# Genetiniai algoritmai komivojažieriaus uždaviniui: negatyvieji ir pozityvieji aspektai*


## Alfonsas Misevičius

Kauno technologijos universiteto
Multimedijos inžinerijos katedros tech. m. dr., docentas
Kaunas University of Technology,
Department of Multimedia Engineering, Dr., Assoc. Prof.
Studentų g. 50–400a/416a, LT-51368 Kaunas
Tel. (8 37) 300 372
El. paštas: alfonsas.misevicius@ktu.lt

## Jonas Blonskis

Kauno technologijos universiteto
Multimedijos inžinerijos katedros tech. m. dr., docentas
Kaunas University of Technology,
Department of Multimedia Engineering, Dr., Assoc. Prof.
Studentų g. 50–400, LT-51368 Kaunas
Tel. (8 37) 300 370
El. paštas: jonas.blonskis@ktu.lt

## Andrius Blažinskas

Kauno technologijos universiteto
Multimedijos inžinerijos katedros doktorantas
Kaunas University of Technology,
Department of Multimedia Engineering, PhD student
Studentų g. 67–504, LT-51392 Kaunas
Tel. (8 37) 300 635
El. paštas: andrius.blazinskas@ktu.lt

## Vytautas Bukšnaitis

Kauno technologijos universiteto
Multimedijos inžinerijos katedros lektorius
Kaunas University of Technology,
Department of Multimedia Engineering, Lecturer
Studentų g. 50–400, LT-51368 Kaunas
Tel. (8 37) 300 370
El. paštas: vytautas.buksnaitis@ktu.lt


*Šiame straipsnyje nagrinėjami klausimai, susiję su genetinių algoritmų taikymu, sprendžiant gerai žinomą kombinatorinio optimizavimo uždavinį – komivojažieriaus uždavinį (KU) (angl. traveling salesman problem). Svarstoma, jog genetinio algoritmo efektyvumui didelę įtaką turi uždavinio specifinės savybės, todėl labai svarbu kūrybiškai sudaryti genetinį algoritmą konkrečiam sprendžiamam uždaviniui. Pateikiami eksperimentų, atliktų su realizuotu genetiniu algoritmu, rezultatai, iliustruojantys skirtingų veiksnių įtaką rezultatų kokybei. Konstatuojama, kad tinkamas genetinių operatorių ir lokaliojo individų (sprendinių) gerinimo derinimas leidžia gerokai padidinti genetinės paieškos efektyvumą.*

## Įvadas

Reikšmingą optimizavimo metodų grupę sudaro genetiniai algoritmai (GA) (angl. *genetic algorithms*), kurie pagrįsti evoliucinių procesų mus supančiame objektyviame pasaulyje imitavimu (Reeves, 2002; Sivanandam, Deepa, 2008). Dėl universalaus pobūdžio ir nepriklausomumo – tai reiškia, jog sprendimo būdas nepriklauso nuo sprendžiamo uždavinio – šie algoritmai gali būti taikomi iš esmės bet kokioms optimizuotinoms struktūroms, pradedant elementariomis matematinėmis funkcijomis ir baigiant sudėtingomis save reguliuojančiomis sistemomis. Natūralu, kad genetiniais algoritmais domisi nemažai tyrinėtojų ir spręsdami tokį sunkų optimizavimo uždavinį kaip

* Darbą remia Lietuvos valstybinis mokslo ir studijų fondas (registracijos Nr. T-09293).

komivojažieriaus uždavinys (KU) (angl. *traveling salesman problem – TSP*) (Applegate ir kt., 2007), kuris jau daugelį metų išlieka tikru iššūkiu inovatyvių optimizavimo metodų kūrėjams (Goldberg, Lingle, 1985; Potvin, 1996; Nguyen ir kt., 2002; Zou ir kt., 2004; Marinakis ir kt., 2005).

Šiame darbe nagrinėjami aspektai, susiję su genetinių algoritmų naudojimu sprendžiant būtent komivojažieriaus uždavinį. Iš pradžių trumpai apžvelgiami bendrieji genetinių algoritmų klausimai. Paskui nagrinėjami kai kurie konkretūs GA variantai komivojažieriaus uždaviniui, taip pat pateikiami atliktų eksperimentų rezultatai. Kartu apsvarstomi negatyvieji ir pozityvieji aspektai, susiję su tų variantų taikymu KU.

## Genetiniai algoritmai: bendrieji klausimai

Nepaisant išmintingos gamtos procesų imitavimo idėjos, aukštas genetinių algoritmų efektyvumas savaime dar negarantuojamas. Pirmiausia, genetiniai algoritmai – kaip ir kiti euristiniu principu besiremiantys metodai – neužtikrina, jog gaunami sprendiniai yra (globaliai) optimalūs, kalbama tik apie artimų optimaliems (*lokaliai optimalių*) sprendinių suradimą per priimtiną laiką. Taip pat pabrėžtina, jog GA efektyvumas daugiau ar mažiau priklauso nuo konkretaus sprendžiamo uždavinio, jo prigimties, ypatumų. GA efektyvumo laipsnis priklauso ir nuo, taip sakant, vidinių paties algoritmo savybių, faktorių. Pagrindiniai GA sunkumai, su kuriais dažniausiai susiduriama, yra šie: *priešlaikinis konvergavimas* (angl. *premature convergence*) ir evoliucinio paieškos proceso *stagnacijos fenomenas* (angl. *stalled evolution* (*stagnation*). Pastarasis savo ruožtu glaudžiai siejasi su diversifikacijos (t. y. genetinio kodo įvairoviškumo praradimo (angl. *loss of genetic variability*)) problema.

Neigiami GA aspektai gali būti daugiau ar mažiau atsveriami jų teigiamų savybių. Visų pirma, genetiniai algoritmai yra labai protinga kopija to, kas jau pačios gamtos sukurta per milijardus metų. Čia turima omenyje pasinaudojimas tiek natūraliosios atrankos, tiek palikuonių kūrimo iš dviejų tėvų, tiek atsitiktinių mutacijų, svarbių naujoviškumui generuoti, principais. Iš tiesų, santykinai determinuotas informacijos paveldėjimas iš tėvų ir perdavimas palikuonims (iš vienos pusės) ir atsitiktinio pobūdžio mutacijos, galimos tiek dėl paties vidinio kryžminimosi, tiek dėl įvairių išorinių veiksnių (iš kitos pusės), yra tie vienas kitą papildantys, sustiprinantys veiksniai, kurie sudaro prielaidas siekti daug didesnio algoritmo veikimo efektyvumo. Svarbi ir ta savybė, jog GA veikimas yra grindžiamas sprendinių rinkinių, t. y. populiacijų, panaudojimu, o kiti (klasikiniai) algoritmai paprastai operuoja vienu sprendiniu. Taigi genetinę paiešką galime asocijuoti su lygiagrečiu procesų vykdymu, savotišku bendru individų būrio judėjimu hipotetinėje paieškos erdvėje, o ne tik atskirų sprendinių manipuliacijomis ir nekoordinuotu „klaidžiojimu“ – tai dar vienas labai svarbus apriorinis GA pranašumas prieš kitus algoritmus (visgi neužmirštant, jog tikriausiai neegzistuoja kuri nors viena geriausia visų uždavinių sprendimo algoritmų klasė). Pagaliau reikia pasakyti, kad genetinių algoritmų konceptualinis turiningumas (įvairių sudedamųjų struktūrinių komponentų egzistavimas) suteikia potencialių galimybių formuoti skirtingus GA realizavimo variantus, kurti įvairias naujas efektyvias algoritmų modifikacijas, kombinacijas, derinius, tarp jų ir vadinamuosius *hibridinius* (arba *memetinius*) algoritmus.

## Genetinio algoritmo variantai komivojažieriaus uždaviniui

Komivojažieriaus uždavinys savo prigimtimi yra kombinatorinio (diskretinio) pobūdžio. Abstraktų kombinatorinio optimizavimo uždavinį formaliai galima apibrėžti pora (*S*, *f*); čia *S* žymi diskretinių sprendinių (kintamųjų) aibę, o *f* – skaliarinę tikslo funkciją, kurios apibrėžimo sritis – *S*, o reikšmių aibė – realieji skaičiai. Tuomet išspręsti uždavinį (*S*, *f*) reiškia surasti sprendinį $\breve{s} \in S$ ir

tokį, kad $\breve{s} \in \breve{S} = \left\{ s^{\nabla} \mid s^{\nabla} = \arg\min_{s \in S} f(s) \right\}$

(laikoma, kad tikslo funkcija turi būti minimizuojama; sprendinys $\breve{s}$ vadinamas (globaliai) optimaliu sprendiniu). Komivojažieriaus uždavinio atveju sprendinių aibės vaidmenį atlieka sveikųjų skaičių nuo 1 iki *n* visų galimų perstatymų aibė

$\Pi_n$, o tikslo funkcija apibrėžiama taip:

$$z(p) = \sum_{i=1}^{n-1} d_{p(i)p(i+1)} + d_{p(n)p(1)}; \qquad (1)$$

čia $p = (p(1), p(2), ..., p(n))$ žymi perstatymą iš aibės $\Pi_n$, kitaip tariant, maršrutą (seką), sudarytą iš $n$ miestų ($i$-asis perstatymo $p$ elementas $j_i = p(i)$ ($i = 1, 2, ..., n$) šiuo atveju nusako $i$-ąjį aplankytą maršruto miestą $j_i$). Perstatymo elementų poros $(p(1), p(2))$, ..., $(p(i), p(i + 1))$, ..., $(p(n), p(1))$ vadinamos maršruto atkarpomis. Skirtumas (arba Hammingo atstumas) tarp maršrutų $p$ ir $p'$ gali būti aprašytas tokia formule:

$$\rho(p, p') = \left| \Omega \right|; \qquad (2)$$

čia aibė $\Omega$ yra sudaryta iš visų galimų porų $(p(i), p((i \bmod n)+1))$ ($i = 1, 2, ..., n$) ir tokių, kad $\neg\exists\, j$ toks, jog:

$$(p(i), p(\, i \bmod n) + 1) = \begin{cases} (p'(j), p'(j+1)), 1 \le j < n \\ (p'(j), p'(1)), \; j = n \end{cases}$$

arba

$$(p(i), p(\, i \bmod n) + 1) = \begin{cases} (p'(j), p'(j-1)), 1 < j \le n \\ (p'(j), p'(n)), \; j = 1 \end{cases} \qquad (3)$$

Atstumai tarp miestų yra aprašyti kvadratinės matricos $\boldsymbol{D} = (d_{ij})_{n\times n}$ elementais (šiuo atveju $d_{p(i)p(i+1)}$, $d_{p(n)p(1)}$ ($i = 1, 2, ..., n-1$) nurodo atstumą tarp atitinkamų gretimų maršruto miestų (atitinkamų maršruto atkarpų ilgį), įskaitant atstumą tarp pirmojo ir paskutiniojo maršruto miestų). Taigi, $z(p)$ apibrėžia bendrą maršruto $p$ ilgį, o komivojažieriaus uždavinys gali būti trumpai formuluojamas kaip tikslas rasti trumpiausią įmanomą (uždarą) maršrutą, kai miestas aplankomas tik vieną kartą.

---

**procedure** Genetinis_Algoritmas_Komivojažieriaus_Uždaviniui;
//pradiniai duomenys: $n$ – uždavinio apimtis (miestų skaičius), $D$ – atstumų matrica, $PD$ – populiacijos dydis, $N_{gen}$ – generacijų skaičius,
// σ – atrankos koeficientas, $GS$ – gerinimo schema, $KV$ – Krosoverio variantas, $DS$ – diversifikacijos slenkstis
//rezultatai: $p^*$ – geriausias rastas sprendinys, t. y. miestų perstatymas (maršrutas)
**begin**
  sukurti pradinę populiaciją $P \subset \Pi_n$ ($|P| = PD$);
  **if** ($GS = 10$) **or** ($GS = 11$) **then**
    optimizuoti kiekvieną populiacijos $P$ sprendinį, panaudojant lokaliojo pagerinimo algoritmą PGN-AP;
  $p^* := \underset{p \in P}{\operatorname{argmin}} z(p)$; //$p^*$ – žymi geriausią populiacijos narį (sprendinį)
  **for** $i := 1$ **to** $N_{gen}$ **do begin** //pagrindinis ciklas
    surūšiuoti populiacijos $P$ narius tikslo funkcijos (maršruto ilgio) didėjimo tvarka; $P' := \varnothing$;
    parinkti „sprendinius tėvus“ $p', p'' \in P$;
    **if** ($KV = 1$) **then** $p''' :=$ Atsitiktinio_Rakto_Krosoveris($p', p''$) **else** $p''' :=$ Dalinio_Atvaizdavimo_Krosoveris($p', p''$);
    **if** ($GS = 01$) **or** ($GS = 11$) **then** optimizuoti gautą palikuonį $p'''$, panaudojant pagerinimo algoritmą PGN-AP;
    $P' := P' \cup \{p'''\}$;
    **if** $z(p''') < z(p^*)$ **then** $p^* := p'''$; //įsimenamas geriausias rastas sprendinys (maršrutas)
    suformuoti naują populiaciją $P$ ($|P| = PD$) iš sąjungos $P \cup P'$, atmetant blogiausią sprendinį;
    **if** (populiacijos $P$ diversifikacijos lygis yra mažesnis už $DS$) **then begin**
      (1) vykdyti mutavimo procedūrą visiems populiacijos $P$ nariams, išskyrus geriausią;
      (2) optimizuoti kiekvieną iš mutuotų sprendinių, panaudojant pagerinimo algoritmą PGN-AP;
      if $z(\underset{p \in P}{\operatorname{argmin}} z(p)) < z(p^*)$ **then** $p^* := \underset{p \in P}{\operatorname{argmin}} z(p)$
    **endif**
  **endfor**
**end.**

---

Pastabos. 1. Atstumų matrica $D$, jei reikia, suformuojama atskiroje programoje. 2. Algoritmo variantai priklauso nuo parametro „$GS$“, kuris gali įgyti šias reikšmes: 00 – be gerinimo; 01 – pradinė populiacija be gerinimo, tačiau naudojamas pokryžminis gerinimas; 10 – generuojama pagerinta pradinė populiacija, tačiau nenaudojamas pokryžminis gerinimas; 11 – naudojama ir pagerinta pradinė populiacija, ir pokryžminis gerinimas. 3. Sukuriant pradinę populiaciją, naudojamas artimiausio kaimyno (AK) algoritmas (Rosenkrantz ir kt., 1977). 4. Lokaliajam pagerinimui naudojamas patobulintas greito nusileidimo-atsitiktinio pakilimo algoritmas (PGN-AP) (Misevičius ir kt., 2007).

*1 pav.* ***Genetinis algoritmas komivojažieriaus uždaviniui***

Sudarytame genetiniame algoritme, kurio detalizuotas aprašymas pateikiamas 1 pav., perstatymai (miestų maršrutai) $p^{(1)}$, $p^{(2)}$, ..., $p^{(k)}$, ... yra tiesiogiai asocijuojami su chromosomomis, taigi papildomo sprendinių kodavimo nereikia. Sprendinio (perstatymo) $p^{(k)}$ ($k$ = 1, 2, ...) elementą $p^{(k)}(i)$ ($i$ = 1, 2, ..., $n$) galime traktuoti kaip atskirą geną (tiksliau, to geno turinį – alelį), esantį $k$-osios chromosomos $i$-ajame lokuse (pozicijoje).

Mūsų algoritmas operuoja labai kompaktiška sprendinių populiacija ($P$), kurią sudaro tik penki maršrutai ($PD$ = | $P$ | = 5). „Sprendinių tėvų“ pozicija surūšiuotoje populiacijoje nustatoma pagal formulę: $u = \lfloor v^{\sigma} \rfloor$; čia $u$ žymi populiacijos pozicijos indeksą, o $v$ yra atsitiktiniu būdu generuojamas skaičius iš intervalo [1, $PD^{1/\sigma}$ ] ($PD$ yra populiacijos dydis, o $\sigma$ – realusis skaičius iš intervalo [1, 2] (atrankos koeficientas)) (Tate, Smith, 1995). „Sprendinių tėvų“ kryžminimui yra išbandytos dvi procedūros: atsitiktinio rakto (AR) (angl. *random key*) krosoveris* (Syswerda, 1989) ir dalinio atvaizdavimo (DA) (angl. *partially-mapped*) krosoveris** (Goldberg, Lingle, 1985) (žr. taip pat 2, 3, 4 pav).

```
function Atsitiktinio_Rakto_Krosoveris(p′, p″);
//pradiniai duomenys: p′, p″ – „sprendiniai tėvai“ (sprendinių apimtis – n); rezultatai: p‴ – „sprendinys palikuonis“
begin
  sugeneruoti atsitiktinį raktą r′, kurio ilgis – n, o reikšmės – iš intervalo
   [1, M] (M > max { p′(1), p′(2), ..., p′(n) });
  sugeneruoti atsitiktinį raktą r″, kurio ilgis – n, o reikšmės – iš intervalo
   [1, N] (N > max { p″(1), p″(2), ..., p″(n) });
  sudaryti raktą r‴, kuris gaunamas, vieną dalį „genų“ kopijuojant iš pirmojo tėvo
  (rakto r′), o likusią dalį – iš antrojo tėvo (rakto r″);
  surūšiuoti rakto r‴ reikšmes (kartu su indeksais) jų didėjimo tvarka;
  perkelti surūšiuotas rakto reikšmes atitinkančių indeksų seką i1, i2, ..., in (ik ∈ {1, ..., n}, k = 1, 2, ..., n)
  į „sprendinį palikuonį“ p‴;
  return p‴
end.
```

*2 p a v*. ***Atsitiktinio rakto krosoveris***

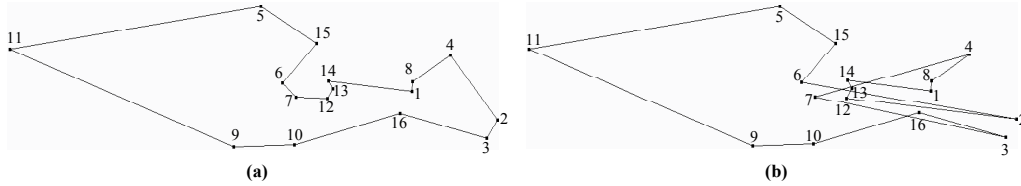
* Atsitiktinio rakto krosoverio veikimo pavyzdys. Tarkime, $n$ = 10 ir kryžminami perstatymai, t. y. maršrutai ($p'$, $p''$) bei sugeneruoti atsitiktiniai raktai ($r'$, $r''$) yra tokie: pirmas maršrutas ($p'$): 3, 4, 5, 1, 10, 9, 6, 7, 8, 2; antras maršrutas ($p''$): 2, 4, 10, 6, 7, 9, 8, 3, 5, 1; raktas $r'$: 42, 35, 50, 18, 12, 71, 64, 69, 75, 38; raktas $r''$: 24, 40, 72, 55, 66, 53, 68, 39, 44, 10. Kopijuojant 5 „genus“ iš pirmojo „rakto tėvo“ ir tiek pat „genų“ iš antrojo, galėtų būti gautas toks rekombinuotas raktas ($r'''$): 42, 35, 50, 18, 12, 53, 68, 39, 44, 10. Surūšiavus šio rakto reikšmes didėjimo tvarka, gaunama tokia surūšiuotas rakto reikšmes atitinkančių indeksų seka (tuo pačiu kryžminimo rezultatas – „maršrutas palikuonis“): 6, 4, 8, 3, 2, 9, 10, 5, 7, 1. Indeksai nesikartoja, taigi gautas maršrutas yra korektiškas; tiesa, skirtumas (žr. (2) formulę) tarp „tėvų“ ir „palikuonio“ yra labai didelis (tarp pirmojo „tėvo“ ir „palikuonio“ jis lygus 8, tarp antrojo „tėvo“ ir „palikuonio“ – 9). Tai yra nemažas šios ir kitų panašių kryžminimo procedūrų, neatsižvelgiančių į maršrutų atkarpų (atkarpų grupių) išsaugojimą, trūkumas.

** Dalinio atvaizdavimo krosoverio veikimo pavyzdys. Tarkime, yra kryžminami tie patys sprendiniai, kaip ir pirmiau pateiktame pavyzdyje. Be to, sakykime, kad atvaizdavimo koeficientas ($\lambda$) lygus 0,4; tuomet atvaizduojamo segemento dydis (miestų skaičius) yra lygus 4. Atlikus dalinį pirmojo sprendinio atvaizdavimą (atvaizdavus 4 miestus, pradedant nuo pirmojo), gaunamas dalinis sprendinys $\langle p \rangle$: 3, 4, 5, 1. Tarkime, jog yra atsitiktiniu būdu parinkta 4-oji sprendinio $p''$ pozicija. Tuomet pakanka tik vieno bandymo, kad būtų suformuotas naujas atvaizduojamas segmentas, kuris sudarytas taip pat iš 4 elementų: 6, 7, 9, 8. Dalinio sprendinio $\langle p \rangle$ apimtis padvigubėja : 3, 4, 5, 1, 6, 7, 9, 8. Pilnas „sprendinys palikuonis“ $p'''$ sudaromas iš dalinio sprendinio $\langle p \rangle$ ir likusių „laisvų“ elementų (miestų): 2, 10. Taigi, galutinis „sprendinys palikuonis“ $p'''$ yra toks: 3, 4, 5, 1, 6, 7, 9, 8, 2, 10. Kaip ir pirmajame pavyzdyje, gautas maršrutas yra leistinas. Tačiau maršrutų skirtumo parametro ($\rho$) reikšmės yra daug mažesnės ($\rho(p', p''') = \rho(p'', p''') = 5 = n/2$). Tai reiškia, kad sukuriamas naujas palikuonis yra artimesnis savo tėvams, geriau perima tėvų genetinę informaciją, jeigu lygintume su AR kryžminimu.

```
function Dalinio_Atvaizdavimo_Krosoveris(p′, p″);
//pradiniai duomenys: p′, p″ – „sprendiniai tėvai“; parametrai: λ – atvaizdavimo koeficientas (0 < λ ≤0,5), q – bandymų skaičius
  (q ≥1)
//rezultatai: p‴ – „sprendinys palikuonis“
begin
  K := max(1, ⌊λn⌋);  //nustatomas atvaizduojamo segmento (atkarpų grupės) dydis (ilgis)
  kopijuoti K elementų iš esamo sprendinio p′ į naujai sudaromą „sprendinį palikuonį“ p‴;
  naujas_segmentas := ∅;
  for iteracija := 1 to q do begin  //bandymų ciklas vykdomas q kartų
    bandomasis_segmentas := ∅; j := random(1, n);  //atsitiktiniu būdu parenkama sprendinio p″ pozicija (j)
    for i := 1 to K do begin
      jeigu elemento p″(j) dar nėra „sprendinyje palikuonyje“ p‴,
      tai bandomasis_segmentas := bandomasis_segmentas ∪ {p″(j)};
      j := (j mod n) + 1 endfor;
    if |bandomasis_segmentas| > |naujas_segmentas| then naujas_segmentas := bandomasis_segmentas;
    if |naujas_segmentas| = K then break endfor;
  perkelti elementus iš gauto segmento naujas_segmentas į „sprendinį palikuonį“ p‴;
  užbaigti formuoti „sprendinį palikuonį“ p‴ atsitiktiniu arba determinuotu (euristiniu)
  būdu;
  return p‴
end.
```

*3 pav.* ***Dalinio atvaizdavimo krosoveris***


*4 pav.* ***Atsitiktinio rakto (AR) ir dalinio atvaizdavimo (DA) kryžminimo iliustracija. (Iliustracijoje vaizduojami maršrutai, gauti sprendžiant KU testinį 16 miestų pavyzdį ulysses16 („Odyssey of Ulysses“ - „Uliso kelionė“) iš KU pavyzdžių (gairių) bibliotekos TSPLIB (Reinelt, 1991))***




*5 pav.* ***Porinio elementų sukeitimo mutacijos iliustracija: (a) - maršrutas prieš mutaciją; (b) - maršrutas po mutacijos (maršruto miestų 2 ir 7 aplankymas sukeistas vietomis). (Iliustracijoje panaudotas tas pats KU pavyzdys ulysses16)***

Mutavimo procedūros, panašiai kaip ir kryžminimo operatoriai, turi būti sudaromos, atsižvelgiant į KU sprendinių specifiką. Iš pirmo žvilgsnio gali pasirodyti, kad mutavimo procedūrai realizuoti yra tinkama standartiniuose genetiniuose algoritmuose dažnai taikoma „bitų apkeitimo“ metodika: čia pakanka sukeisti vietomis du ar daugiau perstatymo elementų. Tačiau trumpai paanalizavus 5 pav. pavaizduotą iliustraciją, kurioje kaip tik pateikiamas vienas iš galimų „bitų apkeitimo“ pavyzdžių, nesunku pastebėti šios metodikos trūkumą: esama maršruto struktūra pernelyg stipriai, chaotiškai deformuojama, nors pati mutavimo operacija yra labai elementari (deformacijos būtų dar akivaizdesnės, padidinus sukeičiamų bitų skaičių). Matyti, kad sprendžiant KU, reikia ieškoti kitokių mutavimo metodų – tokių, kurie būtų grindžiami ne pavienių genų (atskirų miestų) operacijomis, bet stambesnių genų junginių (maršruto atkarpų grupių, segmentų) tam tikrais pertvarkymais. Tokių mutavimo procedūrų pavyzdžiai

yra vadinamoji „keturgubo tilto“ (angl. *four-fold-bridge*) mutacija ir mutavimo algoritmas, besiremiantis dalinių maršrutų, t. y. submaršrutų, transformavimu taikant artimiausio kaimyno (AK) euristiką (angl. *nearest neighbour reconnection*) (Misevičius ir kt., 2007). Taikant pastarąjį mutavimo metodą, turi būti atsižvelgta į tai, kad atsitiktinumo veiksnio nepersvertų AK euristikos „godusis“ principas. Reikiamą stochastiškumo laipsnį galima užtikrinti atsitiktiniu būdu parenkant dalinius maršrutus ir tam tikru mastu randomizuojant AK procedūrą.

Realizuojant šių tipų mutavimo procedūras mūsų algoritme, nukrypstama nuo kanoninių GA taisyklių. Iš tikrųjų, mutavimo procedūros yra panaudojamos ne tiesiogiai pačiame genetiniame algoritme, bet sprendinių pokryžminio lokaliojo gerinimo stadijoje. Šiam tikslui naudojamas į GA inkorporuotas patobulintas greito nusileidimo-atsitiktinio pakilimo algoritmas (PGN-AP) (angl. *enhanced fast descent-random ascent*) (Misevičius ir kt., 2007). Šis algoritmas gali būti naudojamas ir pagerintai pradinei populiacijai konstruoti. Konkreti sprendinių gerinimo schema, sykiu ir GA konfigūracija apibrėžiama valdymo parametru „*GS*“ (žr. 1 pav.).

Mutavimas (būtent randomizuotas submaršrutų transformavimas) vis dėlto yra panaudojamas ir tiesiogiai genetiniame algoritme, bet tik tais atvejais, kai prarandama populiacijos genetinė įvairovė. Kaip genetinio įvairoviškumo indikatorius naudojamas normalizuotas populiacijos diversifikacijos lygis (ξ), kuris apskaičiuojamas pagal tokią formulę:

$$\xi(P) = 2\sum_{k=1}^{PD-1}\sum_{l=k+1}^{PD}\rho(p^{(k)}, p^{(l)}) \Big/ PD(PD-1)n; \quad (4)$$

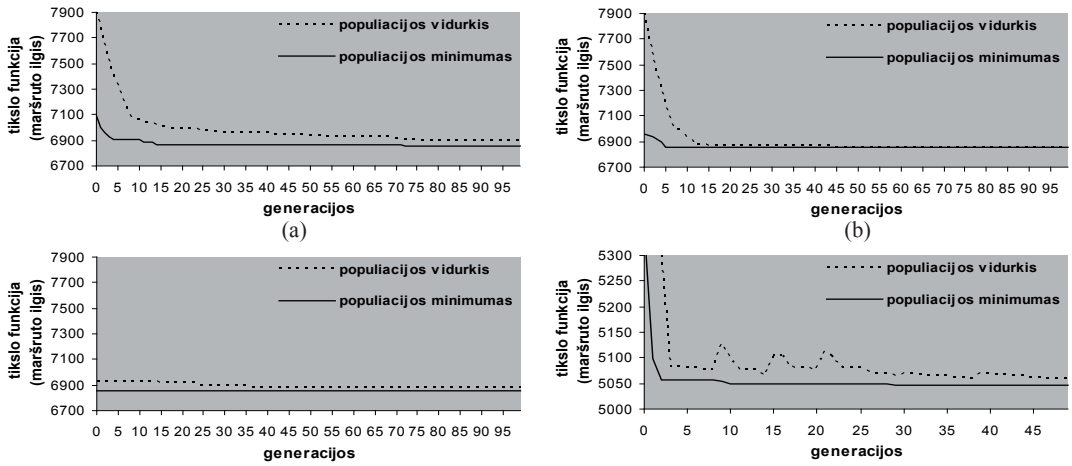
čia $p^{(k)}$, $p^{(l)}$ ($k = 1, ..., PD–1$, $l = k+1, ..., PD$) žymi populiacijos ($P$) atitinkamai $k$-ąjį ir $l$-ąjį sprendinį (maršrutą); maršrutų skirtumo parametras ρ apskaičiuojamas pagal (2) formulę; $PD$ yra populiacijos dydis ($PD = |P|$), o $n$ – uždavinio dydis. Tuomet populiacijos įvairovės praradimo sąlyga galima laikyti situaciją, kai patenkinama nelygybė: $\xi(P) < DS$; čia $DS$ yra iš anksto fiksuotas parametras, daugiau ar mažiau artimas 0 (tuo atveju, jeigu $DS = 0$, duota nelygybė niekuomet negalėtų būti patenkinta ir papildomas mutavimas nebūtų atliekamas). Jei nelygybė patenkinta, mutavimas atliekamas visiems populiacijos sprendiniams, išskyrus geriausią. Rekomenduojama, kad šiuo atveju mutavimo randomizavimo laipsnis būtų didesnis nei įprastai vykdant mutavimą lokaliojo gerinimo algoritme.

## Eksperimentų rezultatai ir išvados

Atliekant eksperimentinius genetinio algoritmo ir jo variantų tyrimus, buvo naudojami testiniai pavyzdžiai (duomenys) iš viešos internete esančios komivojažieriaus uždavinio testinių pavyzdžių (gairių) bibliotekos TSPLIB (Reinelt, 1991). Pagrindiniu algoritmo efektyvumo įvertinimo kriterijumi buvo pasirinktas gaunamų sprendinių kokybės, t. y. tikslo funkcijos (maršruto ilgio), vidutinis santykinis nuokrypis nuo galimos optimalios tikslo funkcijos reikšmės. Vidutinis santykinis nuokrypis $\bar{\delta}$ apskaičiuojamas pagal formulę:

$$\bar{\delta} = 100(\bar{z} - \breve{z})/\breve{z}\ [\%]; \quad (5)$$

čia $\bar{z}$ yra gautų tikslo funkcijos reikšmių vidurkis, kuris apskaičiuojamas atlikus tam tikrą skaičių ($Q$) algoritmo pakartotinių vykdymų; $\breve{z}$ žymi galimą optimalią tikslo funkcijos reikšmę (šios reikšmės, jeigu jos yra nustatytos, pateikiamos bibliotekoje TSPLIB). Papildomi įvertinimo kriterijai yra surandamų galimų optimalių sprendinių skaičius ($\bar{Q}$) ir algoritmo vykdymo laikas ($t$). Buvo išbandyti įvairūs sudaryto genetinio algoritmo variantai. Gautieji rezultatai yra pateikti 6 pav. Mūsų genetinis algoritmas taip pat buvo lygintas su kitais algoritmais (žr. 1 lentelę). Buvo lyginami šie algoritmai: staigiausio nusileidimo 2 atkarpų porų sukeitimo aplinkoje $\Theta_2$ algoritmas (SN-$\Theta_2$) (angl. *2-opt*); algoritmas SN-$\Theta_2$ su pakartotiniais restartais (1000 SN-$\Theta_2$) (restartų skaičius – 1000); staigiausio nusileidimo aplinkoje $\Theta_3$ algoritmas (SN-$\Theta_3$) (angl. *3-opt*); algoritmas SN-$\Theta_3$ su 30 pakartotinių restartų (30 SN-$\Theta_3$) (šių algoritmų konceptualius aprašus galima pateikia C. Nilsson (2003); atkaitinimo modeliavimo algoritmas (AM) (Misevičius, 2008); patobulintas greito nusileidimo-atsitiktinio pakilimo algoritmas (PGN-AP) (Misevičius ir kt., 2007); naujasis genetinis algoritmas (GA). (Algoritmai programiškai realizuoti A. Misevičiaus.)




*6 p a v*. ***Eksperimentų rezultatai, gauti sprendžiant testinį KU pavyzdį gr48 (n = 48, $\bar{z}$ = 5046) iš bibliotekos TSPLIB: (a) – naudota gerinimo schema 00, diversifikacijos slenkstis (DS) – 0; (b) - gerinimo schema – 01, DS = 0; (c) – gerinimo schema –10, DS = 0; (d) – gerinimo schema – 11, DS = 0,1***

*1 l e n t e l ė*. ***Algoritmų palyginimo rezultatai***

| Testinis pavyzdys‡ | $\bar{z}$ | 2-OPT | | | 1000 2-OPT | | | 3-OPT | | | 30 3-OPT | | | AM | | | PGN-AP | | | GA‡‡ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ | $\bar{\delta}$ | $\bar{Q}$ | $t$‡‡‡ |
| gr96 | 55209 | 6,165 | 0 | **0,002** | 0,197 | 2 | 2,2 | 2,927 | 0 | 0,2 | 0,527 | 0 | 7,0 | 0,268 | 1 | 4,4 | 0,000 | 10 | 2,1 | **0,000** | **10** | 0,8 |
| eil101 | 629 | 9,300 | 0 | **0,003** | 0,700 | 0 | 2,5 | 5,072 | 0 | 0,2 | 2,560 | 0 | 7,7 | 0,016 | 9 | 4,9 | 0,000 | 10 | 2,6 | **0,000** | **10** | 1,1 |
| kroa100 | 21282 | 6,953 | 0 | **0,002** | 0,023 | 5 | 2,5 | 3,138 | 0 | 0,3 | 0,265 | 1 | 8,6 | 0,071 | 7 | 4,6 | 0,000 | 10 | 1,7 | **0,000** | **10** | 1,2 |
| lin105 | 14379 | 7,481 | 0 | **0,003** | 0,248 | 2 | 2,8 | 2,728 | 1 | 0,4 | 0,138 | 5 | 10,2 | 0,084 | 7 | 4,8 | 0,000 | 10 | 1,7 | **0,000** | **10** | 1,1 |
| pr107 | 44303 | 5,896 | 0 | **0,003** | 0,056 | 4 | 2,8 | 1,475 | 0 | 0,4 | 0,249 | 3 | 10,9 | 0,000 | 10 | 4,9 | 0,000 | 10 | 2,4 | **0,000** | **10** | 0,9 |
| rat99 | 1211 | 9,587 | 0 | **0,002** | 0,116 | 1 | 2,3 | 5,574 | 0 | 0,2 | 1,841 | 0 | 7,6 | 0,000 | 10 | 4,6 | 0,000 | 10 | 1,7 | **0,000** | **10** | 1,1 |
| rd100 | 7910 | 7,836 | 0 | **0,002** | 0,515 | 0 | 2,5 | 4,502 | 0 | 0,3 | 0,893 | 1 | 8,2 | 0,013 | 8 | 4,6 | 0,043 | 9 | 2,2 | **0,000** | **10** | 1,2 |
| **Vidurkis:** | | 7,603 | 0 | **0,002** | 0,265 | 2 | 2,5 | 3,631 | 0 | 0,3 | 0,925 | 1 | 8,6 | 0,065 | 7 | 4,7 | 0,006 | 10 | 2,1 | **0,000** | **10** | 1,1 |

‡ testinio pavyzdžio pavadinime esantis numeris nurodo miestų skaičių; ‡‡ gerinimo schema – 11, krosoverio variantas – 2, diversifikacijos slenkstis – 0,05); ‡‡‡ pateikiamas vidutinis vieno algoritmo pakartotinio vykdymo laikas sekundėmis (eksperimentams naudotas 3 GHz dažnio kompiuteris).

Eksperimentų rezultatai patvirtina pradines prielaidas, jog gaunamų sprendinių kokybė priklauso nuo to, kiek yra atsižvelgiama į sprendžiamo uždavinio specifinius ypatumus ir kaip universalus metodas adaptuojamas konkrečiam uždaviniui. Konstatuotina, kad gaunami rezultatai yra gana smarkiai veikiami paties genetinio algoritmo savybių ir struktūros, jo vidinės konfigūracijos. Vienas iš pagrindinių veiksnių, lemiančių GA efektyvumo laipsnį, yra tinkamų genetinių operatorių derinimas su lokaliojo optimizavimo (gerinimo) procedūromis. Šis derinimas labai padidina „sprendinių palikuonių“ kokybę ir paspartina evoliucinio proceso konvergavimą. Tokių hibridizuotų genetinių algoritmų ir jų modifikacijų tyrimas, siekiant dar labiau pagerinti rezultatų kokybę, būtų jau ateities darbų tema.

## LITERATŪRA


APPLEGATE, D.L.; BIXBY, R.E.; CHVÁTAL, V.; COOK, W.J. (2007). *The traveling salesman problem: A computational study*. Princeton: Princeton University Press. 606 p.

GOLDBERG, D.E.; LINGLE, R. (1985). Alleles, loci, and the TSP. Iš Grefenstette J.J. (ed.). *Proceedings of the First International Conference on Genetic Algorithms and their Applications*. Hillsdale: Lowrence Erlbaum, p. 154–159.

MARINAKIS, Y.; MIGDALAS, A.; PARDALOS, P. (2005). A hybrid genetic-GRASP algorithm using Lagrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization*, vol. 10, p. 311–326.

MISEVIČIUS, A. (2008). Simulated annealing algorithm for the solution of the traveling salesman problem. Iš Targamadzė A., Butleris R., Butkienė R. (eds). *Proceedings of the 14th International Conference on Information and Software Technologies*, *IT-2008*. Kaunas: Technologija, p. 19–24.

MISEVIČIUS, A.; OSTREIKA, A.; ŠIMAITIS, A.; ŽILEVIČIUS, V. (2007). Improving local search for the traveling salesman problem. *Information Technology and Control*, vol. 36, p. 187–195.

NGUYEN, H.D.; YOSHIHARA, I.; YAMAMORI, K.; YASUNAGA, M. (2002). Greedy genetic algorithms for symmetric and asymmetric TSPs. *IPSJ Transactions on Mathematical Modeling and Its Applications*, vol. 43, p. 165–175.

NILSSON, C. (2003). *Heuristics for the traveling salesman problem*. Tech. Report, Linköping University, Sweden. [Žr. taip pat prieigą per internetą: <http://www.ida.liu.se/~TDDB19/reports_2003/htsp.pdf>.]

POTVIN, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, vol. 63, p. 339–370.

REEVES, C.R. (2002). Genetic algorithms. Iš Glover F., Kochenberger G. (eds.). *Handbook of Metaheuristics*. Norwell: Kluwer, p. 55–82.

REINELT, G. (1991). TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, vol. 3–4, p. 376–385. [Žr. taip pat prieigą per internetą: <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.]

ROSENKRANTZ, D.E.; STEARNS, R.E.; LEWIS, P.M. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, vol. 6, p. 563–581.

SIVANANDAM, S.N.; DEEPA, S.N. (2008). *Introduction to Genetic Algorithms*. Berlin–Heidelberg–New York: Springer. 442 p.

SYSWERDA, G. (1989). Uniform crossover in genetic algorithms. Iš Schaffer J.D. (ed.). *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Mateo: Morgan Kaufmann, p. 2–9.

TATE, D.M.; SMITH, A.E. (1995). A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, vol. 1, p. 73–83.

ZOU, P.; ZHOU, Z.; CHEN, G.; YAO, X. (2004). A novel memetic algorithm with random multi-local-search: A case study of TSP. *Proceedings of the 2004 Congress on Evolutionary Computation CEC-2004*, vol. 2. New York: IEEE Press, p. 2335–2340.


## ON THE GENETIC ALGORITHMS FOR THE TRAVELING SALESMAN PROBLEM: NEGATIVE AND POSITIVE ASPECTS


**Alfonsas Misevičius, Andrius Blažinskas, Jonas Blonskis, Vytautas Bukšnaitis**



Summary

In this paper, we discuss some issues related to the application of genetic algorithms (GAs) to the well-known combinatorial optimization problem – the traveling salesman problem (TSP). The results obtained from the experiments with the different variants of the genetic algorithm are presented as well. Based on these results, it is concluded that the efficiency of the genetic search is much influenced by both the specific nature of the problem and the features of the algorithm itself. In particular, it should be emphasized that the incorporation of the (post-crossover) procedures for the local improvement of offspring has one of the crucial roles in obtaining high-quality solutions.