FUNDAMENTE ALE OPTIMIZĂRII MULTI-OBIECTIV A SISTEMELOR COMPLEXE DE CALCUL

Prof. dr. ing. **Lucian N. VINȚAN**

Universitatea „Lucian Blaga" din Sibiu, Facultatea de Inginerie

REZUMAT. Acest articol prezintă bazele teoretice ale optimizării multi-obiectiv a sistemelor de calcul complexe. Se arată că algoritmii de optimizare multi-obiectiv sunt prea generali. Pentru a obține rezultate bune, aceștia trebuie îmbunătățiți prin utilizarea unor cunoștințe de domeniu, specifice sistemelor de calcul, în cazurile considerate aici. Aceste metode trebuie automatizate și utilizate intensiv în fazele de cercetare – proiectare. Elaborarea unor algoritmi de optimizare multi-obiectiv eficienți, atât din punct de vedere al timpului de rulare cât și din punct de vedere al calității soluțiilor obținute, constituie o mare provocare științifică și tehnică. Se prezintă o abordare a acestei probleme utilizând inclusiv conceptul de meta-optimizare, adică o optimizare bazată pe mai mulți algoritmi de optimizare care lucrează în mod concurent.

Cuvinte cheie: Arhitectura sistemelor de calcul, algoritmi genetici, optimizare multi-obiectiv, suprafața Pareto, logică *fuzzy*, meta-optimizare.

ABSTRACT. In this paper we present some fundamental ideas related to complex computing systems' automatic multi-objective optimization. It is shown that these optimization algorithms are too general. In order to develop some effective optimization methods, such algorithms might be augmented with some domain knowledge. We are also showing that for complex computing systems it would be recommendable to use state of the art optimisation algorithms that will be concurrently orchestrated through a meta-algorithmic abstraction layer. It is analysed a meta-optimization method proposed in order to find the best (Pareto) individuals in a bi-objective space. Developing effective multi-objective optimization methods – particularly meta-optimization methods - represents a great research challenge for further research.

Keywords: Computing Systems, Genetic Algorithms, Multi-Objective (Pareto) Optimization, Fuzzy Logic, Meta-Optimization.

1. INTRODUCERE

Problema optimizării multi-obiectiv a sistemelor de calcul este una complexă, datorită faptului că spațiul de proiectare este, în general, unul enorm, care nu poate fi cercetat și evaluat într-un mod exhaustiv. Spre exemplu, un sistem de calcul cu 50 de parametri (numărul de procesoare din sistem, tipul rețelei de interconectare între acestea, capacitatea cache de nivel 3/2/1 etc.), fiecare dintre aceștia putând avea doar 8 valori discrete, implică un spațiu de căutare de 2^{150} configurații distincte! Problema care se pune este: care dintre aceste configurații sunt cele mai performante? Evident că aceasta este o problemă de complexitate NP-hard. Ca să dăm un alt exemplu, mai practic, simulatorul software M-SIM 2 (procesor superscalar/*Simultaneous Multi-Threading*) are 2,5 milioane de miliarde de configurații posibile. Este evident că o evaluare manuală nu este posibilă în asemenea cazuri. Nici măcar una automată. În realitate, problema este și mai complexă pentru că, din această mulțime enormă de instanțe CPU (în cazul nostru), nu o dorim neapărat pe cea mai performantă, ci, mai degrabă, pe cea mai performantă în condiții de consum energetic minimal, complexitate hardware cât mai mică etc. Așadar, problema de optimizare nu este una cu un singur obiectiv, ci una cu mai multe obiective, în general contradictorii unele față de celelalte (ex. performanța vs. puterea consumată). Problema pusă devine deci și mai complexă, iar soluțiile nu pot fi decât aproximative, generabile prin algoritmi euristici.

Dar nu numai arhitectura hardware trebuie optimizată, ci și compilatorul și programele HLL care vor rula pe aceasta. Aceste programe, spre exemplu, sunt compilate cu diferite opțiuni de optimizare. Este evident că opțiunile optimale de compilare (metodele de scheduling utilizate) sunt și ele dependente de parametrii arhitecturii hardware. În consecință, este nevoie de o co-optimizare hardware-software (*hardware software co-optimization → cross-layer optimization*), ceea ce înseamnă că spațiul de căutare este în realitate și mai mare, fiind dat de totalitatea instanțelor hardware – software ale sistemului de optimizat!

O funcție vector este o funcție care mapează un *t*-uplu de m parametri arhitecturali pe un *t*-uplu de *n* obiective [Vin13]. Mai precis:

- *Min/Max* $y = f(x) = (f_1(x), f_2(x) \ldots f_n(x))$
- $x = (x_1, x_2, \ldots, x_m) \in X$ – vectorul de decizie (nr. de nuclee, tip rețea de interconectare, capacitate cache, nr. unități de execuție per procesor etc.)
- $y = (y_1, y_2, \ldots, y_n) \in Y$ – vector obiectiv (performanță, energie consumată, complexitate hardware, Worst Case Execution Time – WCET etc.).

În figura 1 se prezintă curba Pareto pentru o problemă de minimizare cu doar două obiective (f_1, f_2). Se numește astfel după economistul, sociologul și matematicianul italian *Vilfredo Pareto*. Se doresc identificați acei indivizi, considerați optimali, care minimizează ambele obiective f_1 și f_2. Oricare doi indivizi de pe curba Pareto (A, B) sunt non-dominați. Spre exemplu, individul A domină doar parțial individul B, din punct de vedere al obiectivului f_2 (pentru că $f_2(A) < f2(B)$), în timp ce individul B domină doar parțial individul A, din punct de vedere al obiectivului f_1 (f_1 (B)< f_1 (A)). În schimb, se poate observa că pentru orice individ C, care nu aparține frontului Pareto, există cel puțin un punct pe curba Pareto care să-l domine pe acesta (C), din punct de vedere al ambelor obiective f_1 și f_2. Așadar, indivizii Pareto sunt dominanți, față de cei care nu aparțin curbei respective. Într-un spațiu ortogonal cu 3 obiective, avem suprafața Pareto. Într-un spațiu *n*-dimensional, avem o hiper-suprafață Pareto. Problema unei optimizări multi-obiectiv complexe este aceea de a determina hiper-suprafața Pareto reală, care aproximează pe cât de bine posibil hiper-suprafața Pareto ideală (în general, imposibil de determinat).

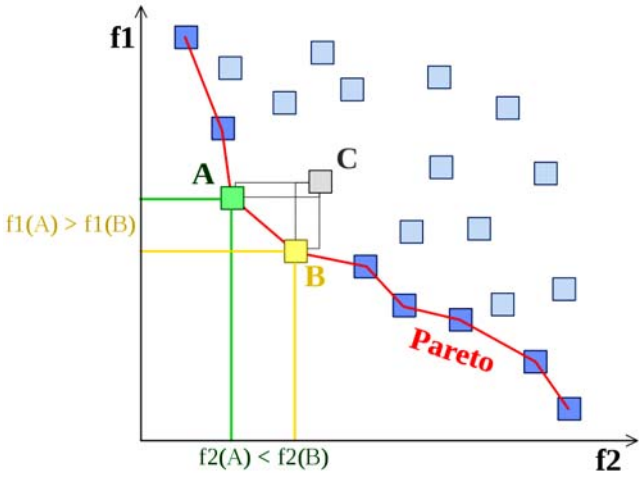

Fig. 1. Curba Pareto pentru o problemă de minimizare cu 2 obiective.

2. ALGORITMI GENETICI ÎN OPTIMIZARE

Pentru a determina Hiper-Suprafața Pareto (HSP) aproximativă, s-au dezvoltat algoritmi euristici de optimizare multi-obiectiv. Printre cei mai cunoscuți astfel de algoritmi sunt algoritmii genetici multi-obiectiv (evolutivi, de tip NSGA-II, SPEA2, CNSGA-II etc.) și algoritmii de tip *Particle Swarm Optimization* (bio-inspirați din inteligența colaborativă a roiurilor de furnici, albine etc.), de asemenea adaptați la multiple obiective. Spre exemplu, biblioteca software gratuită *jMetal* oferă implementări ale unor astfel de algoritmi.

În figura 2 se prezintă schema logică de principiu a unui algoritm genetic. Acesta pornește de la o populație (generație) inițială de N indivizi, numiți cromozomi, prin analogie cu genetica.

Această generație inițială ar putea fi obținută cvasi-aleator sau pe baza experienței utilizatorului, care poate să introducă anumite soluții cunoscute, performante, în ea.

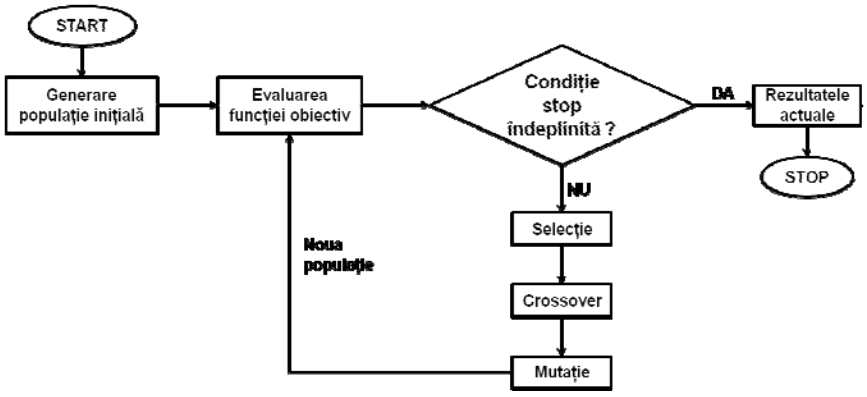

Fig. 2. Schema logică a unui algoritm genetic.

Un cromozom reprezintă, practic, o succesiune de aşa numite gene, concatenate. O genă, în cazul nostru particular, reprezintă un parametru al arhitecturii de calcul care trebuie optimizată (spre exemplu, numărul de procesoare din sistemul de calcul, tipul reţelei de interconectare, capacitatea cache-urilor etc.) Evident că aceste gene sunt codificate într-un anumit mod (deseori binar). Apoi, fiecare individ al populaţiei este evaluat. În cazul nostru, evaluarea se face prin simularea fiecărei instanţe a sistemului de calcul respectiv (din cele N existente), utilizând un simulator dedicat şi unul sau mai multe benchmark-uri (această evaluare este, în general, mare consumatoare de timp; din fericire poate fi paralelizată prin calcul paralel sau distribuit). În urma evaluării, fiecare individ din populaţie va obţine o aşa numită rată de *fitness*, în general normalizată în intervalul [0, 1].

În continuare, se selectează într-un mod stohastic perechi de indivizi spre a fi încrucişaţi, cu o anumită probabilitate. Acestor perechi li se aplică operatorul genetic de încrucişare, numit *Crossover*. Evident, indivizii cu rate de fitness mai mari, vor avea şanse mai mari de a fi selectaţi pentru încrucişare decât cei cu performanţe mai reduse. Există mai multe strategii de *Crossover*. Cea mai simplă (numită *single point crossover operator*), împarte părinţii în două părţi (P11&P12 – părinte 1, P21&P22 – părinte 2), într-un mod cvasi-aleator şi obţine doi noi indivizi (copii, *offsprings*), prin încrucişarea acestor părţi (P11&P22 –offspring 1, P21&P12 – offspring 2). Se obţin astfel încă N indivizi. Acum, celor N indivizi vechi (părinţi), precum şi celor N indivizi nou creaţi (*offsprings*), li se aplică operatorul genetic de mutaţie (*Mutation*), cu o anumită probabilitate prestabilită (în general, mică). Dacă, spre exemplu, probabilitatea de mutaţie este 0,05 atunci doar 5% dintre cei 2N indivizi vor suferi mutaţii, pe anumite gene. Mutaţia *(the bit flip mutation)* operează pe un singur individ, ca mai jos.

1. *Pentru toate genele (parametrii) aferente unui individ (cromozom);*
   - *1.1. Se generează un număr cvasi-aleator (raţional) între 0 şi 1;*
   - *1.2. Dacă numărul generat < probabilitatea de mutaţie;*
     - *1.2.1. Schimbă valoare genei respective în mod cvasi-aleator;*
2. *STOP.*

Există prezentate în literatura de specialitate multe alte variaţiuni ale mutaţiei. Cei N indivizi nou obţinuţi, în urma încrucişărilor şi mutaţiilor, se evaluează şi ei, alocându-se fiecăruia o rată de *fitness* normalizată. Apoi, dintre cei N indivizi ai populaţiei iniţiale, respectiv dintre cei N indivizi nou creaţi, se selectează prin ierarhizare, pe baza ratei lor de fitness, cei mai performanţi N indivizi (cromozomi), care vor forma următoarea generaţie (operatorul genetic de selecţie - *Selection*). Această selecţie este evident una elitistă (pot fi implementaţi şi alţi algoritmi de selecţie). Algoritmul se va repeta până când condiţia de oprire va fi îndeplinită (spre exemplu, se atinge numărul prestabilit de generaţii, performanţa ultimei generaţii nu a mai crescut în mod semnificativ etc.) Există foarte multe variaţiuni ale acestui algoritm euristic. Prin astfel de algoritmi, în generaţia finală se obţin indivizi optimizaţi (adică performanţi) din punct de vedere al ratei de *fitness*. Această ultimă afirmaţie are şi o justificare teoretică solidă, dată de teorema schemei, demonstrată de *Holland*, care justifică faptul că algoritmii genetici converg.

3. ALGORITMI GENETICI MULTI-OBIECTIV

Problema specifică algoritmilor genetici multi-obiectiv este următoarea: cum am putea sorta (ierarhiza) indivizi reprezentaţi într-un spaţiu (ortogonal sau cvasi-ortogonal) multi-obiectiv? Pentru un singur obiectiv, sortarea era trivială, funcţie doar de valoarea ratei de fitness. Desigur, o soluţie ar consta în reducerea algoritmului genetic multi-obiectiv la unul mono-obiectiv, prin agregarea obiectivelor multiple într-unul singur, printr-o formulă de agregare. Ideea este de a micşora, pe cât mai mult posibil, pierderea de informaţie pe care procesul de agregare o implică. Personal nu agreez în mod deosebit asemenea metode. În literatura matematică de specialitate s-au formulat 3 condiţii necesar a fi îndeplinite într-un proces corect de agregare. Astfel, indicatorul global (agregat) trebuie să fie sensibil (senzitiv în mod corect la variaţiile valorilor indicatorilor componenţi). Apoi, acesta trebuie să fie anti-catastrofic, în sensul dat de teoria catastrofelor elaborată de matematicianul francez *Rene Thom* – laureat al Medaliei *Fields* (1958). Intuitiv, aceasta înseamnă că variaţii mici ale valorilor intrărilor, nu trebuie să producă valori mari („catastrofice", discontinue) ale ieşirii. În fine, indicatorul agregat trebuie să fie necompensatoriu, adică modificarea într-un sens (creştere/descreştere) a valorii unui indicator component să nu poată fi compensată de modificarea în sens invers, a altuia. S-a arătat că nu există nicio operaţie de agregare a indicatorilor, care să satisfacă, simultan, cele 3 condiţii de bun-simţ enunţate anterior, ceea ce închide iluzia unei agregări adecvate. În consecinţă nu vom stărui pe abordări de acest fel, deşi ele există.

Prezentăm în continuare câteva abordări native Pareto (multi-obiectiv) la această problemă. Spre exemplu, algoritmul genetic multi-obiectiv NSGA-II soluţionează această problemă de ierarhizare elitistă,

sortând indivizii în fronturi Pareto succesive [Cal10]. După această operație, se selectează indivizii de pe primul front Pareto, P1, al indivizilor non-dominați. Dacă mai este nevoie de indivizi în noua generație, se selectează cei aparținând frontului Pareto P2 ș.a.m.d. Așadar, acest algoritm preferă un individ situat pe un front Pareto (HSP) mai bun. Se pune, totuși, problema sortării indivizilor de pe același front Pareto, Pk. În acest caz, se calculează distanțele între un individ *i* aparținând Pk și vecinii săi, *i* + 1 respectiv *i* – 1 (*Crowding Distance* – CD), pentru toți indivizii aparținând frontului Pk. Algoritmul NSGA-II preferă în acest caz indivizii cu un CD mai mare. Ideea intuitivă care stă la baza acestei alegeri este aceea de a selecta, pe cât posibil, indivizi de pe întreg frontul Pk, asigurându-se astfel o oarecare diversitate a noii generații și o căutare mai uniformă în spațiul de proiectare.

Algoritmul CNSGA-II, spre exemplu, procedează altfel. Acesta preferă să selecteze indivizi de pe toate fronturile Pareto, jertfind deci elitismul selecției NSGA-II, în favoarea unei mai mari diversități a generației următoare (prin includerea, deliberată, a unor indivizi dominați, aparținând unor fronturi Pareto inferioare). Astfel, spațiul de căutare este eșantionat într-un mod și mai uniform.

O metrică de calitate a acestor algoritmi de optimizare multi-obiectiv, utilă și utilizată, este dată de așa numitul hiper-volum (HV). Într-o problemă de minimizare, măsura acestui HV [%] este dată de formula (HVreal/HV ideal)×100%. HVreal reprezintă măsura hiper-volumului cuprins între frontul Pareto real și un așa numit punct de referință. Atunci când frontul Pareto ideal este cunoscut, HVideal este măsura volumului cuprins între frontul Pareto ideal și un așa numit punct de referință. Când însă frontul ideal nu este cunoscut (cazul problemelor complexe), HVideal reprezintă măsura volumului cuprins între axele ortogonale și punctul de referință Acest punct de referință are coordonatele date de valorile maxime ale obiectivelor.

Într-o problemă de minimizare, un HV mai mare înseamnă o calitate mai bună a soluțiilor de pe frontul Pareto real. Măsura HV poate fi folosită și pentru condiția de oprire a algoritmului genetic. Spre exemplu, într-o problemă de minimizare, când măsura HV nu mai crește semnificativ față de generația precedentă, algoritmul s-ar putea opri. O altă metrică este așa numita *Two Set Difference Hypervolume* (TSDH) și este definită ca: $TSDH(X',X'') = H(X'+X'') - H(X'')$, unde $X', X'' \subseteq X$ sunt două mulțimi de vectori de decizie, $H(X)$ este HV-ul aferent vectorului de decizie *X*, iar $X'+X''$ este vectorul de indivizi non-dominați obținuți prin reuniunea mulțimilor X' și X'' (practic, o superpoziție). $TSDH(X',X'')$ calculează HV din hiper-spațiul care este dominat de X', dar nu și de X''. Dacă [$TSHD(X_1, X_2) = 0$ *și* $TSHD(X_2, X_1) > 0$] atunci X_2 este absolut superior lui X_1.

O altă metrică de calitate, numită *Coverage*, compară doi algoritmi multi-obiectiv sau două rulări ale aceluiași algoritm. Această metrică va calcula procentajul de indivizi dintr-o populație care sunt dominați de indivizi din cealaltă populație. Considerând $X', X'' \subseteq X$ două mulțimi de vectori de decizie, funcția C (*Coverage*) mapează perechea ordonată de vectori de decizie (X', X'') în intervalul [0, 1]:

$$C(X',X'') = \frac{\left|\{a'' \in X''; \exists a' \in X' : a' \succeq a''\}\right|}{\left|X''\right|}$$

Dacă toate elementele din X'' sunt dominate (sau egale) de elementele din X' atunci $C(X',X'') = 1$. Dacă $C(X',X'') = 0$, atunci niciun element din *X''* nu este dominat de elemente din *X'*. Acest fapt nu înseamnă neapărat că soluțiile din prima mulțime sunt net mai bune decât cele din prima. În general, $C(X',X'') \neq C(X'',X')$.

4 LOGICI *FUZZY* ÎN ALGORITMII GENETICI DE OPTIMIZARE

Algoritmii de optimizare sunt prea generali, nefiind specializați pe o anumită problematică particulară (sistemele de calcul, la noi). Pentru a obține o convergență mai rapidă a algoritmilor multi-obiectiv de optimizare, dar și pentru o calitate mai bună a soluțiilor, autorul acestei cărți și colaboratorii săi au implementat utilizarea unor cunoștințe specifice de domeniu (*Domain-Knowledge*), exprimabile prin reguli în logici *fuzzy*, în cadrul acestor algoritmi. Această idee a fost publicată pentru prima dată în lucrările [Cal11] și [Jah12] la care și autorul acestei cărți a contribuit. Câteva dintre aceste reguli exprimate în logici fuzzy pentru o arhitectură parametrizabilă de procesor superscalar, sunt de genul: *IF IL1Cache_Size IS small AND DL1Cache_Size IS small THEN UL2Cache_size IS big or IF IL1Cache_Size IS big AND DL1Cache_Size IS big THEN UL2Cache_size IS small, IF Number_of_Physical_Register_Sets IS small/big THEN Decode/Issue/Commit_Width IS small/big* etc. Pentru fiecare astfel de regulă se generează un process de fuzificare → inferențe logice → defuzificare [Cal11].

Logicile *fuzzy* au la bază teoria mulțimilor *fuzzy* dezvoltată de *Lotfi Zadeh* începând din anul 1965. În teoria clasică a mulțimilor, funcția caracteristică asociată unei mulțimi A, pentru un anumit element *x*, este 1 sau 0, după cum x aparține respectiv nu

aparține mulțimii A. În teoria mulțimilor fuzzy această funcție caracteristică este una graduală, putând avea orice valoare reală între 0 și 1. Se pune problema cum se definesc conectorii logici în acest context. Răspunsul dat de Zadeh și de alți logicieni se bazează pe analogia cu algebra logică a lui Boole. Astfel, conectorii logici pot fi definiți astfel:

$$\mu_{\bar{A}} = 1 - \mu_A$$
$$\mu_{A \wedge B} = \mu_A \cdot \mu_B \text{ sau } \mu_{A \wedge B} = \min\{\mu_A, \mu_B\}$$
$$\mu_{A \vee B} = \mu_A + \mu_B - \mu_A \cdot \mu_B$$
$$\text{sau } \mu_{A \vee B} = \max\{\mu_A, \mu_B\},$$

unde μ este funcția caracteristică asociată mulțimii fuzzy respective.

Iată și câteva formule utilizate pentru operatorul de implicație materială în logica fuzzy:

$\mu_{A \to B}(x,y) = \max\big((1-\mu_A(x)), \mu_B(y)\big)$; derivă din logica clasică, unde implicația materială este definită A→B = (~A)VB = max (Valoare(~A), Valoare(B)). S-a considerat că $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$, $\mu_A(x) \in [0,1]$.

$\mu_{A \to B}(x,y) = \max\big[\min\big(\mu_A(x), \mu_B(y)\big), (1-\mu_A(x))\big]$;
L. Zadeh

$\mu_{A \to B}(x,y) = \min\big(1, 1-\mu_A(x) + \mu_B(y)\big)$;
Lukasiewicz

$\mu_{A \to B} = \mu_{\bar{A} \cup B} = 1 - \mu_A + \mu_A \cdot \mu_B$ (*Reichenbach*, naturală prin analogie cu logica clasică)

$\mu_{A \to B}(x,y) = \min\big(\mu_A(x), \mu_B(y)\big)$; *Mamdani*.

Arată ciudat în opinia autorului acestei cărți, pentru că $\mu_{A \wedge B}(x,y) = \min(\mu_A(x), \mu_B(y))$. Această ultimă formulă este naturală, fiind derivată prin analogie cu logica booleană unde: *V(A&B) = min (Valoare(A), Valoare(B))*. În aceste condiții, formula lui Mamdani pentru implicația materială este mai dificil de înțeles. În schimb, poate fi utilizată necontradictoriu în condițiile utilizării formulei alternative $\mu_{A \wedge B} = \mu_A \cdot \mu_B$.

Cu aceste formule logice definite, procesele de inferențe logice (ex. *Dacă A și B atunci C* etc.) sunt deci calculabile și în logicile fuzzy. O introducere intuitivă și rapidă în problematica logicilor fuzzy vizate aici, se găsește în referința web [Vin13]. Spre exemplu, să considerăm două reguli exprimate în logică fuzzy, într-o formă normal conjunctiva, ca mai jos:

- ***R1:*** *IF age IS young AND car-power IS high THEN risk IS high*
- ***R2:*** *IF age IS normal AND car-power IS medium THEN risk IS medium*

Mai întâi se vor defini funcțiile graduale de apartenență aferente variabilelor lingvistice implicate (fuzificare). În urma proceselor de inferențe logice și defuzificare, rezultă valoarea concretă a riscului pentru valorile concrete *age*=a și *car-power*=b, calculând centrul de greutate al funcției de apartenență rezultate. Defuzificarea se face pe baza determinării centrului de greutate (*Center of Gravity - COG*) a funcției de apartenență a obiectivului (obiectivul *risk*, în cazul prezentat), dat de relația

$$COG = \frac{\int x \cdot \mu(x) \mathrm{d}x}{\int \mu(x) \mathrm{d}x}$$

aproximată prin formula

$$COG_{approx} = \frac{\sum_{j=0}^{\max\_x} x_j \cdot \mu_j(x_j)}{\sum_{j=0}^{\max\_x} \mu_j(x_j)},$$

unde $\mu(x)$ este funcția de apartenență rezultată în urma proceselor de inferențe logice fuzzy.

În consecință, operatorul de mutație din algoritmul genetic multi-obiectiv a fost modificat astfel (pseudo-cod):

1. *Pentru toți parametrii dintr-un cromozom individual;*
   - 1.1. *Dacă există o regulă în logică fuzzy pentru parametrul curent care să îl aibă pe acesta consecvent (adică, urmează după THEN în regulă);*
     - 1.1.1. *Calculează centrul de greutate - Center Of Gravity (COG) al acestui parametru, luând în considerare valorile curente ale celorlalți parametri;*
     - 1.1.2. *Calculează valoarea funcției de apartenență $\mu(COG)$ a COG;*
     - 1.1.3. *Generează un număr cvasi-aleator între 0 și 1;*
     - 1.1.4. *Dacă numărul generat aleator anterior este mai mic decât probabilitatea de a efectua o mutație tip fuzzy;*
       - *1.1.4.1. Parametrul curent este setat la o valoare egală cu COG;*
     - 1.1.5. *Jump la următoarea iterație;*
   - 1.2. *Altfel (bit flip mutation);*
     - 1.2.1. *Generează un număr aleator între 0 și 1;*
     - 1.2.2. *Dacă numărul generat aleator anterior este mai mic decât probabilitatea mutației normale;*
       - *1.2.2.1. Modifică valoarea parametrului curent la o valoare aleatoare;*
     - 1.2.3. *Jump la următoarea iterație;*
2. *STOP.*

5. INSTRUMENTE DE OPTIMIZARE AUTOMATĂ, REZULTATE. META-OPTIMIZARE

S-a arătat în mod convingător, că astfel de cunoştinţe de domeniu implementate în algoritmii genetici multi-obiectiv conduc la soluţii mai bune, dar şi la un timp de rulare mai scăzut [Gel09, Gel12, Jah 12, Jah15, Rad13].

În Figura 3, preluată din lucrarea [Gel12], se arată fronturile Pareto într-un spaţiu bi-obiectiv (CPI – *Clocks Per Instruction* şi respectiv energia consumată), cu şi fără cunoştinţe de domeniu (exprimate prin reguli logice fuzzy). Indivizii optimizaţi se referă la anumite instanţe de procesoare superscalare cu procesări predictiv-speculative. Se remarcă faptul că domeniul specific de cunoştinţe a generat un front Pareto mai bun, în special pentru valori medii ale celor 2 obiective.

În vederea optimizării automate a sistemelor de calcul complexe s-a dezvoltat utilitarul FADSE (*Framework for Automatic Design Space Exploration*) de optimizare multi-obiectiv (v. https://github.com/horiacalborean/fadse), sub forma unui cadru software gratuit, şi care înglobează toate metodele succint expuse anterior [Cal10, Cal11]. Scopul lui este de a accelera procesul de DSE automat, nu doar prin utilizarea algoritmilor euristici multi-obiectiv, ci şi prin permiterea evaluării paralele a configuraţiilor.

Aplicaţia integrează şi o bază de date, care îi permite reutilizarea rezultatelor obţinute anterior (indivizi deja simulaţi), conducând deci la o scăderea a timpului necesar pentru explorare. Arhitectura cadrului FADSE este una de tip client – server. Pe server rulează algoritmii multi-obiectiv de optimizare, iar pe partea de client, simulatorul de optimizat. Cele două părţi se interconectează prin intermediul unui conector specific. FADSE permite evaluarea paralelă a indivizilor, prin calcul paralel (*multicore, High Performance Computer*) sau distribuit (reţele LAN). Aşadar, mai multe instanţe ale simulatorului de optimizat pot rula în paralel, în vederea optimizării. FADSE se configurează printr-un fişier XML. Din acest fişier se citesc parametrii simulatorului de optimizat, configuraţia acestuia etc. Există o componentă în FADSE care încarcă din biblioteca jMetal algoritmul de optimizare specificat în fisierul XML.

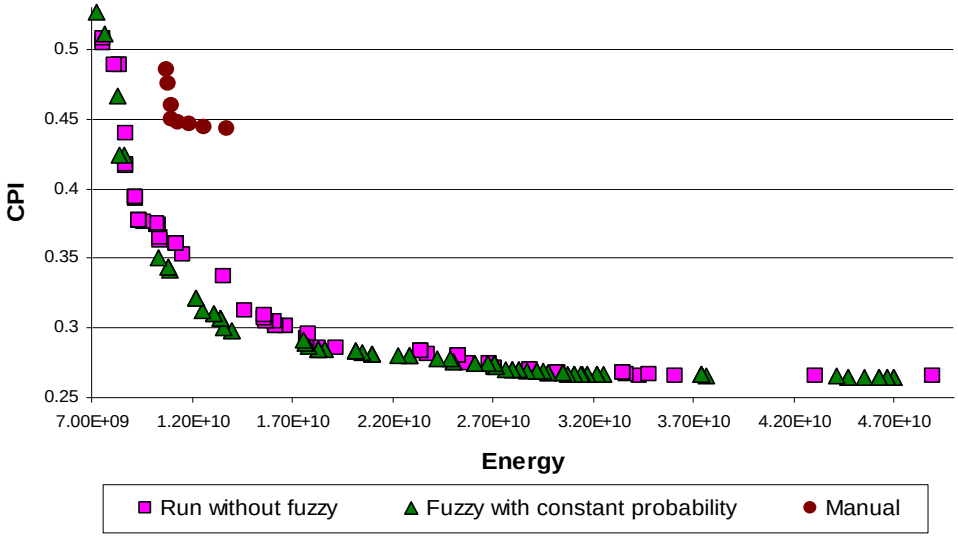

Fig. 3. Fronturile Pareto cu şi fără cunoştinţe de domeniu (fuzzy) [Gel12].

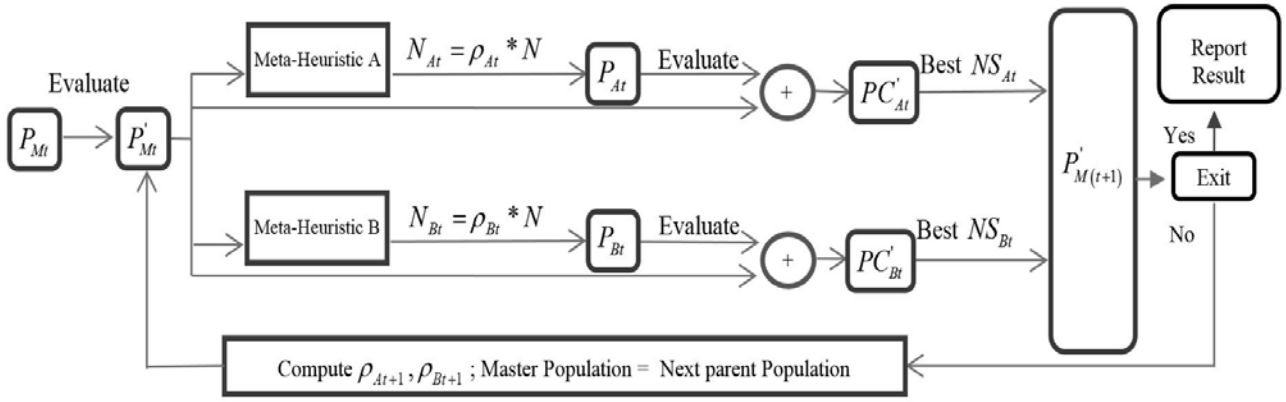

Fig. 4. Schema bloc a unui proces de meta-optimizare.

Figura 4 prezintă în mod intuitiv procesul de meta-optimizare dezvoltat de autor și colaboratorii săi în lucrarea [Vin15]. Populația master curentă, notată cu $P_{M(t)}$, este evaluată în mod automat, utilizând un simulator dedicat arhitecturii de calcul vizate. După acest proces, în general mare consumator de timp (din fericire poate fi paralelizat / distribuit), fiecare individ va avea asociată o rată de *fitness* pentru fiecare obiectiv. Această populație evaluată, notată cu $P'_{M(t)}$, este procesată în paralel de fiecare dintre cei doi algoritmi genetici multi-obiectiv A (NSGA-II în cazul concret) și B (SPEA2), prin intermediul propriilor operatori genetici. Algoritmul de optimizare A va selecta, în concordanță cu propriul algoritm de selecție, $\rho_{A(t)}N$ indivizi, aleși dintre cei mai performanți copii produși de acest algoritm (*offsprings*). Într-un mod analog, algoritmul B va genera $\rho_{B(t)}N=(1-\rho_{A(t)})N$ indivizi. Toți acești N noi indivizi produși, vor fi apoi evaluați în mod automat prin intermediul simulatorului arhitecturii de calcul vizate. Astfel, se stabilesc valorile obiectivelor acestor indivizi. După acest proces de evaluare, fiecare dintre aceste două seturi se vor reuni cu populația master curentă $P'_{M(t)}$. Apoi, prin intermediul mecanismelor specifice de selecție implementate în NSGA-II (A) și SPEA2 (B), vor fi obținuți cei mai buni indivizi din această generație (t), numiți NS_{At} și NS_{Bt}. Aceștia, reuniți, vor forma noua populație master $P'_{M(t+1)}$. S-a considerat că fiecare generație conține un număr de $N = NS_{At} + NS_{Bt}$ indivizi distincți (cromozomi). Această nouă populație master $P'_{M(t+1)}$ este apoi evaluată global, din punct de vedere calitativ, prin intermediul indicatorilor de calitate prezentați anterior, anume *Two Set Difference Hyper-volume (TSDH)* și *Coverage (C)*. Considerând acești doi algoritmi de optimizare (A și B), performanța globală (ρ) reprezintă o agregare între cei doi indicatori de calitate ($\widetilde{TSDH}$) și *Coverage* ($\tilde{C}$), după cum urmează:

$$\rho_{A(t+1)} = \alpha * \widetilde{TSDH}_{At} + (1-\alpha) * \tilde{C}_{At},$$
$$\rho_{B(t+1)} = \alpha * \widetilde{TSDH}_{Bt} + (1-\alpha) * \tilde{C}_{Bt},$$
$$\alpha \in [0,1]$$
$$\widetilde{TSDH}_{At} = 1 - \widetilde{TSDH}_{Bt} \text{ and } \tilde{C}_{At} = 1 - \tilde{C}_{Bt}$$
$$\rightarrow \rho_{A(t+1)} + \rho_{B(t+1)} = 1,$$

unde „*t*" reprezintă indexul generației numărul t.

Din păcate, procesul de selecție a indivizilor din cele două populații combinate, poate conduce la duplicări ale acestor indivizi (populația nu conține doar indivizi distincți). O soluție la această problemă este prezentată în [Vin15]. Conform acesteia, cei mai buni N indivizi selectați pot fi grupați, în acord cu cei doi algoritmi de optimizare, în $NSel_A$ indivizi selectați numai de algoritmul A, respectiv $NSel_B$ indivizi selectați numai de algoritmul B și $NSel_{AB}$ indivizi selectați de ambii. Dacă vom considera că algoritmul A a selectat un număr de NS_A indivizi într-o anumită generație și că algoritmul B a selectat NS_B indivizi, se pot scrie identitățile:

$$NS_A=NSel_A+NSel_{AB}$$
$$NS_B=NSel_B+NSel_{AB}$$

În total s-au obținut $NSel_A+NSel_B+2NSel_{AB}=N$ indivizi, însă numai $NSel_A+NSel_B+NSel_{AB}$ dintre aceștia sunt unici. Un număr de $NSel_{AB}$ dintre ei sunt duplicați. În consecință, este necesar să fie selectați în continuare $NSel_{AB}$ indivizi, astfel încât în final să se obțină N soluții unice. Este necesar să se determine ca algoritmul A să selecteze $nNSel_A$ indivizi în plus și, analog, algoritmul B să selecteze $nNSel_B$ indivizi adiționali. La sfârșit, fiecare algoritm a selectat nNS_A, respectiv nNS_B, indivizi, astfel încât:

$$nNS_A= NS_{A+}\ nNSel_A=NS_{A+}\ \rho_A\ NSel_{AB}$$
$$nNS_B= NS_{B+}\ nNSel_B=NS_{A+}\ \rho_B\ NSel_{AB}$$
$$nNS_A+ nNS_B=N$$

Duplicările pot apărea din nou. În acest caz soluția prezentată anterior se va aplica iterativ până când, în final, se vor selecta N indivizi unici. Această condiție va fi sigur îndeplinită întrucât în cel mai defavorabil caz $P_{M(t+1)} = P_{M(t)}$.

S-a demonstrat că metoda de meta-optimizare propusă generează rezultate mai bune decât cele două metode individuale, precum și în comparație cu fronturile Pareto superpoziționate, generate de cele două metode de optimizare. Mai mult, aceste rezultate superioare au fost obținute simulând doar jumătate din indivizi, în comparație cu metodele individuale. Practic: $HV(\text{Meta-Optimizare}) > HV(A \vee B)$, unde HV = hipervolum, *A (NSGA-II)* și *B (SPEA2)* iar $A \vee B$ semnifică reuniunea populațiilor generate de algoritmii A și B. Arhitectura țintă de optimizat a fost în acest caz procesorul superscalar numit *Grid ALU Processor* (GAP). Detalii se pot găsi în lucrarea [Vin15].

6 CONCLUZII

Având în vedere complexitatea enormă a sistemelor de calcul actuale sunt necesare metode de optimizare multi-obiectiv ale acestora. Abordările trebuie să fie holistice, abordând simultan atât subsistemul hardware cât și cel sofware. Aceste metode trebuie automatizate și utilizate intensiv în fazele de cercetare – proiectare. Elaborarea unor algoritmi de optimizare multi-obiectiv eficienți, atât din punct de vedere al timpului de rulare cât și din punct de

vedere al calității soluțiilor obținute, constituie o mare provocare științifică și tehnică. Metodele de meta-optimizare (optimizări hibride) sunt doar la începutul dezvoltării lor, în domeniul sistemelor de calcul. Acestea ar putea exploata eficient sinergismul diverşilor algoritmi euristici de optimizare multi-obiectiv [Vin16], fiind, credem noi, de mare interes în cercetările viitoare.

REFERINȚE BIBLIOGRAFICE

[Cal10] Calborean H., Vinţan L. – *An Automatic Design Space Exploration Framework for Multicore Architecture Optimizations*, Proceedings of The 9-th IEEE RoEduNet International Conference, pp. 202-207, ISSN 2068-1046**,** Sibiu, June 24-26, 2010**.**

[Cal11] H. Calborean, *Multi-Objective Optimization of Advanced Computer Architectures using Domain-Knowledge (Optimizarea multi-obiectiv a unor arhitecturi avansate de calcul utilizând cunoştinţe de domeniu*), PhD Thesis, „L. Blaga" University of Sibiu, November 25th 2011 (conducător ştiinţific: prof. univ. dr. ing. Lucian Vinţan).

[Gel09] Gellert A., Florea A., Vinţan L. – *Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture*, Journal of Systems Architecture, ISSN: 1383-7621, Elsevier, Volume 55, Issue 3, March 2009.

[Gel12] Á. Gellért, H. Calborean, L. Vinţan, A. Florea – *Multi-Objective Optimizations for a Superscalar Architecture with Selective Value Prediction*, IET Computers & Digital Techniques, United Kingdom, Vol. 6, Iss. 4, ISSN: 1751-8601, 2012.

[Jah12] Jahr R., Calborean H., Vintan L., Ungerer T. - *Boosting Design Space Explorations with Existing or Automatically Learned Knowledge*, The 16-th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB/DFT 2012), March 19-21, 2012.

[Jah15] Jahr R., Calborean H., Vinţan L., Ungerer T. – *Finding Near-Perfect Parameters for Hardware and Code Optimizations with Automatic Multi-Objective Design Space Explorations*, Concurrency and Computation: Practice and Experience, doi: 10.1002/cpe.2975, vol. 27, issue 9, ISSN 1532-0626, John Wiley & Sons, 2015.

[Rad13] C. Radu, Md. S. Mahbub, L. Vinţan – *Developing Domain-Knowledge Evolutionary Algorithms for Network-on-Chip Application Mapping*, Microprocessors and Microsystems, Vol. 37, Issue 1, pp. 65-78, ISSN: 0141-9331, Elsevier, February 2013.

[Vin13] Vinţan N. Lucian – *Automatic Multi-Objective Optimization of Mono-Core and Multi-Core Architectures using Domain-Knowledge* (62 PPT slides), Invited Scientific Presentation, CONTINENTAL R&D Sibiu Branch, March 4th 2013 – disponibilă online la adresa http://webspace.ulbsibiu.ro/lucian.vintan/html/Conti.pdf.

[Vin15] Vinţan L., Chis R., Md. Ali Ismail, Cotofana C. – *Improving Computing Systems Automatic Multi-Objective Optimization through Meta-Optimization***,** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, ISSN: 0278-0070, DOI 10.1109/TCAD.2015.2501299, 2015.

[Vin16] Vinţan L. - *Towards Synergic Meta-Algorithmic Approaches in Complex Computing Systems* (trimisă spre publicare 2016).

Despre autor

Prof. dr. ing. **Lucian N. VINȚAN**
Universitatea „Lucian Blaga" din Sibiu, Facultatea de Inginerie

Este expert în arhitectura sistemelor de calcul, optimizare multi-obiectiv și metode de *text-mining*. A publicat peste 140 de articole științifice în reviste prestigioase de specialitate (ex. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Concurrency and Computation: Practice and Experience, Journal of Systems Architecture, IET Computers & Digital Techniques, Microprocessors and Microsystems* etc.) precum și în conferințe internaționale de referință (*IEEE, ACM*) din România, SUA, Marea Britanie, Italia, Germania, Spania, China etc. Peste 40 dintre acestea sunt indexate/cotate *(ISI) Thomson Reuters*. Lucrările sale au înregistrat până în prezent peste 550 de citări internaționale independente, prin intermediul a peste 350 de publicații de certă ținută științifică. Are un indice *Hirsch H-index=13* (v. http://scholar.google.com/citations?user=9NiMZo4AAAAJ&hl=ro).
A publicat 6 cărți științifice (Editura Academiei Române, Editura Tehnică, Editura Matrix Rom, Editura Universității „Lucian Blaga" din Sibiu), dintre care două au fost redactate în limba engleză, fiind utilizate inclusiv în universități valoroase din străinătate. A finalizat în calitate de director de proiect 10 granturi de cercetare la nivel național/internațional, obținute prin competiții. A condus 6 teze de doctorat finalizate cu succes (una în co-tutelă cu Universitatea din Augsburg, Germania). A obținut titlul onorific de *Visiting Research Fellow* de la *University of Hertfordshire*, Marea Britanie (2002). În anul 2005 i s-a acordat Premiul „*Tudor Tănăsescu*" al Academiei Române, pentru o monografie publicată în 2003. În anul 2005 a fost ales membru corespondent, iar în anul 2012 membru titular al Academiei de Științe Tehnice din România. Din anul 2005 este expert activ al Comisiei Europene în domeniul sistemelor de calcul, fiind implicat în evaluarea și monitorizarea a zeci de proiecte europene de cercetare științifică. Din anul 2012 este membru al *European Network of Excellence on High Performance and Embedded Architecture and Compilation*. A fost membru al comitetelor științifice de program a peste 120 conferințe internaționale, în această calitate realizând sute de recenzii.