Universitatea „Dunărea de Jos” din Galaţi

Școala doctorală de Inginerie

CONTRIBUȚII LA CONDUCEREA OPTIMALĂ A PROCESELOR UTILIZÂND ALGORITMI METAEURISTICI

Rezumat al tezei de doctorat

de

Adrian Emanoil ȘERBENCU

Conducător științific,

Prof. dr. ing. Viorel MÎNZU

Seria I 8: Ingineria sistemelor Nr. 2

GALAȚI

2017

Universitatea „Dunărea de Jos” din Galaţi

Școala doctorală de Inginerie

CONTRIBUȚII LA CONDUCEREA OPTIMALĂ A PROCESELOR UTILIZÂND ALGORITMI METAEURISTICI

Rezumat al tezei de doctorat

de

Adrian Emanoil Șerbencu

Președinte,	Prof univ.dr.ing. Eugen-Victor-Cristian RUSU
Conducător științific,	Prof univ.dr.ing. Viorel MÎNZU
Referenți stiințifici	Prof univ.dr.ing. Nicolae PARASCHIV
	Prof univ.dr.ing. Aurelian STĂNESCU
	Prof univ.dr.ing. Adrian FILIPESCU

Seria I 8: Ingineria sistemelor Nr. 2

GALAŢI

2017

Cuprins

Abrevieri / Abbreviations

	Română	Engleză
ACO	Optimizare cu colonii de furnici	Ant Colony Optimization
AE	Algoritm(i) evolutiv(i)	Evolutionary Algorithm(s)
AG	Algoritm(i) genetic(i)	Genetic Algorithm(s)
ASA	Algoritmul SA	Simulated Annealing Algorithm
BHTPSO	HTPSO binar	Binary HTPSO
BHTPSOM	BHTPSO Modificat	Modified BHTPSO
BLX-α	Încrucișare prin combinare liniară Alpha	Linear Blend Alpha Crossover
DASA	Algoritm diferențial stigmergic	Differential Ant-Stigmergy Algorithm
EDOA	Ecuații diferențiale ordinare și algebrice	Ordinary and Allegorical Differential Equations
GDD	Coborâre deterministă geografică	Geographical deterministic descent
HTPSO	PSO cu topologie hibridă	Hybrid Topology Particle Swarm Optimization
LQP	Problemă liniar pătratică	Linear Quadratic Problem
PCO	Problemă(e) de conducere optimală	Optimal Control Problem(s)
PPFBP	Problemă de producere fed-batch a proteinelor	Protein Production Fed-Batch Problem
PSO	Optimizare cu roiuri de particule	Particle Swarm Optimization
RBM	Regulator bazat pe metaeuristica	Metaheuristic Based Controller
RCAU	Rețea de colectare a apelor uzate	Sewer Network
SA	Simulated Annealing (călire simulată)	Simulated Annealing
SNDOP	Problemă de descărcare optimală a RCAU	Sewer Network Discharge Optimization Problem

Introducere

Tema prezentei lucrări "Contribuții la conducerea optimală a proceselor utilizând algoritmi metaeuristici" pleacă, pe de o parte, de la constatarea că problemele de control optimal degajate din teoria sistemelor dinamice și impuse de practica inginerească sunt probleme dificile - în sensul complexității calculatorii - iar, pe de altă parte, metodele metaeuristice dezvoltate de optimizările din "computer science" sunt concepute tocmai pentru a "sparge" complexitatea problemelor dificile. Noțiunea de metaeuristică, deși larg utilizată, este relativ greu de definit, așa cum rezultă și din încercările unor cercetători consacrați din domeniu.
Metaeuristicile (metodele metaeuristice) sunt euristici de nivel înalt - așa cum arată numele - de la care se așteaptă să acționeze mai bine decât euristicele în rezolvarea problemelor. Procesul de determinare a soluției unei probleme - de obicei optimale - este ghidat de anumite informații sau cunoștințe înglobate în procesul de căutare a soluției. Termenul de "metaeuristici", inventat de Glover în anii '90, desemnează algoritmi inspirați din natură cum ar fi algoritmii genetici, călirea simulată, optimizarea prin colonii de furnici, optimizare prin roiuri de particule, evoluție diferențială, etc. Din acest punct de vedere, *o metaeuristică ne apare ca "un scenariu" cu "roluri" în care "distribuim" elemente ale problemei de rezolvat, astfel încât se creează o dinamică a acestor elemente, ce duce la rezolvarea "optimala" a problemei respective. Acest "scenariu" este exterior universului de discurs al problemei.*

Există probleme de optimizare continue pentru care nu există un algoritm care să determine optimul global cu certitudine și după un număr finit de calcule. Există multe metode clasice "de optimizare globală", dar de multe ori acestea sunt ineficiente dacă funcția obiectiv nu posedă anumite proprietăți structurale, cum ar fi convexitatea.
Dezvoltarea metaeuristicilor aduce o cale de rezolvare a multor probleme de optimizare dificile discrete sau continue. În lucrarea (Siarry, 2014), sunt evidențiate câteva caracteristici comune ale metaeuristicilor care le recomandă pentru probleme de optimizări dificile:

- Metaeuristicile sunt, în general, *stochastice*, ceea ce le permite să facă față exploziei combinatorice a posibilităților ce apar în căutarea soluțiilor optime;
- Metaeuristicile, având un caracter discret, au avantajul foarte important în cazul continuu de a fi *directe*, adică nu recurg la calculul gradientului funcției obiectiv;
- Metaeuristicile sunt inspirate de *analogii fizice* (călire simulată, difuzie simulată,...), *biologice* (algoritmi evolutivi, genetici,...), s*au etologice* (colonii de furnici, roiuri de particule, ...);
- *Convergența* este, în general, bine studiată din punct de vedere teoretic.

Au și deficiențe comune, cum ar fi necesitatea calibrării anumitor parametri ai algoritmilor generați.

Deși ideea de a utiliza metaeuristici în conducerea optimală a proceselor nu este originală, prezenta lucrare se dorește un studiu al utilizării de metaeuristici pentru rezolvarea de probleme de control optimal (PCO), care să atingă câteva obiective:

- studiul implementării unor algoritmi metaeuristici dedicați rezolvării unor PCO. Au fost alese metaeuristici adecvate acestui scop, bine fondate teoretic și cu o bună convergență;
- în ce măsură soluțiile obținute sunt realiste și asigură măcar un caracter quasioptimal;
- care este complexitatea practică a algoritmului ce implementează metaeuristica, până la obținerea unei soluții bune;

- degajarea unor reguli și metode de calibrare a parametrilor ce caracterizează un astfel de algoritm;
- realizarea unui studiu comparativ al utilizării diferitelor metaeuristici în rezolvarea unei aceleași probleme.

De la bun început, menționăm că algoritmii metaeuristici implementați, dezvoltați și analizați precum și structura de conducere în buclă închisă propusă în prezenta lucrare au ca scop utilizarea lor în situațiile în care PCO nu cunoaște alte modalități de rezolvare convenabile. Cauzele pot fi multiple: anumite neliniarități, lipsa unor proprietăți de regularitate, complexitate calculatorie foarte mare, etc. În cazuistica tratată în lucrare, am recurs la niște exemple de PCO, care au rolul de benchmark pentru noi și care au soluții optimale cunoscute teoretic. Acest lucru ne ajută la analiza convergenței și a calității soluțiilor obținute. Primul capitol este dedicat prezentării metodei probabilistice de optimizare "simulated annealing". Este una dintre primele metaeuristici utilizate la rezolvarea PCO. În plus este singura din lucrare bazată pe *evoluția unei singure soluții*. Celelalte capitole descriu metaeuristici ce utilizează *populații de soluții* ale PCO. Au fost analizate, prin exemple, câteva clase de PCO care pot fi rezolvate cu algoritmul simulated annealing (ASA). Clasa PCO *cu stare finală liberă și timp final fixat* este cea mai simplă clasă care poate fi rezolvată cu ASA. Un prim exemplu tratat a fost problema Batch Reactor, care are un criteriu de tip Mayer. Soluția optimală este cunoscută din alte lucrări și reconfirmată de soluția propusă. În acest exemplu s-a implementat tehnica de "step control" care a crescut calitatea soluției fără creșterea semnificativă a numărului de evaluări ale funcției obiectiv. Al doilea exemplu a tratat un sistem dinamic liniar cu criteriu de optimalitate de tip Bolza pentru care s-a obținut o soluție chiar optimă, utilizând aceeași codificare a soluției. Pentru a ilustra posibilitatea de a rezolva cu ASA o PCO *bilocală în raport cu starea și timp final fixat* s-a considerat un sistem liniar cu două variabile de stare. Aspectul nou tratat în acest exemplu este introducerea de termeni suplimentari în funcția obiectiv, corespunzători componentelor legate din variabila de stare. Stabilirea ponderii acestor termeni este problema cheie pe care proiectantul algoritmului trebuie s-o rezolve. *PCO bilocale în raport cu starea și timp final liber* sunt exemplificate prin două probleme de naturi diferite. Soluțiile obținute sunt mai mult decât satisfăcătoare.

În capitolul 2, după analiza generală a algoritmilor evolutivi și a cazului particular al algoritmilor genetici, din literatura de specialitate rezultă că *semnul distinctiv al AG este prezenta unei codificări de tip genotip, indiferent ca este făcută prin șiruri de valori binare sau reale*. În cazul rezolvării unei PCO, codificarea printr-un șir de valori reale a unui profil de comandă este o opțiune preferabilă prin simplitate. De aceea, nu se degajă necesitate unei codificări de tip genotip. Ca urmare, problema poate fi rezolvată printr-un algoritm evolutiv. Au fost considerate trei PCO. Pentru problema de producere fed-batch a proteinelor, a fost dezvoltat un AE cu următoarele caracteristici: selecție cu Stochastic Universal Sampling, utilizare Rang și Scalare Rang pentru selecție, "replacement" în ultimele lambda poziții din cele N ale populației, încrucișare într-un singur punct cu un singur urmaș, mutație neuniformă pe toate genele. Deși simplă, î*ncrucișarea de acest tip este satisfăcătoare pentru acest tip de problemă* și ca urmare soluțiile și viteza de convergență sunt satisfăcătoare. În contrapartidă, pentru Problema Batch Reactor, acest tip de încrucișare nu duce la rezultate satisfăcătoare. De aceea s-a adoptat un operator crossover mai eficient: *BLX-0.5*. Soluțiile și viteza de convergență sunt, cu acest operator, mai mult decât satisfăcătoare. A treia problemă tratată a fost Problema Liniar Pătratică, care, pentru un anumit joc de parametri, are o convergență surprinzător de slabă, cu varianta de AE anterioară. De aceea, am adoptat o Strategia Evolutivă cu Control Adaptiv Global al Variației, ce a utilizat un operator de mutație cu deviație standard normală controlată adaptiv. Rezultatele au fost

spectaculoase, pentru că numărul de generații și numărul de evaluări ale funcției obiectiv sunt de 15-20 ori mai mici decât la variantele celelalte de AE utilizat.
Ca și contribuție mai importantă a acestui capitol, putem aminti analiza, pe o cazuistică formată din PCO diverse, a diferiților operatori de crossover și mutație adecvați problemelor respective. Concluzia ne arată că fiecare problemă are o sensibilitate diferită la utilizarea unui operator sau altul. Au fost implementați și analizați operatori semnificativi pentru implementarea unui AE.
Capitolul 3 este dedicat algoritmilor PSO (Particle Swarm Optimization). Varianta folosită în lucrare se bazează pe o topologie hibridă - Hybrid Topology Particle Swarm Optimization (HTPSO) -, adică utilizează ambele topologii, globală și locală, în același timp. Pentru aplicarea acestei variante în probleme de conducere optimală, au fost implementate tehnici cunoscute în literatură ca eficiente, cum ar fi: stabilirea adaptivă a coeficienților implicați în actualizarea vitezei și poziției particulelor și limitarea poziției prin reflectare față de frontiera domeniului de căutare. Dacă PCO are și un caracter bilocal cu starea, s-a considerat o funcție obiectiv extinsă. Evaluarea funcției obiectiv se realizează cu ajutorul unui model al procesului considerat. Aplicarea algoritmului HTPSO la PCO continue a fost ilustrată pe trei cazuri: procesul neliniar din reactorul Batch, o PCO cu sistem liniar, cu criteriu Lagrange și bilocală pe stare și Problema Liniar Pătratică discretă.
Pentru aplicarea PSO la o PCO discretă-binară a fost realizată și descrisă pe larg o implementare (BHTPSO) adaptată la cazul binar. A fost propusă o modalitate de a ameliora intensificarea algoritmului BHTPSO. Pe lângă considerarea vecinătății sale "sociale", o particulă caută o soluție mai bună și într-o vecinătate "geometrică". În acest scop, a fost propus un algoritm de intensificare "geometrică", determinist, de coborâre locală numit GDD (geographical deterministic descent). Au fost propuse, de asemenea, trei tehnici de ameliorare a algoritmului GDD, care au vizat: apelul recursiv al GDD, pentru mărirea eficienței; minimizarea fenomenului de deviere ("bias") în căutarea soluțiilor optime; apelarea selectivă, la un anumit număr de iterații fără ameliorare a soluțiilor "local best". A fost realizat un algoritm modificat - BHTPSOM. Testarea acestuia a fost făcută pe problema de descărcare optimală a unei rețele de colectare a apelor uzate (RCAU) de mari dimensiuni. Soluțiile obținute au demonstrat realismul acestei abordări, prin caracterul lor optimal sau quasi-optimal, și durata convenabilă a execuției algoritmului.
Capitolul 4 descrie utilizarea algoritmului "Differential Ant-Stigmergy". Stigmergia este un mecanism consensual de coordonare indirectă între agenți sau acțiuni folosind mediul înconjurător. Algoritmul DASA, inspirat de optimizarea pe baza coloniilor de furnici, utilizează distribuții Cauchy în procesul de căutare și transformă problema de căutare într-un spațiu multidimensional continuu într-o problemă de construire a unui drum într-un graf asociat. O proprietate intrinsecă a modului de funcționare a algoritmului DASA face ca acesta să *exploateze* eficient soluția promițătoare identificată în *etapa exploratorie*. Anumite particularități ale algoritmului, precum modul explicit de selectare a preciziei maxime a pasului discret de căutare, sunt analizate în relație cu caracteristicile PCO. Vectorul parametrilor supuși optimizării este reprezentat de eșantioane ale comenzilor aplicabile procesului. Aplicarea algoritmului DASA la rezolvarea unor PCO continue a fost ilustrată pe trei cazuri: procesul neliniar din reactorul Batch, problema de producere fed-batch a proteinelor și sistemul liniar cu criteriu Lagrange dar bilocal pe stare. Ca și contribuții semnificative ale acestui capitol, menționăm:

- Propunerea unei tehnici de redimensionare a problemei de optimizare rezolvabile de către DASA cu scopul creșterii vitezei de convergență către zonele de bună calitate;
- Propunerea a trei variante elitiste pentru algoritmul DASA, care promovează utilizarea mai eficientă a informației transmise prin feromon. Primele două variante de elitism

propuse sunt echivalente unei încercări de căutare după gradient, în sensul încercării de a reutiliza drumul din graf care a generat ultima ameliorare a soluției. Cea de a treia variantă DASA este elitistă în sens probabilistic.

Problema care s-a pus în capitol 5, de a avea o metoda care să ne permită conducerea procesului în buclă închisă, de îndată ce avem un bun algoritm bazat pe o metaeuristică ce rezolvă PCO, este crucială, pentru că dă o perspectivă clară de utilizare a algoritmilor metaeuristici dezvoltați în capitolele anterioare. A fost propusă o structură de conducere în buclă închisă dezvoltată în jurul a două idei. Prima idee este utilizarea unei structuri de conducere care este o adaptare a conceptului de control predictiv. A doua idee este utilizarea unui algoritm capabil să rezolve aproximativ o PCO printr-o metaeuristică. Legătura dintre cele două idei este că structura predictivă are un regulator care poate utiliza algoritmul metaeuristic. A fost propusă o structură de conducere în buclă închisă plecând de la controlul predictiv al proceselor. Regulatorul din structură este un RBM (regulator bazat pe o metaeuristică). La fiecare perioadă de eșantionare, algoritmul rezolvă PCO pentru *un nou orizont de timp* - ce debutează cu momentul de eșantionare respectiv - și pentru *o nouă stare inițială a procesului*, care este starea sa reală presupusă accesibilă. S-a demonstrat faptul că un RBM lucrează în sensul minimizării erorii de predicție și deci că structura propusă este un caz particular al structurii cu control predictiv. S-a propus de asemenea un mod de organizare al controlul în buclă închisă al proceselor cu criterii de optimalitate de tip Lagrange și altul pentru criterii de tip Bolza.

Capitolul 1
Utilizarea metodei Simulated Annealing în probleme de conducere optimală

1.1 Metoda Simulated Annealing

Metoda SA (Simulated Annealing, tradus "călire simulată") propusă în (Kirkpatrick, 1983) și (Cerny, 1985) emulează procesul fizic prin care un corp solid este răcit încet, a.î atunci când procesul este "înghețat", acest lucru se întâmplă la o configurație de energie minimă.

Algoritmul SA generează o realizare a unui lanț Markov discret (în timp) neomogen, $x(t)$. Dacă starea curentă este $x(t) = i$, se alege în mod aleator un vecin j al lui i. Probabilitatea să fie ales $j \in S(i)$ este q_{ij}. Deîndată ce j este ales, starea următoare $x(t+1)$ este determinată astfel:

dacă $J(j) \le J(i)$, *atunci* $x(t+1)=j$.

dacă $J(j) > J(i)$, *atunci* $x(t+1)=j$ cu probabilitatea $\exp\left(-\frac{J(j)-J(i)}{T(t)}\right)$

altfe $x(t+1)=i$

1.2 Implementarea practică a algoritmului SA

Un mecanism important pentru dinamica algoritmului este adaptarea pasului de căutare. Strategia din algoritmul utilizat în această lucrare este de a face raportul $\alpha = N_{accept} / N_{eval}$,

cat mai apropiat de 0.5 pe durata execuției algoritmului, unde N_{accept} este numărul de pași cu acceptare a noii soluții candidat. Implementarea din această teză utilizează următoarea metodă de adaptare a pasului unde *β* este un parametru:

$$\Delta x^{l+1} = \begin{cases} \Delta x^l \left(1+\beta\frac{\alpha-0.6}{0.4}\right) & dacă\ d>0.6 \\ \Delta x^l & dacă\ 0.4\le\alpha\le 0.6 \\ \Delta x^l \left(1+\beta\frac{0.4-\alpha}{0.4}\right)^{-1} & dacă\quad \alpha<0.4 \end{cases} \tag{1.4}$$

1.3 Control optimal pentru sisteme dinamice exprimate prin ecuaţii diferenţiale ordinare şi algebrice

In acest caz, problema de conducere optimală este descrisă de următoarele elemente.
Sistemul dinamic *S* are evoluția modelată de ecuația generală:

$$\frac{dx}{dt} = f(x(t), y(t), u(t), p, t) \tag{1.5}$$

Unde *t* este timpul, $t \in \boldsymbol{R}$;

- $x(t)$ este vectorul variabilelor de stare de dimensiune
- $y(t)$ este vectorul variabilelor algebrice de dimensiune n_y; acestea fac obiectul ecuațiilor algebrice și sunt, de obicei, mărimi de ieșire din proces.
- $u(t)$ este vectorul variabilelor de comandă de dimensiune n_u.
- *p* este vectorul parametrilor invarianți în timp, de dimensiune n_p

Sistemul (1.5) este supus unor restricții de tip egalitate și inegalitate, după cum urmează:

$$x(0) = x_0 \quad \text{(condiții inițiale)}, \tag{1.6}$$

restricții algebrice de tip egalitate $$g\left(x(t), y(t), u(t), p, t\right) = 0, \tag{1.7}$$

restricții pe traiectorie de tip inegalitate $$h\left(x(t), \frac{dx(t)}{dt}, y(t), u(t), p, t\right) \le 0 \tag{1.8}$$

și restricții de timp final t_f, de tip egalitate $$\psi\left(x(t_f), \frac{dx(t_f)}{dt}, y(t_f), u(t_f), p, t_f\right) = 0 \tag{1.9}$$

Funcțiile *f, g, h* și ψ sunt funcții vectoriale de variabile vectoriale respectiv cu dimensiunile n_f, n_g, n_h, n_ψ. Considerăm o primă formulare de problemă de optimizare pentru sisteme dinamice exprimate prin ecuații diferențiale ordinare și algebrice (EDOA) care constă în minimizarea următoarei funcții obiectiv:

$$\min_{x(t), y(t), u(t), p, t_f} J\left(x(t_f), y(t_f), u(t_f), p, t_f\right) \tag{1.10}$$

Minimizarea funcției scalare *J* poate fi exprimată sub forma criteriului Bolza:

$$\min_{x(t), y(t), u(t), p, t_f} M\left(x(t_f), y(t_f), u(t_f), p, t_f\right) + \int_{t_0}^{t_f} L\left(x(t), y(t), u(t), p, t\right) dt \tag{1.11}$$

unde *M* este componenta Mayer a funcției obiectiv, iar *L* este termenul Lagrange.
În marea majoritate a problemelor de optimizare, restricțiile de tip (1.8) sunt tratate ca *restricții de frontieră* de forma

$$x^m \le x(t) \le x^M \; ; \; u^m \le u(t) \le u^M \; ; \; p^m \le p \le p^M \; ; \; t_f^m \le t_f \le t_f^M \qquad (1.8 \text{ bis})$$

Aceste inecuații, indicate pentru o singură componentă a variabilelor respective, utilizează valorile minimă (*m*) și maximă (*M*).
Se definește un spațiu de căutare al soluțiilor optimale prin discretizarea diferitelor variabile. Discretizarea se face pentru *N* momente de timp, de obicei echidistante, care acoperă orizontul de timp:

$$\bar{t} = (t_1, t_2, \cdots, t_N)^T \; ; \; \text{cu } t_N = t_f$$

Necunoscuta principală pentru problema de control optimal este succesiunea de comenzi pentru aceste momente, adică ceea ce se cheamă *profil de comandă*:

$$\bar{u} = (u_1, u_2, \cdots, u_N)^T ; \qquad (1.13)$$

Soluția problemei de conducere optimală (1.5)-(1.10), presupunând că problema are timpul final t_f liber, poate fi prezentată ca vector al variabilelor de optimizare:

$\bar{x} = (\bar{u}, p, t_f)^T$, sau $\bar{x} = (\bar{u}, p)^T$ dacă t_f este legat.

1.4 Clase de control optimal tratate cu algoritmul SA

1.4.1 Control optimal cu stare finală liberă și timp final fixat

Exemplul 1 (Faber, 2005). Într-o reacție chimică ce produce un produs B, o componentă A este consumată. Substtanța B reacționează la temperaturi ridicate și produce produsul secundar C. $2A \xrightarrow{k_1} B \xrightarrow{k_2} C$.

Rata de reacție este funcție de temperatură și concentrație. Obiectivul este maximizarea concentrației de componentă B la sfârșitul orizontului de timp t_f de o oră,

Modelul se inspiră din abordarea Arrhenius și produce trei ecuații diferențiale

$$\frac{dc_A}{dt} = -k_{1,0} \cdot e^{-E_1/RT} c_A^2, \quad \frac{dc_B}{dt} = k_{1,0} \cdot e^{-E_1/RT} c_A^2 - k_{2,0} \cdot e^{-E_2/RT} c_B \qquad (1.16)$$

$$\frac{dc_C}{dt} = k_{2,0} \cdot e^{-E_2/RT} c_B^2 - k_{2,0} \cdot e^{-E_2/RT} c_C$$

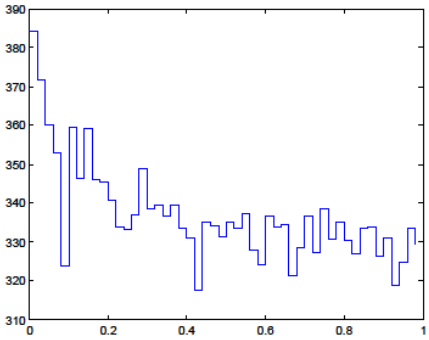
unde c_j , cu *j=A, B, C* în (mol mol^{-1}), $k_{1,0} = 4000\, mol^{-1} s^{-1}$ și $k_{2,0} = 6.2 \times 10^5 \; s^{-1}$, energiile de activare $E_1 = 5000 \,\text{cal mol}^{-1}$ și $E_2 = 10000 \,\text{cal mol}^{-1}$. *R* este constanta universală a gazelor, *R*=1.98721 cal mol^{-1}K^{-1}, iar *T* este temperatura în grade *K*.

Variabila de control este temperatura *T*. La început, reactorul este umplut cu componenta A, deci avem: $c_A(t_0) = 1$, $c_B(t_0) = 0$, $c_C(t_0) = 0$ (1.17)

Restricțiile de frontiera: $298K \le T \le 398K$ (1.18)

Funcția obiectiv este deci: $J\left(x(t_f), y(t_f), u(t_f), p, t_f\right) = J\left(x(t_f)\right) = c_B(t_f)$ (1.19)

Cu implementarea ASA, se obține profilul de comandă optimală din figura 1.3



$c_B(t_f) = 0.6106$

$N_{eval} = 23798$

T_A fina=8.7 10-6

Δx^l_{final} =0.9923

a=0.9

Figura 1.3 Profil de comandă optimală fără controlul pasului

Controlul pasului (*Step Control*)

Conform ASA, schimbările din profilul de comandă $\overline{u}$ se produc în trepte. O modalitate de a menține aceste variații în interiorul unor anumite limite este de a utiliza o tehnică numită controlul pasului (*step control*). Acesta impune niște margini superioare și inferioare ale gradientului variabilelor de conducere $\left|\frac{du(t)}{dt}\right| \le \Delta s_{\max}$, unde $\Delta s_{\max}$ este un parametru ce trebuie stabilit la inițializare. Această restricție nu face parte din cele folosite în problema de optimizare (1.5)-(1.10). Utilizând această tehnică în rezolvarea problemei expuse se ajunge la convergența soluției, lucru dovedit de mai toate execuțiile. De exemplu, pentru un profil de comandă discretizat în N_x=50 puncte pe orizontul de o oră și o inițializare a temperaturii liniar descrescătoare între 398°C și 349°C, se obține profilul de comandă din figura 1.4. $c_B(t_f) = 0.6106$; $N_{eval} = 16684$; T_{Afinal}=4.21·10^{-8}; Δx^l_{final} =6.08·10^{-4}; *a*=0.8

Prin simularea sistemului cu profilul de comandă determinat prin SA, se obține evoluția celor 3 variabile de stare prezentată în figura 1.5

1.4.2 Control optimal bilocal în raport cu starea și timp final fixat

Exemplul 3 - sistem de acționare cu motor electric(Belea, 1985).

$$\ddot{\theta} = u(t),$$

unde θ este unghiul axului motorului, iar u reprezintă un curent, tensiune sau cuplu activ ca mărime de comandă. Se cere comanda optimală $u^*(t)$, $t \in [0,2]$, care transportă sistemul din

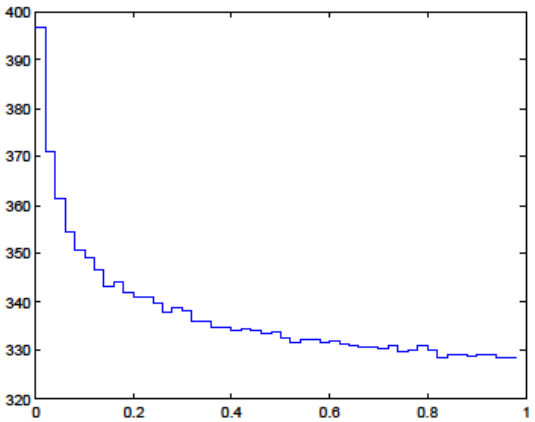


Figura 1.4 Profil de comandă optimală cu controlul pasului

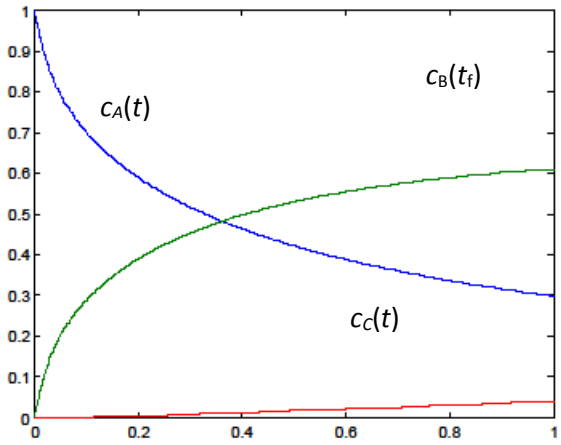


Figura 1.5 Evolutia variabilelor de stare

starea inițială: $t_o = 0$, $\theta_o = 1$, $\dot{\theta}_o = 1$, în starea finală: $t_f = 2$, $\theta_f = 0$, $\dot{\theta}_f = 0$,

a.î. consumul de energie să fie minim $\min_u J = \frac{1}{2}\int_0^{t_f} u^2(t)dt$

Ecuațiile de stare sunt: $\dot{x}_1 = x_2$, $\dot{x}_2 = u(t)$

Tratarea problemelor bilocale în raport cu starea

Aspectul bilocal se poate trata, în principal, prin penalizare suplimentară la nivelul funcției obiectiv. De aceea, se introduc termeni suplimentari în funcția obiectiv, corespunzători componentelor legate din variabila de stare:

$$\min_{u(t),t_f}\left\{c\cdot\left(x_{i_1}(t_f)-x_{i_1}^f\right)^2+\cdots+c\cdot\left(x_{i_r}(t_f)-x_{i_r}^f\right)^2+J\left(x(t_f),y(t_f),u(t_f),p,t_f\right)\right\} \quad (1.20)$$

unde: - $i_1, i_2, \cdots, i_r$ sunt indicii variabilelor de stare care sunt fixate la momentul t_f;

- $c > 0$, constantă de penalizare a neîndeplinirii restricției bilocale la momentul t_f;

- $x_{i_1}^f$, $x_{i_2}^f, \cdots x_{i_r}^f$ sunt valorile impuse variabilelor fixate la momentul t_f;

Minimizarea din (1.20) este actualizată la cazul tratat în această secțiune. De asemenea, în cazul în care restricția bilocală se referă și la momentul inițial, expresia (1.20) poate fi modificată corespunzător.

O problemă importantă este alegerea parametrului de penalizare *c*, care trebuie să penalizeze suficient neîndeplinirea restricției bilocale, dar nu trebuie nici să "eclipseze" minimizarea funcției obiectiv inițiale.Nnoua funcție obiectiv este

$$\min_{u(t),t_f}\left\{50\cdot\left(x_1(t_f)-0\right)^2+50\cdot\left(x_2(t_f)-0\right)^2+\int_0^{t_f}\cdot u^2(t)dt\right\}$$

Considerând un profil inițial de comandă având toate componentele egale cu -0.5, ASA converge la soluția din figura 1.8.

Comanda optimală teoretică este liniară: $u^*(t) = 3t - 3.5$, iar ecuațiile traiectoriilor de stare sunt: $x_1^*(t) = 0.5t^3 - 1.75t^2 + t + 1$, $x_2^*(t) = 1.5t^2 - 3.5t + 1$

În figura 1.9 se prezintă evoluția celor două stări pe intervalul [0, t_f] atât pentru comanda optimală teoretică (cea cu *), cât și pentru comanda "optimă" determinată cu ASA.

Se constată o foarte bună îndeplinire a restricției finale asupra stărilor fixate, care se abat de la cerință cu 3% și respectiv -2%

1.4.3 Control optimal bilocal în raport cu starea și timp final liber

Exemplul 4

Pentru a exemplifica acest caz, prezentăm în această secțiune un exemplu de control optimal pentru sisteme EDOA descris în (Yamashita, 1997), rezolvat în respectiva lucrare printr-un algoritm genetic. Timpul final t_f este liber

$$\begin{cases}\dot{x}_1 = -5x_1 + (1-x_2)\cdot u_1;\\ \dot{x}_2 = (1-x_1)\cdot u_1\end{cases} \qquad J = -\frac{1}{t_f}\int_0^{t_f}(3-4x_1)u_1 dt; \qquad \begin{array}{l}x(0) = [1,0]^T\\ x(t_f) = [0.5, 3.0]^{\mathrm{T}}\\ 0 \le u_1 \le 10\end{array}$$

Codificarea soluției optimale căutate de algoritmul SA

Ținând cont de ecuațiile (1.13) și (1.14) și întrucât problema prezintă timp final t_f liber, trebuie să adoptăm o soluție de forma

$$\bar{x} = \left(\bar{u}, p, t_f\right)^T = \left(\bar{u}, t_f\right)^T = \left(u_1, u_2, \cdots, u_N, t_f\right)^T; \quad (1.21)$$

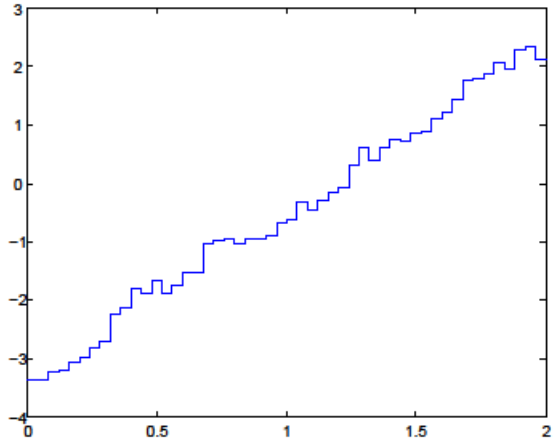


Figura 1.8 Profil de comandă

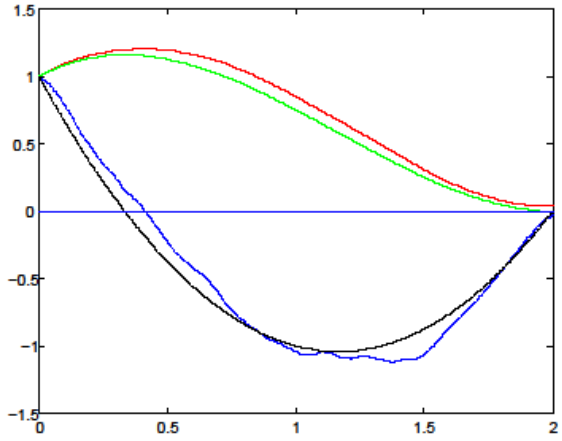


Figura 1.9 Evoluția comparativă a starilor

N_{eval}= 5542; J= 3.4288; T_A final=$5.27\cdot10^{-5}$; $\Delta x^l_{final} = 4.9\cdot10^{-5}$

$x_1(t_f)$= 0.034822; $x_2(t_f)$= -0.029617; $x_1^*(t_f)$= 0; $x_2^*(t_f)$= 0; verde: x_1 optimal teoretic; negru: x_2 optimal teoretic; rosu: x_1 obtinut cu SA; albastru: x_2 obtinut cu SA

Ca urmare necunoscuta t_f este ajustată în cadrul algoritmului, ținând cont și de un factor de scală adecvat.

Simularea sistemului dinamic pe un orizont de timp variabil

Din cauza faptului că timpul final este variabil, apare și necesitatea de a simula sistemul dinamic și de a calcula funcția obiectiv cu timp final variabil. Ambele aspecte pot fi simplificate, prin folosirea unui artificiu: schimbarea de variabila care reprezintă timpul.
Pentru exemplul de mai sus, putem recurge la un artificiu: introducerea unei noi variabile temporale: $\tau = t \cdot \frac{1}{t_f} \Rightarrow dt = t_f \cdot d\tau$

În acest caz, sistemul dinamic și funcția obiectiv devin:

$$\begin{cases} \frac{dx_1}{d\tau} = t_f \cdot [-5x_1(\tau) + (1 - x_2(\tau))u_1(\tau)] \\ \frac{dx_2}{d\tau} = t_f \cdot [(1 - x_1(\tau))u_1(\tau)] \end{cases}; \qquad J = -\int_0^1 (3 - 4x_1)u_1 d\tau$$

Avantajele acestui artificiu sunt: apelarea cu parametri invariabili a funcției Matlab de integrare numerică a sistemului dinamic și calculul cu parametri invariabili a funcției de integrare numerică pentru calculul funcției obiectiv, chiar și în situația în care variabila t_f nu dispare complet din expresia lui *J*. Prezența variabilei t_f se face simțită la nivelul funcției care calculează derivatele variabilelor de stare pentru diferite momente de integrare.
In cazul problemei de mai sus, noua funcție obiectiv este

$$\min_{u(t),t_f} \left\{ 50 \cdot (x_1(1) - 0.5)^2 + 50 \cdot (x_2(1) - 3.)^2 + \int_0^1 (-3 + 4 \cdot x_1(\tau)) \cdot u_1(\tau) d\tau \right\}$$

Majoritatea execuțiilor ASA realizate cu diverși parametri arată convergența soluțiilor găsite ce respectă condiția bilocală. Rezultatele găsite, crorespunzătoare valorilor : N_{eval}: 4154; $T_{A\ finală}$: $3.05\cdot10^{-6}$; Δx^l_{final} : 0.00022995; t_f=1.2924; *J*=-4.3457; $x_1(t_f)$= 0.53773; $x_2(t_f)$= 3.0019, sunt rezultate ce corespund cu cele prezentate în articolul menționat mai sus.

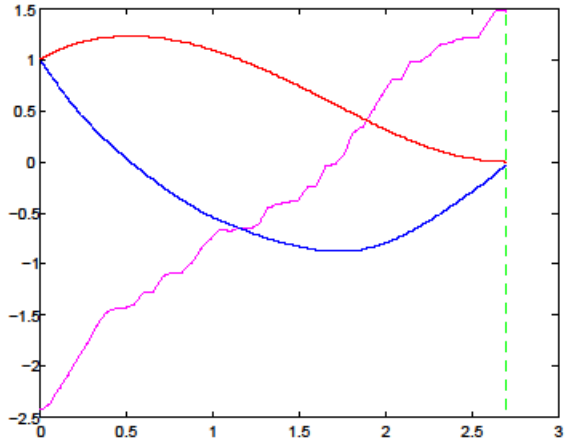
Exemplul 5

Reluăm exemplul 3, bilocal pe stare, dar considerăm o funcție obiectiv diferită, de tip Bolza cu timp final liber: $\min_u J = \frac{1}{2}\int_0^{t_f} u^2(t)dt + t_f$

De asemenea, pentru acest exemplu, putem recurge la introducerea unei noi variabile temporale: $\tau = t \cdot \frac{1}{t_f} \Rightarrow dt = t_f \cdot d\tau$

Sistemul dinamic și funcția obiectiv devin: $\begin{cases} \frac{dx_1}{d\tau} = t_f \cdot x_2(\tau) \\ \frac{dx_2}{d\tau} = t_f \cdot u(\tau) \end{cases}$, $J = \frac{1}{2} \cdot t_f \cdot \int_0^1 u^2(\tau)d\tau + t_f$

Soluția inițială utilizată de ASA este $\underbrace{[-2,-2,\cdots,-2,4]}_{51}$. După convergență, se obține profilul de comandă și evoluția stărilor prezentate în figura 1.12.



N_{eval}: 11329

$T_{A\ finala}$: $1.14 \cdot 10^{-5}$

Δx^l_{final} : $3.81 \cdot 10^{-5}$

t_f=2.7811

J=4.5858

$x_1(t_f)$= 0.0166

$x_2(t_f)$= -0.0154

Figura 1.12 Comanda si starile determinate de ASA

1.5 Concluzii

Secțiunea 1.3 a fost consacrată problemelor de control optimal pentru sisteme dinamice exprimate prin ecuații diferențiale ordinare și algebrice. După o formulare relativ generală a cestui tip de PCO, sunt examinate restricțiile impuse și modul de implementare al acestora. Plecând de la codificarea profilului de comandă, sunt descrise strategiile de căutare a soluției optime: variația secvențială a variabilelor sau căutarea paralelă. In secțiunea 1.4, au fost analizate, prin exemple, câteva clase de PCO care pot fi rezolvate cu ASA.

Clasa PCO *cu stare finală liberă și timp final fixat* este cea mai simplă clasă care poate fi rezolvată cu ASA. Un prim exemplu tratat a fost problema Batch Reactor, care are un criteriu de tip Mayer. Soluția optimală este cunoscută din alte lucrări și reconfirmată de soluția propusă. În acest exemplu s-a implementat tehnica de "step control" care a crescut calitatea soluției fără creșterea semnificativă a numărului de evaluări ale funcției obiectiv. Al doilea exemplu a tratat un sistem dinamic liniar cu criteriu de optimalitate de tip Bolza pentru care s-a obținut o soluție chiar optimă, utilizând aceeași codificare a soluției.

Pentru a ilustra posibilitatea de a rezolva cu ASA o PCO *bilocală în raport cu starea și timp final fixat* s-a considerat un sistem liniar cu două variabile de stare. Aspectul nou tratat în acest exemplu este introducerea de termeni suplimentari în funcția obiectiv, corespunzători

componentelor legate din variabila de stare. Stabilirea ponderii acestor termeni este problema cheie pe care proiectantul algoritmului trebuie s-o rezolve.

PCO bilocale în raport cu starea și timp final liber sunt exemplificate prin două probleme de naturi diferite. Cea din exemplul 4, are un sistem neliniar cu doua variabile de stare și are un criteriu Lagrange, dar timpul final este liber. Prin introducerea unei noi variabile temporale, criteriul nu mai depinde de timpul final, dar acesta rămâne parametru în ecuațiile de stare. În exemplul 5, acest lucru nu mai este posibil. S-a recurs la o codificare care adaugă timpul final pe ultima poziție a vectorului care reprezintă profilul de comandă, cu considerarea unui factor de scală corespunzător. Soluțiile obținute sunt mai mult decât satisfăcătoare.

Contribuțiile din acest capitol sunt:

- Tehnica numita controlul pasului (*step control*) impune niște margini superioare și inferioare ale gradientului variabilelor de conducere. Deși cunoscută, în prezenta lucrare a fost indicată explicit o manieră de implementare în PCO.
- Analiza modului de implementarea a PCO bilocale în raport cu starea și timp final fixat sau liber.

Capitolul 2
Algoritmi evolutivi în probleme de conducere optimală

2.1 Algoritmi evolutivi

Algoritmii evolutivi (AE) sunt metaeuristici stohastice bazate pe o populație de soluții și care adoptă în evoluția acesteia *principiul competiției*. Pe de altă parte, sunt algoritmi de optimizare iterativi care simulează evoluția speciilor prin evoluția unei populații de indivizi.

O populație inițială este generată în mod aleator. Un individ din populație codifică o soluție candidat a problemei de optimizare considerată. O funcție obiectiv asociază fiecărui individ o valoare "fitness" care arată cât de adecvat este individul ca soluție a problemei. La fiecare pas, indivizi sunt selectați pentru a forma părinți, respectând regula conform căreia indivizi cu valori "fitness" mai bune au o mai mare probabilitate să fie selectați. Apoi, aceștia fac obiectul unor operatori de variație cum ar fi crossover-ul și mutația pentru a genera așa-zisele "progenituri" (offsprings). În cele din urmă, se aplică o schemă de înlocuire prin care se decide care indivizi din populație vor supraviețui dintre părinți și progenituri

În lucrarea (Siarry, 2014) se precizează că algoritmii genetici sunt un caz particular de algoritmi evolutivi. Ca și element de diferențiere este faptul ca AG se inspiră din transcrierea genotip-fenotip din genetica naturală.

2.2 Implementarea AE pentru probleme de conducere în timp continuu

2.2.1 Generarea populației inițiale

Caracterul exploratoriu natural al AE trebuie asigurat și prin considerarea unei populații inițiale care să să fie dversă Există patru strategii posibile în generarea soluțiilor inițiale: generarea aleatoare, diversificarea secvențială, diversificarea paralelă și inițializarea euristică.

2.2.2 Metode de selecție

Operatorul de selecție stabilește la fiecare iterație ce indivizi vor fi supusi operatorului de recombinare și câți urmași va avea fiecare. Principiul de funcționare este "Cu cat un individ este mai bun, cu atât trebuie să fie mai mare șansa de a fi părinte". Este un principiu elitist care creează o *presiune de selecție* ce îndreaptă populația spre o componentă cu soluții din ce în ce mai bune. Valoarea funcției fitness dă calitatea unui individ. Dar asocierea acesteia fiecărui individ, poate fi făcută în doua moduri:

Asocierea proporțională cu valoarea funcției obiectiv (Proportional fitness assignement) când unui individ i se asociază valoarea sa de fitness ca atare;

Asocierea cu un rang considerat ca valoare fitness (Rank-based fitness assignement) când o valoare fitness este asociată, dar nu direct pe baza calculului funcției obiectiv, ci în funcție de un rang (de clasificare) în cadrul populației

Pentru selecția părinților în funcție de valoare fitness există patru tehnici larg utilizate în AE: selecție bazata pe principiul ruletei (Roulette Wheel Selection), selecția stocastică universală (Stochastic Universal Sampling), selecție de tip turneu și selecție bazată pe rang (Rank-based Selection) Reducerea derivei generate de Selecția pe principiul Ruletei se poate face cu Selecția Stohastică Universală. Se poate adopta o selecție bazată pe rangul liniarizat, caz în care se ține cont și de presiunea de selecție *s*. Presiunea de selecție se definește în cazul selecției proporționale cu valorile fitness. În acest caz, individului #i i se asociază probabilitatea P(*i*):

$$P(i) = \frac{2 - p_s}{\mu} + \frac{2 \cdot [r(i) - 1] \cdot (p_s - 1)}{\mu \cdot (\mu - 1)}. \tag{2.4}$$

Se alege, de obicei, $1 \le p_s \le 2$.

2.2.3 Mutația în cazul codajului real

Pentru cromozomii care sunt vectori de numere reale, operatorul *Mutație Gaussiană* este cel mai folosit în aplicații. El procedează la adăugarea unui număr aleator fiecărei gene, număr ce este eșantionat dintr-o distribuție normală *N*(0, σ), cu deviația standard σ valabilă pentru toți cromozomii. În acest mod, parametrul σ controlează gradul de împrăștiere a numerelor și poate fi modificat în câteva trepte, dându-i-se valori mai mari, atunci când AE este mai explorator în spațiul de căutare, și valori mai mici atunci când AE este mai orientat spre exploatare, adică spre optimizare locală.

O variantă a acestui operator este *Mutația Gaussiană Auto-adaptivă*. Particularitatea constă în faptul că fiecare cromozom are propria deviație standard.

*Mutația uniformă***:** Se alege aleator o poziție și se modifică gena corespunzătoare acelei poziții. În cazul codajului real, se înlocuiește valoarea genei cu o valoare aleasă aleator, după o lege uniformă, în domeniul de definiție a genei.

*Mutația neuniformă***:** Acest operator, inspirat din metoda „simulated annealing” este definit în maniera următoare:

cromozom tată: cromozom fiu:

$$\left(x_1^t \ldots x_i^t \ldots x_M^t\right) \longrightarrow \left(x_1^{t+1} \ldots x_i^{t+1} \ldots x_M^{t+1}\right)$$

unde avem:

$$x_i^{t+1} = \begin{cases} x_i^t + \Delta(t, UB - x_i^t) \ dacă \ a = 0 \\ x_i^t + \Delta(t, x_i^t - LB) \ dacă \ a = 1 \end{cases} \tag{2.5}$$

unde: $a \in \{0, 1\}$este ales aleator;

- *UB* și *LB* sunt respectiv marginile domeniului de variație ale variabilei x_i^t ;
- $\Delta(t, y)$ este o funcție care întoarce o valoare între 0 și *y*, ce tinde la zero pe măsură ce *t* se apropie de T.

2.2.4 Încrucișarea în cazul codajului real

Din multitudinea de operatori de încrucișare (crossover), în AE propus pentru rezolvarea PCO, ne-am oprit, într-o primă etapă, la două care sunt simple și verificate practic.

Încrucișare simplă . Când încrucișarea este într-un singur punct, ca în cazul codajului binar, se alege o poziție k în mod aleator între 1 și M, pentru cei doi cromozomi părinți. Evident, că se poate practica și o încrucișare în mai multe puncte.

Încrucișarea aritmetică . Este definită ca o combinație lineară a celor doi vectori (cromozomi) părinți. Dacă *x* și *y* sunt cei doi vectori, atunci cei doi fii *x'* și *y'* sunt definiți după cum urmează: $x' = \alpha.x + (1- \alpha).y$; $y' = (1- \alpha).x+\alpha.y$,
unde *α* este o constantă, în cazul încrucișării aritmetice uniforme, sau o variabilă ce depinde de vârsta populației, în cazul încrucișării aritmetice neuniforme (Michalewicz, 1992). O variantă a încrucișării aritmetice este aceea când α este generat aleator în intervalul [0,1].

2.3 Implementarea AE pentru determinarea profilului de comandă

La implementarea Algoritmului Evolutiv propus pentru determinarea profilului de comandă, au fost făcute următoarele opțiuni:

- Populația fiecărei generații are un număr μ de indivizi, care este un parametru al programului (de ex., în Anexa 2, avem μ=60);
- Populația este ordonată descrescător (pentru aflarea unui maxim) în funcție de valoarea funcției obiectiv, a.î cea mai bună soluție se găsește pe prima poziție;
- S-a utilizat selecția cu Stochastic Universal Sampling;
- S-a utilizat selecția bazată pe rang și s-a scalat liniar rangul conform formulei (2.4) ;
- Numărul de cromozomi copii generați la fiecare generație este un parametru al programului ("*nr_fii*"). Avem deci λ= *nr_fii* (de ex., în Anexa 2, avem λ =30);
- Cromozomii copii se obțin din încrucișarea unui cromozom din lista de selecție cu cel mai bun cromozom din populație și se generează un singur copil;
- Încrucișarea inițial adoptată este cea într-un singur punct (cu codificare reală);
- Cei λ copii înlocuiesc ultimii λ cromozomi din populația curentă.

2.3.1 Încrucișarea BLX-α liniară

Pentru una dintre problemele de mai jos, s-a optat pentru alt tip de încrucișare, și anume "Încrucișarea BLX-α liniară" (Linear Blend Alpha Crossover). Daca *x* și *y* sunt două puncte din $\boldsymbol{R}^n$ reprezentând doi indivizi din populația de soluții, un individ *z* rezultă prin aplicarea încrucișării BLX-α liniare conform unei distribuții uniforme pe un segment de dreaptă ce trece prin *x* și *y*.

$$z = x + (y - x) \cdot U(-\alpha, 1 + \alpha)$$

unde $U(-\alpha, 1 + \alpha)$ este un număr tras aleator în intervalul $[-\alpha, 1+\alpha]$

2.3.2 Problema de producere fed-batch a proteinelor (PPFBP)

În lucrarea (Valadi, 2014), la capitolul 8, este prezentată problema de control optimal de mai jos care modelează producția fed-batch de proteină străină indusă prin bacterii recombinate. Prin rezolvarea PPFBP se încearcă determinarea "inductorului" optimal și profilul de alimentare cu nutriment care să maximizeze profitabilitatea fermentatorului. Timpul rezervat unui lot este de 10 ore.

Modelul dinamic are 2 intrări de comandă (u_1 și u_2) și 7 variabile de stare și e descris de următoarele ecuații:

$$\dot{x}_1 = u_1(t) + u_2(t) \tag{2.7}$$

$$\dot{x}_2 = \frac{x_3(t)}{14.35 + x_3(t)\cdot\left(1+\frac{x_3(t)}{111.5}\right)} \cdot \left(x_6 + \frac{0.22\cdot x_7}{0.22 + x_5}\right)\cdot x_2 - (u_1(t)+u_2(t))\cdot\left(\frac{x_2}{x_1}\right), \tag{2.8}$$

$$\dot{x}_3 = \left(\frac{100\cdot u_1}{x_1}\right) - (u_1(t)+u_2(t))\left(\frac{x_3}{x_1}\right)\left(\frac{x_3}{14.35 + x_3\left(1+\frac{x_3}{111.5}\right)}\right)\left(x_6 + \frac{0.22\cdot x_7}{0.22+x_5}\right)\left(\frac{x_2}{0.51}\right) \tag{2.9}$$

$$\dot{x}_4 = \left(\frac{0.233\cdot x_3}{14.35 + x_3\cdot\left(1+\frac{x_3}{111.5}\right)}\right)\cdot\left(\frac{0.005+x_5}{0.022+x_5}\right)\cdot x_2 - (u_1(t)+u_2(t))\cdot\left(\frac{x_4}{x_1}\right) \tag{2.10}$$

$$\dot{x}_5 = \left(\frac{4\cdot u_2(t)}{x_1}\right) - (u_1(t)+u_2(t))\cdot\left(\frac{x_5}{x_1}\right); \dot{x}_6 = \left(-\frac{0.09\cdot x_5}{0.034+x_5}\right)\cdot x_6;\ \dot{x}_7 = \left(\frac{0.09\cdot x_5}{0.034+x_5}\right)\cdot(1-x_7)$$

(2.11-2.13)

Restricții: $0 \le (u_1(t), u_2(t)) \le 1;\quad u_1(t) \ge u_2(t),\ t \in [0, t_f];$ (2.14)

Starea inițială la $t_{initial}$=0: $< x_1, x_2, x_3, x_4, x_5, x_6, x_7 >=< 1, 0.1, 40, 0, 0, 1, 0 >$ (2.15)

Funcția obiectiv: $J = \left\{x_1(t_f)\cdot x_4(t_f) - Q\cdot\int_{t_0}^{t_f} u_2(t)\cdot dt\right\}$ (2.16)

care trebuie maximizată $\max_{u_1(t),u_2(t)} J$ cu $t_f = 10h$. (2.17)

Valadi (2014)consideră cazul Q=0. Pentru acesta, s-a menționat că s-a obținut $J^* = 6.15$.

Profilul de comandă s-a cerut pe ore, a.î. fiecare dintre intrările de comandă, u_1 și u_2, este indicat prin cate un vector de 10 componente.

În figura 2.5 sunt prezentate profilele celor două comenzi. Valoarea maximă a funcției obiectiv obținută frecvent este J^*=6.1577, ceea ce corespunde cu valoarea indicată în (Valadi, 2014). Valoarea se obține în aproximativ 130 generații, adică 3960 apeluri ale funcției obiectiv, ceea ce este pe deplin acceptabil.

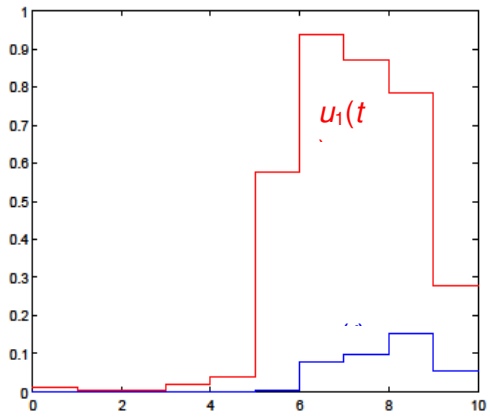


Figura 2.5 Profilul de comandă determinat cu AE pentru PPFBP

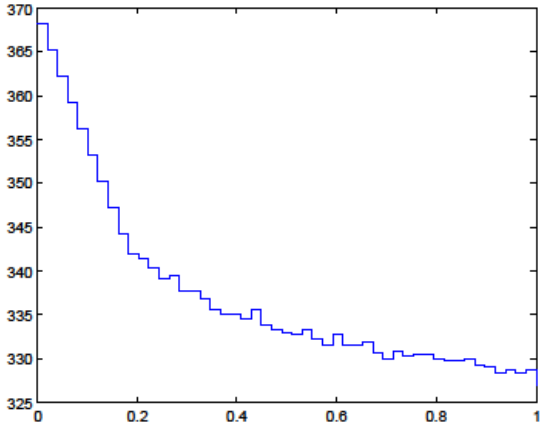


Figura 2.9 Profil de comandă determinat cu AE și Step Control(Problema Batch Reactor)

2.3.3 Problema Batch Reactor

Pentru a realiza o comparație cu algoritmul SA, vom relua această PCO din paragraful 1.4.1 *Exemplul 1*

Aplicarea programului din Anexa 2 nu duce la rezultate satisfăcătoare, în special din cauza operatorului de încrucișare într-un singur punct. De aceea, a fost adoptat operatorul BLX-0.5

După mai multe execuții ale AE, dintre care unele pentru calibrarea parametrilor, au rezultat următoarele valori pentru parametrii utilizați de AE:

- O soluție este codificată printr-un vector $x=[x_1,...,x_n]$, n=50; μ=N=60; λ=nr_fii=30; Nr. maxim generatii:230; $\Delta s_{\max}$ =3.0; (pentru "step control" când această tehnică se folosește)

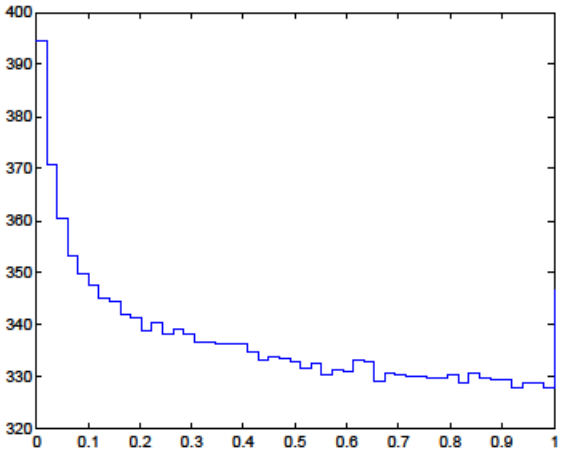


Figura 2.7 Profil de comandă determinat cu AE

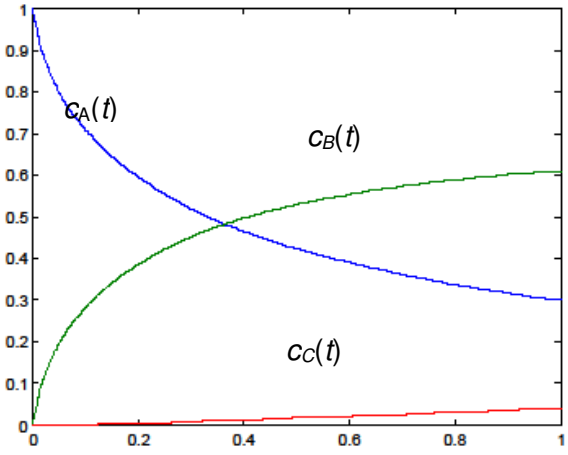


Figura 2.8 Evoluția stărilor

O evoluție tipică a AE conduce la rezultate ca cele din figurile 2.7 și 2.8. Acestea prezintă profilul de comandă și respectiv evoluția stărilor.

Convergența se atinge după evoluția de-a lungul a 200 și 250 generații. Față de algoritmul SA, se constată o complexitate mai redusă până la convergență, în evoluția către soluția optimă, întrucât numărul de evaluări ale funcției obiectiv este mai redus. În plus, valoarea optimală găsită pentru funcția obiectiv este mai mare, adică 0.6107.

În figura 2.7 se observă o evoluție destul de rugoasă a profilului de comandă. Se poate obține o evoluție mai netedă a acestuia, ca în figura 2.9, datorită integrării tehnicii de control al pasului.

2.3.4 Exemplu de Problemă Liniar Pătratică

Instanțele Problemei Liniar Pătratice (LQP) tratate sunt preluate din lucrarea (Michalewicz, 1992), unde acestea sunt rezolvate cu un algoritm genetic.

Fie sistemul discret: $x_{k+1} = a \cdot x_k + b \cdot u_k$, $k = 0, 1, \cdots, N-1$; cu starea inițială: x_0 dată.

Criteriul de optim este: $\min_{u_k,\, k=0,1,\cdots,N-1} \left[q \cdot x_N^2 + \sum_{k=0}^{N-1} \left(s \cdot x_k^2 + r \cdot u_k^2 \right) \right]$, cu parametrii: a, b, q, s, r precizați.

Pentru a compara soluția găsită de algoritmul EA cu cea teoretică, precizăm valoarea optimă a funcției obiectiv $J^* = K_0 \cdot x_0^2$ unde K_k este soluția ecuației algebrice Riccati:

$$K_k = r \cdot a^2 \cdot \frac{K_{k+1}}{r + b^2 \cdot K_{k+1}} + s \text{; cu } K_N = q .$$

LQP a fost rezolvată considerând o valoare N=45 și 4 instanțe ale problemei, la fel ca în articolul respectiv. Aceste instanțe sunt prezentate în tabelul de mai jos:

problema A	a=1; b=1; q=1; r=1; s=1; x_0=6.4
problema B	a=0.01; b=1; q=1; r=1; s=1; x_0=100
problema C	a=1; b=1; q=1; r=10; s=1; x_0=100
problema D	a=0.7; b=1.5; q=2; r=0.5; s=1.2; x_0=100

S-a încercat rezolvarea problemei A cu algoritmului evolutiv prezentat în Anexa 2, adică pe structura prezentată în lucrarea de mai sus, și chiar cu un operator de crossover mai bun, care include crossoverul aritmetic ca un caz particular.
Parametrii principali sunt: N=50; ngene=45; x_{min}=-5; x_{max}=5; presiune de selecție=1.8

Pentru problema A soluția teoretică este K_0=1.61803; J^*=66.2747

Rezultatul tipic obținut este următorul: gen=10000; J*=66.981; Neval =301850;

Cu alte cuvinte, în 10000 de generații se obține un profil de comandă bun, care duce la o valoare a funcției obiectiv situată la 1% de cea optimă, dar cu un număr imens de evaluări ale acesteia. Precizăm ca algoritmul converge întotdeauna, dar este foarte lent. Ca urmare, am adoptat un operator de mutație care să permită un control mai riguros al mărimii pasului de variație al valorilor genelor unui cromozom ($\Delta(t, y)$), variație care are un caracter aleatoriu. La mutația neuniforma exista un *control al variației* pasului de variație (ca variabilă aleatoare), dar această varianță este monoton descrescătoare. De aceea este necesara adoptarea unui alt operator de mutație care să adapteze această varianță pe durata execuției AE.

2.4 Controlul adaptiv global al variației

2.4.1 Strategia Evolutivă Adaptivă

Concluzia secțiunii anterioare se referă la adoptarea unei tehnici împrumutate din strategiile evolutive, care au dus la primele forme de algoritmi evolutivi. În aceste strategii, mutația este mai importantă decât crossoverul, care poate chiar să lipsească din AE. Mutația constă în adăugarea unui vector Δ, de componente Δ_i, i=1,...,n gene. Fiecare element Δ_i este o

realizare a unei variabile aleatoare distribuită normal de medie zero și varianță σ_i^2. Varianța poate fi stabilită pentru fiecare element, sau să fie una pentru întreg cromozomul și poate depinde de indexul generației.

2.4.2 Mutație cu varianță adaptivă

În lucrarea (Siarry, 2016), punctul de vedere asupra implementării procedurii de adaptare a deviației standard σ are elemente diferite, legate de momentele de adaptare. De aceea, în AE modificat pe care l-am utilizat pentru problema LQP, am propus procedura de mutație cu adaptarea deviației standard descrisă de lista 2.8. Parametrul θ, $0 < \theta < 1$, este pragul de la care se modifica deviația standard, la fiecare *M* mutații. O valoare consacrată este θ=1/5, caz în care se spune că se aplică "Regula unei cincimi".

Reluând LQP cu noul AE, s-au obținut rezultatele din figura 2.1(ne rezumăm la problemele A și B). De data aceasta, execuțiile tipice sunt toate convergente, dar și mult mai eficiente. În general, numărul de generații și numărul de evaluări ale funcției obiectiv sunt de 15-20 ori mai mici. Valoarea optimă teoretic este atinsă pentru toate cele 4 probleme.

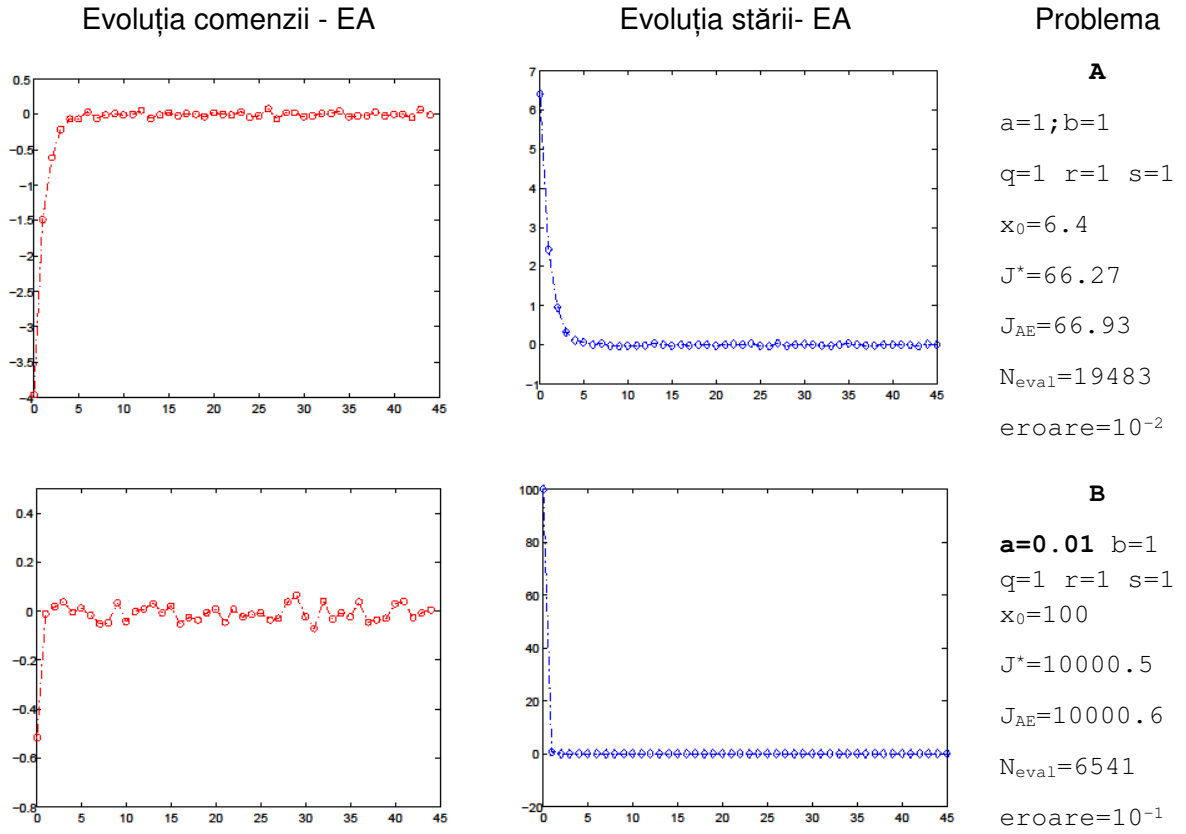


Figura 2.10 Rezultatele tipice ale LQP obținute de un AE folosind un operator de mutație cu varianță auto-adaptiva

2.5 Concluzii

În cazul tratat, adică cel al rezolvării unei PCO, codificarea printr-un șir de valori reale a unui profil de comandă este o opțiune preferabilă prin simplitate. De aceea, nu se degajă necesitate unei codificări de tip genotip. Ca urmare, problema poate fi rezolvată printr-un algoritm evolutiv.

Printr-un demers de "state of the art", în primele două secțiuni ale capitolului au fost trecute în revistă tehnici și operatori semnificativi în dezvoltarea unui AE legate de generarea

populației inițiale, metode de selecție a unor indivizi din populație, tipuri de mutație, tipuri de operatori de încrucișare și strategii evolutive.

Pentru studiul implementării unui AE dedicat determinării profilului de comandă, au fost considerate trei PCO.

Pentru problema de producere fed-batch a proteinelor, a fost dezvoltat un AE cu următoarele caracteristici: Selecție cu Stochastic Universal Sampling, Utilizare Rang și Scalare Rang pentru selecție, "Replacement" în ultimele lambda poziții din cele N ale populației, Încrucișare într-un singur punct cu un singur urmaș, mutație neuniformă pe toate genele.

Deși simplă, î*ncrucișarea de acest tip este satisfăcătoare pentru acest tip de problemă* și ca urmare soluțiile și viteza de convergență sunt satisfăcătoare.

În contrapartidă, pentru Problema Batch Reactor, acest tip de încrucișare nu duce la rezultate satisfăcătoare. De aceea s-a adoptat un operator crossover mai eficient: BLX-0.5. Soluțiile și viteza de convergență sunt, cu acest operator, mai mult decât satisfăcătoare.

A treia problemă tratată în analiza noastră a fost Problema Liniar Pătratică, care pentru un joc de parametri, are o complexitate practică surprinzătoare, cu varianta de AE anterioară. De aceea, am adoptat o Strategia Evolutiva cu Control Adaptiv Global al Variației, ce a utilizat un operator de mutație cu deviație standard normală controlată adaptiv. Rezultatele au fost spectaculoase, pentru că numărul de generații și numărul de evaluări ale funcției obiectiv sunt de 15-20 ori mai mici decât la variantele celelalte de AE utilizat.

Contribuții ale acestui capitol sunt:

- Integrarea unor tehnici eficiente (cum ar fi step control) și selecții care să evite fenomenul de deviere (Selecție cu Stochastic Universal Sampling, Utilizare Rang și Scalare Rang) într-un AE care să tina cont de particularitatea PCO tratate;
- O analiză pe o cazuistică formată din trei PCO, a diferiților operatori de crossover și mutație adecvați problemelor respective. Concluzia ne arată că fiecare problemă are o sensibilitate diferită la utilizarea unui operator sau altul. Au fost implementați și analizați operatori semnificativi pentru implementarea unui AE.

Capitolul 3 Utilizarea metodei Particle Swarm Optimization în probleme de conducere optimală

3.1 PSO în inteligenţa computaţională

Inteligența computațională a *roiurilor* (Swarm) *de particule* este inspirată din natură, de *comportamentul colectiv* al stolurilor de păsări, roiurilor de insecte, colonii de furnici, bancuri de pește, etc. S-a constatat și modelat auto-organizarea acestor grupuri de indivizi (numiți în cele ce urmează "particule"), precum și comportamentul lor emergent ca urmare a interacțiunii locale între particule.

Există două moduri de interacțiune între particule, prin comunicare directă sau indirectă. Când inteligența computațională bazată pe roiuri de particule utilizează în modelele sale *comunicarea directă,* avem de a face cu Particle Swarm Optimization.

Roiul este modelat de un număr de *N* particule, fiecare particulă fiind reprezentată printr-un vector cu 3 componente $(\vec{X}_i, \vec{V}_i, \vec{P}_{best\,i},)$, unde vectorii *m* dimensionali sunt respectiv poziția, viteza și cea mai bună valoare "întâlnită" de particula *i* (cea mai bună experiență personală). Cea mai bună poziție pe întreg roiul ("global best") la un moment dat *t* este $\vec{P}_{gbest}$,

Memoria $\vec{P}_{gbest}$ oferă singură posibilitate de comunicare între particule. Actualizarea vitezei și poziției unei particule *i* se face prin relațiile:

$$\vec{V}_i(t+1) = w \times \vec{V}_i(t) + C_1 \times rand_1 \times \left(\vec{P}_{best\,i}(t) - \vec{X}_i(t)\right) + C_2 \times rand_2 \times \left(\vec{P}_{gbest}(t) - \vec{X}_i(t)\right) \quad (3.1)$$

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1);$$

Există și o altă variantă a algoritmului PSO, cea care se bazează pe o topologie hibridă - Hybrid Topology Particle Swarm Optimization (HTPSO), adică utilizează ambele topologii, globală și locală, în același timp. Ecuația (3.1) este înlocuită prin:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) + C_3(t) \cdot rand_3 \cdot \left(p_{gbest}^d(t) - x_i^d(t)\right) \quad (3.5)$$

unde $rand_3$ este un număr aleator uniform distribuit în intervalul [0 1].

3.2 PSO utilizat în probleme de conducere optimală continuă

3.2.1 Implementarea PSO cu topologie hibridă pentru probleme de conducere

Particularități ale implementării legate de problemele de CO continue

O primă particularitate constă în faptul că evaluarea funcției obiectiv, dată fiind natura PCO, are un caracter mai complex decât în aplicațiile PSO obișnuite pentru că:

- realizează o întreagă simulare a sistemului dinamic pentru profilul de comandă setat de algoritm. Se face, deci o integrare a sistemului dinamic pe orizontul de timp al problemei;
- calculează funcția obiectiv asociată PCO;
- dacă PCO are și un caracter bilocal cu starea, se consideră funcția obiectiv extinsă (1.4.2)

O altă particularitate constă în *integrarea tehnicii step control* în algoritmul HTPSO. Din diferite motive, inclusiv uzarea elementelor de execuție, profilul de comandă trebuie să se apropie de o evoluție continuă și chiar derivabilă. Fără această tehnică, caracterul aleator al stabilirii salturilor comenzii la momente de timp adiacente zădărnicește acest deziderat și comanda "optimală" poate conține salturi puternice. În esență, trebuie implementată restricția de tip diferențial $\left|\frac{du(t)}{dt}\right| \le \Delta s_{\max}$

unde $\Delta s_{\max}$ este implementat prin parametrul `dTdtmax`. Încă o dată, precizam că restricția aceasta nu face parte din PCO. Astfel se limitează saltul de poziție pentru direcții succesive, la aceeași particulă. Ținând cont de codificare, asta înseamnă limitarea comenzii la momente succesive din profilul de comandă.

3.2.2 Exemple de determinare a profilului de comandă prin PSO cu topologie hibridă

Pentru a realiza o comparație cu algoritmul SA, vom relua mai întâi două exemple de PCO din cap. 1. Reluăm mai întâi exemplul 1 al rectorului Batch in care o substanță A este convertita in componentele B și C. Se cere determinarea profilului de temperaură care asigură concentrația maximă a componentei B după o oră.

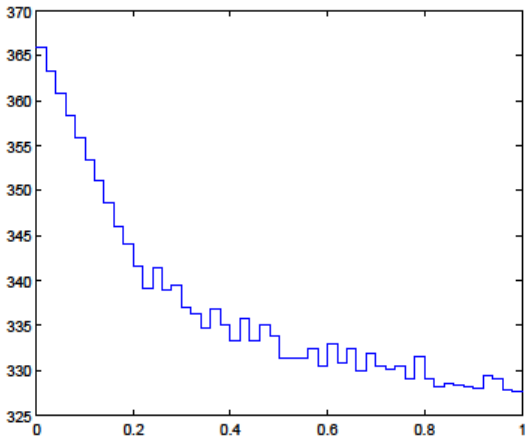


Figura 3.1 Evoluția temperaturii de comandă

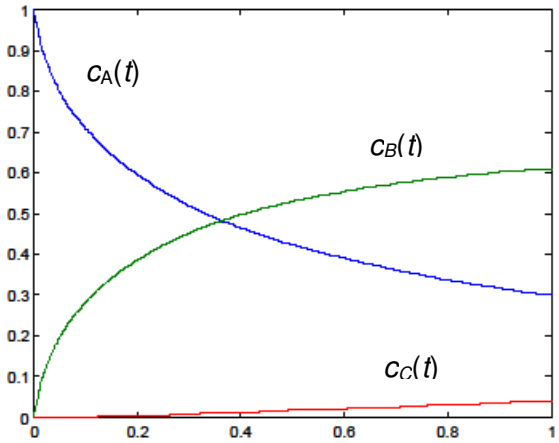


Figura 3.2 Evoluția stărilor

Figura. 3.4 ilustrează o foarte bună evoluție a profilului de comandă, în comparație cu comanda teoretică. Se constată un număr de evaluări sub jumătate din cel utilizat de ASA, în numai 121 de iterații.

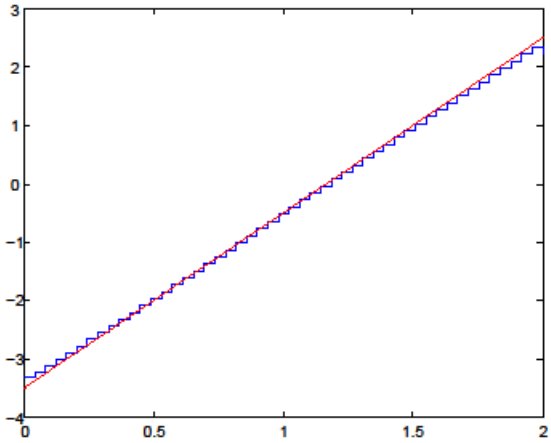


Figura 3.4 Evoluția comenzii
teoretica-roșu HTPSO- albastra

$t=121;$

$N_{eval}= 2420$

$J^*_{teoretic}= 3.25$

$J_{HTPSO}= 3.191$

$J^*=3.118$

Evoluția stărilor sistemului pe intervalul $t \in [0,2]$, cu comanda determinată P_{gbest}, dar și cu comanda optimală teoretică este prezentată comparativ, în figura 3.5.

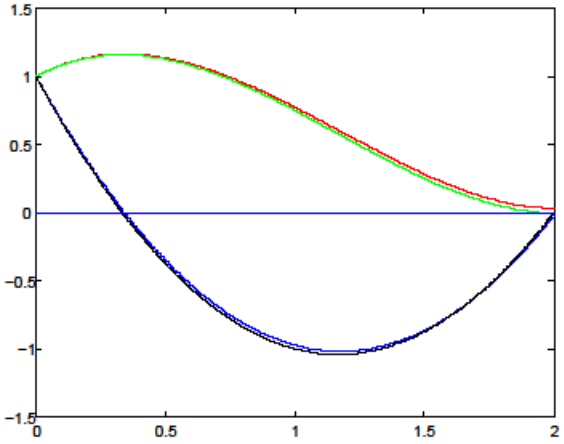


Figura 3.5 Evoluția stărilor - teoretic și cu HTPSO

$x_1(t_f)=0.029412$

$x_2(t_f)=-0.024452$

$x_1^*(t_f)= 0$

$x_2^*(t_f)= 0$

verde: x_1 optimal teoretic

negru: x_2 optimal teoretic

roșu: x_1 obținut cu HTPSO

albastru: x_2 obținut cu HTPSO

Stările evoluează cu cele două comenzi aproape identic, lucru remarcabil care este elocvent pentru eficiența algoritmului HTPSO.

Exemplu de Problemă Liniar Pătratică

Aplicăm aE pe exemplul din paragraful 2.4.2. LQP a fost rezolvată considerând o valoare *N*=45 și cele 4 instanțe ale problemei (combinații ale valorilor parametrilor).

Figura 3.6 prezintă rezultatele celor 4 instanțe de LQP (A, B, C, D). Algoritmul HTPSO a asigurat convergența soluției obținute pentru toate cele 4 instanțe. De menționat că instanța A (`a=1;b=1; q=1; r=1; s=1;`) este singura care nu a dus exact la valoarea teoretică a funcției obiectiv : $J^*=66.27$ și $J_{HTPSO}=68.11$. Adică prezintă o eroare de +2.7%. Celelalte instanțe au fost rezolvate a.î s-a asigurat $J^*= J_{HTPSO}$ chiar pentru toate încercările.

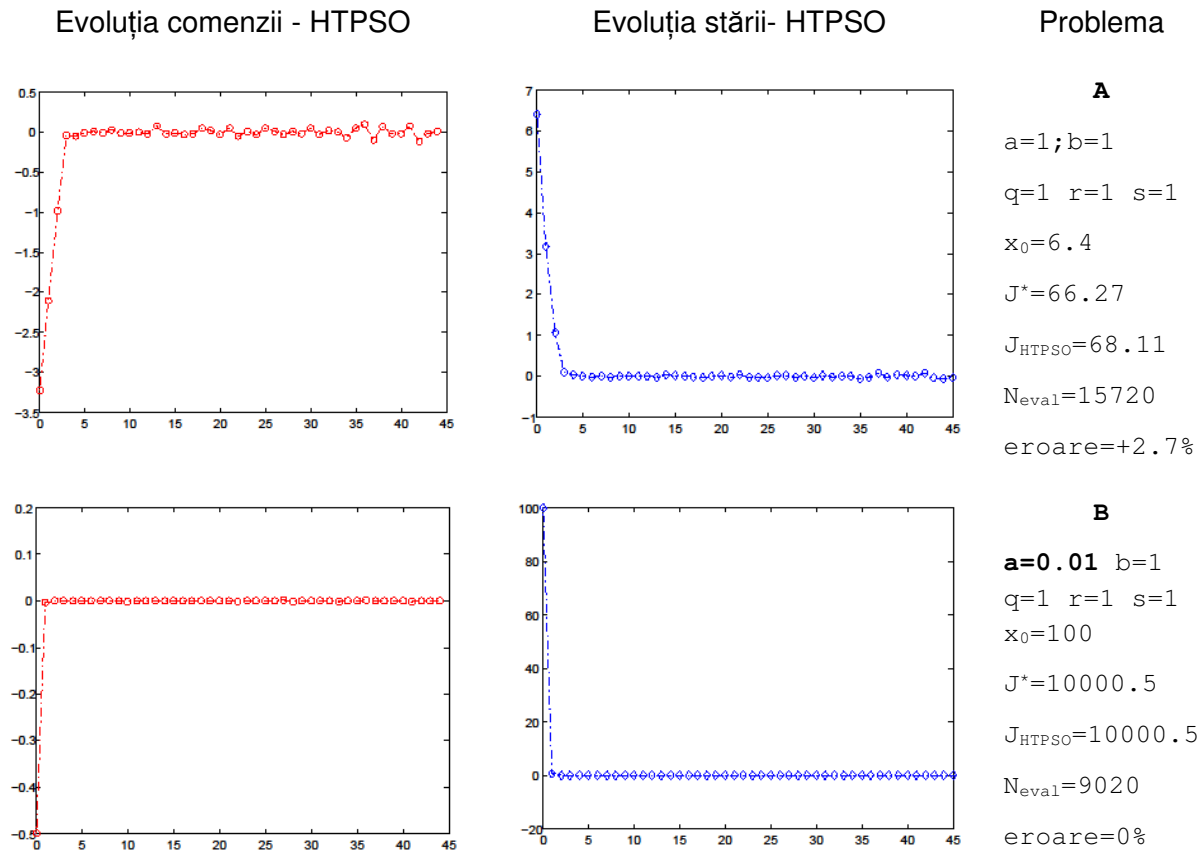


Figura 3.6 Rezultatele tipice ale LQP pentru cele 4 instanțe

3.3 PSO în probleme de conducere optimală cu comandă discretă binară

3.3.1 Adaptarea metaeuristicii PSO cu topologie hibridă la probleme discrete binare

Așa cum am văzut în secțiunea 3.1, PSO este o metaeuristică, bazată pe o populație de soluții, care s-a perfecționat de-a lungul timpului, și care a fost creată să caute soluții "bune" într-un spațiu de căutare continuu. Cu toate acestea, PSO și variantele sale au fost adaptate pentru a fi utilizate în spatii de căutare binare. Lucrarea (Beheshti, 2015) prezintă una dintre cele mai eficiente variante de PSO ce se pretează la a fi utilizată în problemele de optimizare binare.

În versiunea binară a HTPSO, pe care o numim BHTPSO, viteza unei particule este calculată tot cu relațiile (3.3) și (3.5), și doar poziția este altfel determinată, dată fiind natura sa binară. Pentru determinarea poziției pentru o componentă *d*, adică $x_i^d(t) \in \{0,1\}$, se folosește o interpretare probabilistică a vitezei particulei pe direcția *d*:

O valoare mare a vitezei $v_i^d(t+1)$ *este echivalentă cu o probabilitate mare de schimbare a poziției anterioare* $x_i^d(t)$. *Aceasta va fi schimbată la momentul t+1, prin complementare* (*din '0' în '1' sau invers*). *O valoare mică a vitezei* $v_i^d(t+1)$ *este echivalentă cu faptul că poziția este satisfăcătoare și nu este nevoie să se schimbe.*

Ecuației (3.5) capătă mai întâi forma $a_i^d(t+1) = v_i^d(t+1) + C_3(t) \cdot rand_3 \cdot \left(p_{gbest}^d(t) - x_i^d(t)\right)$.

Funcția $S(a) = |\tanh(a)|$ este adecvată să fie considerată o funcție de probabilitate care furnizează probabilitatea de a schimba componenta respectivă a poziției.

Pentru a scăpa de un optim local, se adaugă la $S(a)$ o valoare mică E_i dată de relația:

$$E_i = erf\left(NF_i / T'\right) = (2/\sqrt{\pi}) \int_0^{NF_i/T'} e^{-t^2}\, dt, \tag{3.8}$$

Reunind relațiile (3.7) și (3.8), probabilitatea de a schimba valoarea binară a poziției componentei este dată de (3.9):

$$S\left(a_i^d(t+1)\right) = E_i + (1 - E_i) \times \left|\tanh(a_i^d(t+1))\right| \tag{3.9}$$

Dacă $rand_4$ este un număr uniform distribuit în intervalul [0,1], poziția viitoare a respectivei componente se calculează cu:

$$x_i^d(t+1) = \begin{cases} complement\left(x_i^d(t)\right) & \textbf{dacă } rand_4 < S\left(a_i^d(t+1)\right) \\ x_i^d(t) & \textbf{altfel} \end{cases} \tag{3.10}$$

și pentru $a_i^d(t)$ există o restricție de mărginire: $\left|a_i^d(t)\right| < a_{\max}$ unde $a_{\max}$ se alege pentru fiecare problemă de optimizare în parte. Cu considerentele de mai sus, algoritmul HTPSO poate fi modificat și se obține algoritmul BHTPSO, prezentat în *Lista 3.3*

Algoritm BHTPSO

```
1.  start
2.    Generează populația inițială formată din N particule printr-o selecție binară aleatoare
        în spațiul n-dimensional;
3.    Generează vitezele inițiale ale particulelor ca valori uniform distribuite în intervalul
        [-vmax, vmax]^n
4.    t←1;
5.  ...cât timp condiția de terminare nu este îndeplinită
6.      pentru i=1,..., N       /* fiecare particulă*/
7.        Generează P_lbest_i folosind tehnica descrisă în secțiunea 3.1;
8.        Actualizează viteza particulelor utilizând ecuația (3.3);  /* d=1,..., n */
9.        Actualizează poziția particulelor utilizând ecuațiile (3.6)- (3.11)
10.       val ← EvalFitness( X_i );
11.             dacă (val < BESTi) atunci  P_best_i ← X_i ; ■
12.       dacă (val < GBEST) atunci P_gbest ← X_i ;■
13.       ■
14.     t←t+1;
15.   repetă
16.   return Pgbest
17. stop
```

Lista 3.3 Structura generală a algoritmului BHTPSO

3.3.2 Contribuții la ameliorarea topologiei "local best"

Așa cum am văzut, direcția $\vec{P}_{lbest_i}$ este dată de cea mai bună experiență a particulelor care formează vecinătatea sa "socială". O modalitate de a ameliora intensificarea este aceea de a considera, pe lângă vecinătatea sa "socială", și o căutare a unei soluții mai bune într-o vecinătate "geometrică". În lucrarea (Mînzu, 2015) a fost prezentată o materializare posibilă a acestei idei. S-a propus ca procedura ce determină $\vec{P}_{lbest_i}$ să includă o componentă de intensificare care să ia în considerare poziția curentă $\vec{X}_i$ și care este un algoritm determinist de coborâre locală numit *LDDA* (*local descent deterministic algorithm*). Obiectivul acestui algoritm este să producă o nouă poziție $\vec{X}_i^{'}$, situată în vecinătate "geometrică" a vectorului $\vec{X}_i$, care să aibă o mai bună valoare a funcției obiectiv. Acest nou vector $\vec{X}_i^{'}$ poate să dea un nou $\vec{P}_{best_i}$, sau chiar cea mai bună soluție globală $\vec{P}_{gbest}$. Practic acest algoritm determinist este integrat în procedura de calcul a vectorului $\vec{P}_{lbest_i}$ ca în schema de mai jos.

`Generare_Plbest(`*i*`)`

```
1.    start
2.        Determină indicii j1, j2, j3 ale celor 3      /* pot fi și mai multe "informatoare"*/
          particule "informatoare" ale particulei #i;
3.        (j, val1)← max(BESTj1, BESTj2, BESTj3)
4.    (X_i', val2)←LDDA(X_i);
      dacă val1<val2
5.        atunci Plbest(i)← Pbest(j)
6.        altfel Plbest(i)← X_i'
7.    ■
8.    return Plbest(i)
```

Funcția `Generare_Plbest(`*i*`)` este apelată în linia #7 a Listei 3.3. Linia #3 determină cea mai bună experiență personală pentru cele 3 particule informatoare. Valoarea val_1 este cea mai mică valoare (dacă căutam un minim) dintre cele 3 valori `BEST` și *j* este indicele respectiv.

De remarcat că $P_{lbest}(i)$ se poate întoarce cu valoarea poziției $\vec{X}_i^{'}$, generată de *LDDA*. Pe de o parte, efectul pozitiv este că $\vec{X}_i^{'}$ produce în general o valoare foarte bună pentru funcția obiectiv, ba chiar un minim local. Pe de altă parte, avem de a face cu o anulare a comunicării cu particulele informatoare și roiul nu se mai comportă așa cum este spiritul PSO.

În această lucrare propunem o nouă abordare a introducerii acestei căutări locale deterministe într-o vecinătate geografică. Principiul de bază pe care trebuie să-l implementăm este: *Componenta socială de comunicare locală trebuie să rămână neschimbată, pentru a păstra strategia de intensificare adoptată de BHTPSO. Aplicarea unei componente de intensificare care să ia în considerare poziția curentă (aplicarea un algoritm determinist de coborâre locală, fie el și LDDA) se va face separat de generarea Plbest și simultan pentru toate particulele roiului.*

Algoritm BHTPSOM

```
start
.....
  t←1;
...cat timp condiția de terminare nu este îndeplinita
      pentru i=1,..., N        /* fiecare particula*/
```

Generează $\vec{P}_{lbest_i}$ folosind tehnica descrisă în secțiunea 3.1;

.....

■

pentru *i*=1,..., *N*

$\vec{X}_i \leftarrow$GDD($\vec{X}_i$) /* $\vec{X}_i \leftarrow \vec{X}_i{}'$ */

■

t←*t*+1;

repeta

return *Pgbest*

stop

Lista 3.4 Structura generală a algoritmului BHTPSOM

Din punct de vedere al implementării, se poate vedea în *Lista 3.4* algoritmul BHTPSOM (M de la modificat). *Lista 3.4* este doar scheletul Listei 3.3 la care s-a inserat ciclul încercuit cu linie întreruptă, chiar înainte de trecerea la iterația următoare. Prin acest ciclu, se execută pentru toate particulele algoritmul GDD (geographical deterministic descent).

Procedura GDD($\vec{X}_i$) pornește din poziția $\vec{X}_i$ și are *n* iterații. În fiecare iterație, numai bitul #*d*, *d*=1,…,*n*, este eventual schimbat, dacă aceasta schimbare duce la creșterea funcției fitness. Astfel, se generează o secvență de vectori adiacenți ca în schema următoare:

$$\vec{X}_i \to \vec{X}_i^1 \to \vec{X}_i^2 \to \cdots \to \vec{X}_i^n = \vec{X}_i{}'$$

Unii dintre acești vectori pot fi identici.

Lista 3.5 prezintă o implementare posibilă a acestui algoritm. Valoarea $fitness_i$ este deja calculată în iterația anterioară a algoritmului BHTPSOM.

funcție GDD($\vec{X}_i$)

```
1.   for d=1,...,n
2.      x_i^d ← complement (x_i^d)
3.            val ← EvalFitness(X_i);
4.      dacă (val < fitness_i)
5.          atunci fitness_i ← val;
6.          altfel x_i^d ← complement (x_i^d);   /* revin la valoarea anterioară */
7.            ■
8.   ■
9.   dacă (val < best_i) atunci P_best_i ← X_i
10.                              BEST_i ← val;
11.  ■
12.  dacă (val < gbest) atunci P_gbest ← X_i
13.                              GBEST ← val;
14.  ■
15.  return X_i
```

Lista 3.5 Algoritm determinist de coborâre locală

Funcția *complement*(x_i^d) schimbă valoare bitului x_i^d din 0 în 1, sau vice versa. Dacă valoarea $fitness_i$ nu este îmbunătățită, bitul x_i^d este restabilit la valoarea sa anterioară. Când valoarea $fitness_i$ descrește, bitul complementat este păstrat și GDD va încerca să o micșoreze în

continuare, prin complementarea bitului x_i^{d+1}, ș.a.m.d. Putem considera că această complementare a bitului x_i^d este echivalentă cu mișcarea în direcția "gradientului" pe axa *d*, în raport cu poziția inițială $\vec{X}_i$. Să remarcăm că GDD ia în considerare numai *n* combinații binare adiacente. Schema din *Lista 3.5* poate fi aplicată din poziția inițială $\vec{X}_i$, dar utilizând o altă ordine de tratare a biților ce va produce, în general, alt rezultat. Desigur, dacă poziția curentă $\vec{X}_i$ este un optim local, atunci GDD nu va îmbunătăți valoarea $fitness_i$.

Complexitatea algoritmului BHTPSO se poate calcula ca fiind numărul de apeluri ale funcției obiectiv (i.e. *EvalFitness* în cazul nostru). La iterația *t*, evident, funcția obiectiv este apelată de *N* ori. Dacă ne referim la algoritmul BHTPSOM care integrează algoritmul GDD, complexitatea se ridica la $N+n\cdot N=N\cdot(n+1)$.

Această propunere de integrare a procedurii GDD poate să fie ameliorată, de la caz la caz, prin alte trei modificări posibile.

Modificarea 1: Procedura GDD poate fi apelată în mod recursiv, până când poziția nou generată nu mai generează o ameliorare a funcției obiectiv. Evident, coborârea locală este mai eficientă, dar și mai costisitoare în termeni de număr de apeluri ale funcției *EvalFitness*.

Acest fapt este compensat prin convergenta algoritmului PSO care este mai rapidă în termeni de număr de iterații *t*.

Modificarea 2: Procedura GDD, datorită caracterului sau determinist și asimetric (biții sunt tratați mereu în aceeași ordine fixa) poate introduce fenomenul de deviere (*bias*) în cercetarea spațiului soluțiilor. Acest lucru se poate rezolva renunțând la caracterul determinist al procedurii. Parcurgerea biților se poate face, nu în ordinea $d=1,...,n$, ci se poate genera un vector aleator pentru tratarea celor *n* biți, ce va determina noi relații de adiacență. Se obține o procedura GRD (geographical random descent), pentru care nu mai are sens să se facă un apel recursiv. Testele arată o creștere de eficientă a BHTPSOM fată de varianta fără GRD.

Modificarea 3: Dacă, la iterația *t*, GDD generează poziții locale mai bune și actualizează vectorii de poziție pentru unele particule, la iterația *t*+1, noii vectori de poziție generați de mecanismul BHTPSO pot să nu fie suficient de depărtați de vechile poziții. De aceea, la apelul următor al GDD, noile poziții vor fi atrase de cele anterioare determinate la pasul *t*. Acest fapt arată că apelurile la GDD pot fi ineficiente, în ciuda prețului plătit în termeni de complexitate. Pentru a contracara această situație, nu trebuie apelată procedura GDD la fiecare iterație *t*. După un număr de iterații fără ameliorare a pozițiilor "local best", noii vectori de poziție sunt suficient de depărtați de cei care au generat $\vec{P}_{best_i}\ i=1,\cdots,N$. De aceea există șanse mai mari să se genereze noi vectori "local best". În testele efectuate, am ales să apelăm GDD ori de cate ori numărul de iterații fără ameliorare a pozițiilor "local best" este cel puțin Δ pentru toate particulele, adică avem: $\min_{i=1,...,N}(NF_i)\geq\Delta$ unde Δ este un nou parametru al algoritmului ce trebuie setat.

3.3.3 O problemă de conducere optimală cu comandă discretă binară

In Anexa 4 este prezentat un model neliniar discret în timp, pentru o rețea de colectare a apelor uzate (RCAU), așa cum a fost propus și dezvoltat în (Carp, 2014) și (Mînzu, 2015). Controlul descărcării acestor rețele duce la o problemă de optimizare binară, așa cum este prezentat în aceeași anexă, problemă numită Sewer Network Discharge Optimization Problem (SNDOP). Se consideră rețeaua din figura 3.8.

Datele RCAU sunt: Nr. de bazine (tancuri) $n=10$;

- Perioada de eșantionare: $T_s=120$ sec;

- Orizontul de timp pe care se studiază evoluția RCAU: H=40 (T_s);
- Capacitățile maxime ale bazinelor:

M_1=10 M_2=14 M_3=6 M_4=20 M_5=50 M_6=10 M_7=14 M_8=6 M_9=20 M_{10}=40;

Influentul estimat pentru momentul t și pentru bazinul #i : $D(t, i)$, t=1,...,40 ; i=1,...,10;

În figura 3.9 este ilustrată o predicție pentru debitul de influent din zonele de captare corespunzătoare bazinelor 1 și 2, în cazul unei ploi tipice. În programul BHTPSOM, predicția va fi exprimată în unități de volum (pentru o perioadă de eșantionare), după operațiile de integrare, discretizare și corecție.

Descărcarea RCAU poate fi privită ca o problemă de control optimal binar care să minimizeze deversările totale Q^{over}.

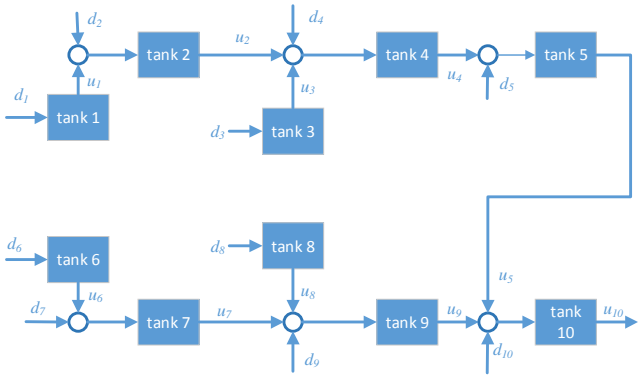


Figura 3.8 Structura unei rețea de colectare a apelor uzate

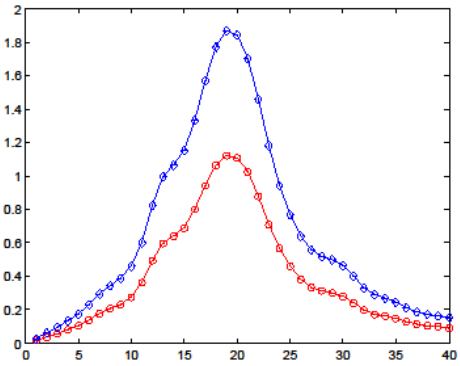


Figura 3.9 Debit influent în zona 1-rosu și 2-albastru

Problema este caracterizată de următoarele elemente:

● Ecuațiile de stare discrete ale RCAU sunt:

$$X(t+1) = f(X(t), U(t)) = \begin{bmatrix} \min(M_1, x_1(t) + Q_1(t) - U_1(t)) \\ \cdots\cdots \\ \min(M_{10}, x_{10}(t) + Q_{10}(t) - U_{10}(t)) \end{bmatrix}$$

unde

$X(t) = [x_1(t)\, x_2(t) \cdots x_{10}(t)]^T$: starea sistemului

$U(t) = [U_1(t)\, U_2(t)\, U_3(t) \cdots U_{10}(t)]^T$: vectorul comenzilor binare

Ecuațiile de stare detaliate pentru fiecare bazin sunt:

$x_1(t+1)=\min(M_1, x_1(t)+ Q_1(t)-U_1(t))$; $\quad Q_1(t) \stackrel{\Delta}{=} D(t,1)$

$x_2(t+1)= \min(M_2, x_2(t)+ Q_2(t)-U_2(t))$; $\quad Q_2(t) \stackrel{\Delta}{=} D(t,2)+U_1(t)$

…….

$x_{10}(t+1)= \min(M_{10}, x_{10}(t)+Q_{10}(t)-U_{10}(t))$; $\quad Q_{10}(t) \stackrel{\Delta}{=} D(t,10)+U_5(t)+U_9(t)$

Ecuația ieșirii este: $y(t) = Q^{\text{effluent}}(t) = U_{10}(t)$

● *condițiile inițiale*: $x_i(t_0) = x_i^0,\ i = 1,\ldots,10;\quad X_0 = [3,3,3,3,3,2,2,2,2,2]$

● *condițiile de frontiera*: $0 \le X_i(t) \le M_i,\ X_i(t) \in \boldsymbol{I},\ i = 1,...,10$; ($\boldsymbol{I}$ - mulțimea întregilor)

● *criteriul de optim*: -

$$Q^{\text{over}} = \min_{U(t)\in U} \left\{ \sum_{t=t_0}^{H-1} [\max(0,\ x_1(t)+Q_1(t)-U_1(t)-M_1)+\cdots+\max(0,x_{10}(t)+Q_{10}(t)-\mathrm{U}_{10}(\mathrm{t})-M_{10})] \right\}$$

Reprezentarea soluției

Pentru algoritmul BHTPSOM, o soluție *x* este vectorul de poziție al unei particule din roi. Pe de altă parte, o soluție trebuie să determine în mod unic traiectoria de stare și deversarea totala Q^{over}, în condițiile în care starea inițială a bazinelor, influentul estimat pe tot orizontul de timp și toți ceilalți parametrii ai RCAU sunt cunoscuți. De aceea o soluție *x* este o secvență a intrărilor de comandă pentru toate bazinele și pentru toate perioadele de eșantionare aparținând orizontului de timp. Dacă presupunem că t_0=1, structura soluției *x* este: $x = [\underbrace{U_1(1)\,U_2(1)\,...U_n(1)}_{t=1} | \cdots | \underbrace{U_1(t_f)\,U_2(t_f)\,...U_n(t_f)}_{t_f=H-1}]$

Se constată că o soluție este un vector binar cu $m = n\times(H-1)$ biți. În cazul nostru, o soluție are *m*=390 biți.

Evaluarea funcției obiectiv

În implementarea noastră, evaluarea funcției obiectiv este realizată de funcția *EvalFitness*(*x*) care simulează evoluția sistemului dinamic (3.12), conform figurii 3.10. Această funcție întoarce valoarea Q^{over}, dar determină și traiectoria de stare i.e. secvența de stări *X*(1), *X*(2), …, *X*(*H*) (ultima intrare de control fiind *U*(*H*-1)).

Parametrii adoptați pentru execuția algoritmului BHTPSOM în vederea rezolvării acestei probleme sunt: `numărul de particule din roi=20; vmax=5.;amax=6.; C1min=0.5; C1max=2.; C2min=1.; C2max=2.; C3min=0.5; C3max=1.5; wmin=0.2; wmax=0.6;`

Pentru cazul soluționat avem `Volumul total de influent=171 (unităţi de volum.)`

Algoritmul converge la iterația *t*=`34` dar cu un număr de apeluri ale f. obiectiv N_{apel}=`266581`.

Comanda optimală găsită generează o deversare totală Q^{over}=0 (unități de volum)

și generează o evoluție a stărilor, dintre care figura 3.11 prezintă doar 5 variabile de stare. Liniile întrerupte marchează capacitățile maxime ale bazinelor.

Pentru o problema de această talie, consideram că rezultatul este remarcabil

Pentru o nouă stare inițială, de exemplu $X_0 = [3,4,2,3,7,2,1,3,6,5]$

comandă optimală este diferită și duce la optimul Q^{over}=1 (unități de volum) și converge într-un număr relativ mare de apeluri ale funcției obiectiv N_{apel}= `446901`. Comanda determinată generează o evoluție a stărilor prezentată în figura 3.12.

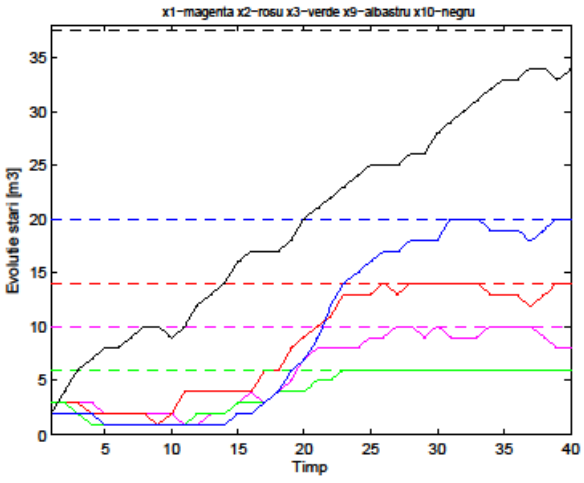


Figura 3.11 Evoluția stărilor determinate de BHTPSO

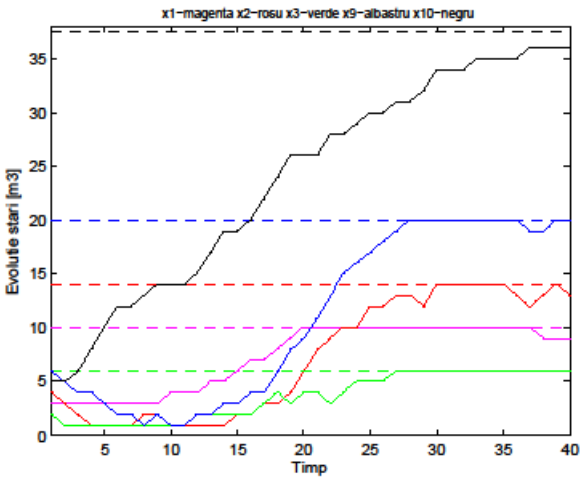


Figura 3.12 Evoluția stărilor pentru noua stare inițială

3.4 Concluzii

Plecând de la cele două variante de bază ale algoritmilor PSO din literatură, a fost descrisă și a treia variantă "standard" a algoritmului PSO, cea care se bazează pe o topologie hibridă - Hybrid Topology Particle Swarm Optimization (HTPSO)-, adică utilizează ambele topologii, globală și locală, în același timp, variantă folosită în această lucrare.

Dacă PCO are și un caracter bilocal cu starea, s-a considerat o funcție obiectiv extinsă. Evaluarea funcției obiectiv, dată fiind natura PCO, se realizează cu ajutorul unui model al procesului considerat.

Aplicarea algoritmului HTPSO la rezolvarea unor PCO continue a fost ilustrată pe trei cazuri: procesul neliniar din reactorul Batch, sistemul liniar cu criteriu Lagrange dar bilocal pe stare și Problema Liniar Pătratică discretă.

Pentru aplicarea PSO la o PCO discretă-binară a fost descrisă pe larg o implementare (cu elemente existente în literatură) de algoritm HTPSO adaptat la cazul binar. S-a adăugat și o accelerație a particulelor pe fiecare componentă, care este legată de o probabilitate de schimbare a valorilor binare din vectorul de poziție. O modalitate de a ameliora intensificarea algoritmului BHTPSO este aceea de a considera, pe lângă vecinătatea sa "socială", și o căutare a unei soluții mai bune într-o vecinătate "geometrică".

Testarea acestui algoritm (BHTPSOM) a fost făcută pe problema de descărcare optimală a unei rețele de colectare a apelor uzate (RCAU) de mari dimensiuni.

Soluțiile obținute au demonstrat realismul acestei abordări, prin caracterul lor optimal sau quasi-optimal, și durata convenabilă a execuției algoritmului.

Contribuțiile din acest capitol sunt:

- Integrarea tehnicii *step control* - care nu corespunde unei restricții specifice PCO - în algoritmului HTPSO pentru a asigura un profil de comandă cu o evoluție continuă și chiar derivabilă, pe un domeniu cat mai larg. Astfel, cel puțin, comanda "optimală" evită salturile puternice;
- S-a propus ca procedura ce determină experiența locală cea mai bună să includă, pe lângă informația "socială", o componentă de intensificare care să ia în considerare poziția curentă (informația "geometrică"). În acest scop, s-a introdus un algoritm de intensificare "geometrică", determinist, de coborâre locală numit GDD (geographical deterministic descent);
- Au fost propuse, de asemenea, trei tehnici de ameliorare a algoritmului GDD, care au vizat:
  - apelul recursiv al GDD, pentru mărirea eficientei;
  - minimizarea fenomenului de deviere ("bias") în căutarea soluțiilor optime;
  - apelarea selectivă, la un anumit număr de iterații fără ameliorare a soluțiilor "local best".

Capitolul 4
Utilizarea algoritmului diferențial stigmergic la probleme de conducere optimală

4.1 Algoritmul ACO. Utilizarea ACO la probleme de optimizare cu parametri continui

Metoda „Optimizare cu colonie de furnici" (Ant Colony Optimization (ACO)(Dorigo, 1995)) a fost dezvoltată pentru a fi utilizată în rezolvarea problemelor de optimizare combinatorică. Avantajul posibilităților de a folosi reprezentări sub formă de graf a spațiului de căutare, de a putea integra în algoritm informații euristice, de a utiliza populații de soluții (ce asigură în general robustețe și evitarea unei convergențe premature) și nu în ultimul rând de a fi ușor de hibridizat, a făcut ca ACO să fie aplicat cu succes în multe probleme din domeniul sistemelor de productică (Șerbencu, 2007), (Șerbencu,2014) și (Șerbencu și Mînzu, 2016).
Mecanismul utilizat de furnicile pentru a transmite informații celorlalți membri ai comunității se bazează pe emiterea și receptarea de feromoni ce sunt depuși în mediu. Coordonarea realizată prin acest mecanism de comunicare poartă numele de comportament stigmergic.
Un algoritm promițător bazat pe structura algoritmului ACO, și care poate fi aplicat problemelor de optimizare cu variabile continue este „Algoritmul diferențial stigmergic" = „Differential Ant-Stigmergy Algorithm" (DASA) propus de Korosec în 2006. În cele ce urmează este realizată o prezentare a DASA, se analizează performanta lui în aplicarea pe PCO și se propun ameliorări ale structurii lui în scopul creșterii performantelor în cazul aplicării la rezolvarea PCO.

4.2 Algoritmul diferenţial stigmergic (DASA)

4.2.1 Forma discretă fin-granulată a domeniului continuu

In cadrul DASA(Korosec 2010), pasul de căutare este discretizat, și nu direct spațiul de căutare. Fiecare direcție de căutare este discretizată folosindu-se *o bază de discretizare* ***b***. Fiecare pas cu care poate fi modificată soluția curentă este calculat sub forma $w \cdot \delta_i$ unde $\delta_i = b^x$ cu $x \in Z$. δ_i este numită *diferență a parametrului* și este aleasă așa cum se detaliază mai jos, iar ω este un factor de ponderare ales ca număr aleator întreg cu valori între 1 și $b-1$. Astfel dacă considerăm p'_i valoarea parametrului *i* din soluția curentă, pe durata căutării valorilor optime ale parametrilor, noua valoare p_i, atribuită parametrului *i* este:

$$p_i = p'_i + \omega\delta_i \tag{4.2}$$

unde δ_i este un element din mulțimea Δ_i de *diferențe* pozitive și negative aplicabile parametrului *i* $\Delta_i = \{\delta^-_{i,k} \mid \delta^-_{i,k} = -b^{k+L_i-1}, k = 1,2,...,d_i\} \cup 0 \cup \{\delta^+_{i,k} \mid \delta^+_{i,k} = +b^{k+L_i-1}, k = 1,2,...,d_i\}$,
cu $d_i = U_i - L_i + 1$. De aceea, pentru fiecare parametru p_i, diferența parametrului, δ_i, ia valori între b^{L_i} și b^{U_i}, iar L_i și U_i se calculează cu: $L_i = \log_b(\varepsilon_i)$, și $U_i = \log_b(\max(p_i) - \min(p_i))$ Parametrul ε_i fixează precizia maximă cu care să fie calculat parametrul p_i și reprezintă un parametru al algoritmului.

4.2.2 Reprezentarea sub formă de graf

Din toate mulțimile, Δ_i, $1 \le i \le D$, unde *D* reprezintă numărul de dimensiuni ale spațiului de căutare, se construiește un graf numit *graf de căutare* G=(***V,E***) cu mulțimea de noduri ***V*** și

mulțimea de arce E intre noduri, conform cu figura 4.1. Fiecare mulțime Δ_i din (4.3) este reprezentată de o mulțime de noduri, $V_i = \{v_{i,1}, v_{i,2}, ..., v_{i,2d_i+1}\}$ și $V = \bigcup_{i=1}^{D} V_i$ (4.6)

În cadrul *grafului de căutare*, fiecare nod din mulțimea $\boldsymbol{V_i}$ este conectat cu toate nodurile care aparțin mulțimii $\boldsymbol{V_{i+1}}$. Se obține astfel un graf orientat, în care orice drum ***v*** de la nodul de start la oricare dintre nodurile de sfârșit are aceeași lungime ***D*** și poate fi definit ca un șir de noduri: $v = (v_1 v_2 \ldots v_D)$ cu $v_i \in V_i, \quad 1 \le i \le D$

Fiecărei mulțimi $\boldsymbol{V_i}$ îi este asociată și o funcție distribuție de probabilitate Cauchy ce va modela feromonul iar utilizarea ei este detaliată în paragraful următor. În figura 4.1 Drumul exemplificat cu albastru în a) corespunde unei posibile eșantionări a distribuțiilor Cauchy asociate din b).

4.2.3 Parametrii DASA

Modul de adaptare a parametrilor distribuției Cauchy reduce, spre finalul căutării, lățimea grafului la doar câteva valori (noduri) dominante (în jurul lui 0), acest lucru determinând și o convergentă în valoare a soluțiilor obținute. Pe de altă parte printr-o alegere adecvată a *bazei discrete* ***b***, se poate controla lățime grafului asociat astfel îmbunătățindu-se de asemenea convergența algoritmului. Valorile recomandate pentru parametrii DASA, în cazul funcțiilor utilizate în analizele realizate în (Korosec, 2010), (Șerbencu, 2011) și (Șerbencu, 2012) sunt utilizate și în testele pe PCO.

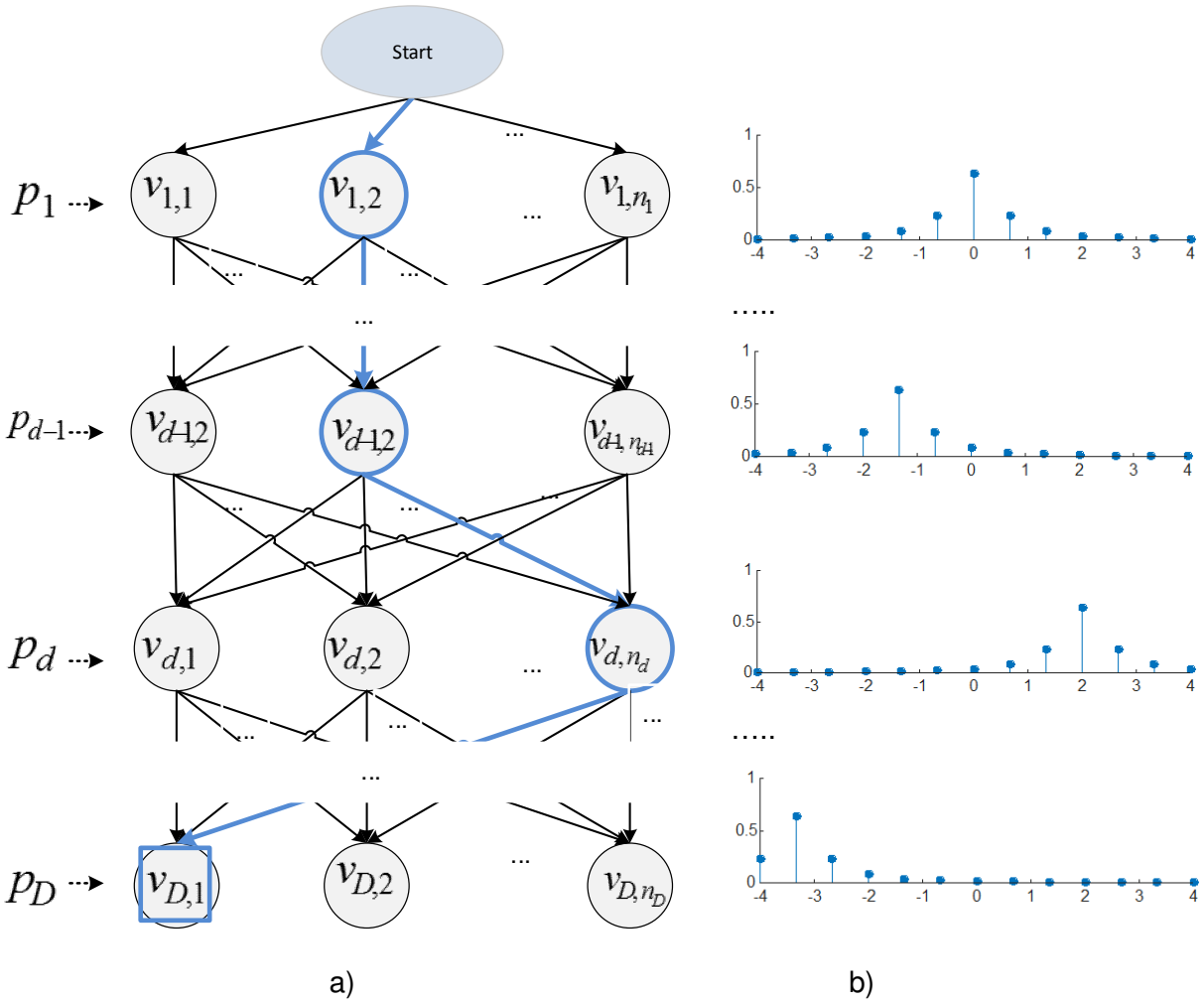


Figura 4.1 a) Graful de căutare asociat unei probleme de optimizare multi-parametru. b) exemplu de distribuții Cauchy asociate

4.2.4 Analiza caracteristicilor DASA

1) Deși bazat pe metafora furnicilor, DASA nu este un algoritm care să genereze direct o populație de soluții. DASA lucrează cu o singură soluție curentă, iar populația de agenți furnică, generează soluții în vecinătatea acesteia.
2) Algoritmul ACO standard poate memora prin feromon variante dispersate „geografic" de componente atractive ce pot fi adăugate la soluția parțială curentă. Prin utilizarea unei singure distribuții Cauchy pentru fiecare direcție de căutare, componentele atractive sunt concentrate în jurul unei singure locații.

   Din aceste prime două caracteristici se poate observa că DASA memorează mai puține informații despre profilul spațiului de căutare comparativ cu AE care memorează o întreagă populație de soluții de calitate, și comparativ cu PSO care suplimentar pe lângă populația de soluții curente mai memorează pentru fiecare particulă cele mai bune soluții traversate și o viteză prezentă.
3) Algoritmul ACO pentru probleme de optimizare combinatorială recomandă integrarea unei informații euristice în procesul de selecție a componentelor de soluție. În forma de bază DASA nu utilizează o astfel de informație. Pentru cazul particular al aplicării la PCO, la care soluția reprezintă o eșantionare a comenzilor ce vor fi aplicate unui proces, un mecanism de tip „step control" poate fi integrat ca informație euristică.
4) Deoarece DASA își controlează intern pasul de căutare în jurul soluției curente, în iterațiile finale ajungându-se, în mod normal, la o căutare cu pași foarte mici în jurul soluției curente, integrarea prin hibridizare a unor metode suplimentare de căutare locală nu este necesară și nici recomandată așa cum se întâmplă în cazul ACO.
5) În cazul în care printr-un pas mare de căutare, se produce o îmbunătățire a soluției curente, dar prin acest pas se traversează în cealaltă parte a văii ce conține un minim ce trebuie exploatat, informația de feromon va scade semnificativ probabilitatea ca în următoarele câteva iterații să se realizeze pași înapoi către vale pe respectiva direcție de căutare. În cadrul unui algoritm de optimizare ce integrează și o strategie de căutare de tip "opositition based" acest aspect al metodei de căutare poate fi adresat direct.

4.3 Aspecte ale aplicării DASA la PCO.

Faptul că DASA are un control direct asupra pasului de căutare, și selecția acestuia afectează în mod direct timpul de execuție al algoritmului prin modificarea numărului de noduri atașat fiecărui parametru optimizat, conduce în cazul PCO la evidențierea următoarei observații.

Observația 4.1: În cazul în care parametrii supuși optimizării reprezintă comenzi ce trebuie aplicate unui proces fizic, și nu simulat, precizia cu care trebuie determinați acești parametri trebuie corelată cu precizia convertorului numeric analogic prezent pe dispozitivul de comandă ce va fi utilizat în conducere. Astfel, dacă se consideră un CNA pe 12 biți, cu el pot fi generate un număr de maxim 4096 comenzi diferite. Pentru un proces ce accepta comenzi în intervalul [-5V,5V] rezultă că este utilă determinarea unei comenzi cu o precizie de 10V/4096= 0.0024V. În funcție de problema rezolvată și algoritmul utilizat o astfel de valoare poate fi considerată de referința în oprirea sau repornirea căutării, sau pentru stabilirea unui pas de discretizare. Suplimentar valorile comenzii pot fi normalizate.

4.4 Aplicarea DASA la PCO

4.4.1 Procesul Batch Reactor - Control optimal cu stare finală liberă și timp final fixat

Considerăm problema „Batch Reactor" prezentată în 1.4.1. Procesul se desfășoară după o reacție de tipul legii lui Arrhenius: $2A \xrightarrow{k_1} B \xrightarrow{k_2} C$. Procesul este comandat prin intermediul profilului de temperatură.
Ca și în cazul ASA, a fost considerată eșantionarea comenzii aplicate *u* în *Nt*=50 eșantioane. Rezultatele obținute la problema Batch-Reactor pentru execuția pe 1000 iterații și 10 furnici (10000 evaluări ale funcției criteriu) sunt cele din figura 4.2 și figura 4.3 . În figura 4.4 este prezentata evoluția relativă a valorii funcției criteriu în cea mai buna soluție găsită în raport cu valoarea la care se oprește căutarea. $(J_k-J_{final})/J_{final}$. Se observă că spre finalul celor 1000 de iterații îmbunătățirea Criteriului scade sub 10^{-5}.

Valoare criteriului pentru soluția găsita este $c_B(t_f) = 0.6107....$ într-un număr de evaluări $N_{eval} = 10000$. Valoarea finală obținută de DASA, mai bună decât cea determinată de HTPSO cu doar 0.05% este rezultatul unui număr mai mare de evaluări ale funcției criteriu. Pentru scăderea numărului de evaluări ale criteriului se propune în secțiunea următoare o metoda de adaptare a pasului de eșantionare a comenzii.

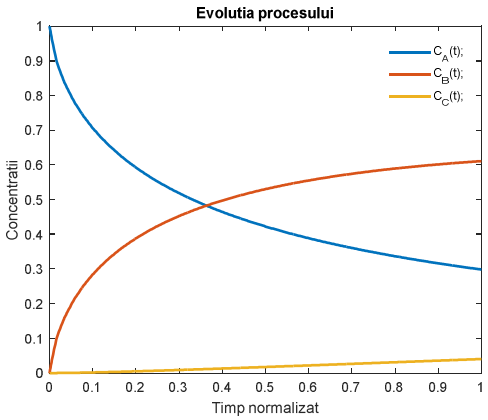


Figura 4.2 Evoluția stărilor. (DASA- Batch Reactor)

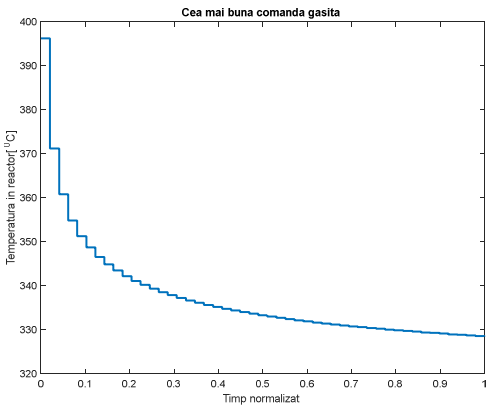


Figura 4.3 Evoluția temperaturii de comanda (DASA – Batch Reactor)

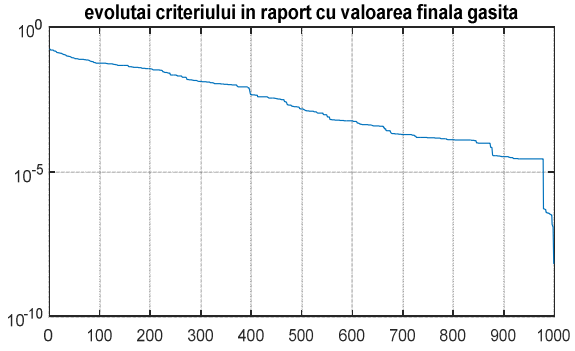


Figura 4.4 Evoluția relativă a valorii funcției criteriu pe durata căutării (J_k-J_{final}) (DASA – Batch Reactor)

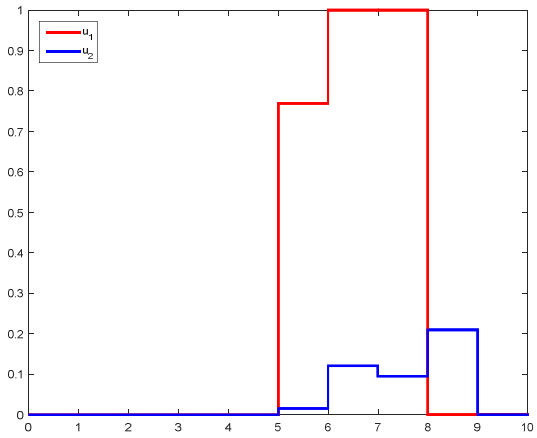


Figura 4.5 Comanda calculată pentru problema reactorului fed-batch în 200 de iterații

4.4.2 Utilizarea DASA la ”Problema de producere fed-batch a proteinelor”

În scopul validării utilizării DASA în probleme de sinteza a unei comenzi în buclă deschisă reconsiderăm problema introdusă în paragraful 2.3.2. În cadrul acestei probleme modelul sistemului are 7 stări x_i i={1,..,7}. Considerăm aceeași inițializare a vectorului de stare și dorim să maximizăm la momentul final cantitatea de proteină produsă cu un consum minim de nutriment și inductor. Comanda este restricționată în intervalul admisibil [0,1]. Aplicând DASA la această problemă și utilizând valorile recomandate pentru parametrii DASA pe doar 2000 de evaluări ale criteriului s-au obținut rezultatele prezentate în figura 4.5. Valoarea obținută de doar 6.0506 poate fi în acest caz satisfăcătoare.

În cazul executării pe 1000 de iterații (tot cu 10 furnici) se obține un rezultat J*= 6.2786. Acest rezultat este semnificativ mai bun decât cel raportat în (Valadi 2014) și pune în evidență abilitățile de căutare locală ale DASA. Justificarea de a obține o comandă cu o precizie de 10^{-15} pentru un sistem de comandă în buclă deschisă este cel mai adesea doar una teoretică.

Evoluția funcției criteriu dea lungul iterațiilor în raport cu valoarea finală la care se oprește algoritmul este data în figura 4.6 iar profilul de comenzi obținute în figura 4.7. După primele 400 de iterații valoarea funcției criteriu nu se mai modifica decât la a 5-cea zecimala, care de multe ori nu se raportează în lucrările din domeniu și nici nu poate fi influențată efectiv de o comandă transmisa printr-un convertor numeric analogic. Astfel putem considera că după cele 400 de iterații algoritmul a convers către o soluție finală (de calitate satisfăcătoare).

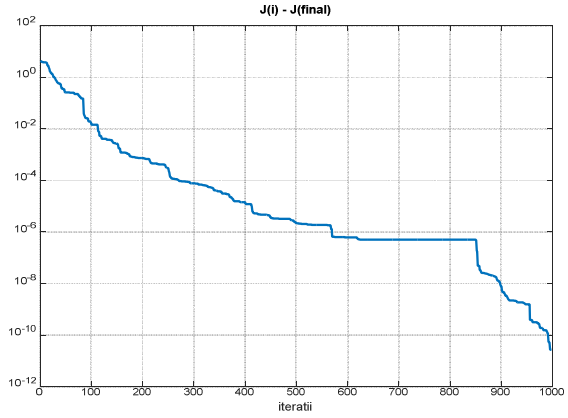


Figura 4.6 Evoluția valorii funcției criteriu în raport cu valoarea finală

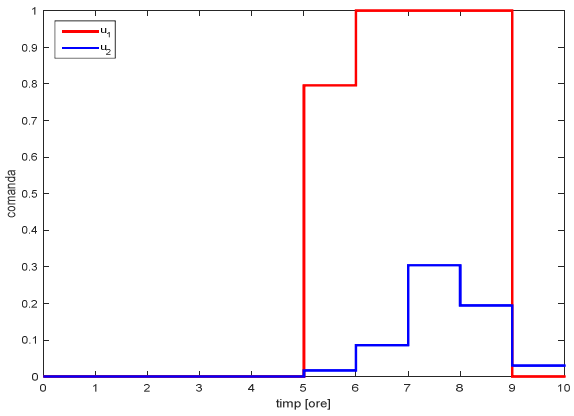


Figura 4.7 Comenzile calculate în 10000 de evaluări ale funcției criteriu.

Performanțe și mai bune s-ar putea obține în cazul în care comanda ar fi eșantionată mai fin, dar formularea problemei în (Valadi 2014) cere determinarea profilului comenzii pe ore. În plus o comandă mult mai potrivită ar fi una în buclă închisă, utilizându-se o structură de control de tipul celei ce va fi introdusă în capitolul 6, însă aceasta ar necesita măsura sau estimarea stărilor reale ale procesului.
Diferența dintre profilul comenzilor din figurile 4.5 și 4.7 se datorează și faptului că funcția criteriu a fost definită doar în raport cu starea finală și nu și în raport cu evoluția comenzii (adică o combinație a consumului de nutriment și substanțe inductoare). Astfel valori similare ale stării finale la momentul t_{final} fixat pot fi accesibile cu comenzi diferite, soluția problemei de optimizare ne fiind în mod necesar unică.

4.5 Contribuţii la ameliorarea performanţelor DASA pentru rezolvarea PCO

Pentru cazul PCO în care spațiul de căutare este reprezentat de eșantioane succesive ale comenzii se propun doua metode de creștere a performanței DASA.

1) Modificarea pasului temporal de eșantionare a comenzii
2) Utilizarea unor variante elitiste ale DASA propuse în (Șerbencu 2011)

4.5.1 Creșterea performanței algoritmului prin redimensionarea spațiului de căutare

Pentru PCO considerate până acum, comanda optimală căutată nu conține componente armonice, fiind cel mai adesea o combinație de componente liniare și exponențiale în raport cu timpul. Pentru aceste cazuri se propune testarea unei reeșantionări cu pas posibil adaptiv a comenzii supuse optimizării, cu scopul de a creste viteza de determinare a unei zone promițătoare din spațiul de căutare. După determinarea unei zone promițătoare comanda determinată este reeșantionată mai fin, procesul putând fi repetat până la obținerea unui pas dorit sau impus de procesul ce urmează a fi condus. Implementarea unei astfel de strategii necesită informații apriori despre ordinul constantelor de timp ale procesului supus optimizării. Efectul aplicării acestei strategii este de creștere a vitezei de convergență prin separarea etapei de explorare de cea de exploatare. În cazul particular al DASA etapa de exploatare poate fi controlată suplimentar prin eliminarea din graf a nodurilor corespunzătoare celor mai mari amplitudini ale pasului de căutare. În continuare se prezintă implementarea acestei strategii prin secvențierea căutării în doar *două etape*. La cazul general secvențierea poate fi realizată în *k* etape.

Secvențierea execuției în două etape.

1) În *etapa 1* se pornește de la o problemă de dimensiune redusă în sensul determinării unei comenzi ce utilizează un pas de discretizare în timp mai mare decât cel impus pentru rezolvarea problemei. După rezolvarea acestei noi probleme și obținerea unei soluții de calitate bună (în raport cu un criteriu de convergență), se trece la *etapa 2*. În *etapa 1* se poate utiliza o precizie mai mică și la nivelul valorilor comenzii, reducându-se astfel numărul de noduri utilizate în cadrul grafului atașat problemei. Acest lucru duce la o modificarea dinamicii algoritmului prin micșorarea probabilității de a alege pași mici de căutare care ar produce îmbunătățiri nesemnificative ale funcției criteriu.
2) *etapa 2.* Soluția de la etapa 1 se reeșantionează cu pasul impus de datele inițiale ale problemei sau cu pasul de eșantionare ce urmează a fi aplicat efectiv în cadrul algoritmului de conducere numerică a procesului. În această *etapă 2*, de rafinare a soluției obținute, se poate crește și precizia în valoare a soluțiilor calculate prin eliminarea nodurile corespunzătoare unor pași mari de căutare, și adăugarea unora de mare finețe. Astfel se obține un comportament intensificator, de căutare locală în jurul soluției generate la *etapa 1.*

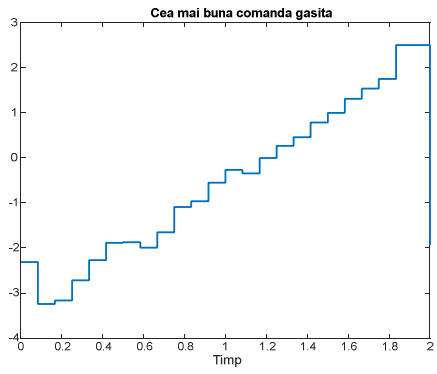
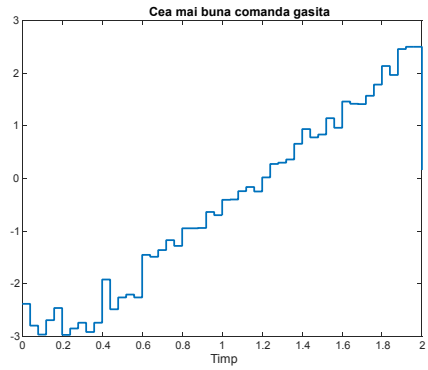
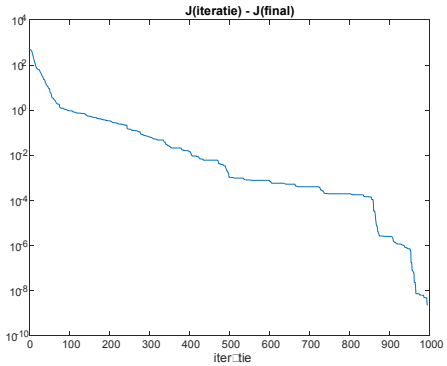
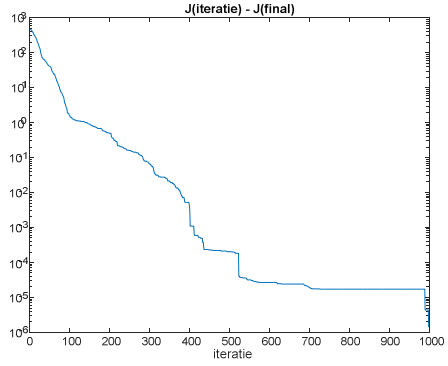
Efectul utilizării unui pas mai mare în *etapa 1* este de a micșora numărul de dimensiuni ale spațiului de căutare. Astfel pentru determinarea unei comenzi cu o precizie dorită, evaluată prin obținerea unei gradient de scădere a valorii funcției obiectiv sub o valoare impusă pentru un număr consecutiv de iterații, va necesita un număr mai mic de iterații. Efectul scăderii preciziei de căutare în *etapa 1* este de a menține ridicată componenta de explorare a spațiului de căutare.

4.5.2 Aplicarea tehnicii de pas variabil de eșantionare a comenzii pe cazul PCO bilocală în raport cu starea

Pentru o analiză a eficientei metodei propuse consideram exemplul de PCO considerat în secțiunea 1.4.2. Aplicăm DASA în varianta lui de bază utilizând o discretizare a orizontului de evaluare a comenzii în 25 și 50 de eșantioane.

Se poate observa ca diferența relativă între valorile funcției criteriu este de aproximativ 0,2%, în favoarea eșantionării comenzii în 25 de pași față de 50 de pași. Faptul că prin eșantionare

cu pas mai mare se poate obține un rezultat mai bun se poate explica prin numărul mai mare de iterații necesare pentru obținerea aceluiași profil al comenzii când trebuie optimizate mai multe eșantioane.

Nr. pași eșantionare	25	50
Comanda u		
J_{final}	3,2311	3,2383
Evoluția criteriului în raport cu valoarea finală		

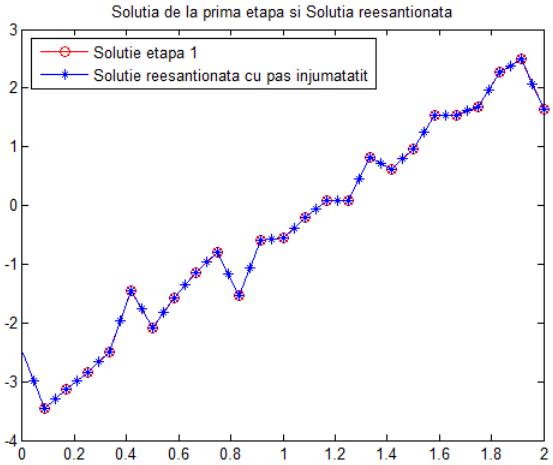


Figura 4.8 Comanda reeșantionată prin înjumătățirea perioadei, folosită ca soluție inițială pentru etapa2

În continuare se prezintă rezultatele unui test în care după 4000 de soluții construite de cele 10 furnici cu un pas de eșantionare de 2/25 secunde, se trece la pasul de eșantionare de 2/50 secunde folosit și cu ASA. Valoarea de 400 de iterații a fost aleasă pe baza evoluției observate a funcției criteriu. Exemplu de comanda reeșantionată prin interpolare liniară a eșantioanelor din soluția etapei 1 este data în figura 4.8 .

În figura 4.9 sunt prezentate doua variante de reeșantionare ale aceleiași comenzi formate doar din 11 eșantioane obținute la etapa 1. În cazul în care în procesul fizic se folosește o comandă constantă în interiorul unui interval de eșantionare, atunci un element de reținere

va fi recomandat să fie utilizat și în procesul de reeșantionare. Altfel, utilizarea unei interpolări de ordinul unu sau superior a comenzilor determinate în prima etapă ar putea aduce îmbunătățiri semnificative asupra calității soluțiilor rafinate în *etapa 2*.

Observație:
În unele teste realizate, rezultate satisfăcătoare au fost obținute și cu un număr mai redus de furnici în fiecare iterație. S-a considerat rezolvarea unei alte probleme cu 5 furnici, menite să crească viteza de convergenta a algoritmului. Reducerea numărului de furnici păstrând nemodificați ceilalți parametri are ca efect o deplasare mai rapida într-o soluție “mai buna” decât cea curentă. Au fost observate și situații în care acest compromis a făcut ca pasul ce a determinat îmbunătățirea și care a actualiza matricea de feromon să nu fie la fel de bun ca cel selectat dintr-un număr dublu de drumuri generate. În general numărul optim de furnici depinde și de profilul spațiului de căutare al problemei supuse rezolvării. Astfel dacă problema este uni-modală, deplasarea se va face în general după o direcție dată de scăderea gradientului pe fiecare direcție de căutare. Dacă în schimb problema prezintă multiple optime locale, utilizarea unui număr de furnici mai mare va favoriza evadarea din bazinele de atracție ale acestora.

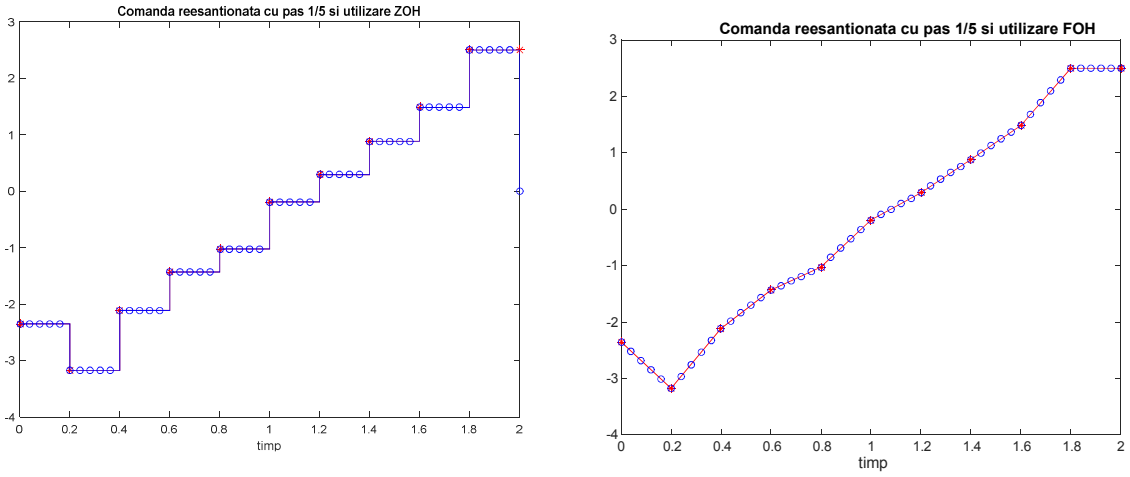


Figura 4.9 Comanda reeșantionată prin micșorarea de 5 ori a perioadei de eșantionare. Durata etapa 1 – 400 iterații

a) Comandă aplicată procesului utilizând interpolare de ordin zero(ZOH).

b) Comandă aplicată procesului utilizând interpolare de ordin unu(FOH).

4.5.3 Ameliorarea convergenței. DASA-*elitist A/B*

În (Șerbencu et al. 2011),(Șerbencu et al. 2012) este propusă ca soluție de creștere a vitezei de convergență a DASA integrarea a trei variante de elitism. Decizia introducerii unui comportament elitist în cadrul algoritmului a fost justificată de observații precum că eșantionarea distribuției Cauchy nu conduce la selecția nodului cu maxim de feromon decât într-un număr de doar 10% din eșantioane pe durata a 1000 de iterații. Prin acest comportament deși sunt favorizate și noduri în jurul celui ce a generat ultima îmbunătățire, această îmbunătățire nu este exploatată într-un mod eficient, având în vedere și variația exponențială a pasului de căutare corespunzătoare la doua noduri adiacente.

Având în vedere această observație, în primele două variante elitiste ale DASA s-a propus folosirea la generația curentă de către una dintre furnici a valorii de vârf a distribuției de feromon. Varianta în care se folosește pasul cu feromon maxim și ponderea *w* este regenerată a fost numită DASA-*eltistA,* iar varianta în care sunt utilizate inclusiv valorile ponderilor care au generat ameliorarea soluției este numită DASA-*eltistB*. Structura

modificată a algoritmului este prezentată în figura 4.10.

In Șerbencu 2011 s-a folosit un set de 6 funcții de test cu parametri continui și s-a testat eficiența unei furnici elitiste comparativ cu una ne-elitistă. S-au utilizat 6 funcții benchmark.

S-a observat că *Furnica-elitistă-B* obține rezultatele cele mai bune. Astfel, în 41,83% din iterații f*urnica-elitistă-B* este cea mai bună a iterației. Dacă celelalte furnici sunt cele mai bune ale iterației în medie într-un procent de (100%-41,83%)/(*m*-1)=6,46% iterații, înseamnă că *furnica-elitistă-B* este mai bună de 41,83/6,46=6,47 ori decât o furnică obișnuită. Prin raportare la varianta DASA standard introdusă de Korosec s-a arătat că varianta DASA-*eltistB* produce o ameliorare ușoară a convergenței doar pentru unele funcții unimodale și multimodale. Testele raportate în (Șerbencu, 2011) au fost realizate pe probleme de dimensiune 100.

4.5.4 Rezolvarea PCO utilizând DASA-*elitist A/B*

Pentru testarea variantelor *elitiste A* și *B*, s-a considerat problema sintezei comenzii pentru PCO bilocală pe stare utilizată și în paragraful 4.5.2. Performanța DASA-*elitist*-A ca valoare a funcției criteriu în soluția finală returnată, este puțin mai bună decât a celui standard obținându-se J_{final}=3,2290, o valoare cu doar 0,06% mai bună. DASA-*elitist-B* produce o performanță similară cu 0,04% mai bună decât DASA standard și un profil al comenzii *u* similar.

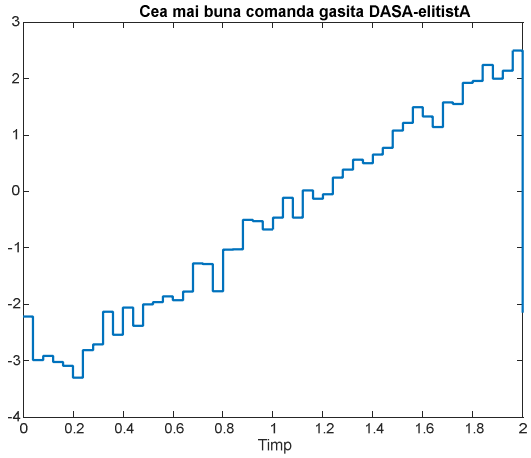


Figura 4.11 Exemplu de comandă determinată utilizând DASA *elitist-A*

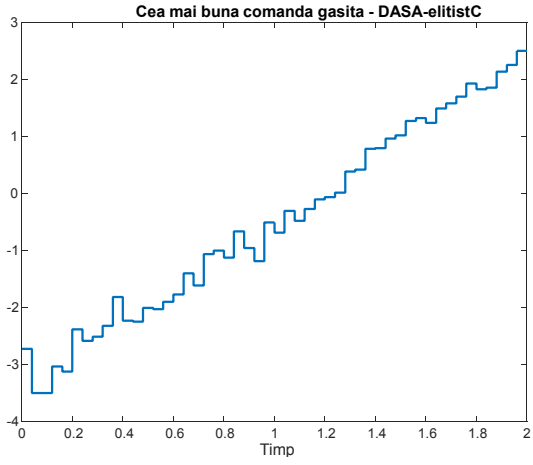


Figura 4.12 Exemplu de comandă determinată utilizând DASA-*elitist-C*

timpul de calcul prin eliminarea eșantionărilor distribuției Cauchy. Această ipoteză nu este însă valabilă în cazul PCO la care evaluarea funcției obiectiv implică cel puțin simularea unui model (de multe ori de tip black-box) al sistemului.

În lucrarea (Șerbencu, 2012) este realizată o comparație între versiunile elitiste ale DASA și PSO. Amândouă metaeuristicile converg spre soluția optimă dar pentru funcțiile considerate, varianta DASA-*elitist-C* are o viteză de convergență superioară.

Cu această variantă de elitism selectând $p_{elitism}$ =0.4 se obține o comandă similară ca performanță, cu o valoare a funcției criteriu de J_{final}=3.2300, o valoare cu doar 0.03% mai bună decât varianta standard. Profilul comenzii determinate este prezentat în figura 4.12.

4.6 Concluzii

În cadrul acestui capitol este utilizat algoritmul „Differential Ant-Stigmergy", care transformă problema de căutare într-un spațiu multidimensional continuu într-o problemă de construire a unui drum într-un graf asociat. Particularitățile algoritmului, precum modul explicit de selectare a preciziei maxime a pasului discret de căutare, sunt analizate în relație cu caracteristicile PCO la care vectorul parametrilor supuși optimizării este reprezentat de eșantioane ale comenzilor aplicabile procesului.

Aplicarea DASA în varianta standard și în două variante ameliorate, la rezolvarea unor PCO continue a fost ilustrată pe trei cazuri: procesul neliniar de tip reactor Batch, problema de producere fed-batch a proteinelor și sistemul liniar cu criteriu Lagrange dar bilocal pe stare. O proprietate intrinsecă a modului de funcționare a DASA face ca acesta să exploateze eficient soluția promițătoare identificată în etapa exploratorie.

Convergența în valoare a soluției identificate de variantele de algoritm propuse este ilustrată prin evoluția criteriului relativ la valoarea finală returnată, evoluții prezentate în figura 4.4 și figura 4.6

Contribuții ale acestui capitol sunt:

- Propunerea unei tehnici de redimensionare a problemei de optimizare rezolvabile de către DASA cu scopul creșterii vitezei de convergență către zonele de bună calitate. Prin rezolvarea într-o primă etapă a unei probleme de dimensiune redusă, se ameliorează probabilitatea algoritmului de a fi prins în optime locale. Soluția problemei de dimensiune redusă este reeșantionată prin interpolare, și se generează astfel o soluție inițială de bună calitate pentru etapa a doua care va rezolva problema inițială.

  Suplimentar, precizia de căutare a fost adaptată celor două etape diferite, în etapa de identificare a zonei promițătoare utilizând-se doar pași "mari", în etapa de intensificare utilizându-se pași corelați cu precizia impusă în rezolvarea problemei inițiale de către parametrii tehnologici a procesului ce urmează a fi condus;

- Propunerea a trei variante elitiste pentru DASA, care promovează utilizarea mai eficientă a informației transmise prin feromon. Primele două variante de elitism propuse sunt echivalente unei încercări de căutare după gradient, în sensul încercării de a reutiliza drumul din graf care a generat ultima ameliorare a soluției.

  Cea de a treia variantă DASA –elitist probabilistic propune utilizarea de fiecare furnică cu probabilitatea $p_{elitism}$ a *diferențelor* ce au ameliorat ultima dată soluția și cu probabilitatea complementară a unor *diferențe* eșantionate conform distribuției Cauchy utilizate de DASA.

Amândouă propunerile de ameliorare a vitezei de convergență au condus la o ameliorare a soluției găsite.

Capitolul 5
Contribuții la conducerea în buclă închisă a proceselor, utilizând optimizarea prin metaeuristici

5.1 Necesitatea unei structuri de conducere în buclă închisă a proceselor optimizate

Cele prezentate în capitolele anterioare demonstrează că diverse PCO pot fi rezolvate prin algoritmi care implementează metaeuristicile respective, ducând la soluții care, dacă nu sunt chiar optimale, sunt foarte aproape de cele optimale. Problema care se pune nu este legată de optimalitatea exactă a soluțiilor obținute, ci de faptul că avem nevoie de o metodă care să ne permită conducerea procesului în buclă închisă. Această metodă trebuie să pornească de la algoritmul generat de metaeuristică, dar să genereze o structură ca cea din figura 5.1, unde $I(t_0)$ este indicatorul de performanță (fără a evidenția toate datele de care depinde calculul acestuia).

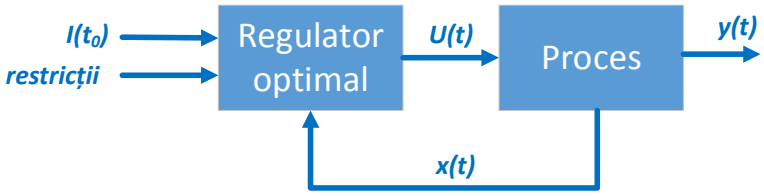


Figura 5.1 Sistem de conducere cu regulator optimal

Să observăm următoarele aspecte:

- Luăm în considerație, în general, procese lente, care ne permit să alegem perioade de eșantionare pentru conducerea lor numerică de ordinul a cel puțin câteva minute;
- Considerăm că stările procesului sunt citibile sau accesibile, lucru care este adevărat în sistemele tratate ca exemplu în capitolele anterioare;
- Regulatorul optimal trebuie să execute algoritmul generat de metaeuristică pentru a determina profilul de comandă "optimală" ce este soluție a PCO definita de <proces; H ; $I(t_0)$; restricții> pe un orizont de timp $[t_0, t_f]$, bine stabilit;
- Într-o implementare numerică, perioada de eșantionare T trebuie să permită regulatorului optimal să rezolve PCO pe un orizont de timp $h \cdot T$, i.e. pe un număr de h perioade de eșantionare.

5.2 Structură de conducere utilizând controlul predictiv

În cele ce urmează, adaptăm conceptul de control predictiv la conducerea în buclă închisă folosind un algoritm ce rezolvă o PCO pe baza unei metaeuristici. Precizăm că este vorba de o adaptare a acestui concept și nu o utilizare integrală a teoriei referitoare la controlul predictiv, dezvoltată în spatii cu bune proprietăți de regularitate. Utilizăm doar caracteristicile general acceptate ale acestui concept. În figura 5.2 este prezentată structura generală a sistemului de conducere propus cu control predictiv, așa cum este descrisă în lucrarea (Lazăr, 1999), și care subliniază caracteristicile acestui concept.

Elementele principale ale metodologiei controlului predictiv sunt prezentate în continuare.

■ La momentul prezent $t=k\cdot T$, vectorii de ieșire $y(k+i)$ cu $i=1,...,H$ sunt predictați, H fiind orizontul de predicție. În cazul nostru, considerăm că vectorul ieșirilor $y(t)$ coincide cu vectorul de stare $x(t)$.

În cele ce urmează vom folosi următoarele notații:

- $x(k+i|k)$, $i=1,...,H$: valorile vectorilor de stare predictate pe baza datelor disponibile până la momentul k;
- $U(k+i|k)$, $i=1,...,H-1$: valorile comenzilor din scenariul viitor de control, stabilite pe baza datelor disponibile până la momentul k.

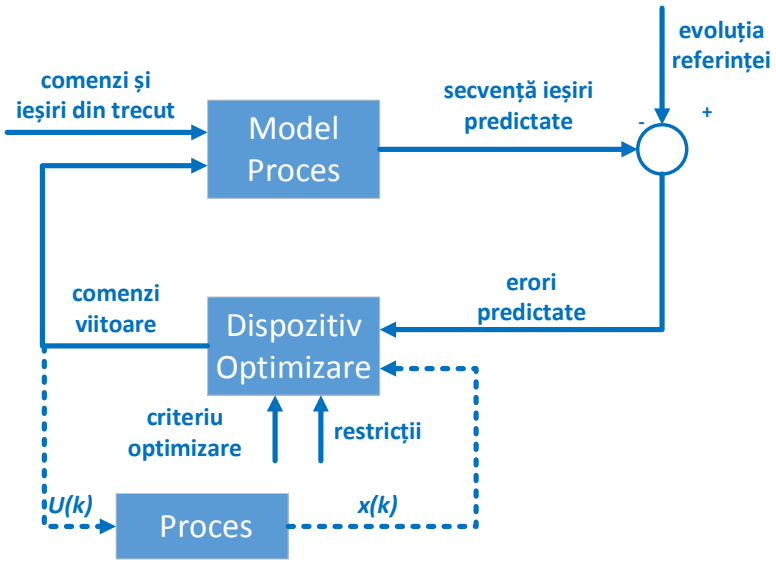


Figura 5.2 Structura generală a sistemului de conducere cu control predictiv

Predicția este realizată utilizând un model al procesului. Valorile predictate $x(k+i|k)$, $i=1,...,H$ depind de valorile $U(k+i|k)$, $i=1,...,H$-1, care se vor aplica începând cu momentul $t=k\cdot T$. Să observăm că:

$$U(k+i|k) \neq U(k+i), \quad k>0\ ,\ i=1,...,H\text{-}1,$$

pentru că $U(k+i|k)$ este o valoare a unei comenzi viitoare, determinată la momentul prezent $t=k\cdot T$, în timp ce $U(k+i)$ este valoarea reală a unei comenzi viitoare la momentul $t=(k+i)\cdot T$, care în mod evident nu este cunoscută la momentul prezent.

■ Primul element din șirul de comenzi optimale (adică din "profilul de comandă "), $U(k|k)$, este aplicat procesului real ca și comandă curentă - în figura 5.2 este notat cu $U(k)$.

Celelalte valori din șirul de comenzi optimale sunt practic uitate, pentru că la următoare perioada de eșantionare, $(k+1)T$, valoarea reală a stării, $x(k+1)$, va diferi de cea predictată, $x(k+1|k)$, și întreaga procedură va fi repetată. O nouă comandă va fi calculată, $U(k+1|k+1)$, a cărei valoare, în general, va fi diferită de cea predictată la momentul curent, $U(k+1|k)$. Pentru că șirul de comenzi optimale este calculat la fiecare perioadă de eșantionare, pentru un orizont de timp H, spunem că abordarea prin control predictiv are un orizont alunecător.

■ Minimizarea erorii de predicție

Valorile $R(k+i|k)$, $i=1,...,H$, definesc referințele pe orizontul de predicție, în ideea că vectorul de stare va evolua de la $x(k)$, $x(k)=\begin{bmatrix}x_1(k) & \cdots & x_n(k)\end{bmatrix}^T$,

către referința sa $R(k)$: $R(k)=\begin{bmatrix}r_1(k) & \cdots & r_n(k)\end{bmatrix}^T$.

Fie Π problema de conducere optimală a cărei soluție dorim să o implementăm prin structura prezentată în figura 5.2. Problema Π poate fi oricare dintre problemele formulate și rezolvate, cu titlu de exemplu, în capitolele anterioare. În general, o PCO este definită de următoarele elemente:

$$\Pi = <f, \text{restricții}, t_0, H, x(t_0), I(t_0, H, x(t_0))>, \tag{5.1}$$

unde f este funcția din ecuațiile de stare ale sistemului dinamic (ecuația (1.5))

$$\frac{dx}{dt} = f(x(t), y(t), u(t), p, t)$$

iar $I(t_0, H, x(t_0))$ este indicatorul de performanță calculat pe intervalul $[t_0, t_0+H]$, din starea inițială $x(t_0)$. Soluția teoretică optimală a problemei Π este un șir de valori ale comenzii care extremizează indicatorul de performanță, în condițiile care definesc problema Π. Este ceea ce am numit profil de comandă optimală:

$$U^*(t_0), U^*(t_1), \cdots, U^*(t_{H-1}); \tag{5.2}$$

Ca urmare, profilul de stare optimal este:

$$x(t_0), x^*(t_1), \cdots, x^*(t_{H-1}), x^*(t_H); \tag{5.3}$$

Fiecare din stările din profilul de stare optimal este de fapt un vector de n componente corespunzătoare stărilor din vectorul de stare:

$$x^*(t) = [x_1^*(t) \quad \cdots \quad x_n^*(t)]^T$$

Pentru a face referire la figura 5.2, vom considera $t_0=k$. Deci profilul de stare optimal este

$$x(k), x^*(k+1), \cdots, x^*(k+H-1), x^*(k+H), \tag{5.4}$$

ca urmare a aplicării profilului de comandă optimală

$$U^*(k), U^*(k+1), \cdots, U^*(k+H-1). \tag{5.5}$$

Să remarcăm că, în secvența (5.4), $x^*(k)$ este de fapt $x(k)$.

Pentru a reface traiectoria optimală, vom considera că aplicăm următoarele referințe:

$$R(k+i|k) = x^*(k+i), \text{ i=1,..., } H \tag{5.6}$$

sau detaliind pe cele n componente, avem

$$\begin{aligned} &r_1(k+i|k)=x_1^*(k+i); \\ &r_2(k+i|k)=x_2^*(k+i); \qquad \text{cu } i=1,\ldots, H \\ &\ldots\ldots\ldots\ldots \\ &r_n(k+i|k)=x_n^*(k+i); \end{aligned} \tag{5.7}$$

■ Șirul de comenzi optimale, $\{U(k+i|k),\ i=0...H-1\}$,

este calculat pe orizontul de predicție, prin minimizarea unui criteriu de optim depinzând de eroarea de predicție.

*În propunerea noastră, regulatorul optimal conține un algoritm **A** ce implementează o metaeuristică care rezolva problema Π*. Doar că acest algoritm nu garantează că, în timpul disponibil, se obțin pofilele de comandă și de stare optime teoretic. Fie profilul de stare obținut prin predicție cu acest algoritm, la momentul k:

$$x(k), x(k+1|k)), \cdots, x(k+H-1|k)), x(k+H|k)), \tag{5.8}$$

Evident $x(k)$ nu este o predicție, ci starea inițială a secvenței. Eroarea de predicție este, deci:

$$\varepsilon(k+i) = x(k+i|k) - R(k+i|k) = \begin{bmatrix} x_1(k+i|k) - x_1^*(k+i) \\ x_2(k+i|k) - x_2^*(k+i) \\ \cdots \\ x_n(k+i|k) - x_n^*(k+i) \end{bmatrix}, \text{ cu } i=1,..,\ H \qquad (5.9)$$

Prin esența metodei, algoritmul ***A*** încearcă să minimizeze eroarea de predicție. Când lucrează bine și are timp să conveargă către soluția optimă teoretic, șirul (5.8) este identic cu șirul (5.4) și atunci, din (5.9) rezultă $\varepsilon(k+i) = 0;$ cu i=1,.., H. (5.10)

În acest caz, prima comandă este $U(k|k) = U^*(k)$ (5.11)

Putem conchide că sistemul cu predicție din figura 5.2 este echivalent cu sistemul de conducere din figura 5.3. În sprijinul acestei afirmații, facem următoarele precizări:

- Algoritmul ***A*** este inclus în Regulatorul optimal pentru a rezolva problema

  Π =<f, restricții, t_0, H, $x(t_0)$, $I(t_0, H, x(t_0))$>

  pe un orizont alunecător H;
- "Modelul procesului" este de asemenea inclus în regulator. Algoritmul ***A*** utilizează acest model pentru evaluarea funcției obiectiv;
- Regulatorul reține doar prima valoare din profilul de comandă, $U(k|k)$, și o trimite ca valoare $U(k)$ către proces;
- Restul valorilor predictate din profilul de comandă sunt uitate, pentru că, la următoarea perioadă de eșantionare, se generează un nou profil de comandă care va începe cu $U(k+1|k+1)$, profil calculat pe baza noii stări reale $x(k+1)$ și tot pe un orizont de H perioade de eșantionare.

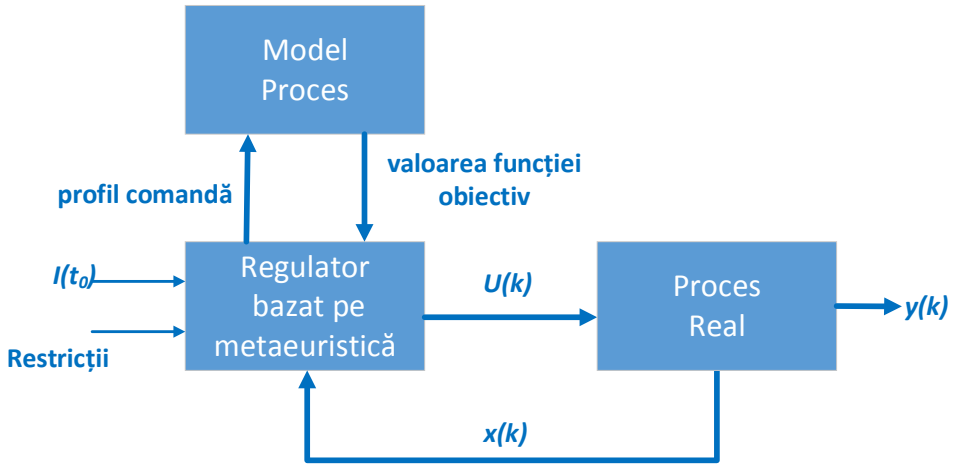


Figura 5.3 Sistem de conducere cu regulator bazat pe o metaeuristică

■ Orizontul de timp de conducere al procesului

Orizontul de conducere al procesului este, în general, diferit de orizontul alunecător $H \cdot T$ - sau H, dacă ne referim la numărul de perioade de eșantionare - care este parametru al algoritmul ***A***.

- Dacă orizontul de conducere este neprecizat, atunci la fiecare moment de eșantionare se apelează algoritmul ***A*** pentru orizontul alunecător H.

- Dacă orizontul de conducere este precizat, vom considera că are lungimea $H_c \cdot T$, cu $H_c > H$. Atunci, în primele $H_c - H$ perioade de eșantionare, se apelează algoritmul ***A*** pe orizontul *H,* iar pentru ultimele *H*-1 perioade, orizontul alunecător este descrescător *H*-1, *H*-2,..., 1. Să observăm că, în total, se apelează algoritmul ***A*** de H_c-1 ori.

Complexitatea calculatorie în general mare a algoritmul ***A*** ne obligă să facem un compromis și să alegem o valoare *H* care să asigure încadrarea sa într-o perioadă de eșantionare.

Observație 5.1:

Algoritmul ***A*** , deși caută valorile optimale, valori ce ar permite regulatorului să asigure o anulare a erorii de predicție, nu le atinge întotdeauna. Acest lucru, în sine, nu este grav, pentru că, finalmente, doar prima comandă din profilul de comandă determinat este luată în considerație și trimisă, ca valoare curentă de comandă, către procesul real. Întrebarea este dacă, procedând astfel, structura de control asigură un comportament al procesului quasi-optimal satisfăcător, pe un orizont de timp mai lung. Considerăm că structura propusă este validă pentru că:

- În condițiile în care nu dispunem de o altă lege de reglare optimală implementabilă, fie datorită proprietăților sistemului, fie datorită complexității numerice a algoritmului rezultat, structura de conducere propusă are un caracter realist prin utilizarea algoritmului ***A*** care "sparge" complexitatea PCO și oferă posibilitatea conducerii în buclă închisă;
- Regulatorul propus, bazat pe metaeuristică (RBM), ține cont de starea curentă a procesului, condiție necesară rejectării unor perturbații din proces;
- Analiza noastră prin simulare a arătat, pe procesele studiate, un comportament quasi-optimal satisfăcător, așa cum se arată în secțiunile următoare.

5.3 Controlul în buclă închisă al sistemelor cu criterii de optimalitate de tip Lagrange

Cele de mai sus pot fi implementate ca atare prin structura de conducere propusă, adică utilizând un algoritm ***A*** și un orizont alunecător *H,* dacă criteriul de performanță are forma de mai jos

$$\min_{x(t),\,y(t),u(t),\,p,t_f} \int_{t_0}^{t_f} L\big(x(t),\,y(t),u(t),\,p,t\big)dt\,, \tag{5.12}$$

adică are numai componentă Lagrange, fără a avea o componentă ce depinde de ieșirea sau starea la un moment final (fixat sau liber), sau au un caracter bilocal. Pentru a exemplifica abordarea considerăm problema SNDOP (Mînzu & Șerbencu, 2016).

Exemplu: Descărcarea optimală a RCAU

Pentru o RCAU având $n \le 5$, o abordare realistă este aceea de a utiliza o structură de control optimal ca cea din figura 5.4. Această abordare a fost prezentată în lucrarea (Mînzu, 2015a) și rezolvă exact PCO printr-un algoritm de programare dinamică discretă. Orizontul de optimizare este de ordinul orelor, de exemplu *N*=40 și *T*=240 sec.

Procesul din RCAU este sub influența influentului real $\boldsymbol{D}_{\text{real}}(t)$ și vectorul curent de stare este trimis regulatorului optimal care va stabili o intrare de control $U(t)$ adecvată. Regulatorul va rezolva PCO definită de stare inițială $x(t)$, șirul de valori ale influentului estimat și orizontul de timp stabilit. Va determina, deci, șirul optimal al intrărilor de comandă, adică "profilul de comandă ". Astfel, intrarea de comandă $U(t)$ adecvată este valoarea primului element din

"profilul de comandă ", adică chiar comanda optimală teoretic cu care începe un proces optimal de lungime *H*.
Profilul de comandă trebuie calculat într-o perioada de eșantionare, ceea ce este o restricție crucială pentru această abordare.
Structura de control a fost testată cu algoritmul de programare dinamică discretă inclus în regulator și, așa cum rezultă din articolul mai sus menționat, comportamentul a fost foarte bun.

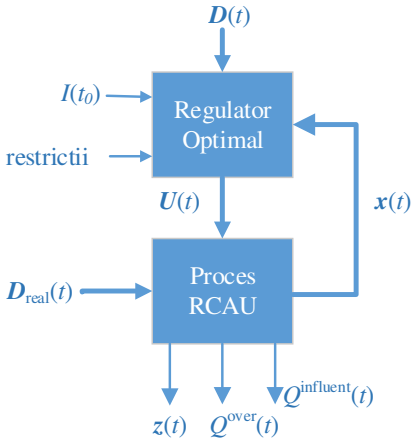


Figura 5.4 Control optimal pentru o RCAU cu n ≤ 5

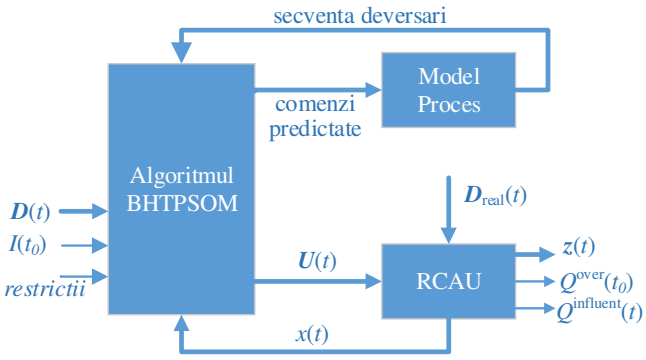


Figura 5.5 Structura de conducere propusa pentru RCAU

În cazul general, când numărul de bazine este mai mare decât 5, e posibil ca algoritmul de optimizare ***A***, bazat pe metaeuristica BHPSO, să necesite mai mult decât o perioadă de eșantionare, pentru a determina comenzile optimale pentru întregul orizont de conducere. De aceea, vom folosi, în cele ce urmează, structura de conducere predictivă prezentată în secțiunea anterioară, adaptată la RCAU de mari dimensiuni.
Se poate arăta că Sewer Network Discharge Optimization Problem (SNDOP) este NP-dificilă. Complexitatea problemei depinde exponențial de numărul de bazine *n*. Instanțe ale acestei probleme cu diferite structuri și dimensiuni au fost rezolvate cu ajutorul algoritmului BHTSPOM (i.e. algoritmul ***A***) obținându-se soluții foarte bune. Complexitatea originară a problemei este modificată de acest algoritm, în sensul diminuării ei, pentru că este un algoritm de aproximare a soluțiilor optime, și de aceea poate să rezolve SNDOP de mari dimensiuni.

În cele ce urmează, prezentăm rezultatele simulării structurii de control aplicată procesului, considerat ca exemplu de RCAU, din secțiunea 3.3.3, unde n=10. În această simulare perioada de eșantionare este T=120 sec. și un orizont de control de 40·T, H_c=40, adică simularea urmărește evoluția pe un orizont de control de 4800 sec. Regulatorul din structura de conducere include algoritmul BHTSPOM (pe post de algoritm ***A***) așa cum rezultă din Anexa 3. Tot de aici rezultă și Modelul Procesului sub formă de program

Prima etapă de simulare

Pentru a face o analiză a impactului orizontului de timp alunecător (H < 40) asupra indicatorului de performanță pe tot orizontul de control, structura de control a fost simulată cu diferite valori ale lui H și considerând

$$\boldsymbol{D}_{\text{real}}(t) = \boldsymbol{D}(t).$$

Rezultatele sunt prezentate în tabelul 5.1 (Mînzu & Șerbencu, 2016). Datorită caracterului nedeterminist al algoritmului BHTSPOM, structura de control a fost simulată prin 30 de rulări independente, pentru fiecare valoare a lui H. În simulările făcute am utilizat o aceeași stare

inițială, $x_0=[3,3,3,3,3,2,2,2,2,2]^T$, la începutul orizontului de control. Simularea furnizează deversarea totală minimă găsită la sfârșitul orizontului de control, notată $Q^{over}(t_0)$ în figura 5.5. În cele 30 de simulări independente, deversarea are valori diferite, deși apropiate. De aceea, tabelul 5.1 indică, în ultimele sale 2 coloane Q_{min} și Q_{max}, adică limitele valorii $Q^{over}(t_0)$.

O singură simulare a structurii de control din cele 30 implică apelul algoritmului BHTPSOM de 39 ori (adică H_c-1). De exemplu, pentru cazul H=11, în primele 29 perioade de eșantionare, algoritmul BHTSPOM este apelat cu parametrul H=11, iar în ultimele 10 perioade de eșantionare, H ia pe rând valorile 10, 9,...,1.

H	$n\cdot(H$-1)	Q_{min}	Q_{max}
6	50	11	14
11	100	7	10
16	150	3	5
21	200	3	4
40	390	1	2

Tabel 5.1 Influența orizontului H asupra deversării minime pe orizontul de control

Șirul de comenzi este utilizat de modelul procesului pentru a determina evoluția predictată a RCAU, adică stări și deversări. Numai valorile $U_1(t_0)$, $U_2(t_0)$,...., $U_n(t_0)$ - evident, t_0 este momentul curent în cadrul simulării - sunt trimise efectiv fiecărui bazin ca valoare reală a comenzii.

Ultima linie din tabel corespunde unui caz particular, când valoarea lui H este egală chiar cu orizontul total de control H_c. De aceea, algoritmul BHTPSO este apelat o singură dată și produce profilul de comandă. *S-a constat că, numai în acest caz, calculul secvenței "optimale" durează mai mult de o perioadă de eșantionare (120 sec.)*. După acest calcul, se trece la simularea structurii de control. La fiecare pas de eșantionare, se extrag din secvența "optimală" primele 10 valori ce sunt trimise către cele 10 bazine ca valori efective ale comenzilor. Simularea furnizează Q_{min}=1 care poate fi considerată ca valoare optimală a indicatorului de performanță pentru SNDOP cu H=40.

Observația 5.2

> Pentru celelalte valori $H \neq H_c$ din tabel, structura de control duce la valori $Q^{over}(t_0) > 1$, pentru că, prin adoptarea unui orizont alunecător, caracterul optimal al profilului de comandă este alterat.

Analizând tabelul 5.1, considerăm ca, în cazul H=16, avem un bun compromis între complexitatea calculatorie a BHTPSO - ce asigură terminarea execuției într-o perioadă de eșantionare - și valoarea finală $Q^{over}(t_0)$. De aceea, am adoptat această valoare pentru a doua etapă a simulărilor efectuate, cele cu "influent real".

A doua etapă de simulare

În această etapă de simulare, RCAU este sub influența influentului real, $\boldsymbol{D}_{real}$, și de aceea valorile stărilor și ale deversărilor vor fi diferite de cele predictate. Dar, la următoarea perioadă de eșantionare, valorile stărilor luate în considerare de regulator vor fi cele reale (din proces), nu cele predictate. Rețeaua reală de Colectare a Apelor Uzate este simulată utilizând același Model de Proces utilizat de Regulator, dar folosind ca matrice de influent pe $\boldsymbol{D}_{real}(t)$, și nu matricea $\boldsymbol{D}(t)$. Deci, perturbația este cea reală.

Pentru simulare, am creat o perturbație reală a.î. $\boldsymbol{D}_{real}(t) = \boldsymbol{D}(t)+\Delta\boldsymbol{D}$,

unde $\Delta \boldsymbol{D}$ este o variabilă aleatoare uniform distribuită în intervalul [0, *L*], valoarea *L* fiind limita superioară a abaterii de la influentul estimat. În testele noastre, am considerat că *L* este 10% din valoarea medie a influentului estimat, pentru fiecare bazin. Și influentul real este supus operațiilor de discretizare și corecție. În simulările făcute am utilizat aceeași stare inițială, ca în prima etapă, la începutul orizontului de control.

Figurile 5.6 și 5.7 ilustrează evoluția volumului de influent către bazinul #2, în ambele cazuri, cel estimat și cel real respectiv. În aceste figuri, graficul albastru arată evoluția continuă a influentului estimat în m^3/s, în timp ce cel roșu și cel verde arată respectiv evoluția după acțiunile de corectare, discretizare și cuantizare. Acțiunea de discretizare poate duce la o pierdere de influent și, de aceea, este necesară o corecție pentru a evita această pierdere. În cazul simulării, influentul estimat pe orizontul de control are un volum total de 171 *ud* (*ud*= unitate de discretizare=12 m^3), în timp ce influentul real are un volum total de 180 *ud*. Deci, influentul real este mai abundent în intervalul 15·*T*-25·*T*.

Figurile 5.8 și 5.9 arată evoluția stărilor predictate și respectiv reale. În linii mari, dinamica variabilelor de stare este practic aceeași, fapt ce arată robustețea structurii de control propuse. Deversarea totală (i.e. suma deversărilor corespunzătoare fiecărui bazin și pentru toate perioadele de eșantionare) în cazul procesului predictat este de 3 *du* în timp ce deversarea reală este de 6 *du*.

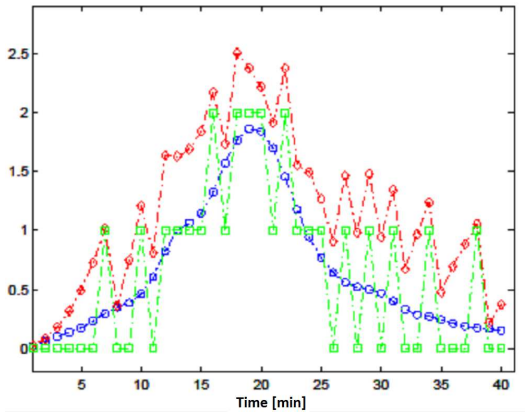


Figura 5.6 Evoluția influentului estimat

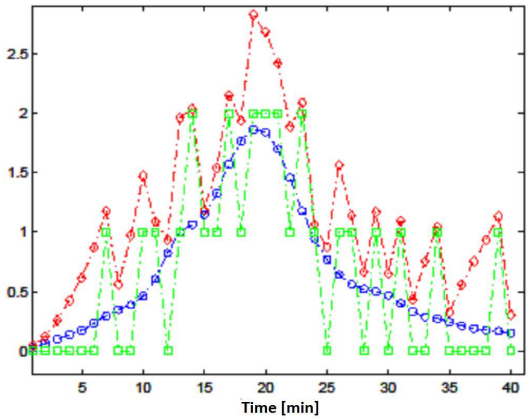


Figura 5.7 . Evoluția influentului real

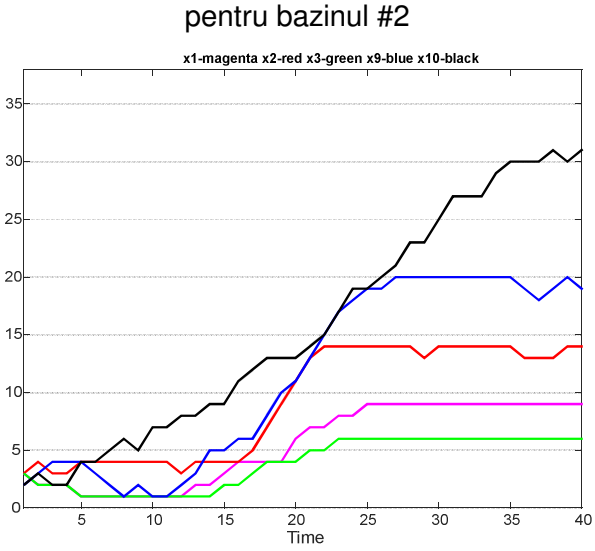


Figura 5.8 Evoluția stării predictate

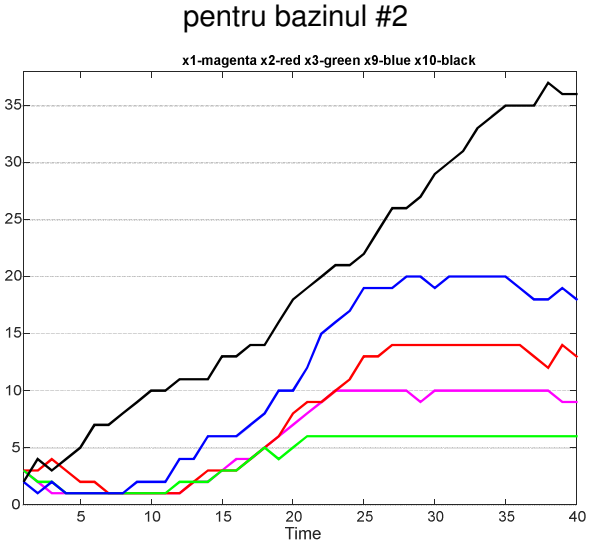


Figura 5.9 Evoluția stării procesului real

În cazul procesului predictat, adică în care influentul $\boldsymbol{D}_{\text{real}}(t)$ este identic cu $\boldsymbol{D}(t)$, valoarea $Q^{\text{over}}(t_0)=3$, adică mai mare decât valoarea minimă considerată $I(t_0, H_c, x_0)=1$, se explică prin observația 5.2. În același timp, structura de control generează un proces real acționând direct asupra RCAU aflată sub influența influentului real $\boldsymbol{D}_{\text{real}}(t)$. Faptul că influentul este mai bogat, 180 *ud* față de 171 *ud*, explică valoarea deversării totale $Q^{\text{over}}(t_0)=6$.

Această etapă de simulare arată că structura de control propusă poate asigura o conducere în buclă închisă cu performanțe bune. Alături de alte simulări, pe aceeași RCAU, cu alte stări inițiale și alte valori ale lui *H*, constituie argumentul cel mai important pentru observația 5.1.

5.4 Controlul în buclă închisă al sistemelor cu criterii de optimalitate de tip Bolza

Dacă criteriul de performanță are forma de mai jos

$$\min_{x(t),y(t),u(t),p,t_f} M\left(x(t_f), y(t_f), u(t_f), p, t_f\right)+\int_{t_0}^{t_f} L\left(x(t), y(t), u(t), p, t\right)dt\,, \qquad (5.13)$$

adică are nu numai o componentă Lagrange, ci și o componentă ce depinde de ieșirea sau starea sistemului la momentul final (fixat sau liber), sau are un caracter bilocal, ideea utilizării unei ferestre alunecătoare nu mai poate fi aplicată, căci momentul final nu este cuprins în orizontul alunecător. De aceea, structura din figura 5.3 poate fi utilizată dar cu o adaptare a orizontului de rezolvare a PCO, la fiecare perioadă de eșantionare. Presupunem că starea curentă a procesului real este accesibilă sau poate fi estimată. În utilizarea acestei structuri, ținem cont de următoarele aspecte:

- Considerăm că orizontul de conducere al sistemului, notat $[t_0, t_f]$, are *n* perioade de eșantionare, 1, 2, ..., *n*;
- Soluția PCO înseamnă determinarea unui șir quasi-optimal $u_1, u_2,..., u_n$ ce constituie profilul de comandă;
- În prima perioadă de eșantionare, plecând din starea inițială (cunoscută sau estimată), algoritmul ***A*** determină un șir $u_1^1, u_2^1, \cdots, u_n^1$ care este o soluție cât mai bună pentru PCO, pe orizontul H_1=1, 2,...,*n*. Calitate acestei soluții depinde de mărimea perioadei de eșantionare, pentru că algoritmul ***A*** trebuie să aibă timp suficient ca să se apropie cât mai mult de soluția optimală. Comanda regulatorului se adoptă ca fiind: $\boldsymbol{U}(1)=u_1^1$;
- În general, în perioada de eșantionare #*i*, se determină mai întâi starea curentă a procesului real $\mathbf{x}_{\text{real}}(i)$. Să remarcăm că această stare, chiar dacă nu este identică, este apropiată de starea teoretică, $\mathbf{x}_{\text{model}}(i)$, spre care evoluează modelul sistemului prin aplicarea comenzii $\boldsymbol{U}(i\text{-}1)=u_{i-1}^{i-1}$: $X_{\mathbf{real}}(i-1) \xrightarrow{\mathbf{U(i-1)}} X_{\mathbf{model}}(i)$

  Din starea $\mathbf{x}_{\text{real}}(i)$, pe orizontul $H_i=i,...,n$, algoritmul ***A*** determină un șir $u_i^i, \cdots, u_n^i$ care este o soluție cât mai bună pentru PCO. Acest șir este determinat prin ameliorarea șirului $u_i^{i-1}, \cdots, u_n^{i-1}$, deja cunoscut din perioada de eșantionare anterioară și care are un caracter quasi-optimal în raport cu starea $\mathbf{x}_{\text{model}}(i)$. Șirul $u_i^{i-1}, \cdots, u_n^{i-1}$ este folosit ca soluție inițială a PCO pe orizontul curent. Comanda regulatorului se adoptă ca fiind: $\boldsymbol{U}(i)=u_i^i$;

- Ultima comandă pe orizontul de conducere considerat face parte din șirul de 2 elemente u_{n-1}^{n-1}, u_n^{n-1}. Valoarea acesteia este, deci $\boldsymbol{U}(n\text{-}1)=u_{n-1}^{n-1}$.

Constatăm că PCO, rezolvată de algoritmul ***A*** cu orizontul cel mai lung în perioade de eșantionare, caracterizează prima perioadă de eșantionare. De aceea caracterul "optimal" al acesteia este discutabil, dar și influența asupra părții Mayer a funcției obiectiv

$$M\left(x(t_f), y(t_f), u(t_f), p, t_f\right)$$

este mai redusă, distanța față de momentul t_f fiind maximă. Cu alte cuvinte, prin comenzile următoare se încearcă compensarea caracterului quasioptimal al primelor comenzi ale regulatorului.

Din punctul de vedere al încadrării într-o perioadă de eșantionare a execuției algoritmului ***A*** pentru orizontul H_1=1, 2,...,*n*, soluția este considerarea unui număr maxim de evaluări N_{eval}, verificat prin simulare, drept condiție de oprire a algoritmului. Pentru perioadele de eșantionare următoare, acesta va acoperi și mai bine căutarea secvenței de comenzi optimale, pentru că orizontul scade.

Reluăm, ca exemplu, sistemul din secțiunea 1.4.2, pentru că funcția obiectiv trebuie să conțină o componentă legată de caracterul bilocal al problemei, chiar dacă lipsește componenta Mayer. Reluăm aici modelul dinamic: $\dot{x}_1 = x_2$, $\dot{x}_2 = u(t)$.

Se cere comanda $u^*(t)$, $t \in [0,2]$ - unde *t* este exprimat în ore - care transportă sistemul din

starea inițială: $t_o = 0$, $\theta_o = 1$, $\dot{\theta}_o = 1$, în starea finală: $t_f = 2$ ore, $\theta_f = 0$, $\dot{\theta}_f = 0$,

a.î. consumul de energie să fie minim; $\min_u J = \frac{1}{2}\int_0^{t_f} u^2(t)dt$

Caracterul bilocal al problemei, ne obligă să considerăm următoarea funcție obiectiv pe parcursul căutării soluției optimale:

$$f(u) = 50 \cdot \left(x_1(t_f) - 0\right)^2 + 50 \cdot \left(x_2(t_f) - 0\right)^2 + \int_0^{t_f} \cdot u^2(t)dt\,.$$

Regulatorul utilizează algoritmul ***A***, identic cu cel folosit în secțiunea 3.2.2, adică un algoritm HTPSO. În cele ce urmează, vom simula funcționarea sistemului de conducere din figura 5.3, cu RBM.

Prima etapă de simulare

În această etapă, vom simula Procesul Real printr-un Model al Procesului Real. Desigur, este rațional să considerăm, într-o primă etapă, ca Model al Procesului Real, chiar Modelul Procesului, utilizat în RBM. În felul acesta, *vom analiza doar efectul utilizării unor ferestre de timp din ce în ce mai mici, asupra caracterului de optimalitate al soluției implementate de regulator, fără a mai considera și alte efecte legate de un model imperfect sau de perturbații*.

Modelul Procesului realizează următoarea tranziție a stărilor la perioada *k* de eșantionare, $X_{\mathbf{model}}(k) \xrightarrow{\mathbf{U(k)}} X_{\mathbf{model}}(k+1)$. Dată fiind ipoteza adoptată, avem $\mathbf{x}_{\text{real}}(k)=\mathbf{x}_{\text{model}}(k)$, care este echivalentă cu tranziția stărilor $X_{\mathbf{real}}(k) \xrightarrow{\mathbf{U(k)}} X_{\mathbf{real}}(k+1)$

În figura 5.10, este indicată în albastru evoluția tipică obținută pentru comanda cu RBM, în comparație cu comanda optimală teoretică în buclă deschisă (soluția PCO). În figura 5.11,

avem evoluția stărilor când procesul este comandat în buclă închisă (roșu: x_1 obținut cu RBM; albastru: x_2 obținut cu RBM) versus evoluția teoretică în buclă deschisă (soluția PCO; verde: x_1 optimal teoretic; negru: x_2 optimal teoretic)

Numărul de apelări ale funcției obiectiv este `Apeluri fitness=4740,` ceea ce este mulțumitor pentru 50 perioade de eșantionare. Examinând aceste figuri, rezultă că structura de control în buclă închisă este o soluție realistă care dă rezultate foarte bune.

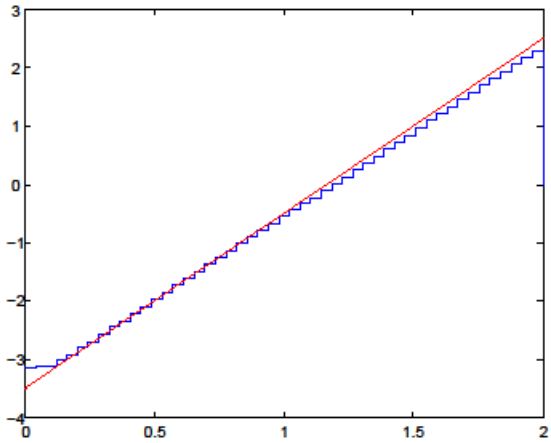


Figura 5.10 Evoluția comenzii teoretic si cu regulator

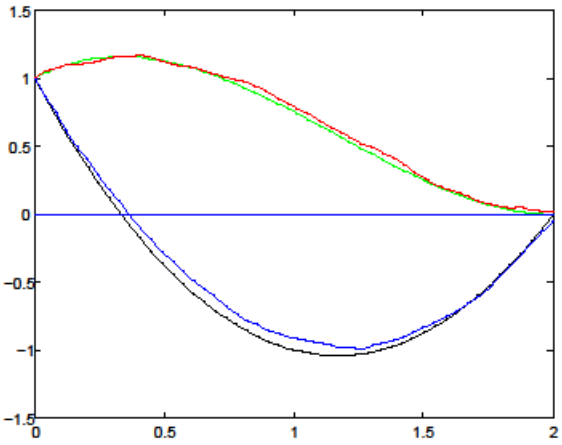


Figura 5.11 Evoluția stărilor teoretic si cu regulator

A doua etapa de simulare

În această etapă, vom simula Modelul Procesului Real utilizând Modelul Procesului, dar alterând stările acestuia (presupuse accesibile sau posibil de estimat) printr-un zgomot aditiv.

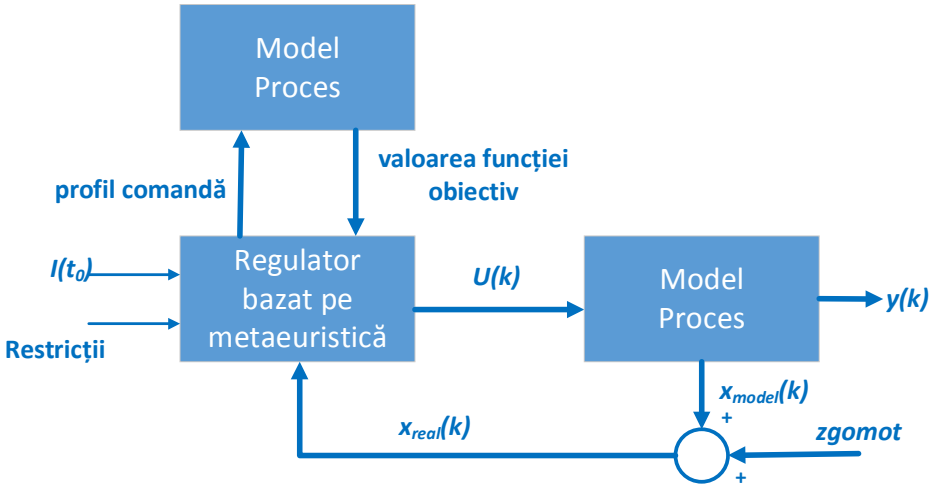


Figura 5.12 Simulare cu Model Proces Real

Structura de control simulată este prezentată în figura 5.12. Zgomotul considerat pentru cele doua stări este uniform distribuit în intervalul $[-L_k^1, L_k^1]$ respectiv $[-L_k^2, L_k^2]$, unde

$$L_k^1=0.05 \cdot|x^1(k)|; \qquad L_k^2=0.05 \cdot|x^2(k)|;$$

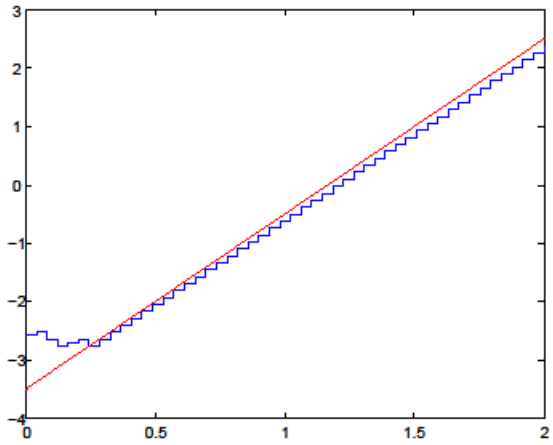


Figura 5.13 Evoluția comenzii teoretic și cu regulator

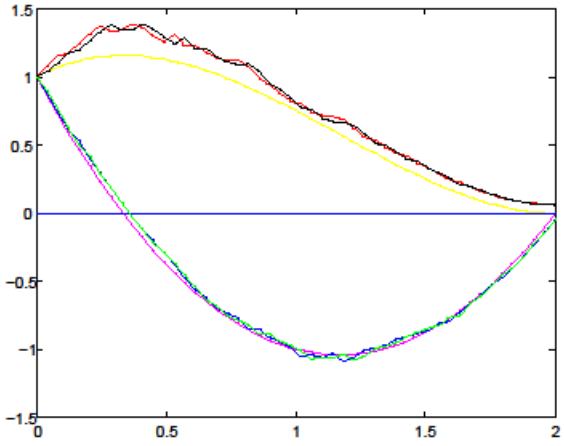


Figura 5.14 Evoluția stărilor teoretic, cu zgomot și fără zgomot

În figura 5.14, avem evoluția stărilor când procesul este comandat:

- în buclă închisă fără zgomot(negru: x_1 obținut cu RBM; verde: x_2 obținut cu RBM);

- în buclă închisă cu zgomot(roșu: x_1 obținut cu RBM; albastru: x_2 obținut cu RBM);

- în buclă deschisă - evoluția teoretică - (soluția PCO; galben: x_1 optimal teoretic; magenta: x_2 optimal teoretic).

Observăm evoluția asemănătoare a stărilor obținute cu RBM, atât cu zgomot cât și fără. Numărul de apelări ale funcției obiectiv este `Apeluri fitness=5000,` ceea ce este remarcabil de puțin pentru 50 perioade de eșantionare. Examinând aceste figuri, rezultă că structura de control în buclă închisă, cu reacție pe stare reală, este o soluție realistă care dă rezultate mai mult decât satisfăcătoare.

5.5 Concluzii

Problema care s-a pus în acest capitol este de a avea o metodă care să ne permită conducerea procesului în buclă închisă, de îndată ce avem un bun algoritm bazat pe o metaeuristică ce rezolvă PCO (algoritmul ***A***).
A fost propusă o structură de conducere în buclă închisă dezvoltată în jurul a două idei. Prima idee este utilizarea unei structuri de conducere care este o adaptare a conceptului de control predictiv. A doua idee este utilizarea unui algoritm capabil să rezolve aproximativ o PCO printr-o metaeuristică. Legătura dintre cele două idei este că structura predictivă are un regulator care poate utiliza algoritmul respectiv.

Rezultatele care sunt și contribuții ale acestui capitol sunt următoarele:

- A fost propusă o structură de conducere în buclă închisă plecând de la controlul predictiv al proceselor. Regulatorul din structură este un RBM (regulator bazat pe o metaeuristică), adică folosește un algoritm ***A*** și un model al procesului condus. La fiecare perioadă de eșantionare, algoritmul ***A*** rezolvă PCO pentru *un nou orizont de timp* -ce debutează cu momentul de eșantionare respectiv - și pentru *o nouă stare inițială a procesului*, care este starea sa reală presupusă accesibilă;
- S-a demonstrat faptul că un RBM lucrează în sensul minimizării erorii de predicție și deci că structura propusă este un caz particular al structurii cu control predictiv;

- S-a propus un mod de organizare al controlul în buclă închisă al proceselor cu criterii de optimalitate de tip Lagrange, cu orizont de control nedefinit sau cu timp final stabilit. În această situație, orizontul de timp pe care RBM caută soluția cea mai bună are un număr constant de perioade de eșantionare, dar este alunecător și începe cu momentul de eșantionare respectiv;
- S-a propus un mod de organizare al controlului în buclă închisă al proceselor cu criterii de optimalitate de tip Bolza, sau care duc la probleme bilocale. Modul de rezolvare al problemelor bilocale cu criteriu de optimalitate Lagrange, duce, de fapt, așa cum s-a arătat în capitolele anterioare, la considerarea unui criteriu de tip Bolza pe parcursul căutării soluției optimale, cu scopul de a asigura îndeplinirea condițiilor finale. În această situație, orizontul de timp pe care RBM caută soluția cea mai bună este descrescător ca număr de perioade de eșantionare, dar conține întotdeauna momentul final.

Capitolul 6
Concluzii generale, contribuţii originale şi perspective

Pe parcursul a 5 capitole, prezenta lucrare propune o analiză a rezolvării unor PCO cu ajutorul unor algoritmi metaeuristici, dar și o structură de conducere în buclă închisă care să implementeze controlul propriu-zis. Consider că au fost atinse obiectivele propuse:

- studiul implementării unor algoritmi metaeuristici dedicați rezolvării unor PCO;
- analiza măsurii în care soluțiile obținute sunt realiste și asigură măcar un caracter quasioptimal;
- analiza complexității practice a algoritmului ce implementează metaeuristica, până la obținerea unei soluții bune;
- realizarea unui studiu comparativ al utilizării diferitelor metaeuristici în rezolvarea unei aceleași probleme;
- dezvoltarea și validarea prin simulare a unei structuri de conducere în buclă închisă utilizând acești algoritmi

Consider că algoritmii metaeuristici implementați, dezvoltați și analizați precum și structura de conducere în buclă închisă propusă au o importanță practică considerabilă în situațiile în care pentru PCO supuse rezolvării nu se cunosc alte modalități de rezolvare și implementare convenabile.

6.1 Contribuţii

Voi indica, pentru fiecare capitol, aspectele științifice care se pot constitui în contribuții legate de metaeuristicile respective și utilizarea lor la rezolvarea unor PCO.

6.1.1 Ameliorarea performanţelor algoritmului Simulated Annealing în rezolvarea PCO

- Tehnica numită controlul pasului (*step control*) impune niște margini superioare și inferioare ale gradientului variabilelor de conducere. Deși cunoscută, în prezenta lucrare a fost indicată explicit o manieră de implementare în PCO.

- Analiza modului de implementarea a PCO bilocale în raport cu starea și timp final fixat sau liber cu evidențierea unor soluții practice.

6.1.2 Analiza pentru ameliorarea performanțelor algoritmilor evolutivi în rezolvarea PCO

- Integrarea într-un AE unor tehnici eficiente (cum ar fi step control) și metode de selecție care să evite fenomenul de deviere (Selecție cu Stochastic Universal Sampling, Utilizare Rang și Scalare Rang), ținându-se astfel cont de particularitățile PCO tratate.
- Analiză pe o cazuistică formată din trei PCO, a diferiților operatori de crossover și mutație adecvați problemelor respective. Concluzia ne arată că fiecare problemă are o sensibilitate diferită la utilizarea unui operator sau altul. Au fost implementați și analizați operatori semnificativi pentru implementarea unui AE.

6.1.3 Contribuții la utilizarea PSO în rezolvarea PCO

- Integrarea tehnicii *step control* - care nu corespunde unei restricții specifice PCO - în algoritmul HTPSO pentru a asigura un profil de comandă cu o evoluție continuă și chiar derivabilă, pe un domeniu cât mai larg. Astfel, cel puțin, comanda "optimală" evită salturile puternice.
- S-a propus ca procedura ce determină experiența locală cea mai bună să includă, pe lângă informația "socială", o componentă de intensificare care să ia în considerare poziția curentă (informația "geometrică"). În acest scop, s-a introdus un algoritm de intensificare "geometrică", determinist, de coborâre locală numit GDD (geographical deterministic descent).
- Au fost propuse, de asemenea, trei tehnici de ameliorare a algoritmului GDD, care au vizat:

- apelul recursiv al GDD, pentru mărirea eficienței;
- minimizarea fenomenului de deviere ("bias") în căutarea soluțiilor optime;
- apelarea selectivă, la un anumit număr de iterații fără ameliorare a soluțiilor "local best".

6.1.4 Variante ameliorate ale meta-euristicii „Algoritm diferențial stigmergic”

În prezenta teză s-au propus două direcții de îmbunătățire a performanțelor meta-euristicii „Algoritm diferențial stigmergic” în rezolvarea PCO.

- Primă direcție validată a fost rezolvarea PCO în două etape. Problema supusă rezolvării este transformată inițial într-o problemă cu dimensiune redusă prin considerarea unui pas de granularitate crescută în eșantionarea comenzii. Astfel se scade complexitatea computațională a problemei și se realizează o *explorare* într-un spațiu mai *granulat*, cu o convergență mai bună către zona cu optimul căutat. Soluția problemei de dimensiune redusă este reeșantionată prin interpolare, și se generează astfel o soluție inițială de bună calitate pentru etapa a doua care va rezolva problema inițială. Suplimentar dimensiunea spațiului de căutare în cele două etape este controlată direct prin precizia cu care este discretizat spațiul continuu și transformat în graful asociat. În etapa de identificare a zonei promițătoare se folosesc pași ”mari” de căutare, iar în etapa de intensificare se utilizează pași corelați cu precizia impusă în rezolvarea problemei inițiale și cu parametrii tehnologici a procesului ce urmează a fi condus.
- A doua direcție validată de ameliorare a performanțelor DASA la rezolvarea PCO a fost introducerea unor elemente de elitism în strategia de căutare. Cele trei variante elitiste

ale DASA propuse promovează o utilizare mai eficientă a informației transmise prin feromon. Primele două variante de elitism propuse sunt echivalente unei încercări de căutare după gradient prin utilizarea unei furnici elitiste la nivelul populației de furnici. Aceste variante ale meta-euristicii a fost denumite „DASA-elitist-pur". A treia variantă, a fost cea de introducere a componentei de elitism în cadrul procesului de construcție a soluției. Meta-euristica obținută a fost denumită „DASA-elitist-probabilistic".

6.1.5 Contribuții la conducerea în buclă închisă a proceselor, utilizând optimizarea prin metaeuristici

A fost propusă o structură de conducere în buclă închisă dezvoltată în jurul a două idei. Prima idee este utilizarea unei structuri de conducere care este o adaptare a conceptului de control predictiv. A doua idee este utilizarea unui algoritm capabil să rezolve aproximativ o PCO printr-o metaeuristică. Legătura dintre cele două idei este că structura predictivă are un regulator care poate utiliza algoritmul respectiv.

Rezultatele care sunt și contribuții ale acestui capitol sunt următoarele:

- A fost propusă o structură de conducere în buclă închisă plecând de la controlul predictiv al proceselor. Regulatorul din structură este un RBM (regulator bazat pe o metaeuristică), adică folosește un algoritm ***A*** și un model al procesului condus. La fiecare perioadă de eșantionare, algoritmul ***A*** rezolvă PCO pentru *un nou orizont de timp* -ce debutează cu momentul de eșantionare respectiv - și pentru *o nouă stare inițială a procesului*, care este starea sa reală presupusă accesibilă;
- S-a demonstrat faptul că un RBM lucrează în sensul minimizării erorii de predicție și deci că structura propusă este un caz particular al structurii cu control predictiv;
- S-a propus un mod de organizare al controlului în buclă închisă al proceselor cu criterii de optimalitate de tip Lagrange, cu orizont de control nedefinit sau cu timp final stabilit. În această situație, orizontul de timp pe care RBM caută soluția cea mai bună are un număr constant de perioade de eșantionare, dar este alunecător și începe cu momentul de eșantionare respectiv;
- S-a propus un mod de organizare al controlul în buclă închisă al proceselor cu criterii de optimalitate de tip Bolza, sau care duc la probleme bilocale. Modul de rezolvare a problemelor bilocale cu criteriu de optimalitate Lagrange, duce, de fapt, așa cum s-a arătat în capitolele anterioare, la considerarea unui criteriu de tip Bolza pe parcursul căutării soluției optimale, cu scopul de a asigura îndeplinirea condițiilor finale. În această situație, orizontul de timp pe care RBM caută soluția cea mai bună este descrescător ca număr de perioade de eșantionare, dar conține întotdeauna momentul final.

6.2 Direcții viitoare de cercetare

În cazul meta-euristicii „Differential Ant-Stigmergy Algorithm" - DASA, o direcție de cercetare ce va fi investigată este cea în care distribuția feromonului ar fi realizată prin altă distribuție decât cea Cauchy.Altă direcție este cea de adaptare a modelului feromonului pentru a permite memorarea direcțiilor promițătoare de căutare pe termen mai lung.

O direcție conexă cu tema acestei lucrări este utilizarea unor metaeuristici într-o procedură de acordare on-line a parametrilor unor regulatoare.

6.3 Diseminarea rezultatelor

Rezultatele obținute ca urmare a activității de cercetare a autorului și care au stat la baza realizării prezentei teze au fost diseminate în cadrul următoarelor lucrări din reviste sau manifestări științifice:

13 lucrări la conferințe internaționale, indexate în următoarele BDI: SCOPUS (13), IEEE Xplore (7), Science direct-IFAC-PapersOnLine (1);

7 din cele 13 lucrări sunt indexate ISI proceedings Web of Science (WoS);

1 lucrare în revistă indexată BDI.

Mai jos, este prezentată lista cu lucrările prin intermediul cărora au fost diseminate rezultatele:

Șerbencu, A., Minzu, V., & Șerbencu, A. (2007). An ant colony system based metaheuristic for solving single machine scheduling problem. *The Annals of Dunarea De Jos University of Galati*, *3*, 19-24. (Revistă indexată BDI)

Șerbencu, A. E., Șerbencu, A., Cernega, D. C., & Mînzu, V. (2010, July). Particle swarm optimization for the sliding mode controller parameters. In *Control Conference (CCC), 2010 29th Chinese* (pp. 1859-1864). IEEE.

Șerbencu, A., Cernega, D. C., Șerbencu, A. E., & Susnea, I. (2009, August). Path following problem for patrolbot solved with fuzzy control. In Automation and Logistics, 2009. ICAL'09. IEEE International Conference on (pp. 2005-2010). IEEE.

Șerbencu, A. E., & Șerbencu, A. (2012, October). A comparison of particle swarm optimization and differential ant stigmergy algorithm. In *System Theory, Control and Computing (ICSTCC), 2012 16th International Conference on* (pp. 1-6). IEEE.

Șerbencu, A. E., Mînzu, V., & Șerbencu, A. (2014, October). Precedence constraints treatment in ant colony optimization. In *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference* (pp. 87-92). IEEE.

Mînzu, V., & Șerbencu, A. (2016, October). Control structure for the optimal sewer network discharge. In *System Theory, Control and Computing (ICSTCC), 2016 20th International Conference on* (pp. 61-66). IEEE.

Șerbencu, A., & Mînzu, V. (2016, December). Hybridized Ant Colony System for Tasks to Workstations Assignment. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on* (pp. 1-7). IEEE.

Șerbencu, A., Șerbencu, A., Mînzu, V., & Cernega, D. Multiagent Systems Used For Discrete Optimization Problems. *Buletinul Universitatii Petrol-Gaze din Ploiesti*, Vol. LV, No. 2/2003, ISSN 1221-9371

Serbencu, A., & Serbencu, A. E. (2015, October). Two mobile robotic systems synchronous servicing an Assembly/Disassembly production line. In *System Theory, Control and Computing (ICSTCC), 2015 19th International Conference on* (pp. 81-86). IEEE.

Șerbencu, A., Mînzu, V., & Cernega, D. C. (2006). XML-nets for simulation of the supply chain processes. *IFAC Proceedings Volumes*, *39*(3), 597-602.

Cernega D. C., Bratcu A. I. and Şerbencu A., Supervisory Control Problem for Protocol Conversion, *In Proceedings of The 12-th International Symposium on Modeling, Simulation and System's Identification*, 2004, 200-205

V. Mînzu, L. Beldiman, Adrian Serbencu, Adriana Serbencu, M. Barbu; "Virtual workshop for academic training", *Proceedings of the 14-th Conference on Control Systems and Computer Science*, 2-5 July 2003, Bucuresti, pp. 557-562

Adrian Emanoil Șerbencu, Viorel Mînzu, Adriana Șerbencu, Daniela Cernega: Two Elitist Variants of Differential Ant-stigmergy Algorithm. In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* ICINCO (1) 2011: 136-141, ISBN: 978-989-8425-74-4

Șerbencu A., Șerbencu A.E., Cernega D.C. (2010) Evolutionary Strategies Used for the Mobile Robot Trajectory Tracking Control. In: Diamantaras K., Duch W., Iliadis L.S. (eds) Artificial Neural Networks – ICANN 2010. ICANN 2010. *Lecture Notes in Computer Science*, vol 6353. Springer, Berlin, Heidelberg http://link.springer.com/chapter/10.1007/978-3-642-15822-3_36

Capitol de carte

D. C. Cernega, A. I. Bratcu and A. Șerbencu – Discrete Event Supervision for Communication Protocol Conversion. In: Intelligent Systems at the Service of Mankind II (Eds. W. Elmenreich, J. Tenreiro Machado, I. J. Rudas), UBooks Verlag, Augsburg (Germany), 2005, ISBN 3-86608-052-2, pp. 227-238.