimale, care uneori nu sunt totuşi cerute în mod expres, ci mai degrabă la soluții satisfăcătoare. În realitate, calea cea mai promițătoare s-a dovedit cea a combinației sinergice dintre IA şi metodele de optimizare şi simulare.

Algoritmii genetici fac parte din categoria algoritmilor euristici, ei aplicându-se cu succes în cazul problemelor ce nu admit algoritmi în timp polinomial. Algoritmii genetici, după cum sugerează şi numele, sunt inspirați din natură, mai precis din felul în care prin recombinări genetice se îmbunătățeşte o specie.

Ideea algoritmilor genetici este de a reprezenta soluțiile posibile ale problemei sub forma unor cromozomi şi de a lucra, la fiecare pas, cu un număr fix de cromozomi, care formează o populație. Astfel, se incearcă îmbunătățirea populației de cromozomi în limita timpului disponibil, în sensul apropierii cât mai mult de soluția optimă.

În acest caz, populația este reprezentată de un sistem colaborativ, iar cromozomii de activitățile desfăşurate în cadrul sistemului. Soluția optimă rezultată din aplicarea unui algoritm genetic se referă la obținerea unui grad maxim de paralelism, adică determinarea cât mai multor activități care se derulează în paralel.

La primul pas, se aleg aleator *n* activități A_1, A_2, …, A_n din totalul activităților. Pentru evaluarea sistemului de activități se calculează:

- funcția obiectiv, prin convertirea fiecărei activități într-o valoare reală şi determinarea valorii funcției în punctul reprezentat de activitatea A_i;
- suma valorilor funcțiilor obiectiv;
- probabilitatea de selecție pentru fiecare activitate;
- probabilitatea cumulativă de selecție pentru fiecare activitate. Aceste probabilități se notează cu p_1, p_2, ..., p_n, reprezentând un şir crescător a cărui ultimă valoare este 1. Cu cât A_{i+1} conține o valoare pentru care se obține o valoare mai mare pentru funcția obiectiv, cu atât diferența dintre p_{i+1} şi p_i va fi mai mare. Astfel, şirul probabilităților cumulative de selecție reprezintă o diviziune a intervalului [0,1]. Pentru crearea unui sistem intermediar de activități, selectam *n* numere uniform aleatoare în intervalul (0,1]. Daca un număr se gaseşte în intervalul $(p_i, p_{i+1}]$, atunci activitatea A_{i+1} este selectată. Se observă că probabilitatea ca o activitate să fie selectată este cu atât mai mare cu cât intervalul $(p_i, p_{i+1}]$ este mai mare.

Se observă uriaşa asemănare între algoritmii genetici şi viața de toate zilele. Deşi cromozomii valoroşi au şanse mai mari să ajungă în noua populație, există şi posibilitatea ca unii dintre ei să se piardă. Important nu este atât cromozomul, ci populația de cromozomi.

7. CONCLUZII

Ideea de sistem colaborativ se referă la un sistem eficient, în care oamenii şi echipamentele cooperează în scopul atingerii anumitor obiective. Este vorba despre Human-Human-Interaction şi Human-Computer-Interaction. Activitățile care se desfăşoară într-un astfel de sistem trebuiesc eficientizate prin analiza şi separarea lor. Scopul final este obținerea unui paralelism maxim, adică execuția cât mai multor activități în paralel, în vederea optimizării sistemului.

BIBLIOGRAFIE

[ARBA05] Raluca ARBA – *Collaborative Electronic Marketplace*, in International Workshop „Collaborative Support Systems in Business and Education”, Cluj-Napoca, october 2005, pag. 11.

[BRES05] Vasile Paul BRESFELEAN, Mihaela Antonia BURCA – *Collaborative E-learning Systems,* in International Workshop „Collaborative Support Systems in Business and Education”, Cluj-Napoca, october 2005, pag. 17.

[DOBR05] Ovidiu DOBRICAN – *An Example of Collaborative System*, in International Workshop „Collaborative Support Systems in Business and Education”, Cluj-Napoca, october 2005, pag. 48.

[IVAN99] I. IVAN, A. KARADIMOU, A. LICURICEANU, G. LUPU – *Metrici de complexitate software bazate pe dependențele instrucțiunilor,* Revista Informatică Economică, nr. 11, 1999.

[MICA05] Daniel MICAN – *Collaborative System in Handling Freelancer IT Projects*, in International Workshop „Collaborative Support Systems in Business and Education”, Cluj-Napoca, october 2005, pag. 196.

[MOCE05] Loredana MOCEAN, Robert BUCHMANN, Ciprian-Nicuşor DRAGOTĂ – *The Proceedings of the International Workshop on COLLABORATIVE SUPPORT SYSTEMS IN BUSINESS AND EDUCATION,* October 28-29, 2005, Risoprint, Cluj-Napoca, Romania.

[MUNT05] Mihaela MUNTEAN – *Collaboration – A Business Strategy in The Global Economy,* in International Workshop „Collaborative Support Systems in Business and Education”, Cluj-Napoca, october 2005, pag. 245.