Liet. matem. rink, **47**, *spec. nr.*, 2007, 479–483

Genetinio algoritmo taikymas ir parametrų nustatymo problemos gamybinių tvarkaraščių sudarymui

Edgaras ŠAKUROVAS, Narimantas LISTOPADSKIS (KTU)
el. paštas: edgaras.sakurovas@stud.ktu.lt, narlis@ktu.lt

Reziumė. Genetiniai algoritmai priklauso aproksimacinių metaeuristinių algoritmų klasei. Šiame darbe nagrinėjamas jų taikymas trijų tipų gamybinių tvarkaraščių (darbų, srautinio ir atvirojo fabriko) sudarymui. Algoritmų tyrimas atliktas, siekiant nustatyti parametrų įtaką visų trijų tipų tvarkaraščių uždavinių sprendimui. Remiantis tyrimo rezultatais trys algoritmo strategijos ir jas atitinkančios parametrų kombinacijos yra pasiūlytos.

1. Įvadas

Viename iš klasikinių kombinatorinio optimizavimo apibrėžimų [1] teigiama, kad kombinatoriniame optimizavime ieškoma objekto iš baigtinės ar galimai baigtinės aibės. Šis objektas paprastai yra sveikasis skaičius, poaibis, permutacija ar grafinė struktūra, t.y. turime kombinatorinio optimizavimo problemą $P = (S, f)$; kintamųjų aibę $X = \{x_1, \ldots, x_n\}$; kintamųjų sritis $D_1, \ldots, D_n$; apribojimus tarp kintamųjų; tikslo funkciją f, kurią reikia minimizuoti (maksimizuoti), kur $f\colon D_1 \times \ldots \times D_n \to \mathbf{R}^+$; visų galimų priskyrimų aibę $S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n) | v_i \in D_i, s$ tenkina visus apribojimus$\}$. Čia S – paieškos (ar sprendinių) erdvė. Išspręsti kombinatorinę problemą, vadinasi rasti sprendinį $s^* \in S$ su minimalia tikslo funkcijos reikšme, t.y. $f(s^*) \leqslant f(s)$ $\forall s \in S$. s^* vadinamas globaliai optimaliu (S, f) sprendiniu ir aibė $S^* \subseteq S$ vadinama globaliai optimalių sprendinių aibe.

Tradiciniai lokalios paieškos metodai dauguma atvejų yra neefektyvūs, todėl taikomi metaeuristiniai algoritmai (gr. *heuriskein* – ieškoti, *meta* – virš, aukštesniame lygyje). Tai strategijos (dažniausiai turinčios realaus pasaulio analogiją), kurios „vadovauja“ paieškos procesui. Metaeuristiniai algoritmai yra apytiksliai ir paprastai nedeterminuoti.

Metaeuristikų klasei priklauso šie žymesni algoritmai: Skruzdžių Kolonijos Optimizavimas, Evoliuciniai Skaičiavimai, įskaitant Genetinius Algoritmus, Iteracinė Lokali Paieška, Dirbtinis Grūdinimas, Tabu Paieška ir kt.

Į klausimą, kurią strategiją taikyti gana vaizdžiai atsako viena iš „*No free lunch*“ teoremų [2], kuri teigia, kad vidutinis visų paieškos algoritmų visoms problemoms našumas yra lygus. Tokiu būdu, idėja yra ta, jog reikia naudoti tinkamą algoritmą tinkamai problemai.

2. Genetinio algoritmo taikymas

Genetiniai algoritmai priklauso evoliucinių skaičiavimų klasei. Jie yra paremti evoliucinės biologijos principais, tokiais kaip paveldėjimas, mutacija, atranka ir kryžminimas.

Anksčiau minėtąjį kanoninį genetinį algoritmą pritaikėme analizuojamam uždaviniui.

Kodavimas. Taikytas permutacinis kodavimas, kuriame pradinės $m \times n$ formato duomenų matricos A ir B transformuojamos atitinkamomis eilutėmis į $2 \times (m \cdot n)$ matricą X ($c = m \times n$ ilgio chromosomą), kuri atitinka darbų atlikimo eiliškumo tvarką. Akivaizdu, jog šiuo atveju iš viso yra $(m \cdot n)!$ galimų sprendinių.

Populiacija. Generuojame populiaciją iš pradžių sukurdami $p \times (m \cdot n)$ formato matricą P (iš matricų X), nagrinėjame jos eilutes (X), t.y. tų eilučių elementus X_j (stulpelius) nuo pradžių ($j = 1$), ties kiekvienu stulpeliu sugeneruojame atsitiktinį sveikąjį skaičių $rr \in (1; c)$, ir sukeičiame j-tąjį stulpelį su rr-tuoju, tada einame prie sekančio stulpelio ir taip iki tol, kol prieiname paskutinį. Tokiu būdu užpildoma visa matrica (populiacija).

Įvertinimo funkcija. Sukuriame matricą-stulpelį Pf, kuris susideda iš p eilučių. Kiekvieną iš eilučių užpildome $Pf_i = f(P_i)$, kur $1 \leqslant i \leqslant p$, t.y. matrica Pf yra visų populiacijos chromosomų įverčių matrica. Tam sukuriame įvertinimo funkciją $f(X)$, kuri įvertina kiekvieną chromosomą X, t.y. papildomai suformuojamos matricos stulpeliai Tr iš m eilučių, kuriame bus fiksuojamas i-tosios mašinos darbo pabaigos laikas duotuoju momentu ($1 \leqslant i \leqslant m$), ir D iš n eilučių, kuriame bus fiksuojamas j-tojo darbo pabaigos laikas duotuoju momentu ($1 \leqslant j \leqslant n$). Pradiniu laiko momentu abiejų matricų elementai yra 0. Iš eilės tikriname nagrinėjamos chromosomos stulpelius $k (1 \leqslant k \leqslant c)$. Pagal chromosomos kodavimą, 1-a jos eilutė yra mašinos numeris, o 2-oji – darbo numeris, vadinasi k-tajame tikrinime gauname dvi reikšmes: $e = X_{0,k}$ ir $s = X_{1,k}$, jas įsistačius į matricą, gauname s-tojo darbo trukmę e-tojoje mašinoje, t.y. Tuomet turime tokias sąlygas:

$$\begin{cases} Tr_e = D_s + t, & \text{jei } Tr_e < D_s, \\ Tr_e = Tr_e + t, & \text{jei } Tr_e \geqslant D_s, \end{cases}$$

bei $D_s = Tr_e$, t.y. jei nagrinėjamas s-tasis darbas tuo metu dar vykdomas kurioje nors kitoje mašinoje, vadinasi, reikia palaukti jo pabaigos ir tik tada pradėt vykdyt e-tojoje mašinoje, priešingu atveju s-tasis darbas iškart vykdomas e-tojoje mašinoje. Toks įvertinimas tinka atvirojo fabriko problemoms. Darbų ir srautinio fabriko problemoms reikia papildomai įvesti matricą-stulpelį Se iš n eilučių, kuriame būtų saugomi mašinų numeriai, kuriose atitinkamas darbas dar nebuvo, t.y. ar laikomasi mašinų eiliškumo tvarkos.

Galiausiai abiem atvejais chromosomos įvertis yra $\max_{0 \leqslant i \leqslant m}(Tr_i)$, kas atitinka vėliausiai pasibaigusį darbą i-tojoje mašinoje.

Atranka. Populiacijų atrankai įgyvendinti naudojama ruletės rato atrankos metodo modifikaciją, kurios esmė yra ta, kad net ir blogiausią įvertį turinti chromosoma turi šansą (nors ir nedidelį) pakliūti į atranką. Iš pradžių apskaičiuojamos reikšmės $l_{\max} = \max_{1 \leqslant i \leqslant p}(Pf_i)$, $\max = l_{\max} + l_{\max} \cdot 0.1$, $suma = \sum_{i=1}^{p}(\max - Pf_i)$, $koef =$

$100/suma$, tada sugeneruojamas atsitiktinis realus skaičius r iš intervalo $[0; 100)$ ir tikrinama sąlyga:

$$\begin{cases} nr1 = j, & \text{jei } \sum_{i=1}^{j}(\max - Pf_i)\cdot koef \geqslant r, \\ j = j+1, & \text{priešingu atveju,} \end{cases} \quad \text{čia } j = \overline{1, p}.$$

Kryžminimas. Turime dvi populiacijos P chromosomas P_{nr1} ir P_{nr2}. Sugeneruojame du atsitiktinius skaičius r ir rr, sveikąjį ir realų, atitinkamai iš intervalų $[1; c-1)$ ir $[0; 1)$. Tikriname, ar $rr \leqslant K$, jei taip atliekame chromosomų kryžminimą - tradicinį, t.y. iš P_{nr1} chromosomos paimame r pirmų stulpelių ir nukopijuojame į naująją chromosomą P'_{k+1}taip, kad elementai nesikartotų. Jei kryžminti nereikia, tai chromosomos-palikuonys P'_k ir P'_{k+1} yra tikslios chromosomų-tėvų atitinkamai P_{nr1} ir P_{nr2} kopijos.

Mutacija. Chromosomos mutacijai atlikti, iš pradžių, sugeneruojame du atsitiktinius sveikuosius skaičius $p1$ ir $p2$ iš intervalo $[1; c]$ bei vieną realų r iš intervalo $[0; 1)$. Tikriname, ar $r \leqslant M$, jei taip, vadinasi, sukeičiame mutuojančios chromosomos stulpelius, kurių indeksai $p1$ ir $p2$ vietomis.

Pakeitimas. Senoji populiacija P naująja populiacija P' pakeičiama chromosomų atrankos metodo pabaigoje: iš pradžių atrenkamos chromosomos kryžminimui, jos sukryžminamos ir įtraukiamos į naująją populiaciją P'. Iš viso atliekama $(p - Es)/2$ kryžminimų. Tada $p - Es$ populiacijos P' chromosomų mutuoja/nemutuoja, o likusioji dalis užpildoma geriausiomis chromosomomis iš P, neatliekant jokių pakeitimų, t.y. tomis chromosomomis, kurių įvertis yra mažiausias. Tokia strategija vadinama elitarizmu

Algoritmo pabaigos sąlyga yra apibrėžtas iteracijų skaičius $IterSk$.

3. Tyrimas

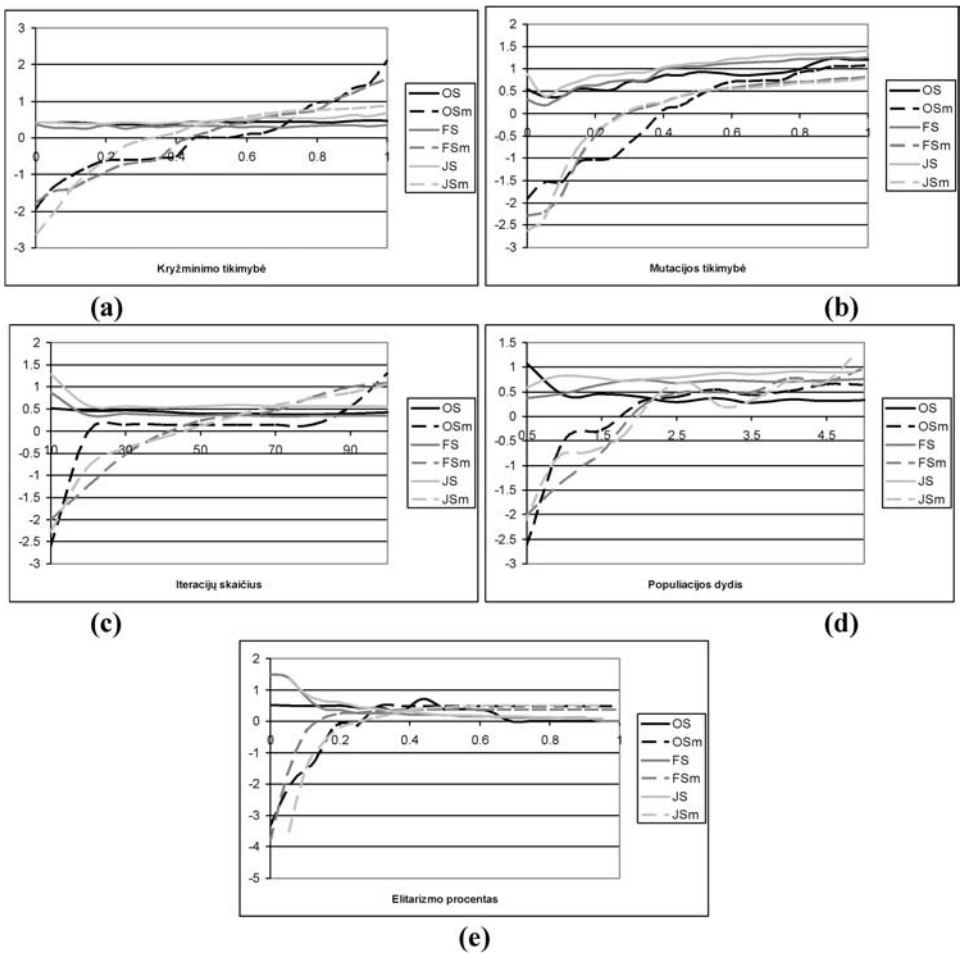
Algoritmo parametrų tyrimui buvo sukurta C++ kalbos programa „CGA-partsun“, joje įgyvendintas praeitame skyriuje minimas genetinis algoritmas, kuris pritaikytas klasikinėms gamybinio tvarkaraščių sudarymo problemoms. Duomenys programai paimti iš Vakarų Šveicarijos taikomųjų mokslų universiteto profesoriaus *Éric Taillard* puslapio [3].

4. Rezultatai

Grafiniai rezultatai pateikti 1 pav. Juose vientisa kreivė vaizduoja standartizuotų populiacijų sprendinių vidurkį, o punktyrinė – standartizuotus populiacijų minimumus, kintant atitinkamam parametrui. Juodos spalvos kreivė – atvirajam fabrikui (OS), pilkos – srautiniam (FS), šviesiai pilkos – darbų fabrikui (JS), jų minimumams – atitinkamai OSm, FSm, JSm kreivės.

Kryžminimo tikimybė. Iš 1 pav. (a) dalies pastebime, jog kintant kryžminimo tikimybei bendrasis populiacijos pajėgumas išlieka stabilus, tačiau minimumai akivaizdžiai gerėja, didėjant chromosomų kryžminimo tikimybei. Visgi, jeigu nenaudojama elitarizmo strategija, kryžminimo tikimybė turėtų būti $K < 1$.

Mutacija. Didinant mutacijos tikimybę (1 pav. (b) dalis), tiek populiacijos pajėgumas, tiek surastas minimumas gerėja. Tačiau tokią situaciją šiek tiek iškraipo elitarizmo strategija, kadangi surasti geriausi sprendiniai paliekami.



1 pav. a) Kryžminimo tikimybės parametro įtaka; b) Mutacijos tikimybės parametro įtaka; c) Iteracijų skaičiaus parametro įtaka; d) Populiacijos dydžio skaičiaus parametro įtaka; e) Elitarizmo procento parametro įtaka.

Elitarizmas. Iš 1 pav. (e) dalies pastebime, kad esant pradiniam parametrų parinkimui geriausia, kai visos chromosomos „dalyvauja“ populiacijos reprodukcijoj. Tačiau gali būti situacijų, kai geriausi sprendiniai nepaklius į sekančią kartą, todėl norint to išvengti reiktų, kad $E > 0\%$.

Iteracijų skaičius. Kuo didesnis iteracijų skaičius, tuo daugiau šansų, kad algoritmo metu bus išnagrinėta didesnė sprendinių erdvės dalis. Rasti minimumai didėjant iteracijų skaičiui, turi tendenciją didėti (1 pav. (c) dalis). Tačiau iteracijų skaičius tiesiogiai priklauso ir nuo laiko sąnaudų, kas realiose situacijose verčia palikti iteracijų skaičių vienodą arba priklausomą nuo užduoties sunkumo.

1 lentelė. Siūlomos parametrų reikšmės

Nr.	**Strategija**	***IterSk***	***P***	***K***	***M***	***E***, %
1	Stabilioji	50–100	$m \cdot n$	0.6-0.8	0.05-0.1	10–30
2	Godžioji	20–30	20–30	0.8-0.95	0.1-0.5	1–10
3	Mišrioji	50–100	$m \cdot n$	06-0.8 (0.8-0.95)	0.05-0.1 (0.1-0.5)	1–10

Populiacijos dydis. Kaip ir iteracijų skaičiaus atveju, pagrindinis kriterijus parenkant populiacijos dydį yra laiko sąnaudos, tačiau kaip ir matyti iš 1 pav. (d) dalies jis neturėtų būti per mažas ir geriausiai, jei tiesiogiai priklausytų nuo problemos sunkumo.

5. Išvados

Išanalizavus teoriją, atlikus tyrimą bei remiantis individualia patirtimi siūlome tokias parametrų kombinacijas – algoritmo strategijas (1 lentelė):

- chromosomų kryžminimas ir mutacija yra esminiai genetinio algoritmo operatoriai sprendinių sklaidos ir algoritmo konvergavimo atžvilgiu. Įvairios jų tarpusavio kombinacijos gali gerokai pagreitinti, tačiau ir sulėtinti algoritmo konvergavimą;
- elitarizmo procentas iš esmės nėra toks kritinis ir jis veikia daugiau kaip buferinis parametras norint būsimose populiacijoje neprarasti geriausio iš surastų lokalių minimumų;
- iteracijų skaičius ir populiacijos dydis mažiau svarbūs teoriškai, bet labai svarbūs praktiškai. Tarpusavy susiję ir turi būti siejami su sprendžiamo uždavinio kodavimu ir skaičiavimo resursais;
- gamybinių tvarkaraščių sudarymo atveju optimalių parametrų parinkimas galimas, bet visoms trims problemoms tai daryti yra netikslinga.

Literatūra

1. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, *ACM Computing Surveys*, **35**(3), 268–308 (2003).
2. `http://www.no-free-lunch.org/`
3. `http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html`

SUMMARY

E. Šakurovas, N. Listopadskis. Application of genetic algorithm to industrial scheduling and problems of parameters evaluation

Genetic algorithms are widely used in various mathematical and real world problems. They are approximate metaheuristic algorithms, commonly used for solving NP-hard problems in combinatorial optimisation. Industrial scheduling is one of the classical NP-hard problems. We analyze three classical industrial scheduling problems: job-shop, flow-shop and open-shop. Canonical genetic algorithm is applied for those problems varying its parameters. We analyze some aspects of parameters such as selecting optimal parameters of algorithm, influence on algorithm performance. Finally, three strategies of algorithm – combination of parameters and new conceptual model of genetic algorithm are proposed.

Keywords: scheduling problem, genetic algorithm, job shop, open shop, flow shop.