Utilizarea arhitecturilor paralele în TSP folosind Ant colony

Ionuț Balan, Ștefan Gheorghe Pentiuc

Abstract—**Optimization is an important tool in making decisions and analysis of physical systems. Currently, a wide variety of real-life problems requires a multitude of calculations to solve them. Parallel architectures are used quite often in these calculations to obtain the best results in shortest possible time. Population-based methods offer the greatest degree of competition, so a large number of objective function evaluations can be achieved in parallel, achieving the desired solution after attending a number of generations. This paper aims to emphasize the results obtained by using a hibrid ant colony algorithm in travelling salesman problem.**

Index Terms—**ACO, mutation, TSP, MPI, random, pheromone trail, parallel**

I. INTRODUCERE

TEORIA optimizării constă în studierea matematică a unei probleme date, în cadrul cărei se caută o valoare de maxim sau de minim pentru o funcție obiectiv într-un domeniu dat. Aceasta include căutarea unei soluții posibile, a proprietăților problemei studiate, precum şi a aspectelor algoritmului propus pentru rezolvarea acesteia. Importanța studierii teoriei optimizării a crescut cu trecerea timpului. Aceasta s-a datorat, în special, multitudinii de domenii în care aceasta acționează, incluzând matematica aplicată, știința calculatoarelor, ingineria, economia, etc.

Ant Colony Optimization este o paradigmă folosită în rezolvarea problemelor de optimizare combinatorială. Primul algoritm ce poate fi inclus în această categorie a fost prezentat în 1991, şi de atunci, mai multe variante a acestui principiu au fost menționate în literatura de specialitate[1]. Aceşti algoritmi sunt incluşi în categoria algoritmilor metaeuristici. Această categorie de algoritmi ne ajută la obținerea unor rezultate pozitive prin evitarea unor optime locale fie prin generarea unui punct din spațiu de căutare şi adăugarea, pe rând, a câte un punct până la obținerea unei soluții optime, fie prin generarea unei soluții complete, urmată de modificarea iterativă a unor elemente din ea pentru obținerea unei soluții corespunzătoare. Comportamentul emergent [2], caracterizat drept nondeterministic al sistemelor naturale poartă această denumire deoarece se formează din comportamentele multor agenți simpli care urmăresc un set limitat de reguli simple bazate doar pe viziunea locală a acestora asupra sistemului. Agenții individuali nu sunt conştienți de starea globală a sistemului sau de scopurile funcționale finale ale sistemului.

Comunicația (interacțiunea) este imperativă atât în cazul sistemelor deterministice, artificiale cât şi cele emergente, naturale. În domeniul natural, studiul coloniilor de furnici a relevat faptul că schimbul de feromoni joacă un rol foarte important în comunicația în interiorul muşuroiului.

În analogie cu biologia, Ant Colony Optimization este bazată pe comunicația indirectă dintr-o colonie de agenți, denumiți (artificial) furnici, mediată de intensitatea feromonilor artificiali [3]. Intensitatea feromonilor, pentru această categorie de algoritmi, reprezintă informație numerică, distribuită, folosită de către furnici pentru construcția probabilistică a unei soluții pentru problema ce se dorește a fi rezolvată, care în cele din urmă suferă anumite modificări pe parcursul execuției algoritmului în funcție de experiența dobândită de agenți.

În rezolvarea problemei, fiecare furnică traversează un graf care reprezintă o instanță a problemei şi se foloseşte de două tipuri de informații care sunt comune întregii colonii [4]:

informația euristică, care depinde de instanța problemei specificate. Este creată înainte de rularea algoritmului şi rămâne fixă pe parcursul lui. Reprezintă, de fapt, valoarea asociată fiecărei muchii (i, j) fiind notată cu η_{ij};

informația ce descrie intensitatea feromonilor, care este modificată pe parcursul rulării algoritmului şi depinde de numărul de furnici care traversează fiecare muchie şi de numărul de soluții generate de acestea. Această informație este reprezentată sub forma unei matrici de feromoni, $\tau = [\tau_{ij}]$, ce mimează feromonul real pe care îl depun furnicile în deplasările lor.

Folosind acest comportament al furnicilor putem rezolva o serie de probleme de optimizare cum ar fi: travelling salesman problem, job-shop scheduling problem, vehicle routing problem, assignement problem, set problem şi altele. Pentru a evidenția avantajele oferite prin folosirea acestei tehnici, hibridizate, prin introducerea anumitor tehnici specifice algoritmilor genetici, în această lucrare am recurs la rezolvarea travelling salesman problem. Travelling salesman problem este o problemă de optimizare combinatorială studiată destul de

Ionuț Bălan este preparator în cadrul Catedra de Informatică a Facultății de Științe Economice şi Administrație Publică din cadrul Universității „Ştefan cel Mare" Suceava. (e-mail: ionutb@seap.usv.ro). În prezent este doctorand în cadrul Universității „Ştefan cel Mare" Suceava, domeniul Calculatoare şi Tehnologia Informației.

Ştefan Gheorghe Pentiuc este profesor doctor inginer în cadrul Catedrei de Calculatoare a Facultății de Inginerie Electrică şi Știința Calculatoarelor, Universitatea "Ştefan cel Mare" Suceava (e-mail: pentiuc@eed.usv.ro).

des în ştiința calculatoarelor. Fiind dată o listă de oraşe şi distanțele dintre acestea, scopul acestei probleme constă în determinarea celui mai scurt drum care trece prin fiecare oraş o singură dată. Specificul acestei problemei este dat de faptul că orașul de început al drumului de parcurs trebuie să coincidă cu orașul de sfârşit.

Lucrarea de față este structurată pe 6 capitole: capitolul 2 ne prezintă o scurtă descriere a ant colonies optimization, capitolul 3 un mod de hibridizare a tehnicilor specifice ant colonies, capitolul 4 descrie modul de paralelizare a algoritmului folosit, capitolul 5 rezultatele obținute cu interpretarea lor, iar capitolul 6 reprezintă concluziile referitoare la această lucrare.

II. ANT COLONY OPTIMIZATION

Algoritmul Ant Colony Optimization este o tehnică probabilistică pentru rezolvarea problemelor computaționale care se rezumă la aflarea celor mai scurte drumuri într-un graf. Propus inițial de Marco Dorigo în 1992, în teza sa de doctorat, primul algoritm a avut drept scop căutarea drumului optim într-un graf ; căutare bazată pe comportamentul furnicilor în găsirea unui drum între colonia de care aparțineau şi o sursă de mâncare. Din acel moment, ideea a suferit mai multe modificări, scopul acestora fiind posibilitatea rezolvării unei game variate de probleme.

Furnicile artificiale au un dublu scop [5]. Pe de o parte, sunt o abstractizare comportamentală a furnicilor reale care caută cel mai scurt drum spre o sursă de hrană. Pe de altă parte, ele dispus de anumite capacități ce nu sunt întâlnite în cazurile reale. De fapt, folosim furnicile artificiale pentru optimizarea problemelor dificile, ce nu se pot rezolva prin metodele obişnuite, drept urmare, suntem îndreptățiți să atribuim acestora anumite capabilități ce nu pot fi întâlnite în cazurile reale, ceea ce duce la creşterea gradului de eficiență a sistemelor bazate pe aceste tehnici.

În cazul furnicilor artificiale feromonul exprimă experiența coloniei în găsirea drumului celui mai scurt, în timp ce memoria şi informațiile euristice exprimă cunoştințele uzuale despre problema pe care un astfel de sistem trebuie s-o rezolve. În acest sens, probabilitatea de alegere a unei variante în determinarea soluției optime este dată de [6]:

$$p_{ij}^{k}(t)=\begin{cases}\tau_{ij}^{\alpha}*\eta_{ij}^{\beta}\Big/\sum_{k\notin\Gamma}^{m}\tau_{ij}^{\alpha}*\eta_{ij}^{\beta} & \text{dacă } j\notin\Gamma\\ 0 & \textit{in caz contrar}\end{cases} \quad (1)$$

unde τ_{ij} este concentrația de feromoni pe muchia (i,j), η_{ij} este o funcție euristică, în timp ce Γ este o listă tabu. Funcția euristică η contribuie la determinarea soluției de căutare prin furnizarea unor informații specifice problemei de optimizat, iar lista tabu conține toate muchiile pe care furnicile le-au vizitat şi care nu trebuie alese de două ori în același drum, având rol de memorie a furnicilor. Parametrii α şi β măsoară importanța relativă a căilor cu feromoni (experiența) şi, respectiv, euristica locală (cunoştințele).

La fiecare iterație o nouă colonie de furnici (număr dat de specificul problemei) este trimisă în căutare de hrană. După alegerea unui anume drum, furnicile depozitează o anumită cantitate de feromoni, de obicei la terminarea traseului, pe calea aleasă. Un tur este o rută completă dintre muşuroi şi sursa de hrană, iar o iterație este echivalentă cu durata parcursă de cele n furnici de la muşuroi la sursa de hrană. Actualizarea concentrației de feromoni pe muchiile grafului este dată de următoarea expresie [6]:

$$\tau_{ij}(t+m*g)=\tau_{ij}(t)*(1-p)+\sum_{k=1}^{g}\Delta\tau_{ij}^{k} \quad (2)$$

unde $p\in[0,1]$ reprezintă fenomenul de evaporare a feromonului, iar $\sum_{k=1}^{g}\Delta\tau_{ij}^{k}$ reprezintă cantitatea de feromon ce se stochează pentru muchia (i,j) parcursă de g furnici într-o iterație, care este definită în felul următor [6]:

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{1}{f_k} & \textit{daca arcul } (i,j) \textit{ a fost parcurs de furnica } k\\ 0 & \textit{in caz contrar}\end{cases} \quad (3)$$

unde f_k este rezultatul unei funcții de evaluare pentru fiecare furnică k într-o problemă de minimizare.

În cazul furnicilor reale, timpul are un rol important în găsirea drumului optim, deoarece ele nu pot porni în căutarea soluției în același timp, în schimb, în cazul furnicilor artificiale, timpul este același, deoarece toate furnicile sunt trimise în același timp şi găsesc o soluție parcurgând drumul în același interval de timp.

III. HIBRIDIZARE ALGORITM

Formulându-se un anume model, un algoritm de optimizare poate fi utilizat pentru găsirea soluțiilor lui. De obicei, atât algoritmul cât şi modelul sunt destul de complicate, astfel încât avem nevoie de ajutorul unui sistem de calcul modern pentru implementarea acestor procese. Algoritmii de optimizare sunt diferiți de la problemă, la problemă, ei nefiind universali. În alegerea lor, un rol important îl are utilizatorul, care se orientează după tipul problemei de optimizare studiate. Astfel, în funcție de metoda aleasă, o problemă poate fi rezolvată mai repede, sau mai lent, ori poate da un rezultat corect (optim global) sau unul eronat (optim local).

Față de alți algoritmi, metoda de optimizare ce foloseşte tehnica ant colony ne oferă soluții destul de bune. Cu toate acestea, prin folosirea acestei tehnici în găsirea unei soluții optime, putem ajunge într-un punct de optim local, punct din care sunt şanse mici să mai ieşim.

Pentru îmbunătățirea rezultatelor aferente unei probleme

rezolvată prin tehnica ant colony, am recurs la hibridizarea acestei tehnici, prin introducerea unui element specific algoritmilor genetic, şi anume operația de mutație.

Prin introducerea acestei operații, algoritmul nou obținut poate fi descris prin pseudocodul următor:

```
/*ant colonies secvențial*/
    Initializari
Pentru i de la 1 la numarul de pasi executa
   Pentru k de la 1 la numarul de furnici
       Genereaza un oras
       Selecteaza pe rand cele mai apropiate
       orase in functie de matricea de
       probabilitati(feromoni)
   //sfarsit pentru k
   Aplica mutatii
   Modifica matricea probabilitatilor in
   functie de traseele parcurse
//sfarsit pentru i
Afisare solutie optima
/*sfârşit program*/
```

Considerând că un drum parcurs de o furnică este asemănător ca structură unui cromozom din algoritmii genetici, această mutație poate fi efectuată destul de uşor. Astfel, în cazul de față, mutațiile afectează anumite trasee parcurse de furnici într-o iterație. Aceste trasee sunt generate aleator, iar numărul lor va depinde de specificul problemei. Mutația are loc în felul următor:

- genele supuse operației de mutație sunt generate aleator (două la număr);
- genele generate anterior sunt interschimbate;
- din genele intermediare rezultate se vor alege pe rând doar acele gene ce sunt mai apropiate nodului curent (oraşului curent);

Un exemplu de mutație este reprezentat în figura 1:

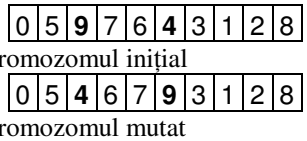

Fig. 1. Operatorul de mutația aplicat cromozomului.

IV. PARALELIZARE ALGORITM

Creşterea continuă a complexității aplicațiilor executate de sisteme de calcul a impus căutarea unor soluții adecvate pentru asigurarea unor performanțe corespunzătoare. Reducerea timpului de execuție s-ar putea realiza prin folosirea unei tehnologii mai rapide, a unei arhitecturi paralele, a unor algoritmi mai performanți (paraleli) sau o combinație a tuturor acestor elemente [7]. Prin intermediul arhitecturilor paralele putem obține rezultate satisfăcătoare în cazul anumitor probleme.

În cazul de față am folosit arhitecturile paralele pentru obținerea unor rezultate satisfăcătoare pentru seturile de date studiate. Astfel, am implementat un ant colony insular (populațiile au evoluat în paralel, comunicând între ele la un moment dat, trimițând unul altuia cea mai bună rută) ce poate fi descris prin pseudocodul ce urmează:

```
/*ant colonies paralel*/
Initializari
    Pentru i de la 1 la numarul de pasi executa
       Pentru k de la 1 la numarul de furnici
executa
          Genereaza un oras
          Selecteaza pe rand cele mai apropiate
          orase in functie de matricea de
          probabilitati(feromoni)
       //sfarsit pentru k
       Aplica mutatii
       Modifica matricea probabilitatilor in
       functie de traseele parcurse
       Daca numarul de generatie= m*1/5 *nr total
       de generatii
         //m=1,2,3,4,5
          Comunicatii intre procese
       //sfarsit daca
    //sfarsit pentru i
    Afisare solutie optima
    /*sfârşit program*/
```

Comunicațiile dintre procese se realizează folosind librăria MPI (Message Passing Interface) [8], implementată cu ajutorul limbajului C.

V. REZULTATE OBȚINUTE

Pentru a scoate în evidență îmbunătățirile aduse atât de hibridizare, cât şi de paralelizare am prezentat în tabelul numărul 1 rezultatele obținute pentru patru variante de algoritmi: AS – varianta ant colony secvențială, AP – varianta ant colony paralelă, AHS – varianta ant colony hibridizată secvențială, AHP – varianta ant colony hibridizată paralelă. Datele de intrare pentru toate variantele studiate reprezintă distanțele stradale dintre 29 de oraşe din Bavaria (bays29) [9]. Distanța optimă, cunoscută, pentru acest set de date este 2020 km. Numărul de furnici ce caută soluția optimă pe fiecare maşină, indiferent de variantă este 25, numărul de paşi (iterații) în care se execută problemă este de 200. Pentru variantele hibride, la fiecare iterație se execută 20 de mutații, în timp ce, pentru variantele paralele, se folosesc 5 noduri pentru obținerea rezultatului final, comunicațiile dintre noduri efectuându-se la iterațiile: 40, 80, 120, 160 şi 200.

TABEL 1. REZULTATELE OBȚINUTE PENTRU PROBLEMA DE TEST

	AS km	AS paşi	AP km	AP paşi	AHS km	AHS paşi	AHP km	AHP paşi
1	2160	20	2043	20	2056	29	2022	63
2	2225	38	2081	41	2069	103	2020	71
3	2344	1	2101	6	2063	19	2022	103
4	2317	1	2101	6	2038	198	2022	89
5	2342	152	2047	83	2022	54	2022	107
6	2356	1	2047	124	2094	147	2022	43
7	2223	6	2047	50	2044	195	2020	182
Avg	2281	31	2067	47	2055	106	2021	94
Best	2160	152	2043	124	2022	198	2020	182

Comparând cele patru variante din punct de vedere al distanței obținute se observă clar că variantele hibride oferă rezultate mult mai bune decât celelalte două. Chiar dacă am recurs la paralelizarea variantei simple, varianta hibridă secvențială ne oferă rezultate mult mai bune (figura 2).

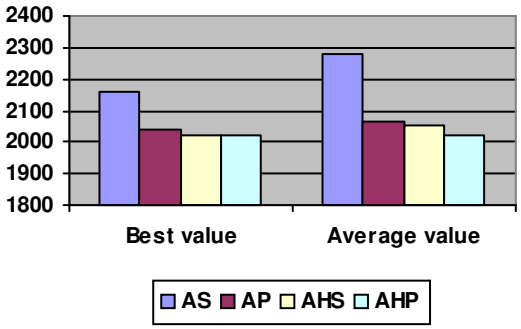

Fig. 2. Cele mai bune valori şi mediile obţinute în cele patru cazuri considerate.

Se observă că cele mai bune rezultate ne sunt oferite de către algoritmii hibrizi ce descriu comportamentul furnicilor, iar metodele paralele sunt mult mai eficiente decât cele secvenţiale pentru fiecare caz în parte. În cazul ant colony hibrid se observă că cel mai bun rezultat obţinut secvenţial este apropiat de cel mai bun rezultat obţinut în paralel, dar media rezultatelor obţinute în paralel pentru varianta simplă şi cea hibridă este net superioară mediei rezultatelor obţinute secvenţial pentru cele două variante, de aici putând trage concluzia că metoda paralelă este superioară celei secvenţiale.

Din aceste rezultate putem trage şi concluzii referitoare la eficienţa folosirii resurselor de către algoritmul. Această eficienţă ne va fi dată de către numărul de paşi în care se obţine o soluţie (numărul de paşi din tabelul 1). Astfel dacă avem la dispoziţie 200 de paşi, iar o soluţie (fie ea şi optim local) s-a obţinut chiar la primul pas, considerăm ca metoda utilizează ineficient resursele avute la dispoziţie, adică celelalte 199 de iteraţii nu contribuie la îmbunătăţirea rezultatului final. Cu cât o soluţie finală e îmbunătăţită spre ultima iteraţie, cu atât vom considera acea metodă mai eficientă din punctul de vedere al utilizării resurselor. La o primă privire asupra datelor din tabelul 1 (asupra rezultatului Best aferent numărului de paşi), am putea considera AHS (198 paşi) ca fiind cel mai eficient din punctul de vedere precizat mai sus, iar AP (124 paşi) ca fiind cel mai ineficient. Privind rezultatele din punctul de vedere al valorii medii al paşilor la care se obţine rezultatul final se observă că ierarhia se modifică, astfel, cel mai eficient va fi considerat AHS (106,43 paşi), iar cel mai ineficient fiind AS (31,29). Nici una dintre cele două afirmaţii de mai sus nu descrie adevărul faptic. Pentru a scoate în evidenţă adevărata ierarhie în ceea ce priveşte eficienţa considerată vom lua medianele acelor variante ca puncte de reper. Ele sunt prezentate în tabelul numărul 2.

TABEL 2. MEDIANELE NUMĂRULUI DE PAŞI ÎN CARE SE OBŢINE REZULTATUL

	AS (paşi)	AP (paşi)	AHS (paşi)	AHP (paşi)
Median	6	41	103	89

Din acest al doilea tabel rezultă că cea mai eficientă variantă este cea hibridă secvenţială, urmată de cea hibridă paralelă, simplă paralelă, pe ultimul loc fiind cea simplă secvenţială.

O altă explicaţie a cestor afirmaţii ar putea fi dată de incapacitatea soluţiei simple secvenţiale de a evolua spre soluţia optimă cunoscută, ceea ce conduce la ocuparea ultimei poziţii din punctul de vedere al eficienţei, iar poziţia a doua ocupată de către varianta hibridă paralelă este dată de obţinerea optimului posibil înainte de limită, ceea ce duce la rularea fără rezultat a algoritmului până la terminarea numărului de paşi consideraţi iniţial.

VI. CONCLUZII

Ant colony este o tehnică des întâlnită în anumite probleme de optimizare. Totuşi folosind numai această tehnică nu suntem siguri că vom obţine o soluţie satisfăcătoare. Pentru îmbunătăţirea rezultatelor obţinute prin această metodă se poate recurge la hibridizarea ei prin introducerea de elemente aferente altor tehnici cunoscute sau mai puţin cunoscute.

O altă metodă de a îmbunătăţii rezultate este apelarea la arhitecturile paralele, soluţie prin care putem obţine rezultate ce sunt mult mai bune decât cele obţinute prin variantele secvenţiale. Prin combinarea celor două metode: hibridizare şi paralelizare obţinem ant colony ce ne furnizează soluţii bune, atingându-se în anumite situaţii optimul cunoscut în literatura de specialitate.

În lucrările ce urmează, dorim să îmbunătăţim rezultate obţinute prin aplicarea algoritmilor genetici asupra problemelor de optimizare, prin hibridizarea acestora; astfel, vom înlocui etapa de selecţie a cromozomilor ce vor supravieţui cu o tehnică inspirată din ant colony.

BIBLIOGRAFIE

[1] Maniezzo, V., Gambardella, L.M., De Luigi, F., - Ant Colony Optimization, Optimization Techniques in Engineering. Springer-Verlag, Addison-Wesley, 2004

[2] Cioargă R.D., - Comportament emergent în medii colaborative robotizate, MELISSEVS Project: Model for rEpresentation of coLlaborative robotic and Intelligent Sensor Systems in EnVironment exploration and Supervision applications, DSPLabs Timisoara, 2007

[3] Dorigo, M., Stutzle, T., - The Ant Colony Optimization Metaheuristic: Algorithms, Application, and Advances, Handbook of Metaheuristics, Kluwer Academic Publishers, 2002

[4] Cordon, O., deViana, I.F., Herrera, F., Moreno, L., - A New ACO Model IntegratinG Evolutionary Computation Concepts: The Best-Worst Ant System, 2000

[5] Gasbaoui, B., Allaoua, B., - Ant Colony Optimization Applied on Combinatorial Problem for Optimal Power Flow Solution, Leonardo Journal of Science, Issue 14, ISSN 1583-0233, 2009, p.1-17

[6] Silva, C.A., Runkler, T.A., - Ant Colony Optimization for dynamic Traveling Salesman Problems, ARCS –2004 – organic and pervasive computing, workshops Proceedings, Ausburg, Germany, GI, Gesellschaft für Informatik, Bonn, 2004, p.259-266

[7] Almasi, G.S., Gotlieb, A., - Higly parallel Computing, Benjamin-Cummings, Redwood Cliffs, NJ, 1989

[8] The Message Passing Interface (MPI) standard, [http://wwwunix.mcs.anl.gov/mpi/], 2000

[9] University of Heidelberg.Interdisciplinary Center for Scientific Computing, [http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSP LIB95/tsp/]