Universitatea Babeş-Bolyai Cluj-Napoca
Facultatea de Matematică şi Informatică

Metode evolutive pentru rezolvarea problemelor de optimizare

Candidat:
Flavia Zamfirache

Coordonator:
Prof. Dr. Militon Frenţiu

Rezumat
2011

Abstract

Metodele evolutive sunt utilizate cu succes pentru a rezolva probleme de optimizare statică, a căror scop este de a găsi soluţia optimă globală. În cazul problemelor de optimizare în medii dinamice şi incerte scopul algoritmului este de a urmări modificările din mediu sau de a găseasi un comportament robust. În cazul dinamic, algoritmii pot întâmpina probleme din cauza convergenţei premature a metodelor evolutive. Diferite mecanisme sunt propuse pentru a rezolva problemele care apar în mediile dinamice şi incerte, metode care permit menţinerea diversităţii populaţiei în timpul rulării, utilizarea arhivelor de memorie sau multi-populaţii.

Multe probleme din lumea reală sunt dinamice, proprietăţile lor se schimbă/ variază în timp. Această teză se bazează pe trei probleme reale: planificarea taskurilor în grid, gruparea datelor şi extragerea de informaţii din date, analizând în acelaşi timp comportamentul lor în medii incerte şi dinamice. Incertitudinea în problemele de optimizare poate fi cauzată de diferite surse: funcţia obiectiv poate fi afectată de zgomot, funcţia obiectiv poate fi aproximată sau schimbată de-a lungul timpului.

În această teză am propus şi analizat comportamentul unor algoritmi inspiraţi din natură în medii incerte şi dinamice. Am propus un mecanism simplu de perturbare a poziţiei unei particule pentru euristica Particle Swarm Optimization şi introducerea de elemente aleatoare în populaţie în cazul euristicii Differential Evolution pentru a încetini convergenţa euristicilor şi am testat metodele propuse pe două probleme de benchmark. În cazul problemei de planificare a datelor pe grid, am propus şi studiat comportamentul unui planificator bazat pe algoritmi evolutivi în cazul planificării online. De asemenea, pentru planificatorul evolutiv şi pentru planificatorul bazat pe euristica Ant Colony Optimization am studiat robusteţea şi am analizat comportamentul într-un mediu dinamic. În cazul problemei de grupare a datelor, am studiat influenţa zgomotelor şi valorilor lipsă din date pentru euristica `AntClust` si am propus un

mecanism de calcul al densităţii care îmbunătăţeşte rezultatele grupării în cazul datelor afectate de zgomot. În cazul problemei de extragere a informaţiilor din date am propus un algoritm evolutiv multi-obiectiv care încorporează evaluarea regulilor de către experţii umani şi am analizat influenţa valorilor lipsă în date şi impactul intervenţiei utilizatorului în procesul de căutare. De asemenea, am propus utilizarea unui clasificator bazat pe o selecţie evolutivă de reguli generate folosind tehnica prototipurilor neimbricate ca alternativă la algoritmii de interschimbare din cadrul unui planificator.

Cuprins

Capitolul 1

Introducere

1.1 Cuvinte cheie

Algoritmi evolutivi, Particle Swarm Optimization, Ant Colony Optimization, Planificare pe grid, Determinarea de reguli pentru date medicale, Determinarea de reguli pentru aplicarea algoritmilor de planificare

1.2 Motivaţie

Multe probleme întâlnite în lumea reală sunt supuse schimbărilor. Aceste schimbări pot fi determinate de simulări, erori de măsură sau aproximare, ceea ce duce la apariţia zgomotelor în funcţia obiectiv. De asemenea, variabilele care caracterizează problema sau condiţiile din mediu pot fi perturbate sau schimbate de-a lungul timpului. De exemplu, în cazul unei probleme de planificare a unor aplicaţii, noi aplicaţii pot să apără altele pot să dispără, maşinile pot funcţiona mai încet, materialele îşi pot schimba calitatea, blocaje de trafic pot să apără în cazul problemelor de rutare, etc. În aceste cazuri, când rezolvăm probleme de optimizare, va trebui să luăm în considerare faptul că se execută în medii incerte şi dinamice. În acest caz, obiectivul problemei nu este de a găsi un optim global, ci de a urmări modificările în mediul dinamic sau de a găsi soluţii robuste când mediul este incert.

Metodele evolutive au fost folosite cu succes pentru a rezolva probleme de optimizare. Ele sunt o alternativă la algoritmi determinişti, care găsesc soluţii suboptimale ale problemelor. Printre avantajele lor se numără robusteţea raportată la dimensiunea problemei (o bună alternativă în cazul problemelor mari), la instanţele problemei şi variabilele aleatoare. Calitatea

algoritmilor evolutivi depinde de balansul dintre proprietăţile de exploatare şi explorare; exploatarea permite îmbunătăţirea soluţiei în timp ce explorarea încurajează căutarea în regiuni noi ale spaţiului de căutare. Există diferite tehnici de căutare ce aparţin metodelor evolutive: algoritmii genetici, strategiile evolutive, Particle Swarm Optimization, Ant Colony Optimization, sistemele imune, etc. Metodele evolutive s-au dovedit a fi potrivite pentru a rezolva probleme de optimizare în medii incerte şi dinamice, deoarece permit adăugarea de mecanisme ce duc la obţinerea de rezultate bune fără a fi nevoie de restartarea algoritmului.

În această teză se analizează abilitatea unor metode evolutive de a rezolva diferite probleme de optimizare în medii incerte şi dinamice. Această problemă este una de actualitate deoarece multe probleme care se regăsesc în practică sunt dinamice şi diferiţi factori pot să influenţeze calitatea soluţiei, fezabilitatea soluţiei depinzând de criterii multiple.

1.3 Structura tezei

Teza este structurată în următoarele capitole:

Capitolul 2 realizează o prezentare generală a conceptelor dezbătute în această teză: probleme de optimizare, medii incerte şi dinamice. Acest capitol conţine de asemenea o motivare a faptului că problemele de grupare şi găsire de reguli pot fi formulate ca probleme de optimizare.

Capitolul 3 conţine o prezentare generală a euristicilor folosite în această teză: algoritmii evolutivi, Ant Colony Optimization şi Particle Swarm Optimization. De asemenea, sunt descrise câteva mecanisme ce permit adaptarea euristicilor prezentate anterior în medii dinamice. Acest capitol conţine şi un exemplu de aplicare a două euristici: Differential Evolution şi Particle Swarm Optimization într-un mediu dinamic pentru două probleme de optimizare.

Capitolul 4 dezbate problema unui planificator pe grid în medii statice şi dinamice. Capitolul conţine o trecere în revistă a metodelor evolutive aplicate problemei de planificare în grid, formalizarea problemei şi descrierea particularităţilor a două euristici inspirate din natură: algoritmii evolutivi şi Ant Colony Optimization, necesare pentru a mapa euristicele pe problema de planificare. Rezultatele numerice prezentate în acest capitol se referă la robusteţea şi comportamentul celor două euristici în medii statice şi dinamice. De asemenea, este analizat comportamentul euristicelor şi în cazul online.

Capitolul 5 descrie o problemă de grupare a datelor afectate de zgomot şi valori lipsă, aceasta fiind abordată folosind un algoritm inspirat din comportamentul furnicilor `AntClust`. În Secţiunea 5.1 se regăseşte o trecere în revistă a algoritmilor de grupare inspiraţi din compor-

tamentul furnicilor. De asemenea, este prezentat algoritmul `AntClust`, câteva recomandări în setarea parametrilor şi variante ale acestuia. Experimentele numerice prezintă comportamentul algoritmul `AntClust` în cazul: datelor afectate de zgomot, în acest caz propunânduse un mecanism de îmbunătăţire a performanţei; şi în cazul datelor medicale incomplete.

Capitolul 6 prezintă o abordare evolutivă interactivă, multiobiectiv pentru problema găsirii de reguli în datele medicale. Capitolul conţine o prezentare a problemei de găsire a regulilor, măsuri utilizate pentru evaluarea regulilor, câteva metode de corectare a valorilor lipsă din date şi un review al abordărilor evolutive utilizate în găsirea de reguli. Sunt prezentate de asemenea rezultate numerice pentru găsirea regulilor de asociere pentru date de test (UCI Machine Learning Repository) şi date reale, colectate de la un spital de Obstetrică-Ginecologie pe parcursul unui an. S-a analizat influenţa diferitelor tehnici de tratare a valorilor lipsă din date asupra regulilor. Acest capitol prezintă un clasificator hibrid bazat pe determinarea regulilor folosind prototipuri generalizate neimbricate şi o selecţie a atributelor şi prototipurilorlor care utilizează un algoritm evolutiv. Experimentele numerice s-au bazat pe determinarea acurateţii algoritmului faţă de alte tehnici de clasificare şi interpretarea rezultatelor pentru a determina caracteristicile taskurilor care determină selecţia unui anumit algoritm de planificare sau a altuia.

Capitolul 7 prezintă concluziile şi câteva direcţii pentru studiile viitoare.

1.4 Contribuţii

Această teză tratează problema aplicării metodelor evolutive pentru rezolvarea problemelor de optimizare. Teza se bazează pe aplicarea următoarele metode evolutive: algoritmi evolutivi, ant colony optimization and particle swarm optimization în medii incerte. Deoarece incertitudinea în probleme poate apărea din diferite motive am analizat următoarele situaţii: influenţa zgomotului în procesul de grupare a datelor, robusteţea în cazul algoritmilor de planificare, comportamentul unor planificatoare evolutive într-un mediu dinamic şi descoperirea de reguli de asociere într-un mediu interactiv.

Contribuţiile principale ale acestei teze legate de aplicarea algoritmilor inspiraţii din natură pentru probleme de optimizare în medii incerte sunt:

- pentru a încetini convergenţa euristicii Differential Evolution într-un mediu dinamic am propus utilizarea de elemente aleatoare (Capitolul 3 pagina 15);

- pentru a încetini convergenţa euristicii Particle Swarm Optimization într-un mediu dinamic am propus utilizarea unei perturbaţii care afectează poziţia particulei (Capitolul 3 pagina 15);
- s-a propus un algoritm evolutiv de planificare bazat pe populaţii şi s-a analizat comportamentul lui in cazul planificării online (Capitolul 4 pagina 20) şi influenţa strategiilor de perturbare a planificării (Capitolul 4 pagina 22) şi structura populaţiei iniţiale asupra calităţii soluţiei.
- pentru un algoritm evolutiv şi un algoritm bazat pe modelul coloniilor de furnici am analizat comportamentul într-un mediu static şi dinamic (Capitolul 4 pagina 22) şi robusteţea euristicilor (Capitolul 4 pagina 23);
- am analizat abilitatea euristici `AntClust` de a grupa date afectate de zgomot, un parametru bazat pe densitate a fost propus pentru a îmbunătăţi rezultatele grupării (Capitolul 5 pagina 29);
- am analizat abilitatea euristicii `AntClust` de a grupa date medicale cu valori lipsă (Capitolul 5 pagina 31);
- s-a propus un algoritm evolutiv multiobiectiv interactiv de găsire de reguli în date medicale, care pe lângă selecţia automată implică şi utilizatorul în selecţia datelor (Capitolul 6 pagina 38). Particularitatea abordării noastre constă în introducerea unei distanţe de crowding între reguli, care se comportă ca un criteriu de menţinere a diversităţii, astfel încât mulţimea Pareto este stimulată să aleagă elemente din zone mai puţin dense pentru a le adăuga în arhivă (Capitolul 6 pagina 36);
- s-a analizat influenţa câtorva mecanisme de gestionare a datelor cu valori lipsă în cazul găsirii de reguli (Capitolul 6 pagina 40);
- s-a propus un mecanism de selecţie evolutiv pentru clasificarea regulilor generate pe baza euristicii NNGE şi s-a aplicat pentru selectarea automată a algoritmului de planificare pentru un set de taskuri (Capitolul 6 pagina 37).

Capitolul 2

Probleme de optimizare în medii dinamice şi incerte

Obiectivul unei probleme într-un mediu dinamic este de a urmări modificările optimului când au loc modificări în mediu şi de a găsi soluţii robuste. Obiectivul unei probleme într-un mediu incert este de a găsi cea mai bună soluţie globală care va fi în continuare cea mai bună soluţie şi atunci a când mediul se schimbă.

2.1 Probleme de optimizare în medii incerte

Există diferite motive care pot induce incertitudine în mediu:

- *Zgomotul.* Funcţia de evaluare este afectată de zgomot;
- *Perturbările.*Variabilele problemei pot fi perturbate sau se pot schimba după ce s-a găsit soluţia optimă;
- *Aproximarea funcţiei obiectiv.* Evaluarea funcţiei obiectiv este costisitoare sau nu există o expresie analitică a acesteia; în aceste cazuri, funcţia este aproximată pe baza unor date generate prin experimente sau simulări.
- *Funcţia obiectiv se modifică în timp.* Funcţia obiect se poate determina în orice moment de timp, dar depinde de acesta.

2.2 Probleme de optimizare în medii dinamice

Datorită convergenţei premature a algoritmilor care rezolvă probleme de optimizare, identificarea optimelor care se schimbă este o problemă pentru algoritmi. Astfel, algoritmii trebuie să găsească un balans între exploatarea şi exploraea spaţiului de căutare. Dacă proprietatea de exploatare este prea mult încurajată, există posibilitatea blocării algoritmului într-un punct de minim local şi transformarea căutării într-o căutare locală. Dacă proprietatea de explorare este prea mult încurajată, există posibiliatea de a transforma căutarea într-una aleatoare. Particularităţile unei probleme dinamice sunt determinate de: reprezentarea problemei, caracteristicile mediului şi măsurile de calitate utilizate pentru a evalua performanţa algoritmului.

2.2.1 Probleme de optimizare continue

Problemele de optimizare continuă se ocupă de funcţii definite pe domenii continue, care trebuie optimizate (maximizate sau minimizate). În acest caz, optimul poate urma o traiectorie *continuă* sau *discontinuă* în spaţiul de căutare.

2.2.2 Probleme de optimizare combinatoriale

Problemele de optimizare combinatorială (alocarea resurselor, planificare, etc) se ocupă de o alocare eficientă a unui set finit de resurse pentru a realiza un obiectiv.

2.3 Gruparea datelor - problemă de optimizare

Gruparea datelor este o tehnică a cărei scop este de a identifica grupuri de date similare, fără a avea informaţii despre aceste grupuri, prin măsurarea similarităţii (sau disimilarităţii) dintre date. Există o multitudine de metode de grupare care diferă prin: (i) informaţiile necesare despre problemă (ex. numărul de clustere); (ii) modul de identificare şi construire a grupurilor; (iii) modul în care tratează datele (numerice, categorice sau ambele); (iv) măsuri de similaritate utilizate. Referitor la modul în care algoritmii de grupare interpretează datele, există două abordări principale: (i) grupurile sunt mulţimi compacte, distante de date (această abordare poate fi văzută ca o *problemă de optimizare globală*); (ii) grupurile corespund unor regiuni dense de date separate prin regiuni mai puţin dense de date (această abordare poate fi văzută ca o

problemă de optimizare locală). Algoritmii de grupare pot fi de asemenea clasificaţi în funcţie de numărul de criterii de grupare ca tehnici cu un singur obiectiv sau cu obiective multiple.

2.3.1 Abordări ale problemelor de grupare a datelor

Problemele de grupare a datelor pot fi clasificate în:

- **Probleme de optimizare globală**
- **Probleme de optimizare locală**
- **Probleme de optimizare multi-obiectiv**

  În [ZZN+07] am analizat influenţa diferitelor criterii de calitate asupra grupării documentelor. Analiza comparativă se bazează pe un algoritm de grupare inspirat de algoritmul Differential Evolution care permite atât estimarea numărului de clustere cât şi a reprezentanţilor. Rezulatele ilustrează particularităţile diferitelor criterii de grupare şi abilitatea abordării propuse de a identifica atât numărul de clustere cât şi reprezentanţii clusterelor, sugerând că cel mai bun comportament este obţinut prin combinarea măsurilor de conectivitate şi separabilitate.

2.3.2 Gruparea datelor afectate de zgomot

Zgomotul în date poate fi introdus de erori aleatoare (codare necunoscută, valori care nu se încadrează în domeniul de definiţie, date inconsistente) sau variaţii în măsurarea variabilelor. Pentru înlăturarea zgomotului, se pot folosi diferite metode precum: verificarea consistenţei, informaţii despre domeniu, metode statistice, etc. În cazul metodelor de grupare, pentru eliminarea zgomotelor se pot folosi măsuri bazate pe densitatea datelor. Printre algoritmii de grupare care se bazează pe clasificare putem aminti: DBSCAN, DENCLUE şi UNC (Unsupervised Niche Clustering). În [ZZ05b] am propus un mecanim bazat pe densitate care permite îmbunătăţirea performanţelor algoritmului `AntClust` (capitolul 5).

2.4 Găsirea de reguli - problemă de optimizare

Extragerea de informaţii din date (data mining) este un proces automat sau semi-automat de găsire de informaţii necunoscute, potenţial utile şi de şabloane din bazele de date. Una din

problemele care se regăseşte în extragerea de informaţii din date este găsirea de reguli de asociere. O regulă de asociere este o regulă de forma $(A \Rightarrow C)$ unde A (antecedent) şi C (consecvent) sunt mulţimi disjunte. Cum regulile sunt combinaţii de atribute, regulile de asociere pot fi văzute ca probleme de optimizare combinatorială. În acest caz, scopul problemei de optimizare este de a optimiza diferite măsuri de calitate ale regulilor (ex. suport, încredere). În capitolul 6 este prezenatat în detaliu un algoritm de extragere de reguli de asociere multi-obiectiv.

2.5 Concluzii

În acest capitol am introdus contextul în care această teză este dezvoltată: probleme de optimizare, medii incerte, gruparea datelor şi extragerea de reguli din date. Am început cu o prezentare a proprietăţilor care pot induce incertitudine în mediu: evaluarea funcţiei obiectiv poate conţine zgomote, variabilele problemei pot fi perturbate, funcţia obiectiv este aproximată sau funcţia obiectiv se schimbă de-a lungul timpului. Au fost detaliate apoi proprietăţile unui mediu dinamic, măsuri de evaluare, tipuri de probleme şi benchmark utilizate pentru testare. Am prezentat apoi problemele de grupare a datelor şi de extragere de informaţii din date ca probleme de optimizare.

Capitolul 3

Metaeuristici inspirate din natură utilizate în medii dinamice

Problemele de optimizare în medii incerte şi dinamice sunt complexe şi dificile şi de multe ori algoritmii clasici bazaţi pe programarea dinamică sau abordările matematice reuşesc să rezolve doar instanţe mici ale problemelor. Din acest motiv, în ultimi ani, metaeuristici ca Ant Colony Optimization, Evolutionary Computation, Simulated Annealing, Tabu Search şi altele, s-au dovedit a fi alternative fezabile la abordările clasice.

3.1 Metaeuristici inspirate din natură

În această teză am folosit trei metaeuristici inspirate din natură: algoritmi evolutivi, Ant Colony Optimization şi Particle Swarm Optimization. Am ales aceste metaeuritici datorită abilităţii lor de a se adapta la modificările din mediu, fapt care le face potrivite pentru experimentele numerice descrise în următoarele capitole.

3.1.1 Algoritmi evolutivi

Algoritmii evolutivi (EA) sunt tehnici de căutare euristice care se bazează pe ideea de evoluţie naturală (selecţie, supravieţuirea celui mai bun individ). Procesul de învăţare colectivă a populaţei, auto-adaptarea şi robusteţea sunt câteva din avantajele algoritmilor evolutivi asupra tehnicilor globale de optimizare. Cei mai cunoscuţi algoritmi evolutivi sunt: *algoritmii genetici* (Holland 1975), *programarea evolutivă* (Fogel 1966), *strategiile evolutive*(Rechenberg 1973, Schwefel 1981), *programarea genetică* (Koza 1999). Aplicaţii ale algoritmilor evolutivi

se pot găsi în probleme ca: gruparea datelor, planificare, proiectare, control, simulare şi identificare, etc.

3.1.2 Ant Colony Optimization

Ant Colony Optimization (ACO) este o metaeuristică inspirată din comportamentul coloniilor de furnici care sunt capabile să găsească cel mai scurt drum de la muşuroi la o sursă de mâncare. Acest comportament este determinat de interacţiunea colectivă care duce la găsirea celui mai scurt drum. Euristica ACO a fost propusă de Dorigo şi a fost larg folosită pentru a rezolva probleme de optimizare (ex. problema comis voiajorului, rutarea în graf, colorarea grafurilor, etc) şi a devenit rapid o euristică populară.

3.1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) este o tehnică de optimizare bazată pe populaţii propusă de Eberhart şi Kennedy în 1995, inspirată de comportamentul social al stolurilor de păsări şi al bancurilor de peşti. PSO este o tehnică bazată pe populaţii în care indivizii sunt numiţi particule şi sunt grupaţi în stoluri. Fiecare particulă din swarm reprezintă o soluţie candidat a problemei de optimizat, care "zboară" în spaţiul de căutare urmărind particula optimă de la momentul curent. PSO este aplicat în diferite domenii ca: extragerea de reguli din reţele fuzzy, recunoaşterea de imagini, optimizarea de reţele electrice, etc.

3.2 Mecanisme de adapatare la medii dinamice

Problemele de optimizare dinamice sunt probleme ale căror constrângeri, obiective sau reprezentare se modifică în timp de-a lungul procesului de optimizare. În cazul problemelor în medii statice, scopul algoritmului de optimizare este de a găsi cât mai repede optimul, în schimb, în cazul dinamic, scopul este de a detecta schimbarea şi de a urmări optimul de-a lungul timpului. Euristicile inspirate din natură întâmpină probleme legate de convergenţa prematură în medii dinamice care nu le permit descoperirea de noi optime, deoarece populaţia pierde din diversitate în procesul iterativ. Cea mai simplă soluţie de a rezolva convergenţa prematură este de a reporni algoritmul atunci când se observă o schimbare în mediu, dar acestă abordare are dezavantajul că pierde toate informaţiile stocate de populaţie în momentul repornirii. Astfel, s-au propus diferite tehnici care să rezolve această problemă.

3.2.1 Algoritmi evolutivi

Există diferite tehnici de păstrare a proprietăţilor populaţiei, de a reacţiona la schimbările din mediu fără a fi necesară repornirea alogoritmului [JB05]: menţinerea diversităţii populaţiei, memorii sau multi-populaţii.

Există două direcţii principale în cazul menţinerii diversităţii în populaţie: una reactivă şi una proactivă. Prima abordare constă în apelarea unui mecanism de inducere a diversităţii în populaţie când se observă o schimbare în mediu (ex. hypermutaţia). Abordarea proactivă constă în păstrarea diversităţii de-a lungul întregului proces de căutare. Acest lucru împiedică convergenţa prin diminuarea presiunii selecţiei (tehnici de crowding sau sharing) sau prin stimularea directă a diversităţii (prin imigranţi aleatori în fiecare generaţie).

În cazul abordărilor bazate pe memorii, algoritmii evolutivi conţin o memorie care permite refolosirea informaţiilor dobândite de generaţiile anterioare, aceasta fiind extrem de utilă dacă optimul se reîntoarce în locaţiile anterioare. Abordările bazate pe memorii se pot împărţi în: *explicite* şi *implicite*. În cazul memoriilor explicite, trebuie definită o strategie de stocare şi scoatere de informaţii (ex. readăugarea în populaţie a indivizilor care s-au comportat bine în situaţii similare). În cazul memoriilor implicite, EA conţine o reprezentare redundantă a populaţiei (ex. diploidy).

Un alt mecanism de păstrare a diversităţii este bazat pe populaţii multiple. Dacă fiecare populaţie conţine informaţii despre diferite zone promiţătoare ale spaţiului de căutare procesul de căutarea poate urmări uşor optimul. Această abordare poate fi reactivă dacă populaţia este reiniţializată când se detectează o schimbare în mediu sau proactivă când populaţiile sunt mereu încurajate să caute în diferite spaţii ale mediului.

3.2.2 Ant Colony Optimization

În cazul problemelor dinamice, s-au propus diferite strategii pentru a îmbunătăţi performanţele algoritmului ACO: multi-populaţii, strategii de modificare a matricii de feromon sau noi proprietăţii ale furnicilor (ex. sensibilitate).

3.2.3 Particle Swarm Optimization

S-au propus multe strategii pentru aplicarea algoritmului PSO în medii dinamice deoarece PSO întâmpină probleme legate de neactualizarea memoriei şi pierderea diversităţii. Problemele de

neactualizare a memoriei apar datorită faptului că informaţiile stocate în elemente nu mai sunt valide şi pot ghida în mod greşit căutarea. În acest caz, se consideră de obicei că se ştie când se realizează o modificare în mediu iar elementele sunt reevaluate. Pentru a detecta modificările din mediu se aleg câteva particule aleatoare, *santinele*, care sunt reevaluate. Pe lângă particulele aleatoare, s-a propus folosirea celei mai bune şi celei de a doua cea mai bună particulă pentru a fi considerate santinele. Pentru evitarea convergenţei populaţiei, s-au propus diferite strategii: introducerea diversităţii după detectarea unei schimbări, menţinerea diversităţii de-a lungul căutării sau multipopulaţii.

3.3 Experimente numerice

Existenţa aşa multor mecanisme de menţinere a diveristăţii în populaţii, ridică problema găsiri mecanismului potrivit pentru o anumită problemă. În afară de a fi efectiv, un mecanism ar trebui să fie cât mai simplu, pentru a nu creşte costul computaţional al algoritmului. În aceeaşi timp acelaşi mecanism ar putea avea un impact diferit asupra diferitelor abordări bazate pe populaţii. Astfel în [ZZ06] am analizat nişte mecanisme simple de păstrare a diversităţii aplicate la două metode de optimizare bazate pe populaţii: Differential Evolution (DE) [SP95] şi Particle Swarm Optimization (PSO) [KE95].

3.3.1 Differential Evolution

Differential evolution (DE) a fost introdusă în [SP95] şi este o metodă de optimizare bazată pe populaţii care utilizează un operator de recombinare aleator şi o selecţie bazată pe competiţia dintre copil şi părinte.

3.3.2 Mecanisme de adaptare la mediul dinamic

Pentru a împiedica convergenţa prematură a celor doi algoritmi analizaţi, DE şi PSO în medii dinamice, am analizat impactul unor mecanisme de menţinere a convergenţei. În cazul DE, am analizat o schemă de adaptare a parametrilor, introducerea de elemente aleatoare şi multipopulaţii; pentru PSO au fost testate de asemenea o simplă schemă de perturbare şi multipopulaţii.

Mecanisme de adaptare pentru DE

În [ZZ06], ca măsură de diversitate am folosit varianţa. Pentru a menţine diversitatea algoritmului la acelaşi nivel pe tot parcursul procesului, la fiecare generaţie, t, este calculat raportul $c_j = \gamma \cdot Var(x^i)/Var(z^i)$. Un alt mecanism testat în [ZZ06] este o abordare cu multipopulaţii ce se caracterizează prin faptul că pentru fiecare populaţie se aplică algoritmul adaptiv DE, periodic un proces de migrare aleatoare este început: fiecare element al fiecărei populaţii, cu o anumită probabilitate, este interschimbat cu un element ales aleator dintr-o sub-populaţie arbitrară. Chiar dacă în cazul static aceste mecanisme împiedică convergenţa prematură, aceste mecanisme nu sunt suficient de bune în cazul dinamic.

Problema pe care am analizat-o în [ZZ06] este de a găsi cel mai simplu mecanism care să fie suficient de puternic încât să permită urmărirea optimului care se modifică. În cazul modificărilor continue ale poziţei optimului (fără salturi mari), ar fi suficient să perturbăm cel mai bun element pentru a ajunge cu populaţia în zonele unde se află noul optim. Folosirea unui "nor" de elemente care oscilează în jurul optimului ar asigura posibiliatea algoritmului de a urmări optimul.

În DE cu elemente aleatoare, populaţia este împărţită în două părţi: o mulţime de elemente care urmează regurile DE şi o altă mulţime $\{x_1, \ldots, x_\mu\}$ de elemente care oscilează în jurul celui mai bun element, x_*, al populaţiei anterioare:

$$y_i^j = x_*^j + K_j \cdot N(0,1),\ i = \overline{\mu+1, m},\ j = \overline{1, n} \tag{3.1}$$

unde $N(0,1)$ este o variabilă normală aleatoare cu o distribuţie standard şi parametrul $K_j > 0$ controlează amplitudinea perturbării.

Mecanisme de adaptare pentru PSO

Algoritmul PSO are probleme de convergenţă prematură când viteza, v_i devine prea mică şi particulele nu se mai mişcă. În [ZZ06] am analizat două mecanisme de împiedicare a convergenţei: creşterea vitezei particulei şi perturbarea poziţiei particulei. În [LA05], este propus un mecanism de perturbare a vitezei particulei: dacă valoarea vitezei este mai mică decât o valoare prag v_c, atunci este înlocuită cu o valoare aleatoare $U(-1,1)V_{max}/\rho$. Varianta propusă în [LA05] se numeşte Turbulent Particle Swarm Optimization (TPSO). Noi am propus o variantă mai

simplă, nu de perturbare a vitezei ci de perturbare a poziţie particulei staţionare:

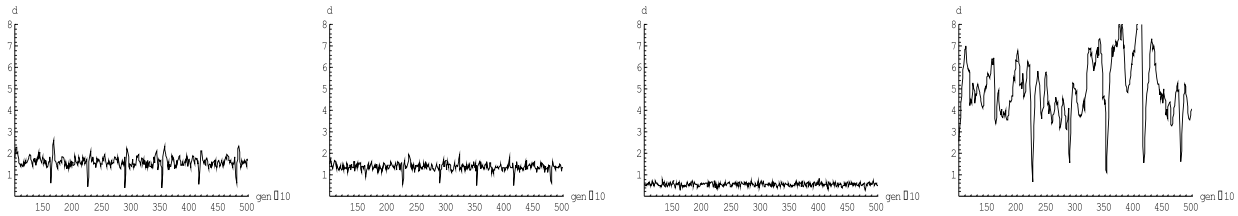
$$x_i^j(t+1) = \begin{cases} x_i^j(t) + v_i^j(t+1), & \text{if } v_i^j(t+1) \geqslant v_c \\ x_i^j(t) + K_j \cdot N(0,1), & \text{if } v_i^j(t+1) < v_c \end{cases} \tag{3.2}$$

unde N(0,1) este o variabilă aleatoare normală cu o distribuţie standard şi parametrul $K_j > 0$ controlează amplitudinea perturbării. Cum orice modificare în poziţie influenţează viteza particulei, particula poate ieşi din optimul local.

3.3.3 Rezultate numerice

Pentru a analiza impactul mecanismelor de păstrare a diverităţii în medii dinamince am folosit pentru teste o variantă dinamică a funcţiei Ackley şi o variantă a funcţiei de test Moving Peaks Benchmark [Bra].

Evoluţia distanţei medii dintre cel mai bun element din populaţie şi optimul curent este ilustrată în Figura 3.1 pentru câteva variante ale algoritmului. Pentru valori mici a lui γ (e.g. $\gamma = 1$), varianta adaptivă de DE nu reuşeşte să urmărească optimul (Figure 3.1 (d)). Pe de altă parte, folosirea unui procent între 25% şi 50% de elemente aleatoare îmbunătăţeşte semnificativ comportamentul algoritmului. S-a remarcat că pentru a urmări optimul este necesar să se reevalueze elementele populaţei curente la fiecare generaţie.



(a) (b) (c) (d)

Figura 3.1: Comportamentul DE pentru varianta dinamică a funcţiei Ackley. (a) DE cu parametri ficşi ($p = 0.5$, $F = 0.5$); (b) DE adaptiv cu $\gamma = 1.25$; (c) DE cu parametri ficşi D ($p = 0.5$, $F = 0.5$) şi 50% elemente aleatoare; (d) DE adaptiv cu $\gamma = 1$

Pentru algoritmul DE, am analizat impactul adaptării parametriilor şi elementelor aleatoare. După cum se observă în Figura 3.2, varianta adaptivă DE are un comportament acceptabil dar nu la fel de bun ca în cazul folosiri unui mic procent de elemente aleatoare.

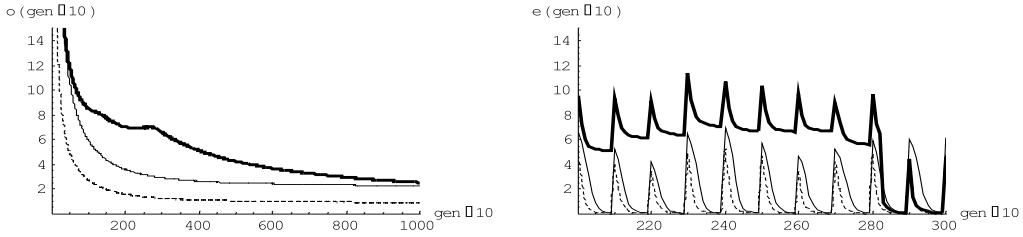


Figura 3.2: Comportamentul DE pentru un optim care îşi schimbă poziţia: eroarea offline (stânga), eroarea curentă (dreapta). Variante DE: DE adaptiv ($\gamma = 1$) + 50% elemente aleatoare (linia groasă), DE adaptiv ($\gamma = 1$) + 25% elemente aleatoare (linia întreruptă), DE adaptiv ($\gamma = 1.25$) fără elemete aleatoare (linia subţire).

Cum convergenţa algoritmului PSO este mai lentă decât a algoritmului DE, este de aşteptat ca algoritmul PSO clasic să poată urmări optimele care se mişcă lin. Figura 3.3 sugerează că pentru probleme dinamice simple nu este nevoie de un algoritm suplimentar de menţinere a diversităţii (precum perturbarea vitezei/poziţiei sau elemente aleatoare).

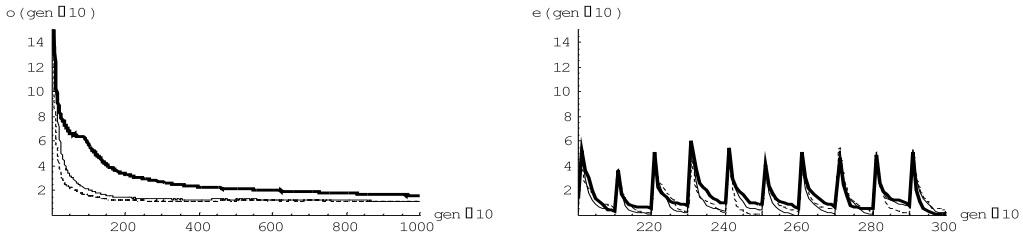


Figura 3.3: Comportamentul PSO pentru un optim care îşi schimbă poziţia: eroarea offline (stânga), eroarea curentă (dreapta). Variante PSO: clasic (linia întreruptă), PSO perturbat(linia subţire), PSO + 25% elemente aleatoare (linie normală).

3.4 Concluzii

Atât Turbulent PSO introdus în [LA05] cât şi simpla perturbare pe care am propus-o în [ZZ06] s-au dovedit a fi eficiente în evitarea convergenţei premature. În ce priveşte multipopulaţiile, experimantele numerice au arătat că încetinesc convergenţa prematură dar nu reuşec să o evite.

În cazul optimelor care se schimbă lent s-au obţinut rezultate bune prin utilizarea unui mecanism simplu de menţinere a diversităţii (ex. adapatarea parametriilor cu valori mari ale varianţei sau adăugarea de elemente aleatoare în cazul DE) sau chiar nefolosirea nici unui

mecanism suplimentar (în cazul PSO). Dar dacă severitatea schimbării este mare, scheme suplimentare ar trebui folosite (ex. excluziunea [MM05], charged PSO [BB02]).

Capitolul 4

Planificarea taskurilor în medii dinamice distribuite

Problema de planificare a taskurilor a fost larg studiată în ultimul timp, drept consecinţă există mulţi algoritmi care rezolvă diferite variante ale problemei de planificare.

4.1 Problema planificării pe grid

Problema de planificare pe grid este o problemă combinatorială NP-complexă. S-au propus diferite variante ale problemei în funcţie de proprietăţile taskurilor, particularităţile mediului şi ale procesului de planificare.

Formalizare. Scopul problemei de planificare este de a găsi o alocare a resurselor la taskuri astfel încât constrângerile takurilor (hardware, software) şi constrângerile de precedenţă să fie îndeplinite şi să se optimizeze criteriile de calitate. Atribuirea taskurilor la resurse se face pe baza unei estimări a timpului de execuţie a taskului pe diferite resurse. Fie un set de n taskuri, $\{t_1, \ldots, t_n\}$, care urmează să fie planificate pe un set de $m < n$ procesoare, $\{p_1, \ldots, p_m\}$. Presupuneam că pentru fiecare pereche (t_i, p_j) ştim o estimare $ET(i, j)$ a timpului necesar ca taskul t_i să se execute pe procesorul p_i. O planificare este o mapare a taskurilor la resurse, $S = (p_{j_1}, \ldots, p_{j_n})$, unde $j_i \in \{1, \ldots, m\}$ şi p_{j_i} reprezintă procesorul la care taskul t_i este atribuit. Dacă $\mathcal{T}_j$ reprezintă un set de taskuri asignate la procesorul p_j şi T_j^0 reprezintă momentul de timp de la care procesorul j este liber atunci timpul de terminare al acestui procesor va fi $CT_j = T_j^0 + \sum_{i \in \mathcal{T}_j} ET(i, j)$. Makespanul este timpul maxim de terminare a tuturor procesoarelor, $makespan = \max_{j=\overline{1,m}} CT_j$. Problema care rămâne de rezolvat este de a minimiza valoarea makespanului.

4.2 Abordări într-un mediu static

Deoarece soluţiile suboptimale ale acestei probleme sunt acceptate în practică, noi am testat două euristici inspirate din natură: algoritmii evolutivi (Simple Population-based Scheduler - SPS) şi unul inspirat de euristica Ant Colony Optimization. Experimentele realizate au fost făcute în medii statice şi dinamice.

4.2.1 Simple Population-based Scheduler

În [ZFZ11] am propus un planificator numit *SPS - SimplePopulationScheduler*. Elementul cheie al acestui algoritm este strategia de perturbare (mutaţie), care este modulul care defineşte strategia de îmbunătăţire a planificării. Un alt element care influenţează performanţele planificatorului este pasul de iniţializare.

Iniţializarea populaţiei. Construirea de planificări (sub)-optimale se bazează pe crearea unei planificări iniţiale care este îmbunătăţită iterativ. Când se construieşte o planificare iniţială se iau două decizii: (i) ordinea în care taskurile sunt atribuite la procesoare; (ii) criteriul folosit pentru a selecta procesorul corespunzător fiecărui task. În funcţie de aceste criterii, există câteva strategii care pot fi aplicate: aleator, OLB, MET, MCT, Max-Min şi Min-Min. Fiecare din aceste strategii generează planificări, care ar putea suferi îmbunătăţiri. Din această cauză, ar fi util să nu folosim doar o singură strategie ci o populaţie de strategii construite folosind diferite strategii. Adăugarea de planificări generate prin diferite euristici în populaţia iniţială a fost des utilizată [BSB+01, PKN10, Xha07].

Perturbarea. Planificările iniţiale construite de euristicele de planificare sunt de obicei ne-optimale şi pot fi îmbunătăţite prin mutarea sau interschimbarea de taskuri între resurse. În funcţie de criteriul utilizat, pentru a selecta resursele sursă, destinaţie şi taskul care va fi realocat, se pot construi o mulţime de strategii de perturbare a planificării [XA10]. Majoritatea operatorilor de perturbare care se regăsesc în euristicele de planificare se bazează pe două operaţii: mutarea (”move”) unui task de pe o resursă pe alta şi interschimbarea (”swap”) a două taskuri între resursele lor. Strategiile de perturbare prezentate în Tabelul 4.1 au fost selectate pe baza simplicităţii, eficienţei şi balansului între caracterul greedy şi aleator. Perturbarea ”random move” corespunde operatorului ”local move” [Xha07] şi este asemănător cu operatorul de mutaţie folosit de algoritmii evolutivi. Operatorul ”greedy move” este asemănător cu operatorul ”steepest local move” [Xha07], dar implică un grad mai mare de lăcomie deoarece implică întotdeauna cel mai încărcat procesor. Operatorul ”greedy swap” este similar cu ”steepest local

Sursă		Destinaţie		Strategie
Procesor	Task	Procesor	Task	
Aleator	Aleator	Aleator	-	*Random Move*
Cel mai încărcat (max CT)	Aleator	Cea mai bună îmbunătăţire	-	*Greedy Move*
Cel mai încărcat (max CT)	Aleator	Cel mai puţin încărcat (min CT)	Random	*Greedy Swap*

Tabelul 4.1: Caracteristicile strategiilor utilizate pentru a perturba planificarea

swap" [Xha07], dar este mai puţin lacom şi costisitor deoarece nu implică o căutare peste întregul set de taskuri. Dacă aplicăm la fiecare pas doar un tip de perturbare (random move, greedy move or greedy swap), vom numi strategia de perturbare *SimplePerturbation*, dacă aplicăm o variantă care combină celor trei strategii perturbarea va fi numită *HybridPerturbation*.

Deoarece doar un pas de aplicare a perturbaţiei nu duce neapărat la o îmbunătăţire a calităţii perturbaţiei, am optat pentru o aplicare iterativă a pasului de perturbare, până când fie n iteraţii sunt executate (fiecare task are şansa să fie mutat) sau până când un număr, g_p, maxim de perturbaţi s-au executat cu eşec.

4.2.2 Ant Colony Optimization

Această secţiune prezintă particularităţile planificatorului inspirat de comportamentul coloniilor de furnici Ant Scheduler, inspirat de algoritmul introdus în [RL04].

Iniţializarea Feromonului. Pentru a ajuta construirea de planificări bune am utilizat ideea de încorporare a informaţiei corespunzătoare unei planificări generate cu euristica Min-Min. Ideea de a adăuga informaţii obţinute de euristici greedy este o abordate comună în cazul planificatoarelor evolutive [CX06] şi îmbunătăţeşte şi comportamentul planificatorului bazat pe ACO.

Căutarea locală. Pe baza concluziilor oferite de studiile anterioare [RL04], căutarea locală poate îmbunătăţi semnificativ comportamentul planificatorului bazat pe ACO. Astfel, am aplicat un pas de îmbunătăţire a soluţiei optime la fiecare epocă a algoritmului. Pasul de îmbunătăţire este bazat pe un operator de rebalansare, care este asemănător cu operatorul de mutaţie (*HybridPerturbation*), utilizat de algoritmul evolutiv. Operaţia este repetată până cand nu se mai observă o îmbunătăţire a soluţiei.

4.2.3 Rezultate numerice

În cazul algoritmului Simple Population-based Scheduler (SPS) am analizat influenţa strategiilor de perturbare [ZFZ11]. De asemenea am analizat abilitatea SPS şi a planificatorului bazat pe ACO (AS) de a construi planificări într-un mediu static pentru trei tipuri de medii: consistent (C), semi-consistent (S) şi inconsistent (I) [ZZC10a, ZZC10b].

Mediu de test. Datele de test utilizate au fost introduse în [BSB+01], şi oferă matrici care conţin valori pentru o estimare a timpul de execuţie (ET) generate pe baza diferitelor presupuneri asupra heterogenităţii taskurilor, heterogenităţii şi consistenţei resurselor.

Analiza strategiilor de perturbare. Scopul studiului numeric în cazul mediilor statice în [ZFZ11] a fost de a analiza influenţa diferitelor strategii de perturbare asupra performanţelor algoritmului Simple Population-based Scheduler (SPS). De asemenea, am analizat câteva strategii de iniţializare: (i) aleatoare; (ii) introducerea în populaţie a unor euristici de planificare; (iii) utilizarea unor euristici de planificare perturbate aleator; (iv) utilizarea euristicii Min-Min şi a elementelor perturbate ale ei. După cum ne aşteptam, au fost obţinute cel mai bun comportament atunci când populaţia iniţială conţine elemente generate folosind alte euristici de planificare, iar cel mai rău comportament s-a observat în cazul elementelor aleatoare.

Am comparat patru variante de perturbare (aleatoare, bazată pe mutarea unui task, bazată pe interschimbarea a două taskuri şi una hibridă) cu algoritm memetic hibridizat cu tabu search (MA+TS) [Xha07]. Chiar dacă se bazează pe operatori mai simpli, algoritmul SPS găseşte soluţii calitativ similare cu algoritmul MA+TS, în cazul datelor inconsistente ("u_i_**" problems), algoritmul propus găseşte rezultate mai bune.

Analiza planificatoarelor SPS şi AS în mediu static. În [ZZC10a, ZZC10b] am analizat comportamentul planificatoarelor Simple Population-based Scheduler (SPS) şi a planificatorului Ant Scheduler (AS) într-un mediu static. În cazul algoritmului SPS, am folosit pentru teste o strategie simplă de perturbare - GreadyMove. Ca date de test am folosit datele caracterizate prin taskuri şi procesoare cu un grad mare de heterogenitate în diferite tipuri de medii: consistent ("u_c_hihi"), inconsistent ("u_i_hihi") şi semi-consistent ("u_s_hihi"). Rezultatele obţinute în [CX06] sunt puţin mai bune decât cele obţinute de planificatoarele noastre SPS şi AS în cazurile consistent şi semiconsistent, în timp ce în cazul inconsistent planificatoarele noastre se comportă mai bine.

4.3 Analiza robusteţii euristicilor

În [**Z**ZC10b], am studiat robusteţea algoritmilor de planificare prezentaţi anterior. O planificare bună ar trebui să satisfacă trei proprietăţi: (i) ar trebui să se obţină relativ repede; (ii) ar trebui să se adapteze uşor într-un mediu cu modificări line; (iii) ar trebui să fie robustă, adică să fie puţin sensibilă la modificările din timpul rulării [CJSZ08]. Rezultatele obţinute sugerează, în cazul mediilor consistente şi semiconsistente, că euristica Min-Min duce la rezultate mai robuste decât planificatoarele SPS şi AS, în timp ce în cazul inconsistent, planificatoarele SPS şi AS dau rezultate mai bune.

4.4 Planificare Online

În cazul planificării online, taskurile vin secvenţial şi sunt planificate atunci când ajung; această abordare diferă faţă de abordările anterioare unde taskurile vin sub formă de grupuri (batch). În [**Z**FZ11] câteva euristici dinamice care se bazează pe îmbătrânire au fost testate în comparaţie cu variantele lor bazate de populaţii (euristica specifică a înlocuit operatorul de perturbare din SPS). Comportamentul acestor algoritmi a fost de asemenea comparat cu o variantă a algoritmului SPS care utilizează o variantă de aplicare neiterată a perturbaţiei (pentru fiecare generaţie, perturbarea este aplicată o singură dată). Printre algoritmii testaţi se află o variantă a algoritmului DMECT descris în [Fr9], algoritmul MinQL descris în [FMC09] şi euristicele Min-Min şi Max-Min la care s-a adăugat un mecanism de îmbătrânire.

Mediu de test. Pentru planificare online, am considerat un model de simulare în care timpii de execuţie a taskurilor (ET) urmăresc o distribuţie Paretro. Rata de venire este simulată folosind un polinom de gradul 8 determinat pe baza activităţii unei reţele reale [Fei10]. A fost generat un număr de 500 de taskuri pentru fiecare test. Replanificarea a fost executată la fiecare 250 unităţi de timp utilizând un min de 1000 de unităţi pentru ET. Toate testele au fost repetate de 20 de ori.

Rezultate numerice. Au fost testate câteva euristici dinamice de planificare la care s-a adăugat un mecanism de îmbătrânire (DMECT, MinQL) versus variantele lor care folosesc scheletronul algoritmului SPS dar înlocuiesc pasul de perturbare cu propriul comportament (pDMECT, pMinQL). Comportamentul lor a fost de asemenea comparat cu o versiune a algoritmului SPS în care pasul de perturbare se aplică o singură dată.

Tabelul 4.2 prezintă beneficiile abordării bazate pe populaţii (pDMECT, pMinQL) în cazul planificării online. Variantele bazate pe populaţii ale algoritmilor DMECT şi MinQL s-au dovedit mai bune decât variantele simple, iar varianta pDMECT are un comportament similar cu SPS (cele mai bune valori din tabelul 4.2 au fost validate folosind un test t-test cu un nivel de semnificaţie egal cu 0.05). Singura diferenţă notabilă este legată de viteza de execuţie. pDMECT a avut nevoie de aproape 30 de secunde pentru a construi o planificare în timp ce SPS doar 3 secunde în medie.

	DMECT	pDMECT	MinQL	pMinQL	SPS	Max-Min	Min-Min
MS	66556.20	**49409.11**	76564.40	54332.89	**46996.76**	61165.15	68774.87
	±15097.85	±9522.13	±18114.51	±9891.15	±8812.87	±11936.19	±15101.05
Timp	66.56	28343.04	3.06	2254.64	2777.70	684.49	669.21
(ms)	± 15.50	±10702.15	±2.52	± 314.45	± 578.22	±242.15	±209.99

Tabelul 4.2: Valoarea makespan (MS) obţinută de euristicele de planificare în cazul planificatorului online

4.5 Adaptarea în mediul dinamic

În această secţiune vom prezenta analiza făcută în [ZZC10a, ZZC10b] referitoare la utilizarea unor mecanisme de păstrare a informaţiilor din paşi anteriori de planificare pentru algoritmii inspiraţi din natură prezentaţi anterior.

4.5.1 Simple Population-based Scheduler

În cazul mediului dinamic, procesul de planificare constă în evenimente de planificare consecutive. La fiecare eveniment de planificare, lista de procesoare diferă de lista de procesoare de la pasul anterior. Dacă diferenţa dintre listele de procesoare nu este prea mare, putem reutiliza informaţiile găsite în evenimentul anterior de planificare.

Pentru a putea reutiliza informaţiile găsite în evenimentele anterioare de planificare în [ZZC10a, ZZC10b] am analizat comportamentul a două variante: utilizarea planificărilor bune din evenimentul anterior de planificare şi folosirea unei arhive care conţine cele mai bune planificatoare din evenimentele anterioare.

4.5.2 Ant Colony Optimization

Matricea de feromon asigură comunicare între furnici, deoarece este accesată de toate furnicile în fiecare epocă a fiecărui eveniment de planificare. Din această cauză, pare a fi utilizată pentru a fi folosită ca metodă de comunicare între evenimente de planificare diferite. Acest lucru presupune că matricea de feromon nu este reiniţializată la fiecare eveniment de planificare când lista de procesoare se schimbă. Valorile obţinute în evenimentul anterior de planificare sunt menţinute. Particularitatea acestei abordări este că valorile matricii de feromon care aparţin procesoarelor care nu sunt prezente în lista de procesoare de la evenimentul curent de planificare rămân neschimbate în pasul curent.

4.5.3 Experimente în mediul dinamic

Analiza experimentală este realizată pe datele de test prezentate în [BSB+01] pentru estimările timpilor de execuţie şi caracterul dinamic al mediului, prin faptul că unele procesoare devin neutilizabile în unele momente ale procesului de planificare.

Mediu de test. Pentru a simula comportamentul dinamic al mediului, am generat pentru fiecare eveniment de planificare o listă nouă de procesoare disponibile prin înlăturarea aleatoare a unui procent de procesoare din lista iniţială de 16 procesoare. Procentele utilizate în experimente au fost de 10%, 20% şi 40%.

Rezultate numerice. În cazul mediului dinamic am analizat comportamentul algoritmilor dinamici SPS şi AS. Pentru algoritmul evolutiv am folosit o strategie de perturbare simplă, *Greedy Move*.

Scopul experimentelor a fost de a analiza impactul mecanismelor de memorie (refolosirea planificărilor anterioare sau matrici de feromon) asupra abilităţii planificatorului evolutiv şi a celui bazat pe comportamentul furnicilor de a se adapta în mediul dinamic. Tablelul 4.3 conţine raportul dintre numărul de evenimente de planificare în care varianta dinamică s-a comportat mai bine ca varianta statică şi de asemenea raportul dintre numărul de evenimente în care varianta statică s-a comportat mai bine decât cea dinamică. Putem observa că, comportamentul celor doi algoritmi este diferit. În cazul SPS, atât timp cât diferenţa dintre maşinile indisponibile este mai mică decât 12%, folosirea arhivei de memorie aduce beneficii pentru toate tipurile de mediu (consistent, semi-consistent, inconsistent). Totuşi, beneficiul este mai mare în cazul mediilor consistente şi mai mic în cazul mediilor inconsistente. Pentru algoritmul AS, mecanismul de memorie bazat pe conservarea matricii de feromon nu aduce îmbunătăţiri în cazul

mediului consistent. Pe de altă parte, dacă mediul este inconsistent, varianta dinamică de AS este mai bună decât varianta statică şi de asemenea depăşeşte performanţele variantei dinamice obţinute cu SPS.

Problema	Procesoare neutilizabile (%)	SPS		AS	
		Dinamic	Static	Dinamic	Static
C	10	42/50	0/50	0/50	49/50
C	20	5/50	3/50	2/50	39/50
C	40	1/50	3/50	0/50	30/50
S	10	35/50	1/50	0/50	49/50
S	20	5/50	2/50	5/50	43/50
S	40	0/50	3/50	12/50	35/50
I	10	31/50	0/50	49/50	0/50
I	20	4/50	1/50	39/50	1/50
I	40	0/50	1/50	20/50	2/50

Tabelul 4.3: Raportul dintre numărul de evenimente de planificare când există o diferenţa statistică între variantele din mediul static şi dinamic ale algoritmilor AS şi SPS

4.6 Concluzii

În acest capitol am prezentat doi algoritmi de planificare: un planificator evolutiv (Simple Population-based Scheduler) şi unul inspirat din comportamentul furnicilor. Am analizat comportamentul planificatoarelor în medii statice [ZZC10a, ZZC10b, ZFZ11], dinamice [ZZC10a, ZZC10b] şi în cazul planificării online [ZFZ11].

Capitolul 5

Algoritmi de grupare a datelor inspiraţi din natură

Pe lângă tehnicile clasice de grupare ierarhică şi partiţională, tehnicile inspirate din natură, ca gruparea de date evolutivă sau gruparea de date bazată pe swarms, sunt aplicate cu succes şi în gruparea datelor.

5.1 Coloniile de furnici şi gruparea datelor

Comportamentul coloniilor de furnici a inspirat dezvoltarea diferitelor tehnici de grupare. Diferite abordări sunt bazate pe diferite aspecte ale comportamentului furnicilor: (i) organizarea cimitirelor şi sortarea larvelor; (ii) recunoaşterea chimică a vecinilor de muşuroi [LMV02]; (iii) abilitatea de a construi punţi din furnici; (iv) strategiile de vânare a prădătorilor la specia Pachycondyla Apicalis.

5.2 Algoritmul `AntClust`

Algoritmul `AntClust` propus în [LMV02], simulează aşa numita "apropiere colonială" bazată pe feromon în coloniile de furnici. Feromonul este o substanţă chimică pe care furnicile o posedă şi care le permite să facă diferenţa dintre colegii de colonie şi intruşi. Fiecare furnică deţine o informaţie despre "mirosul" coloniei, care este mereu modificată. Pornind de la această idee, Labroche et al. a propus un model de furnică artificială care reuşeşte să clasifice date.

Fiecare furnică, i, are următoarele caracteristici: (1) o dată asociată, x_i, care este un element unic ce nu se modifică în timpul procesului de grupare; (2) o etichetă, L_i, care este un număr ce identifică clusterul; (3) un prag de similaritate, T_i, care este utilizat pentru a decide dacă două furnici sunt suficient de similare ca să aparţină aceleiaşi colonii (4) vârsta, A_i, care de fapt este egală cu numărul de întâlniri la care a participat furnica (5) un parametru adaptiv, M_i, care reprezintă dimensiunea coloniei aşa cum o percepe furnica, (6) un parametru adaptiv, M_i^+, care reprezintă gradul de acceptanţă al furnicii în colonie. Cu cât gradul de acceptanţă este mai mare cu atât furnica este mai bine integrată în colonie.

Procesul de grupare se realizează în trei paşi: pasul de învăţare a pragului de similaritate, pasul de întâlniri aleatoare şi pasul de rafinare a clusterelor mici. În pasul în care se realizează întâlnirile se aplică cinci reguli în funcţie de proprietăţile furnicilor: (i) crearea de noi colonii, (ii) acceptarea unei furnici într-o colonie, (iii) întâlnirea pozitivă între două furnici care aparţin aceleiaşi colonii ¸ (iv) întâlnirea dintre furnici care nu aparţin aceleiaşi colonii.

5.3 Tratarea zgomotelor în tehnicile de grupare inspirate de comportamentul furnicilor

În [ZZ05b] am analizat comportamentul algoritmului `AntClust` pentru două seturi de date generate artificial. Primul set este format din 6 clustere elipsoidale, generate utilizând o distribuţie normală bidimensională, peste care s-a suprapus un zgomot cu o distribuţie uniformă (Figura 5.1(a)). Al doilea set este format din 2050 de puncte grupate în 4 clustere care au forme geometrice diferite (punctele au fost generate uniform în interiorul formelor) şi 750 de puncte uniform distribuite în afara formelor geometrice, reprezentând zgomotul (Figura 5.1(d)).

Algoritmul `AntClust` (aplicat pentru $k_M = m^2/10$ pentru primul set de date şi $k_M = m^2/2$ pentru al doilea set) identifică 6 şi respectiv 4 clustere cum se observă în Figura 5.1(d). Cum M^+ este o măsură a acceptanţei furnicii în colonie, este natural să o interpretăm ca pe un nivel de separare a datelor de zgomote. În acest caz, datele care au valori mici pentru M^+ pot fi considerate ca zgomote. Problema care apare în acest caz este de a găsi un prag care face separarea dintre date şi zgomote. Deoarece toate valorile M^+ se găsesc în intervalul $[0, 1)$ am putea folosi o valoare absolută (ex. 0.1). De asemenea se poate aplica un criteriu bazat pe o ordonare statistică. Rezultatele care s-au obţinut prin considerarea unui sfert din datele M^+ asociate ca fiind zgomote sunt prezentate în Figurile 5.1 (b), (c) şi 5.1 (e), (f). În cazul

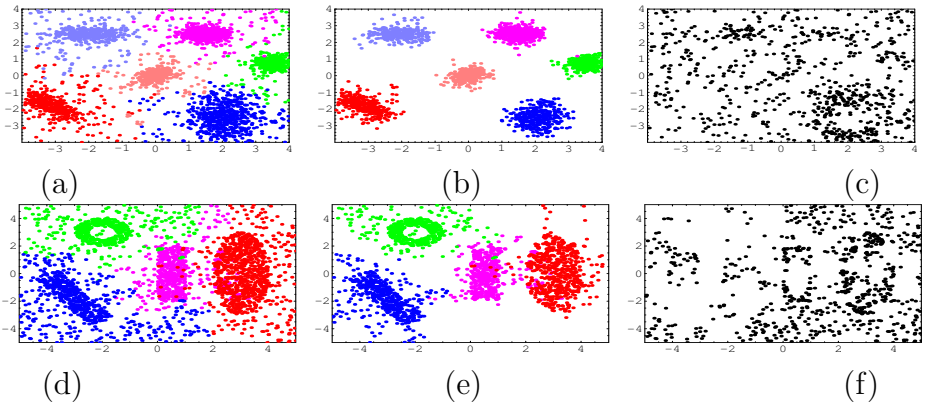


Figura 5.1: (a),(d) Rezultate obţinute prin aplicarea algoritmului `AntClust` iniţial; (b),(e) Clusterele identificate prin ignorarea punctelor pentru care parametrul de acceptanţă (M^+) este mai mic decât un sfert din toate valorile lui M^+; (c),(f) Punctele ignorate (estimarea zgomotului)

.

clusterelor elipsoidale, rezultatele sunt acceptabile, dar în cazul datelor geometrice, zgomotul nu este bine definit. Acest lucru ne sugerează folosirea unei măsuri legate de densitate.

5.3.1 Adăugarea informaţiei de densitate la algoritmul `AntClust`

Pentru a introduce informaţia legată de densitate în algoritmul `AntClust` am propus adăugarea unui nou parametru D_i pentru fiecare furnică. Acest parametru este o estimare a densităţii regiunii în care se află furnica. Acest parametru este iniţializat cu 0 şi este ajustat la fiecare întâlnire a furnici i cu o altă furnică j cu $D_i := D_i + \Delta_i$ unde

$$\Delta_i = \begin{cases} exp\left(\frac{(1-S(i,j))^2}{2\sigma_i^2}\right) & \text{if } 1 - S(i,j) \leqslant \sigma_i \\ 0 & \text{if } 1 - S(i,j) > \sigma_i \end{cases} \tag{5.1}$$

unde σ_i sunt parametrii care controlează aria de influenţă a fiecărei date. De obicei $\sigma_i = \sigma$ pentru toţi i. Această măsură de densitate este similară cu cea folosită de algoritmul DENCLUE, dar în loc de a o calcula folosind o căutare sistematică a vecinilor, ne folosim de întâlnirile aleatoare la care participă furnicile.

După terminarea pasului de întâlniri, parametru D_i se împarte la vârsta furnici, A_i, informaţia de densitate fiind păstrată în intervalul $[0, 1)$. Prima întrebare care apare este dacă informaţia stocată în D_i este diferită de cea din M^+.

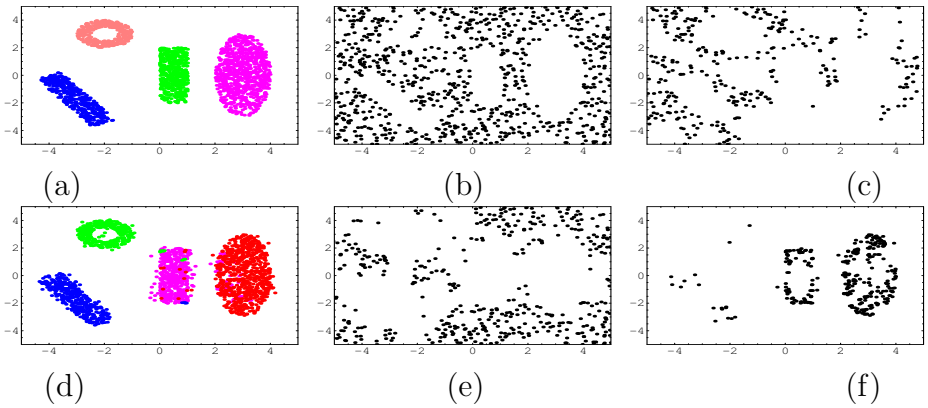


Figura 5.2: Clustere geometrice (a) clusterele originale; (b) zgomotul suprapus pe clustere; (c) datele în a treia categorie; (d) clusterele identificate (a patra categorie); (e) zgomotul identificat (prima categorie); (f) datele în a doua categorie.

Un prim mod de utilizare a valorilor D_i este în pasul de rafinare a clustelor: elementele fără cluster sunt asignate unui cluster doar dacă valoarea de densitate este mai mare decât un sfert din valorile de densitate. Când căutăm elementul cel mai similar utilizat pentru a fi asignat la un cluster, se iau în considerare doar elementele care aparţin clusterului şi au densitatea mai mare decât un sfert din valorile densităţilor. În acest mod apare o nouă clasă, cea a valorilor neclasificate care sunt considerate a fi zgomote.

Parametru σ joacă un rol important în definirea clusterelor. Am analizat influenţa acestui parametru asupra comportamentului algoritmului `AntClus` pentru setul de date care conţine clustere sub formă de figuri geometrice (Figura 5.1 (d)) şi zgomot distribuit uniform în exteriorul figurilor geometrice. Algoritmul ar trebui să identifice 5 clase, patru corespunzând datelor utile şi una zgomotului. Pentru a evalua calitatea grupării am calculat eroarea de măsură introdusă în [LMV02].

Pe de altă parte, rolurile diferite pe care D_i şi M_i^+ le joacă ne sugerează folosirea ambelor măsuri pentru a separa zgomotul de datele utile. Acest lucru presupune împărţirea datelor în patru categorii pe baza valorii parametrilor D_i şi M_i^+. Separarea se realizează cu ajutorul a două praguri T_M (prag pentru gradul de acceptanţă) şi T_D (prag pentru densitate). Cele patru categorii sunt: (1)*Prima categorie.* Conţine toate datele i pentru care $M_i^+ \leqslant T_M$ şi $D_i \leqslant T_D$ şi corespunde datelor care pot fi considerate zgomote (Figura 5.2(e)); (2)*A doua categorie.* Conţine toate datele i pentru care $M_i^+ \leqslant T_M$ şi $D_i > T_D$ şi corespunde datelor care au o estimare mare a densităţii dar un grad mic de acceptanţă. Aceste date nu au putut fi clasificate chiar dacă aparţin unor regiuni dense. De obiecei aceste date care se află pe marginea clusterelor (Figura 5.2(f)) şi sunt similare cu punctele graniţa ale algoritmului DBSCAN; (3)*A*

treia categorie. Conţine toate datele i pentru care $M_i^+ > T_M$ şi $D_i \leqslant T_D$ şi care corespunde la date cu un grad mare de acceptanţă la cluster, estimate a se afla în regiuni rare. De obicei aceste date corespund unor regiuni rare, dar au participat la multe întâlniri (Figura 5.2(c)). Existenţa acestei categorii se explicată prin diferitele praguri utilizate în estimarea lui M_i^+ (T_i) şi prin modul în care se calculează D_i (σ); (4)*A patra categorie.* Conţine toate datele i pentru care $M_i^+ > T_M$ and $D_i > T_D$ şi corespunde datelor care aparţin unor regiuni dense şi au fost acceptate în clusterele lor (Figura 5.2(d)).

5.4 AntClust aplicat în cazul datelor medicale

În [ZZ05a] am analizat comportamentul algoritmului `AntClust` propus de Labroche et al. [LMV02] în cazul datelor medicale caracterizate atât de atribute numerice cât şi categoriale, dar şi de date lipsă. Algoritmii de grupare bazaţi pe comportamentul furnicilor a fost aplicaţi cu succes în gruparea documentelor, web mining, dar comportamentul în cazul datelor medicale care conţin atribute numerice şi categoriale precum şi elemente lipsă nu a fost analizat. În [LMV02], `AntClust` a fost aplicat în cazul datelor medicale care conţin date numerice. Toate datele conţin doar atribute numerice şi nu conţin valori lipsă.

5.4.1 Măsuri de similaritate/disimilaritate potrivite în cazul datelor medicale

Un element cheie în procesul de clasificare este alegerea unor măsuri adecvate de similaritate/disimilaritate. Când alegem astfel de măsuri, trebuie să luăm în considerare natura datelor care vor fi procesate, deoarece o măsură potrivită datelor numerice nu ar fi potrivită şi datelor categoriale. Pe de altă parte, datele medicale sunt de multe ori caracterizate prin valori lipsă. Există diferite metode de corectare a acestor măsuri: înlăturarea datelor lipsă, strategii bazate pe calcularea unor distanţe parţiale sau cel mai apropriat prototip.

Datele medicale conţin frecvent atribute categoriale, exprimând nivele (grade) de prezenţă a unor proprietăţi (ex. simptom). Codificarea clasică în acest caz este: 0-dacă simptomele nu sunt prezente şi 1,2,3 ... pentru nivelele de prezenţă ale simptomului. Uneori, o secvenţă de valori consecutive nu este cea mai potrivită codare deoarece diferenţa dintre 0 (lipsa simptomului) şi primul nivel (cel mai mic nivel al prezenţei simptomului) ar trebui să fie mai mare decât diferenţa dintre restul de nivele. Acest lucru se întâmplă deoarece aceste nivele codează nişte informaţii subiective care nu sunt reflectate corect de valorile care le codează.

Am studiat aceste probleme pentru o bază de date de test - Dermatology - de la UCI Machine Learning Repository. Am studiat influenţa pasului de învăţare a măsurii de similaritate pentru diferite formule de calcul şi adecvarea unei codări sau măsuri de similaritate.

5.5 Concluzii

În acest capitol am analizat abilitatea unei tehnici de grupare inspirate din natură, `AntClust`, de a se descurca în cazul datelor afectate de zgomot şi a datelor medicale incomplete. Algoritmul `AntClust` a fost inspirat de abilitatea furnicilor de a realiza noi muşuroaie, de acceptare a furnicilor în muşuroaie şi de recunoaştere a veciniilor din colonie. Secţiunea 5.2 conţine descrierea algoritmului şi unele recomandări de setare a parametrilor.

A fost analizată abilitatea algoritmului `AntClust` de a separa datele de zgomot. A fost realizată o analiză a utilităţii parametrilor M^+ dar şi a parametrului de densitate D introdus. Calculul acestui nou parametru şi faza de post procesare introdusă nu au modificat semnificativ timpul de execuţie al algoritmului. Cum parametrul σ poate fi calculat pe baza pragului de similaritate, nu este nevoie de setarea unui nou parametru. Rezultatele preliminare sunt încurajatoare, dar multe lucruri legate de informaţiile stocate de furnică nu sunt complet clarificate. Valorile folosite pentru setarea pragurilor T_M şi T_D au fost determinate pe baza unor experimente, nu pe paza unui rezultat analitic. De asemenea, se doresc rezultate referitoare la estimarea parametrilor M, M^+ şi D.

De asemenea, am analizat abilitatea algoritmului `AntClust` de a identifica clustere în datele medicale. Importanţa codificării datelor categoriale a fost de asemenea analizată şi comparată cu algoritmul K-Means. În acest caz, am observat că dacă parametrii şi codificarea este aleasă cu grijă algoritmul `AntClust` se aplică cu succes în gruparea de date medicale. Pe de altă parte, când sunt implicate atribute care conţin valori clasificate pe nivele, o codificare adecvată a valorilor îmbunătăţeşte rezultatele grupării.

Capitolul 6

Abordări bazate pe algoritmi evolutivi pentru extragerea de reguli

Procesul de extragre de reguli (data mining) este unul din paşii cheie al KKD, care presupune aplicarea câtorva metode de procesare pentru a facilita aplicarea algoritmilor de data mining pentru a îmbunătăţi şi rafina procesul de descoperire a cunoştinţelor.

6.1 Extragerea de reguli din date

O regulă poate fi prezentată în următoarea formă: *IF "câteva condiţii aplicate atributelor prezise sunt adevărate" THEN "unele condiţii aplicate atributelor ţintă sunt adevărate"*. Dacă există doar un atribut ţintă şi acesta specifică o clasă, atunci problema descrisă este una de *clasificare*, adică data care satisface condiţia antecedent (IF) aparţine unei clase specificate de consecvent (THEN). Când atributele ţintă nu exprimă o clasă, vom avea de a face cu reguli de *predicţie*, exprimând ipoteze despre dependinţele dintre antecedentul şi consecventul regulii. Dacă setul de atribute consecvent şi antencedent nu sunt predefinite, vorbim despre reguli de *asociere* care exprimă asocierea diferitelor atribute. Descoperirea şi selectarea de reguli este un proces de căutare ghidat de măsuri de calitate: *acurateţea*, *sensul* şi *înteresul* regulii. Aceste măsuri de multe ori intră în conflict, de exemplu regulile care au o acurateţe mare nu sunt neaparat interesante şi uşor de citit, din acest motiv procesul de căutare trebuie să fie multicriterial. Pe de altă parte, datele care se regăsesc în practică (de ex. datele medicale studiate în [LZZ08]) conţin un număr mare de valori lipsă (MVs), care implică diferite tehnici de corecţie pentru a putea fi evaluate.

6.1.1 Măsuri de evaluare

Calitatea regulilor de clasificare, predicţie sau asociere poate fi contorizată utilizând diferite măsuri statistice, fiecare dintre ele specificând anumite caracteristici ale regulilor. Regulile extrase din date ar trebui să caracterizeze datele, să fie inteligibile şi să conţină informaţii noi şi interesante, astfel măsurile de calitate se pot împărţi în următoarele clase:

- măsuri de acurateţe - cuantifică gradul în care regulile exprimă datele: suport (*Supp*), încredere (*Conf*), acurateţe (*Acc*), specificitate (*Spec*) şi sensitivitate (*Sens*);
- măsuri de înţelegere - cuantifică gradul în care regula este uşor înţeleasă;
- măsuri de interes - cuantifică gradul în care regula conţine noi informaţii sau informaţii deja ştiute (coeficientul Phi (Φ), procentul de neînţelegere (*OR*) şi măsura cosinus (*cos*)).

Regulile au următoarea structură: *IF* (AT_1,AT_2, ..., AT_k) *THEN* (CT_1,CT_2, ..., CT_l), unde AT_i este antecedentul regulii iar CT_j este consecventul regulii. Fiecare termen conţine cel puţin un atribut şi este un triplet $\langle a, op, valoare/interval\rangle$ unde a este un atribut, op este un operator (egal, diferit, în, not în, mai mic, mai mare) şi $valoare$ este o posibilă valoare sau set de valori ale atributului.

6.1.2 Valori lipsă în date

Când evaluăm o regulă pentru un set incomplet de date, trebuie să decidem cum vom trata valorile lipsă. Cea mai radicală abordare de a trata valorile lipsă (MV) este de a elimina observaţiile care conţin valori lipsă (de ex. datele incomplete) din analiză. Variantele pe care le-am analizat în [LZZ08] se bazează pe ideea de a trata datele incomplete în timpul procesului de extragere de cunoştinţe. În acest caz, valorile lipsă intră în discuţie în momentul când verificăm dacă o înregistrare se potriveşte cu o anumită regulă.

În cazul datelor incomplete, când o regulă R este evaluată, valorile lipsă pot interfera cu procesul de evaluare numai dacă atribute sunt implicate în regulă. Va trebui să decidem cum trebuie modificate calculele în acest caz. Am analizat două metode.

Metoda 1. În acestă variantă, am încercat să limităm penalizarea regulii care implică atribute incomplete.

Metoda 2. În acestă variantă, nu am ignorat datele incomplete ci le-am penalizat utilizând valori non-crisp. În loc de valoare 0 de potrivire pentru atributele lipsă, fiecărei valori atribut lipsă a atributului i s-a atribuit o probabilitate de a satisface termenul regulii.

Aceasta este o variantă de estimare a probabilităţii care arată că o valoare lipsă va satisface termenul regulii, bazată pe presupunerea că valorile atributelor sunt distribuite uniform în domeniul lor de definiţie. Prin utilizarea altor modele de distribuţie pentru valorile atributelor, s-ar putea obţine diverse valori de potrivire. Diferenţa dintre prima variantă şi cea curentă constă în faptul că în primul caz $cover(R, S) \in \{1, \ldots, cardS\}$, în timp ce în al doilea caz $cover(R, S) \in [1, cardS]$.

6.2 Un algoritm evolutiv pentru extragerea de reguli din date

În [ZL**Z**08] ne-am propus să implicăm utilizatorul în procesul de căutare, deoarece găsirea unei combinaţii de măsuri de calitate predefinite este dificilă; de asemenea utilizatorii se pot răzgândi asupra calităţii regulii în timpul procesului de optimizare [OAT+99]. Abordarea propusă este bazată pe un algoritm evolutiv multi-obiectiv (MOEA).

Am aplicat tehnica de extragere de reguli pentru datele de test oferite de repositoriul UCI şi pentru un set de date obstetrice colectate timp de un an de la un spital de Obstetrică-Ginecologie. Deoarece scopul studiului nostru a fost de a ajuta medicii specialişti în a lua decizii, trebuie să luăm în considerare feedbackul oferit de specialişti şi să le permitem să intervină în procesul de extragere de reguli.

Codarea. Fiecare element (cromozom) al populaţiei corespunde la o regulă şi este format dintr-o listă de componente (gene) care corespund la toate atributele din setul de date. Fiecare componentă este constituită din trei câmpuri: $\langle$ *flag de prezenţă*, *operator*, *valoare* $\rangle$. Un element este o listă de valori mixte (binare, întregi, reale şi interval). Diferenţa dintre atributele antecedent şi consecvent este realizată doar în pasul de evaluare a elementului.

Operatori evolutivi. În fiecare generaţie, o nouă populaţie este creată din cea curentă folosind câţiva operatori evolutivi.

Prin *încrucişare*, o nouă regulă este construită pornind de la două reguli selectate aleator din populaţia curentă.

Mutaţia deţine rolul de a modifica regulile obţinute prin încrucişare. Pentru fiecare atribut, operatorul de mutaţie este aplicat cu o probabilitate (de ex. $p_m = 1/n$) şi poate afecta doar un câmp (de ex. flag-ul de prezenţă, operatorul sau valoarea) şi doar unul la fiecare pas de mutaţie.

Selecţia şi arhiva. După ce o nouă populaţie este creată prin intermediul mutaţiei şi încrucişării, este aplicat un pas de selecţie (tipic pentru MOEA). Strategia de selecţie este similară cu cea utilizată de algoritmul NSGA-II [DK07], ceea ce presupune că elementele din populaţia reunită (părinţii şi copiii) sunt aranjate în funcţie de o relaţie de nedominanţă. Pentru a stimula diversitatea în frontul Pareto rezultat, o distanţă de crowding este utilizată ca un criteriu secund de selecţie: pentru două elemente care au acelaşi rang, este selectat cel cu o distanţă de crowding mai mare (sugerând că aparţine la zone mai puţin dense). Distanţa de crowding poate fi definită în spaţiul variabilelor de decizie sau obiectiv. O caracteristică particulară a abordării noastre este legată de distanţa de crowding dintre reguli.

Am analizat două tipuri de distanţe, una care exprimă diferenţa structurală dintre reguli şi cealaltă exprimă diferenţa dintre dată şi subsetul de date acoperit de regulă.

După un anumit număr de generaţii, se creează o arhivă care conţine elemente nedominate. Nu toate elementele nedominate din populaţia curentă sunt transferate în arhivă, ci ele sunt filtrate astfel încât distanţele structurale şi bazate pe cover dintre două elemente ale arhivei să fie mai mari decât un prag (în analiza noastră am folosit 0.01).

6.3 Interacţiunea utilizatorului cu procesul de extragere de informaţii

O căutare iterativă permite utilizatorului să interfereze cu procesul evolutiv pentru a-l ghida spre zone interesante din spaţiul de căutare. Ideea de a permite utilizatorului să interfereze cu procesul evolutiv a mai fost explorată în contextul optimizării multi-obiectiv evolutive [DK07] prin faptul că cere utilizatorului să furnizeze un aşa numit punct de referinţă în spaţiul obiectiv.

În varianta interactivă, procesul de căutare este alcătuit din câţiva paşi; la fiecare pas, populaţia evoluează un număr de generaţii şi o arhivă cu regulile nedominate este oferită utilizatorului împreună cu toate măsurile obiectiv utilizate pentru setul testat (măsuri nu neaparat limitate la cele utilizate ca criterii în procesul de optimizare). În implementarea noastră am folosit următoarele măsuri: suportul, încrederea, acurateţea, specificitate, sensibilitate,

înţelegere, gradul de înţelegere, lift, uncovered negative şi riscul relativ [OAT+99]. Pe baza acestor criterii şi a evaluării subiective, utilizatorul poate decide dacă o regulă este neinteresantă sau neînţeleasă. Apoi utilizatorul poate marca aceste reguli şi se poate trece la noul pas al căutării. Efectul marcării regulilor ca nenecesare este următorul: în primul rând, elementele populaţiei marcate sunt înlocuite cu reguli generate aleator. Acest lucru permite algoritmului să exploreze noi regiuni ale spaţiului de căutare, asemănătoare cu tehnica imigranţilor aleatori; în al doilea rând, regulile marcate sunt adăugate unei liste de reguli interzise (L_p), creându-se o arhivă care conţine părţi ale spaţiului de căutare care sunt considerate a fi neinteresante de către utilizator.

Un posibil efect secundar al utilizării unui criteriu suplimentar de optimizare este că de obicei duce la un număr mare de elemente nedominate. Pentru a evita acest efect, lista de reguli neinteresante poate fi utilizată pentru a introduce unele constrângeri, de ex. toate regulile similare cu cele din listă sunt considerate nefezabile. Fezabilitatea regulii intervine când se verifică relaţia de dominanţă dintre două reguli: dacă o regulă este fezabilă şi cealaltă nu este, prima regulă o domină pe cea de a doua, ignorând valorile criteriului de calitate; dacă ambele sunt fezabile sau nefezabile, dominanţa este decisă pe baza criteriului de calitate.

6.4 O abordare evolutivă de reducere a regulilor extrase din date

Algoritmul NNGE (Non-Nested Generalized Exemplars) este un algoritm hybrid de învăţare bazat pe instanţe care deduce din date reguli de clasificare reprezentate ca hiper-dreptunghiuri neimbricate şi nesuprapuse. După ce un set de hiperdereptunghiuri $\mathcal{H}$ se generează folosind algoritmul NNGE, acesta este postprocesat utilizând un algoritm evolutiv pentru a încerca reducerea dimensiunii şi a acurateţi clasificării [ZPNZ11].

6.5 Experimente numerice

Experimentele numerice sunt realizate pe două tipuri de date: date medicale şi date despre caracteristicile taskurilor care urmează a fi planificate. În cazul datelor medicale, este analizat un algoritm evolutiv multi-obiectiv. Pentru datele de test de planificare, am analizat comportamentul unui algoritm evolutiv de postprocesare utilizat în selecţia regulilor generate cu algoritmul NNGE.

6.5.1 Setul de date medicale

Rezultatele numerice conţin rezultate obţinute pentru date medicale de test de la repository-ul UCI şi un set de date colectat de la un spital de Obsterică Ginecologie pe parcursul unui an. În [ZLZ08] scopul experimentelor a fost de a valida abilitatea abordării evolutive de a descoperi reguli precise, şi de a analizea impactul intervenţiei utilizatorului în procesul de căutare. Pe de altă parte, în [LZZ08] am analizat influenţa câtorva metode de tratare a datelor incomplete în procesul de găsire a regulilor de clasificare din date medicale.

Date de test

Setul de date *ObGyn* este format din date colectate timp de un an (2006) într-un spital de Obstetrică-Ginecologie. Setul de date conţine 2686 de înregistrări care sunt împărţite în două clase: naşteri înainte de termen (370 înregistrări, reprezentând 13.77%) şi naşteri la termen (2316 înregistrări, reprezentând 86.23%). Fiecare înregistrare conţine 63 de atribute corespunzătoare diferitelor caracteristici ale mamei şi copilului. Procentul de valori lipsă din date este de 23% şi sunt distribuite neuniform printre atribute.

Extragerea de reguli de asociere

Pentru a valida abilitatea algoritmului multi-obiectiv propus de a extrage reguli valoroase din date, l-am testat prima dată în cazul problemelor de clasificare. Rezultatele au fost obţinute pentru seturile de date *Pima Indians Diabetes* şi *Wisconsin Breast Cancer* (1991), bazate pe două criterii de optimizare; acurateţea (*Acc*) şi uncovered negative (*UN*). Rezultatele MOEA sunt comparabile cu cele obţinute prin aplicarea altor clasificatori implementaţi în WEKA: clasificatori simpli (ZeroR, OneR), clasificatori de reguli conjuctive (CR), clasificatori bazaţi pe tabele de decizie (DT), clasificatori bazaţi pe învăţarea incrementală regulilor (JRIP), clasificatori bazaţi pe cel mai apropiat vecin îmbinat cu o tehnică de neimbricare a prototipurilor (NNge), arbori de decizie parţială (PART).

Impactul interacţiunii cu utilizatorul

Două variante de includere a evaluării utilizatorului în procesul de căutare au fost implementate şi testate: criterii utilizator şi constrângeri utilizator. *Criteriile utilizator*: pentru fiecare element, un criteriu suplimentar de optimizare este adăugat între distanţele minimal structural şi

cover-based pentru toate elementele din lista marcată. *Constrângerile utilizatorului*: un element care este structural sau semantic similar cu cel puţin o regulă marcată este considerat nefezabil. Fezabilitatea regulii intervine când se verifică relaţia de dominanţă dintre două reguli: o regulă nefezabilă este întodeauna domină o regulă fezabilă; dacă ambele sunt fezabile sau nefezabile, dominanţa este decisă pe baza criteriului de calitate.

Impactul acestor două variante asupra numărului de reguli nedominate este ilustrat în tabelul Table 6.1, pentru setul de date *Wisconsin Breast Cancer* (fiecare pas execută 100 de generaţii). Intervenţia utilizatorului constă în marcarea calităţii tuturor regulilor cu valoari de calitate mai mici decât un prag (de ex. încrederea, sensibilitatea, specificitatea sau acurateţea mai mici decât 0.75). Rezultatele din Tabelul 6.1 demonstrează că intervenţia utilizatorului a condus la găsirea unor reguli mai sensibile şi precise. După cum ne-am aşteptat, varianta II *criteriile utilizatorului* au permis unui număr mai mare de reguli să evolueze în comparaţie cu varianta III (*constrângerile utilizatorului*).

Var/ Pas	Nr. reguli	Coef.Marcare reguli	Conf. interval	Sens. interval	Spec. interval	Acc. interval	Lift interval
I/1	16	–	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
I/2	6	–	[0.23, 1]	[0.25, 0.55]	[0.88, 1]	[0.81, 0.93]	[2.88, 12.4]
I/3	4	–	[0.97, 1]	[0.28, 0.53]	[0.99, 1]	[0.94, 0.96]	[12.1, 12.4]
II/1	16	10	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
II/2	36	16	[0.75, 1]	[0.07, 0.98]	[0.97, 1]	[0.92, 0.98]	[10.06, 12.4]
II/3	32	19	[0.21, 1]	[0.21, 0.99]	[0.88, 1]	[0.84, 0.98]	[2.6, 12.4]
III/1	16	10	[0.42, 1]	[0.25, 0.98]	[0.31, 1]	[0.54, 0.93]	[1.24, 2.9]
III/2	15	6	[0.23, 1]	[0.28, 0.98]	[0.83, 1]	[0.81, 0.99]	[2.8, 12.4]
III/3	11	2	[0.75, 1]	[0.27, 0.98]	[0.97, 1]	[0.96, 0.99]	[9.3, 12.4]

Tabelul 6.1: Breast data set: intervalele măsurilor de calitate ale regulilor găsite pentru trei scenarii ale algoritmului MOEA. Varianta I: fără interaţiunea utilizatorului. Varianta II (*criterii utilizator*) şi III (*constrângeri utilizator*): la fiecare pas, utilizatorul marchează regulile care au măsurile de confidenţă, sensibilitate, specificitate, sau acurateţe mai mici de 0.75.

De asemenea am folosit variantele interactive de MOEA pentru a explora spaţiul de căutare în cazul datelor obstretice (setul de date *ObGyn*), scopul experimentului fiind găsirea potenţialilor factori de risc pentru naşteri premature (de ex. naşterea înainte de 37 de săptămâni gestaţionale). Modelele de risc pot fi exprimate ca reguli de clasificate (IF "condiţiile antecedent sunt sat-

isfăcute" THEN clasa=prematur) sau ca reguli de predicţie (IF "condiţiile antecedent sunt satisfăcute" THEN "vârsta gestaţiei este mai mică decât o valoare").

Strategia propusă de implicare a utilizatorului în procesul de căutare este doar un prim pas în dezvoltarea unui sistem interactiv a cărui scop este de a ajuta medicii să exploreze datele şi să extragă noi, posibil neaşteptate, cunoştinţe. Rezultatele obţinute în cazul datelor obstretice nu au fost suficient de relevante din diferite motive: pe de o parte, particularităţiile datelor (multe valori lipsă şi eronate); pe de altă parte, limitările strategiei evolutive. Folosirea de valori numerice pentru valorile continue a fost utilizată pentru a evita o discretizare preliminară, dar acest lucru a condus la un spaţiu de căutare foarte mare şi descoperirea de reguli care nu sunt uşor de interpretat. Folosirea de variabile fuzzy în loc de variabile crisp ar putea îmbunătăţi calitatea regulilor, mai ales în cazul datelor medicale.

Influenţa tratării valorilor lipsă din date pentru regulile de clasificare

Deoarece datele de test de la UCI nu conţin valori lipsă, am pregătit datele pentru teste prin eliminarea aleatoare a $mv\%$ din valorile atributelor ($mv \in \{10, 20, 30\}$ şi reprezintă un procent din numărul total al atributelor din setul de date). Pentru a realiza validarea încrucişată, datele au fost împărţite în patru exemplare. Valorile lipsă au fost introduse doar în datele utilizate pentru antrenare, setul de validare conţinând valorile originale ale atributelor.

Pentru a analiza influenţa valorilor lipsă din date am utilizat trei metode: *Metoda 1* şi *Metoda 2* sunt descrise în seţiunea 6.1.2 şi *Metoda 3* se bazează pe o simplă strategie de imputare (datele sunt pre-procesate astfel încât valorile lipsă sunt înlocuite cu mediana valorilor existente pentru atributele corespunzătoare). Prima metodă este varianta care încearcă să limiteze penalizarea regulilor şi a doua variantă foloseşte valori non-crip pentru potrivire.

Experimentele au ilustrat că prin utilizarea de metode diferite de tratare a valorilor lipsă din date s-au obţinut diferite seturi de reguli. Pe de altă parte, diferenţa statistică dintre calitatea regulilor obţinute cu diferite metode nu este semnificativă. Rezultate puţin mai bune s-au obţinut pentru metodele care ajustează calculul valorilor de calitate pentru a trata valorile lipsă din date (Metoda 1 şi Metoda 2).

De asemenea am realizat o analiză şi asupra datelor obstretice (obGyn). Scopul a fost găsirea de reguli care corespund naşterilor premature. Numărul mare de atribute duce la utilizarea unui spaţiu mare de căutare, creeînd dificultăţi algoritmului evolutiv în căutarea de reguli de caliate ridicată. Din această cauză, am ales diferite seturi de atribute care sunt implicate în reguli. Rezultatele prezentate în Tablul 6.2 au fost obţinute când am folosit în antecedentul regulii de

clasificare informaţia despre sarcinile anterioare, avorturi provocate, avorturi naturale şi despre înălţimea fătului. Am utilizat produsul dintre specificitate şi sensibilitate ca criteriu unic de optimizare. Din cauză că am folosit un singur criteriu de optimizat la fiecare rulare, a rezultat o singură regulă. Toate regulile obţinite au conţinut termeni care corespund la înălţimea "fundus uterus"(de ex. înălţimea "fundus uterus" $\leqslant 29$) chiar dacă acest atribut conţine valori lipsă în cazul unor date. Acest lucru poate fi explicat prin faptul că atributul care conţine înalţimea "fundus uterus" apare de obicei în cazul naşterilor premature. Pe de altă parte, după cum sugerează Tabelul 6.2, rezultatele obţinute de cele trei metode nu diferă semnificativ. Cu toate acestea, prima metodă oferă rezultate puţin mai bune decât celelalte două.

	acc	spec	sens	UN	lift
Method 1	0.70±0.09	0.73±0.13	0.52±0.17	0.58±0.12	1.92±0.63
Method 2	0.64±0.08	0.64±0.12	0.59±0.18	0.55±0.1	1.57±0.34
Method 3	0.64±0.05	0.66±0.07	0.54±0.11	0.57±0.06	1.54±0.29

Tabelul 6.2: Rezultate pentru setul de date obstretice

6.5.2 Setul de date utilizat pentru planificarea taskurilor

O alternativă la algoritmii de interschimbare este *selecţia utilizând forţa brută* (BS). În acest caz, fiecare strategie de planificare (SA) este testată pe setul de date, ceea ce este o abordare care consumă timp. O alternativă ar fi aplicarea stategiei BS doar pe un set de date de antrenare. Setul de date de antrenare conţine câteva caracteristici ale platformei pentru scenariul de planificare (taskurile şi resursele relaţionate) împreună cu cea mai bună SA (eticheta clasei) pentru o configurare specifică, găsită de BS. Acest set de date poate fi folosit pentru a antrena un sistem de clasificare. Apoi, când apare o nouă configurare, clasificatorul generat în pasul anterior este utilizat pentru a găseasi SA corespunzător. În [ZF11a, ZF11b] am testat câteva strategii de clasificare pentru a găsi strategia care asigură cea mai bună acurateţe de clasificare şi identificarea caracteristicilor evenimentelor de planificare care influenţează cel mai mult alegerea algoritmului de planificare.

Date de test

Setul de date a fost generat artificial utilizând modelele descrise în [Fei10]. Fiecare instanţă a setului de antrenare este formată din următoarele atribute: *timpul când planificarea se termină*,

media timpului estimat de execuţie (ETT); *media deviaţiilor standard a EET*; *media timpului estimat de terminare (ECT)*; *media deviaţiei standard a ECT*; *media dimensiunii taskurilor*; *media deviaţiilor standard a mărimii taskurilor*; *numărul total de taskuri*; şi *numărul de taskuri lungi* utilizate în experiment. Pe lângă informaţia pentru fiecare configuraţie, cea mai bună strategie de planificare găsită de BS a fost adăugată petru a identifica clasa. Şapte SA au fost utilizate pentru a determina cea mai bună strategie: Max-Min [MAS+99], Min-Min [MAS+99], Sufferage [CLZB00], MinQL [FMC09], MinQL-Plain [FMC09], DMECT [Fr9] şi DMECT2 [Fr9].

Clasificatorii utilizaţi în teste. În experimentele noastre am utilizat câţiva clasificatori implementaţi în WEKA data mining toolkit (o reţea neuronală multi-perceptron (MPL), Radial Basis Function (RBF) network, Non-Nested Generalized Exemplars (NNGE)), un clasificator ne-supervizat Fuzzy C-Mean şi un clasificator hibrid care combină algoritmul NNGE cu un algoritm evolutiv de selecţie a atributelor şi exemplarelor (EP-NNGE descris în Secţiunea 6.4).

Rezultatele testării. O primă analiză a fost realizată pentru a determina timpul necesar strategiilor de planificare pentru selecţia celui mai potrivit planificator. Timpul de execuţie pentru euristicele de clasificare a fost sub 2.5s (antrenare+clasificare), în timp ce în cazul strategiei BS pentru 7 SA timpul mediu a fost de aproximativ 6 secunde pentru fiecare eveniment de planificare. Acurateţea mare a clasificării şi timpul scurt de învăţare fac din tehnicile de clasificare un candidat potrivit pentru a înlocui politica de interschimabare a planificatoarelor şi strategia BS.

O a doua analiză a fost realizată pentru a determina cea mai bună tehnică de clasificare. Comportamentul clasificatorilor propuşi EP-NNGE şi EPA-NNGE este similar pentru cele trei seturi de date şi mai bun decât restul de euristicilor de clasificare.

6.6 Concluzii

În acest capitol am prezentat un algoritm evolutiv multi-obiectiv pentru extragerea de reguli din datele medicale şi o strategie evolutivă de selecţie pentru determinarea de reguli de clasificare care asigură o bună alegerea a unei strategii de planificare pentru un eveniment de planificare.

Capitolul 7

Concluzii şi direcţii viitoare

Această teză se bazează pe aplicarea euristicilor inspirate din natură în medii incerte; sunt discutate trei probleme de optimizare concretă (planificare pe grid, gruparea datelor, extragerea de reguli din date) care pot fi formulate ca probleme de optimizare în medii dinamice şi incerte şi câteva funcţii de test.

În cazul problemelor de optimizare dinamică am analizat doi algoritmi: Particle Swarm Optimization şi Differential Evolution. Pentru a îmbunătăţii comportamentul euristicilor în medii dinamice am introdus un mecanism de perturbare a poziţiei particulei în cazul algoritmului PSO (Secţiunea 3.3.2) şi elemente aleatoare care oscilează în jurul optimului în cazul DE (Secţiunea 3.3.2).

În cazul problemei de planificare pe grid am propus un planificator simplu bazat pe populaţii si am analizat robusteţea lui (Secţiunea 4.3), în cazul planificatoarelor batch comportamentul într-un mediu static (Secţiunea 4.2.3) şi dinamic (Secţiunea 4.5.3), în cazul planificări online (Secţiunea 4.4). Deasemenea am studiat influenţa a diferite strategii de iniţializare (Secţiunea 4.2.1) şi perturbare (Secţiunea 4.2.3) a planificatorului bazat pe populaţii.

În cazul problemei de grupare de date am adăugat un parametru de densitate (Secţiunea 5.3.1) algoritmului `AntClust` pentru a-i îmbunătăţi performanţele în cazul grupării datelor afectate de zgomot şi am analizat performanţele euristicii în cazul datelor medicale care conţin atât atribute numerice cât şi categoriale afectate de valori lipsă (Secţiunea 5.4).

În cazul problemei de extragere de reguli din date, am propus introducerea expertului uman (Secţiunea 6.3)pe lângă evaluarea automată a regulilor algoritmului evolutiv. O particularitate a acestei abordări constă în adăugarea unei distanţe de crowding între reguli, care se comportă ca un criteriu de diversitate, iar mulţimea Pareto este stimulată să aleagă elemente din zone mai puţin dense în arhivă (Secţiunea 6.2). De asemenea, am propus metode de tratare a valorilor

lipsă de date şi am analizat influenţa acestora în procesul de extragere de reguli (Secţiunea 6.5.1).

O altă problemă abordată în cazul extragerii de reguli din date se referă la clasificarea aplicaţiilor care sunt trimise planificatoarelor astfel încât, atunci când apare un nou eveniment de planificare selectarea algoritmului potrivit să se facă pe baza caracteristicilor taskurilor. Contribuţia în acest caz este legată de introducerea planificatorului hibrid bazat pe exemplare neimbricate şi aplicarea unei selecţii evolutive asupra atributelor şi exemplarelor (Secţiunea 6.4).

Câteva direcţii viitoare de cercetare sunt prezentate la sfârşitul fiecărui capitol. În cazul problemei de planificare pe grid, studiile viitoare vor trata cazul taskurilor relaţionate sau folosirea altor metrici ca Total Processing Consumption Cycle care este o alternativă la makespan şi este independentă de hardware. În cazul problemei de grupare a datelor, se doreşte o analiză a comportamentului alogritmului pentru alte date dar şi pentru datele reale. În cazul problemei de descoperire a regulilor din date, se poate realiza un studiu mai amănunţit a diferitelor metode de tratare a valorilor lipsă din date şi influenţa lor asupra designului algoritmului evolutiv. Deoarece datele utilizate pentru testare în cazul algoritmilor de planificare sunt nebalansate, se doreşte realizarea unui studiu al diferitelor metode de balansare asupra tehnici de căutare propuse.

Deoarece problemele de optimizare în medii incerte şi dinamice se regăsesc printre problemele reale iar euristicele inspirate din natură sunt candidate bune pentru a rezolva aceste probleme, studiile viitoare se vor axa pe analiza unor algoritmi din natură având ca scop rezolvarea unor probleme de optimizare în medii incerte şi dinamice.

Bibliografie

[BB02] T. M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 19–26, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[Bra] J. Branke. Moving peaks benchmark. *www.aifb.uni-karlsruhe.de/jbr/MovPeaks/movpeaks/*.

[BSB+01] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Lasislau L. Bölöni, Muthucumara Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, Bin Yao, Debra Hensgen, and Richard F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput.*, 61:810–837, June 2001.

[CJSZ08] Louis-Claude Canon, Emmanuel Jeannot, Rizos Sakellariou, and Wei Zheng. Comparative evaluation of the robustness of dag scheduling heuristics. In Sergei Gorlatch, Paraskevi Fragopoulou, and Thierry Priol, editors, *Grid Computing*, pages 73–84. Springer US, 2008. 10.1007/978-0-387-09457-1-7.

[CLZB00] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *Procs. 9th Heterogeneous Computing Workshop (HCW)*, pages 349–363, Cancun, Mexico, May 2000.

[CX06] J. Carretero and F. Xhafa. Using genetic algorithms for scheduling jobs in large scale grid applications. *Journal of Technological and Economic Development - A Research Journal of Vilnius Gediminas Technical University*, 12:11–17, 2006.

[DK07] Kalyanmoy Deb and Abhishek Kumar. Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 781–788, New York, NY, USA, 2007. ACM.

[Fei10] D.G. Feitelson. Workload modeling for computer systems performance evaluation.

URL http://www.cs.huji.ac.il/ feit/wlmod/, 2010.

[FMC09] M. Frîncu, G Macariu, and A. Cârstea. Dynamic and adaptive workflow execution platform for symbolic computations. *Pollack Periodica*, 4(1):145–156, 2009.

[Fr9] Marc E. Frîncu. Dynamic scheduling algorithm for heterogeneous environments with regular task input from multiple requests. In *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, GPC '09, pages 199–210, Berlin, Heidelberg, 2009. Springer-Verlag.

[JB05] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *Evolutionary Computation, IEEE Transactions*, 9(3):303–317, 2005.

[KE95] J. Kennedy and R.C. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, pages 1942– 1948, 1995.

[LA05] Hongbo Liu and Ajith Abraham. Fuzzy adaptive turbulent particle swarm optimization. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pages 445–450, Washington, DC, USA, 2005. IEEE Computer Society.

[LMV02] N. Labroche, N. Monmarche, and G. Venturini. A new clustering algorithm based on the chemical recognition system of ants. *Harmelen, F. van (Ed.) Proc. of the 15th European Conference on Artificial Intelligence, Lyon, France*, pages 345–349, 2002.

[LZ**Z**08] Diana Lungeanu, Daniela Zaharie, and **Flavia Zamfirache**. Influence of missing values handling on classification rules evolved from medical data. In *Industrial Conference on Data Mining - Posters and Workshops'08*, pages 86–95, 2008.

[MAS+99] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59:107–131, 1999.

[MM05] R. Mendes and A. Mohais. Dynde: a differential evolution for dynamic optimization problems. *Proc. of CEC2005. Edinburgh, Scotland, September 2-5*, pages 2808–2815, 2005.

[OAT+99] M. Ohsaki, H. Abe, S. Tsumoto, H. Yokoi, and T. Yamaguchi. Discovering interesting prediction rules with a genetic algorithm. *Proc. of of Congress on Evolutionary Computing (CEC 1999)*, pages 1322–1329, 1999.

[PKN10] Andrew J. Page, Thomas M. Keane, and Thomas J. Naughton. Multi-heuristic

dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *J. Parallel Distrib. Comput.*, 70:758–766, July 2010.

[RL04] G. Ritchie and J. Levine. A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. *Proceedings the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*, pages 1–7, 2004.

[SP95] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012, ICSI, March*, 1995.

[XA10] Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Gener. Comput. Syst.*, 26:608–621, April 2010.

[Xha07] F. Xhafa. A hybrid evolutionary heuristic for job scheduling on computational grids. In Ajith Abraham, Crina Grosan, and Hisao Ishibuchi, editors, *Hybrid Evolutionary Algorithms*, volume 75 of *Studies in Computational Intelligence*, pages 269–311. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-73297-6-11.

[**Z**F11a] **Flavia Zamfirache** and Marc Frîncu. Automatic selection of scheduling algorithms based on classification models. *STUDIA*, (2), 2011.

[**Z**F11b] **Flavia Zamfirache** and Marc Frîncu. Automatic selection of scheduling algorithms based on classification models. *Proc. of International Conference on Knowledge Engineering: Principles and Techniques, Cluj-Napoca*, 2011.

[**Z**FZ11] **Flavia Zamfirache**, Marc Frîncu, and Daniela Zaharie. Population-based metaheuristics for tasks scheduling in heterogeneous distributed systems. In *Proceedings of the 7th international conference on Numerical methods and applications*, NMA'10, pages 321–328, Berlin, Heidelberg, 2011. Springer-Verlag.

[ZL**Z**08] Daniela Zaharie, Diana Lungeanu, and **Flavia Zamfirache**. Interactive search of rules in medical data using multiobjective evolutionary algorithms. In *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, GECCO '08, pages 2065–2072, New York, NY, USA, 2008. ACM.

[ZPNZ11] D. Zaharie, L. Perian, V. Negru, and F. Zamfirache. Evolutionary pruning of non-nested generalized exemplars. *Proc. of 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011) to be held on May 19-21, 2011 in Timisora, Romania*, pages 57–62, 2011.

[Z**Z**05a] Daniela Zaharie and **Flavia Zamfirache**. Ant-based clustering of medical data. *HCMC 2005, First East European Conference on Health Care Modelling and Computation (eds: F. Gorunescu, E. El-Darzi, M. Gorunescu)*, pages 332–343, 2005.

[Z**Z**05b] Daniela Zaharie and **Flavia Zamfirache**. Dealing with noise in ant-based clustering. In *Congress on Evolutionary Computation'05*, pages 2395–2401, 2005.

[Z**Z**06] Daniela Zaharie and **Flavia Zamfirache**. Diversity enhancing mechanisms for evolutionary optimization in static and dynamic environments. *Proc. of 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence*, pages 460–471, 2006.

[**Z**ZC10a] **Flavia Zamfirache**, Daniela Zaharie, and Ciprian Craciun. Evolutionary task scheduling in static and dynamic environments. *Proc. ICCC-CONTI, Timisoara, Romania*, pages 619–624, 2010.

[**Z**ZC10b] **Flavia Zamfirache**, Daniela Zaharie, and Ciprian Craciun. Nature inspired metaheuristics for task scheduling in static and dynamic computing environments. *Scientific Buletin of Politehnica University of Timisoara, Romsnis, BS-UPT TACCS*, 55(69)(3):133–142, 2010.

[Z**Z**N$^+$07] Daniela Zaharie, **Flavia Zamfirache**, Viorel Negru, Daniel Pop, and Horia Popa. A comparison of quality criteria for unsupervised clustering of documents based on differential evolution. *Proc. of International Conference on Knowledge Engineering: Principles and Techniques*, pages 114–121, 2007.