CONTRACTOR: Universitatea Babeş-Bolyai **nr.contract: 477**

Programul: **IDEI**
Tipul proiectului: **Proiecte de cercetare exploratorie**
Cod proiect: **ID_2286**

Sinteza lucrării

CERCETĂRI ÎN DIRECŢIA OPTIMIZĂRII ADAPTIVE A SISTEMELOR INFORMATICE FOLOSIND TEHNICI DE ÎNVĂŢARE AUTOMATĂ ŞI SISTEME MULTIAGENT

Etapa unică an 2011

A. Sinteza cercetării. Rezultate obţinute

În cele ce urmează vom prezenta o sinteză a rezultatelor originale obţinute în urma cercetărilor efectuate în cadrul proiectului în scopul îndepliniriii obiectivelor ştiinţifice propuse în planul de realizare a proiectului pe anul 2011. Vom indica, pentru fiecare obiectiv prevăzut în planul de realizare pe anul 2011, modul în care au fost îndeplinite activităţile aferente. De asemenea, vom menţiona rezultatele originale obţinute în vederea realizării obiectivelor.

Menţionez faptul că obiectivele planificate pe anul 2011, cât şi activităţile aferente acestora au fost realizate în totalitate, şi desfăşurate conform cu planul de realizare al proiectului. De asemenea, criteriul minim de performantă prevăzut (**1** articol ISI şi **3** articole BDI) a fost îndeplinit, aşa cum se poate vedea în secţiunea legată de modul de diseminare al rezultatelor.

1. Elaborarea unor noi metode pentru problema introducerii şabloanelor de proiectare în sisteme informatice existente folosind tehnici de învăţare automată (clustering, reţele cu autoorganizare -SOM, reţele neuronale)

Fie $S = \{C_1, C_2, ..., C_l\}$ un sistem informatic orientat obiect, unde $C_i, 1 \leq i \leq l$ este o clasă a sistemului.

Fără a restrânge generalitatea abordării, vom considera că vrem să identificăm în sistemul S locaţiile unde ar trebui introdus un anumit şablon de proiectare p. Pentru început, ne-am orientat asupra şabloaneor de proiectare structurale.

Un *şablon de proiectare* p din sistemul informatic S poate fi privit ca fiind o pereche, $p = (\mathcal{C}_p, \mathcal{R}_p)$, unde

- $\mathcal{C}_p = \{C_1^p, C_2^p, \ldots, C_{nc_p}^p\}$ este o submulţime a mulţimii S, $\mathcal{C}_p \subset S$, şi reprezintă mulţimea claselor care fac parte din şablonul de proiectare p. nc_p reprezintă numărul de clase din şablon.
- $\mathcal{R}_p = \{r_1^p, r_2^p, \ldots, r_{nr_p}^p\}$ este mulţimea restricţiilor (relaţiilor) existente între clasele din $\mathcal{C}_p$, restricţii care caracterizează şablonul de proiectare p. Ca urmare, fiecare restricţie $r_i^p, \forall 1 \leq i \leq nr_p$ din mulţimea $\mathcal{R}_p$ este o relaţie definită pe o submulţime de clase din $\mathcal{C}_p$, iar nr_p reprezintă numărul de restricţii care caracterizează şablonul de proiectare p.

Menţionăm faptul că toate restricţiile din mulţimea $\mathcal{R}_p$ pot fi exprimate sub forma unor restricţii binare (sunt două clase participante la restricţie).

Problema identificării locaţiilor unde ar trebui introdus şablonul p se poate reduce la problema identificării unor submulţimi de clase din S al căror comportament este similar cu cel indicat de şablonul p. Pentru a ilustra acest grad de similaritate, am introdus o măsură Sim care indică similaritatea între o mulţime de clase $\{Cl_1, ..., Cl_k\} \subset S$ şi şablonul $p = (\mathcal{C}_p, \mathcal{R}_p)$.

Ca urmare, problema studiată am redus-o la a determina submulţimi ale lui S a căror similaritate Sim în raport cu p este mai mare decât un prag $Prag$ dat. Pentru rezolvarea acestei probleme, care poate fi văzută de fapt ca şi o problemă de satisfacere a restricţiilor, am introdus un algoritm de căutare sistematică de tip BackJumping [12].

În continuare, vom descrie pe scurt algoritmul BJ. BJ este similar cu algoritmul backtracking cronologic (BT), exceptând doar că este mai eficient atunci când nu se găseşte nici o instanţiere consistentă pentru variabila curentă (dead-end). C şi la BT, variabilele $x_1, x_2, ..., x_N$ care trebuie instanţiate pentru a determina soluţia, sunt instanţiate secvenţial, în ordinea $x_1, x_2, ...$, astfel încât, la orice moment instanţierea partţială săfie consistentă. Când apare un dead-end, BJ sare la variabila de indice maxim care a fost conflict cu variabila curentă.

Verificările consistenţei între instanţierea variabilei curente şi instanţierea variabilei trecute sunt îndeplinite pe măsură ce ordinea originală a instanţierilor este $(x_1, x_2,)$. Verificările se opresc în momentul în care o verificare a consistenţei eşuează. Variabila max_check[i] reţine indicele maxim al variabilei care a fost verificată faţă de instanţierea curentă x_i . Această valoare este folosită pentru determinarea punctului de backtracking la

finalul funcţiei main. Problema principal cu BJ este c se fac salturi doar din nodurile terminale ale arborelui (dead-ends). Toate celelalte căutări fac parte din backtracking-ul cronologic, deci acolo este sigur o mulime de trashing-uri. Pe de alt parte, costurile sunt mai mici la BJ.

Implementarea, evaluarea şi validarea algoritmului BJ

După cum am arătat anterior, a fost implementat în JADE un prototip al sistemului *DSSEM*, în care a fost adăugată funcţionalitatea de introducere a şabloanelor de proiectare în sisteme informatice existente.

Pentru evaluare experimentală, am considerat un studiu de caz simplu, un sistem informatic având 5 clase în care , în scopul îmbunătăţirii design-ului ar trebui introdusă o instanţă a şablonului de proiectare *Proxy*. *Proxy* este un şablon de proiectare structural care surprinde o tehnică des folosită în reţele: un obiect local comunică cu un proces aflat la distanţă, dar se doreşte ascunderea detaliilor despre procesul la distanţă sau protocolul de comunicare. Obiectul *proxy* permite accesul la serviciile la distanţă cu aceeaşi interfaţă a proceselor locale. De fapt, atunci când o anumită *Operaţie* este solicitată obiectului proxy, acesta deleagă execuţia operaţiei solicitate obiectului *SubiectReal*. Deoarece atât *Proxy* cât şi *SubiectReal* sunt subclase ale lui *Subiect*, acestea exportă aceeaşi interfaţă pentru *Operaţie*. Pentru a putea apela metodele obiectului *SubiectReal*, *Proxy* necesită o asociere cu acesta.

În urma aplicării algoritmului BJ pe care l-am descris anterior, a fost identificată o sumbmulţime formată din 3 clase care ar trebui să reprezinte o instanţă a şablonului de proiectare *Proxy*. În continuare, ne propunem să îmbunătăţim şi să extindem abordarea propusă în acest capitol în direcţia aplicării abordării propuse pe sisteme soft reale.

2. Dezvoltarea de noi algoritmi de învăţare automată (clustering, reţele cu autoorganizare -SOM, reţele neuronale) în scopul autoadaptării din punct de vedere comportamental al sistemelor informatice

2.1 Dezvoltarea de algoritmi bazaţi pe tehnici de învăţare automată pentru autoadaptarea structurilor de date folosite la fluxul de date din sistem

Este bine ştiut faptul că tipurile abstracte de date reprezintă nucleu loricării aplicaţii soft, iar o utilizare eficientă a acestora este o cerinţă esenţială pentru dezvoltarea unor sisteme robuste şi eficiente. Structurile de date sunt esenţiale în obţinerea unor algoritmi eficienţi, având o importanţă majoră în procesul de dezvoltare de soft. Selectarea şi crearea celei mai potrivite structuri de date pentru implementarea unui tip abstract de date poate avea un impact mare asupra performanţei şi eficienţei sistemului soft. Nu este o problemă trivială pentru un dezvoltator soft, deoarece sunt greu de anticipat toate scenariile de utilizare ale aplicaţiei analizate, iar o selectare statică înaintea execuţiei sistemului nu este, în general, foarte precisă.

În articolul [4] am studiat problema selectării dinamice a structurilor de date eficiente pentru implementarea tipurilor abstracte de date, folosind o abordare supervizată. În scopul selectării dinamice a celei mai potrivite reprezentări pentru un container de date în funcţie de contextul curent al execuţiei sistemului, vom folosi o reţea neuronală. Am evaluat tehnica propusă pe un studiu de caz, evidenţiind avantajele modelului propus în comparaţie cu abordări similare existente în literatură.

Consideraţii teoretice

Fie $S = \{s_1, s_2, ..., s_n\}$ un sistem informatic orientat obiect, unde $s_i, 1 \leq i \leq n$ poate fi o clasă, o metodă dintr-o clasă sau un atribut al unei clase. Ne vom referi la un element al sistemului informatic S ca la o *entitate*.

Să considerăm următoarele:

- $Class(S) = \{C_1, C_2, \ldots, C_l\}$, $Class(S) \subset S$, este mulţimea claselor din sistemul informatic S.
- Fiecare clasă C_i $(1 \leq i \leq l)$ este o mulţime de metode şi atribute, altfel spus, $C_i = \{m_{i1}, m_{i2}, \ldots, m_{ip_i}, a_{i1}, a_{i2}, \ldots, a_{ir_i}\}$, $p_i \leq n,\ 1 \leq r_i \leq n$, unde m_{ij} $(\forall j,\ 1 \leq j \leq p_i)$ sunt metodele şi a_{ik} $(\forall k,\ 1 \leq k \leq r_i)$ sunt atributele din clasa C_i.
- $Meth(S) = \bigcup_{i=1}^{l} \bigcup_{j=1}^{p_i} m_{ij}$, $Meth(S) \subset S$, este mulţimea metodelor din toate clasele sistemului informatic S.
- $Attr(S) = \bigcup_{i=1}^{l} \bigcup_{j=1}^{r_i} a_{ij}$, $Attr(S) \subset S$, este mulţimea atributelor din toate clasele sistemului informatic S.

Pe baza notaţiilor anterioare, sistemul informatic S poate fi definit ca în Ecuaţia (1):

$$S = Class(S) \bigcup Meth(S) \bigcup Attr(S). \tag{1}$$

Fie $\mathcal{T}$ un tip abstract de date având în interfaţa sa o mulţime de operaţii $\mathcal{O}$ care pot fi efectuate asupra unei instanţe a tipului $\mathcal{T}$. De asemenea, considerăm o mulţime $\mathcal{D}_{\mathcal{T}} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ de structuri de date care pot fi utilizate în sistemul S pentru implementarea lui $\mathcal{T}$. Dacă o anumită instanţă c a clasei $C \in Class(S)$

foloseşte $\mathcal{T}$, selectarea celei mai potrivite structuri de date din $\mathcal{D}$ care să fie folosită pentru implementarea (şi instanţierea) lui $\mathcal{T}$ depinde de *contextul de execuţie* curent (starea curentă a lui c). În Definiţia 1 vom introduce noţiunea de *context de execuţie* a lui $\mathcal{T}$.

Definiţia 1 ***Context de execuţie a lui $\mathcal{T}$.***
Fie c o instanţă a clasei $C_i \in Class(S)$ care foloseţe $\mathcal{T}$. Contextul de execuţie *a lui $\mathcal{T}$ reprezeintă starea clasei C_i la execuţie, atunci când $\mathcal{T}$ este iniţializat. Ca urmare, este definit ca mulţimea $\mathcal{EC}_\mathcal{T} = \{v_{i1}, v_{i2}, \ldots, v_{ir_i}\}$, unde r_i reprezintă numărul de atribute din clasa C_i şi $\forall 1 \leq j \leq r_i$, v_{ij} reprezintă valoarea atributului a_{ij} în pasul curent de execuţie.*

În Definiţia 2 vom defini noţiunea de *implementare potrivită* a lui $\mathcal{T}$ pe baza contextului de execuţie.

Definiţia 2 ***Cea mai potrivită implementare a lui $\mathcal{T}$ pe baza contextului execuţiei $\mathcal{EC}_\mathcal{T}$.***
Fie $\mathcal{T}$ un tip abstract de date şi $\mathcal{EC}_\mathcal{T}$ contextul execuţiei lui $\mathcal{T}$ într-un anumit scenariu de utilizare. Cea mai potrivită implementare a lui $\mathcal{T}$ pe baza contextului $\mathcal{EC}_\mathcal{T}$ este structura de date $\mathcal{D}_i \in \mathcal{D}$ care furnizează o implementare eficientă a lui $\mathcal{T}$ în scenariul de execuţie, adică timpul necesare pentru efectuarea operaţiilor lui $\mathcal{T}$ în contextul $\mathcal{EC}_\mathcal{T}$ este minimizat.

De fapt, cea mai potrivită implementare a lui $\mathcal{T}$ într-un anumit scenariu de execuţie depinde de tipul şi numărul de operaţii din $\mathcal{O}$ care sunt efectuate asupra lui $\mathcal{T}$ în timpul execuţiei. Ca urmare, putem conclude că problema selectării dinamice a celei mai potrivite implementări a lui $\mathcal{T}$ este, de fapt, problema predicţiei, pe baza contextului curent al execuţiei, numărului şi tipului operaţiilor efectuate asupra lui $\mathcal{T}$ într-un anumit scenariu de execuţie. Pentru predicţia supervizată a celei mai potrivite implementări a tipului abstract de date $\mathcal{T}$ am folosit în [4] o reţea neuronală.

Evaluare experimentaă

În scopul evaluării acurateţei modelului ANN propus anterior, am simulat un experiment pentru simularea celei mei potrivite structuri de date pentru implementarea tipului abstract de date *List*. Setul de date considerat pentru analiză constă în rezultatele unei analize chimice a vinurilorcare cresc într-o anumită regiune a Italiei, dar obţinută de ladiferiţi cultivatori. Analiza efectuată a determinat 13 constituenţie găsiţi în fiecare tip de vin [13]. Pentru testare, s-a folosit o validare încrucişată de tipul 2/3-1/3 (2/3 instanţe pentru antrenare, 1/3 instanţe pentru testare). S-au folosit 50 de episoade de testare. O acurateţe globală de 0.98133 a fost obţinută pe cele 50 de episoade. Rezultatele obţinute de-a lungul epsioadelor sunt stabile, fiind obţinută o deviaţie standard de 0.01231079 pe setul de precizii obţinut. Valoarea scăzută a deviaţiei standard indică precizia abordării noastre.

2.2 Dezvoltarea unei metode bazate pe învăţare supervizată şi agenţi pentru predicţia comportamentului utilizatorului

În trecut, utilizatorii sistemelor informatice erau cei care le şi concepeau, astfel încât nu existau probleme de utilizabilitate. În prezent, în orice domeniu al vieţii eonomice sunt utilizate calculatoarele, iar nivelul de expertiză al utilizatorilor scade, o dată cu creşterea complexităţii sarcinilor. În acest context, este necesară identificarea de soluţii care să ajute utilizatorii novici în îndeplinirea sarcinilor. Soluţia propusă în acest sens constă în dezvoltarea unui agent inteligent, denumit **LIA** (**L**earning **I**nterface **A**gent) pentru predicţia comportamentului utilizatorului. Rolul acestui agent inteligent este acela de a învăţa, în baza experienţei anterioare şi a contextului actual al interacţiunii, să prezică următoarea acţiune pe care ar trebui să o efectueze utilizatorul pentru a realiza sarcina.

În cele ce urmează, considerăm că agentul **LIA** monitorizează interacţiunile utilizatorilor cu o aplicaţie interactivă $\mathcal{SA}$ în timp ce realizează o sarcină $\mathcal{T}$. Vom nota prin $\mathcal{A}$ mulţimea $\{a_1, a_2, \ldots, a_n\}$ tuturor acţiunilor posibile care pot apărea în timpul interacţiunii cu $\mathcal{SA}$. O acţiune poate fi: apăsarea unui buton, selectarea unei opţiuni dintr-un meniu, completarea unui câmp text. În timpul interacţiunii cu $\mathcal{SA}$ pentru efectuarea sarcinii $\mathcal{T}$ se generează *urme utilizator* ale interacţiunii. O *urmă utilizator* este o secvenţă de acţiuni utilizator. Vom spune că o urmă utilizator este *cu succes* dacă sarcina este îndeplinită. În celelalte cazuri avem o urmă utilizator *fără succes*. În abordarea curentă vom lua în considerare doar urmele cu succes.

Scopul urmărit este acela de a înzestra agentul **LIA** cu capacitatea de a prezice la un moment dat următoarea acţiune pe care utilizatorul ar trebui să o efectueze pentru a realiza sarcina $\mathcal{T}$. Pentru a atinge acest scop, propunem o tehnică de învăţare supervizată care constă în doi paşi: *antrenarea*, respectiv *predicţia*. Pe parcursul antrenării, agentul **LIA** monitorizează interacţiunea unui set de utilizatori reali în timp ce realizează sarcina $\mathcal{T}$, folosind aplicaţia $\mathcal{SA}$ şi îşi construieşte o bază de cunoştinţe $\mathcal{KB}$. Interacţiunea este monitorizată folosind AOP. O abordare mai generală poate fi obţinută prin construirea a două baze de cunoştinţe în timpul antrenării: una cu urme utilizator cu succes, cealaltă cu urme utilizator fără succes.

După ce antrenarea a avut loc, în pasul de predicţie se urmăreşte predicţia comportamentului pentru un nou utilizator U pe baza datelor culese în timpul antrenării, folosind un model probabilistic. După fiecare acţiune *act* efectuată de un utilizator U, exceptând prima sa acţiune, agentul **LIA** va prezice următoarea acţiune, a_r ($1 \leq r \leq n$), care ar trebui efectuată cu o anumită probabilitate $P(act, a_r)$, folosindu-se de $\mathcal{KB}$.

Probabilitatea $P(act, a_r)$ este dată de Egalitatea (2).

$$P(act, a_r) = max\{P(act, a_i),\ 1 \leq i \leq n\} \tag{2}$$

Pentru a calcula aceste probabilităţi vom introduce conceptul de *scor* între două acţiuni. Scorul dintre acţiunile a_i şi a_j, notat prin $score(a_i, a_j)$ indică gradul în care a_j trebuie să-i urmeze lui a_i într-o execuţie cu succes a sarcinii $\mathcal{T}$. Acest fapt sugerează că valoarea $score(a_i, a_j)$ este maximă atunci când a_j trebuie să urmeze imediat după a_i într-o execuţie cu succes a sarcinii. Scorul dintre o acţiune dată *act* şi o acţiune utilizator a_q, $1 \leq q \leq n$, $score(act, a_q)$, se calculează ca în Egalitatea (3).

$$score(act, a_q) = max\left\{\frac{1}{dist(t_i, act, a_q)},\ 1 \leq i \leq m\right\} \tag{3}$$

unde $dist(t_i, act, a_q)$ reprezintă *distanţa* dintre două acţiuni *act* şi a_q într-o urmă t_i, calculată folosind $\mathcal{KB}$.

$$dist(t_i, act, a_q) = \begin{cases} length(t_i, act, a_q) - 1 & \text{dacă } \exists\ sub_{t_i}(act, a_q) \\ \infty & \text{altfel} \end{cases} \tag{4}$$

$length(t_i, act, a_q)$ defineşte distanţa minimă dintre *act* şi a_q în urma t_i.

$$length(t_i, act, a_q) = min\{\|s\| \mid s \in \mathcal{SUB}_{t_i}(act, a_q)\} \tag{5}$$

În abordarea noastră, $length(t_i, act, a_q)$ reprezintă numărul minim de acţiuni efectuate de utilizatorul U în urma t_i, pentru a ajunge de la acţiunea *act* la acţiunea a_q, adică, lungimea minimă dintre toate suburmele posibile $sub_{t_i}(act, a_q)$.

Din Egalitatea (3), rezultă că $score(act, a_q) \in [0, 1]$ şi că valoarea lui $score(act, a_q)$ creşte odată cu descreşterea *distanţei* dintre *act* şi a_q în urmele din $\mathcal{KB}$.

Pe baza *scorurilor*, $P(act, a_i)$, $\forall 1 \leq i \leq n$, se calculează după cum urmează:

$$P(act, a_i) = \frac{score(act, a_i)}{max\{scor(a_i, a_j) \mid 1 \leq j \leq n\}} \tag{6}$$

Pe baza Ecuaţiei (6), probabilităţi mai mari sunt asociate acţiunilor cele mai potrivite pentru a fi executate.

Rezultatul predicţiei agentului este acţiunea a_r care satisface Egalitatea (2). Menţion-ăm că într-un caz nedeterministic (când mai multe acţiuni au aceeaşi probabilitate maximă P) se poate folosi o metodă de selecţie suplimentară. Pentru a evalua acurateţea predicţiei agentului **LIA**, comparăm secvenţa de acţiuni efectuate de utilizatorul U care duc la îndeplinirea sarcinii cu secvenţa de acţiuni prezise de agent. Considerăm că predicţia unei acţiuni este precisă dacă probabilitatea predicţiei este mai mare decât un prag fixat. În acest scop, am definit o măsură de calitate, $ACC(\mathbf{LIA}, U)$, denumită *ACCuracy* (precizie). Evaluarea acurateţii predicţiei agentului **LIA** s-a realizat pe un studiu de caz, folosind măsurile de calitate anterior definite. Am ales spre evaluare o aplicaţie interactivă de dimensiuni medii.

3. Alte rezultate obţinute

În lucrarea [3] am introdus o abordare originală bazată pe reţele cu autoorganizare şi algoritmul *SOMSYR* (*Self-Organizing Map for SYstems Restructuring*) , al cărui scop este determinarea refactorizărilor necesare pentru îmbunătăţirea structurii de clase a sistemelor soft. Pentru construcţia reţelei cu autoorganizare, am folosit topologia de *tor*. Evaluarea algoritmului s-a făcut pe sistemu open source JHotDraw si un sistem informatic real.

Optimizarea combinatorială are ca obiect căutarea de soluţii în spaţiul discret şi bine-definit al unei probleme. Metodele de optimizare au o deosebită importanţă practică, în special în procesele de proiectare, în experimentele ştiinţifice, precum şi în procesele decizionale în afaceri. Datorită importanţei practice majore a problemelor de optimizare combinatorială, o serie de metode au fost propuse pentru abordarea lor. Una dintre acestea este generarea de soluţii inteligente folosind tehnici de învăţare prin întărire. Învăţarea prin întărire este o abordare a instruirii automate, în care un agent poate învăţa să se comporte într-un anumit fel prin alegerea unor acţiuni şi primirea de pedepse, sau recompense, în funcţie de acţiunile alese. Principalul obiectiv al lucrării [10] este propunerea şi prezentarea unei interfeţe de programare pentru rezolvarea problemelor de optimizare combinatorială, folosind tehnici bazate pe învăţarea prin întărire. Această interfaţă permite dezvoltarea simplă de aplicaţii pentru rezolvarea problemelor de optimizare combinatoriaă, folosind învăţarea prin întărire. De asemenea, lucrarea oferă şi un experiment pentru rezolvarea problemei bidimensionale a plierii proteinei, proiectat cu ajutorul interfeţei propuse.

În lucrarea [8] propunem un model bazat pe învăţarea prin întărire pentru rezolvarea problemelor de optimizare combinatorială. Rezolvarea acestui tip de probleme în mod optim este o sarcină dificilă, motiv pentru care se încearcă îmbunătăţirea soluţiilor care se pot obţine. Ne concentrăm în special pe problema plierii proteinei, o bine-cunoscută problemă NP-completă, de mare importanţă în multe domenii, cum ar fi bioinformatica,

biochimia, biologia moleculară şi medicina. Introducem un model bazat pe învăţarea prin întărire pentru determinarea structurii bidimensionale a proteinei în modelul simplificat hidrofob-polar. Modelul propus în această lucrare poate fi extins cu uşurinţă pentru rezolvarea altor probleme de optimizare. Oferim, de asemenea, o validare matematică a modelului propus, bazat pe învăţarea prin întărire, indicând astfel potenţialul acestuia.

Lucrarea [9] se axează pe cercetarea modelului bazat pe învăţarea prin întărire, introdus pentru determinarea structurii bidimensionale a proteinelor, folosind modelul simplificat hidrofob-polar. Modelul propus este fondat pe o abordare de tip *Q-learning*, bazată pe agenţi. Evaluarea experimentală confirmă o bună performanţă a modelului bazat pe învăţarea prin întărire, evidenţiind potenţialul propunerii noastre. Cercetările efectuate în vederea propunerii unei abordări bazate pe un model distribuit de învăţare prin întărire pentru rezolvarea problemelor de optimizare combinatorială sunt prezentate în lucrarea [5]. Ne focalizăm atenţia pe problema bidimensională a plierii proteinei, care se referă la determinarea structurii bidimensionale a proteinei, folosind secvenţa primară de aminoacizi. Modelul pe care îl propunem se bazează pe o abordare distribuită de tip *Q-learning*, care extinde către o arhitectură distribuită un model pe care l-am propus în prealabil [8]. Experimentele au fost realizate în vederea obţinerii de soluţii pentru problema bidimensională a plierii proteinei, iar rezultatele obţinute sunt promiţătoare.

Reconstrucţia unei molecule de ADN, utilizând fragmente de ADN, sau, altfel spus, problema asamblării fragmentelor de ADN este o problemă importantă în numeroase domenii, cum ar fi bioinformatica, biologia computaţională, biotehnologia sau medicina. Datorită faptului că ea este NP-completă, au fost propuse şi dezvoltate o serie de tehnici computaţionale pentru rezolvarea ei, printre acestea numărându-se şi diferiţi algoritmi inteligenţi. Deoarece asamblarea fragmentelor de ADN este de o importanţă esenţială în orice tip de proiect de secvenţiere a ADN-ului, o mare parte din cercetători se concentrează pe direcţia dezvoltării de algoritmi performanţi de asamblare. Scopul principal al lucrării [7] este propunerea unui model bazat pe învăţarea prin întărire pentru rezolvarea problemei asamblării de fragmente. Ne focalizăm atenţia asupra problemei asamblării fragmentelor de ADN. Noul model introdus este axat pe o abordare bazată pe agenţi, de tip *Q-learning*. Rezultatele experimentelor realizate demonstrează potenţialul direcţiei propuse, precum şi faptul că aceasta merită a fi cercetată în continuare.

După cum s-a menţionat mai sus, problema asamblării fragmentelor de ADN este o problemă NP-completă, prin care se încearcă reconstituirea unei molecule de ADN pornind de la un număr mare de fragmente, fiecare dintre acestea conţinând câteva sute de nucleotide. Modelul distribuit propus în [2] pentru abordarea problemei asamblării fragmentelor de ADN extinde modelul de învăţare prin întărire propus în [7] şi este de tip *Q-learning* concurent, în care mai mulţi agenţi cooperează în scopul de a obţine soluţii optime. Lucrarea oferă şi o evaluare experimentală a sistemului introdus, ale cărei rezultate sunt încurajatoare.

De asemenea,ne-am orientat cercetările spre imbunătăţirea performanţei folosind calculul paralel. Pentru aplicarea cu succes a calculului paralel se impune existenta a trei componente distincte: prima este reprezentata de hardware-ul pe care se vor executa aplicatiile; a doua este maşina abstracta sau modelul de programare in care sunt scrise aplicaţiile, iar a treia este reprezentată de procesul de proiectare care permite crearea aplicaţiilor din specificaţii.

O alegere posibilă pentru un model general de programare paralelă este BSP – *Bulk Synchronous Parallelism*. Un avantaj important al BSP este dat de faptul că măsurile de cost asociate estimează în mod real costurile. Pentru că separa partea de comunicare de sincronizare este in mod particular clar si simplu. Această separare permite o dezvoltare simplă si formalizarea procesului de constructie al softului. Pentru această formalizare propunem o metodă bazata pe specificaţii parametrizate si dezvoltare formală [6]. Un program este considerat ca fiind un număr de procese parametrerizate cu structura similară care cooperează. Metoda foloseste pre- şi post- conditii parametrizate şi consideră distribuţia datelor chiar de la începutul procesului de construcţie. Posibilitatea de a estima numărul de comunicaţii din postconditii ne permite sa facem o evaluare a costului chiar de la primele stagii de dezvoltare si aceasta este importantă pentru luarea deciziilor de construcţie eficientă.

Abordarea noastră pentru construcţia unui proces parametrizat $S.q$ din specificaţiile functionale este similară cu metodele de derivare folosite in programarea secvenţială. Se folosesc regulile clasice de derivare pentru programarea secvenţială si introducem reguli noi pentru compunere paralelă şi comunicare. Se foloseste metoda clasică (bazata pe calcul) de obţinere a unui invariat. Programele se derivează prin calcularea condiţiilor necesare menţinerii invarianatului. Pentru că procesele sunt parametrizate se vor deriva şi invarianţi parametrizaţi. In modelul BSP, partiţionarea datelor este crucială spre deosebire de maparea partiţiilor rezultate care este irelevantă. Alegerea unei distribuţii adecvate de date este un mijloc important de influenţare a performanţei programului. Folosim în acest scop distribuţii de date simple dar şi multivoce şi evidenţiem importanţa alegerii adecvate pe mai multe exemple de dezvoltare de programe paralele.

Prin folosirea conceptului de *tip abstract de date*, o structură de date poate fi definită mai riguros şi mai formal. Un pas mai departe in aceasta direcţie s-a făcut odata cu introducerea programării orientate-obiect (OOP). Folosind OOP putem defini nu doar structuri de date generice prin polimorfism şi genericitate, dar putem de asemenea să separăm definiţiile de implementări prin folosirea interfeţelor; interfeţele descriu tipuri abstracte de date specificând operaţiile corespunzătoare lor. Sabloanele de proiectare au dus lucrurile mai departe permiţând obţinerea unui grad sporit de flexibilitate si reutilizabilitate in implementarea structurilor de date. Pentru a construi o structura de date extensibilă şi reutilizabilă este necesară decuplarea comportamentului

ei primitiv şi intrinsec de comportamentul specific adaptat aplicaţiei care o manipuleaza.

Orice tip de structură de date de tip container este formată dintr-un număr de elemente care în general au acelaşi tip. Un container particular are proprietăţi si comportament care nu este dependent de tipul elementelor constitutive.

Prin studirea diferitelor structuri de date am evidenţiat urmatoarea clasificare a acestora [11]:structuri de date de *nivel primar* sau fundamentale; structuri de date de *nivel secundar* care sunt caracterizate de faptul ca implementările lor folosesc structurile de date fundamentale. Tablourile si reprezentarile înlantuite pentru liste şi arbori sunt considerate structuri de date fundamentale. Pentru a implementa o mulţime sau un dicţionar putem folosi un tablou sau o listă inlantuita sau un arbore; astfel mulţimile si dicţionarele sunt considerate exemple de structuri de date de nivel secundar.

In proiectarea propusă structurile de date fundamentale implementeaza o interfata Storage pentru a fi evidenţiată proprietatea lor de a putea fi folosite pentru stocarea elementelor structurilor de date de nivel secundar. Procesul poate fi simplificat prin folosirea şablonului *Bridge* impreuna cu sabloanele *Abstract Factory* si *Singleton*. Aceste şabloane asigura independenţa de stocare a structurilor de dare de nivel secundar [11]. Prin separarea reprezenării unei structuri de date de comportamentul ei dat de tipul ei, introducem un nou nivel de indirectare şi prin urmare si de abstractizare. Avantajul este dat de posibilitatea de a implementa structuri de date bazându-ne pe diferite structuri de date fundamentale fără a fi nevoie de a construi mai multe clase. Se introduce de asemenea şi un nou nivel de genericitate. Interfeţele de tip - Storage - pot introduce o altă clasificare a structurilor de date. Orice structură de date poate fi folosită ca o posibilitate de stocare pentru altă structura de date sau ca si o componenta de stocare intr-un program generic. Programarea modernă foloseşte medii de programare care simplifică munca programatorului foarte mult. Aceste medii oferă un nivel înalt de abtractizare şi astfel se permite crearea programelor prin specificarea modului în care componentele de baza sunt compuse. In acest context un nou nivel de abtractizare a datelor manipulate de aceste programe este de dorit. Putem să considerăm un scenariu in care programatorul specifică doar tipul (comportamentul) al entităţii de stocare: simplu, secvenţial, sortat, etc. Pe baza reprezentării propuse şi, de asemenea, folosind mecanisme automate cum sunt tehnicile de inteligenţă artificială se pot alege tipurile concrete cele mai eficiente pentru aplicaţia dezvoltată.

B. Activităţi organizatorice

Activităţile organizatorice au avut ca scop menţinerea unui cadru propice activităţii de cercetare şi îndeplinirii cerinţelor de această natură stipulate în contractul de finanţare. Pagina web a proiectului, dedicată prezentării proiectului, a echipei de cercetare şi a rezultatelor obţinute, http://www.cs.ubbcluj.ro/∼raoss, a fost permanent actualizată. S-au efectuat întâlniri periodice ale echipei de cercetare, în scopul realizării cu succes a activităţilor prevăzute în plan şi diseminarea în cadrul grupului a rezultatelor obţinute.

C. Diseminarea rezultatelor

Diseminarea rezultatelor ştiinţifice obţinute în anul 2011 în cadrul proiectului şi prezentate în secţiunie anterioare, s-a realizat prin publicarea a 12 articole de specialitate, după cum urmează. **7** publicaţii ISI: **4** în reviste cotate ISI - Science Citation Index Expanded (lucrările [1], [2], [3], [4]) şi **3** la conferinţe ISI - Conference Proceedings Citation Index (lucrările [5], [6], [7]) şi **4** articole în reviste indexate BDI (lucrările [8]-[11]). De asemenea, membrii echipei au realizat periodic rapoarte tehnice conţinând rezultatele obţinute în vederea atingerii obiectivelor proiectului. Menţionăm faptul că factorul de impact cumulat al publicaţiilor în revistele cotate ISI (calculat pe 2010) este 4.985. Ca urmare, criteriul minim de performantă prevăzut pe anul 2011 (**1** articol ISI şi **3** articole BDI) a fost îndeplinit.

D. Concluzii

Sintetizăm rezultatele obţinute în cadrul proiectului pe anul 2011 ca fiind următoarele: elaborarea unor noi metode bazate pe satisfacerea constrângerilor pentru problema introducerii şabloanelor de proiectareîn sisteme informatice existente; dezvoltarea de noi algoritmi de învăţare supervizată pentru adaptarea dinamică a structurilor de date dintr-un sistem informatic; dezvoltarea unei metode bazate pe învăţare supervizată şi agenţi pentru predicţia comportamentului utilizatorului. Conform celor prezentate anterior, obiectivele planificate pe anul 2011, cât şi activităţile aferente acestora au fost realizate în totalitate, şi desfăşurate conform cu planul de realizare al proiectului. De asemenea, criteriul minim de performantă prevăzut a fost îndeplinit. De asemenea, obiectivele proiectului pe cei trei ani de derulare a acestuia (2009-2011) au fost realizate, criteriile minime de performanţă fiind îndeplinite.

Menţionăm faptul că factorul de impact cumulat (pe perioada de derulare a proiectului 2009-2011) al publicaţiilor în revistele cotate ISI este 6.512.

Bibliografie

[1] Czibula, G., Cojocar, G.S., Czibula, I.G., *Evaluation Measures For Partitioning Based Aspect Mining Techniques*, International Journal of Computers, Communications and Control, 6(1), 2011, pp. 72-80 (ISI - Science Citation Index Expanded) - IF = 0.650 (pe 2010)

[2] Bocicor, M. I., Czibula, G., Czibula, I.G., *A Distributed Q-Learning Approach to Fragment Assembly*, SIC Journal, Studies in Informatics and Control, Vol. 20, Issue. 3, 2011, to be published (ISI - Science Citation Index Expanded) (IF=0.671 pe 2010)

[3] Czibula, G., Czibula, I.G., *Unsupervised Restructuring of Object-Oriented Software Systems using Self-Organizing Feature Maps*, International Journal of Innovative Computing Information and Control, Japan, 2011, to be published (ISI - Science Citation Index Expanded) (IF=1.664 pe 2010)

[4] G. Czibula, I.G. Czibula, and R.D. Gaceanu. Intelligent Data Structures Selection Using Neural Netorks. Knowledge and Information Systems, 2011, to be published (ISI - Science Citation Index Expanded) (IF=2.00 pe 2010).

[5] Czibula, G., Bocicor, M.I., Czibula, I.G., *A Distributed Reinforcement Learning Approach for Solving Optimization Problems*, in Recent Researches in Communications and IT, Proceedings of the 5th International Conference on Communications and Information Technology (CIT '11), Greece, 2011, pp. 25-30 (ISI - Conference Proceedings Citation Index)

[6] Niculescu, V., *Formal Refinement of BSP Programs with Early Cost Evaluation*, Proceedings of the 10th International Symposium on Parallel and Distributed Computing, ISPDC 2011, to be published (ISI - Conference Proceedings Citation Index)

[7] Bocicor, M. I., Czibula, G., Czibula, I.G., A Reinforcement Learning Approach for Solving the Fragment Assembly Problem, Proceedings of the 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2011, to be published (ISI - Conference Proceedings Citation Index)

[8] Czibula G., Bocicor, M.I., Czibula, I.G., *A Reinforcement Learning Model for Solving the Folding Problem*, IJCTA - International Journal of Computer Technology and Applications, Vol. 2, Issue 1, 2011, pp. 171-182 (indexed Index Copernicus)

[9] Czibula, G., Bocicor, M.I., Czibula, I.G., *An Experiment on Protein Structure Prediction using Reinforcement Learning*, Studia Babes-Bolyai Informatica, LVI (1), 2011, pp. 25-34 (indexed Mathematical Reviews)

[10] Czibula, I. G., Czibula, G., Bocicor, M.I., *A Software Framework for Solving Combinatorial Optimization Tasks*, Studia Universitatis "Babes-Bolyai", Informatica, Special Issue, LVI(3), pp. 3-8, 2011 (indexed Mathematical Reviews)

[11] Niculescu, V., *Storage Independence in Data Structures Implementation*, Studia Universitatis "Babes-Bolyai", Informatica, Special Issue, LVI(3), pp. 21-26, 2011 (indexed Mathematical Reviews)

[12] Gabriela Czibula. *Sisteme inteligente. Instruire automata*, Editura RisoPrint, Cluj-Napoca, 2008

[13] Wine data set. http://archive.ics.uci.edu/ml/datasets/Wine

Cluj-Napoca, 15 noiembrie 2011

Director de proiect,
Prof. dr. CZIBULA GABRIELA