KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Šarūnas Putrius

Atkaitinimo modeliavimo algoritmo komivojažieriaus uždaviniui sudarymas ir tyrimas

Magistro darbas

Darbo vadovas
Doc., dr. Alfonsas Misevičius

Kaunas
2008

Turinys

Pratarmė

Kombinatorinio optimizavimo metodai ir algoritmai yra svarbi informatikos mokslo sritis. Reikšmingą kombinatorinio optimizavimo metodų dalį sudaro euristiniai algoritmai, kuriais siekiama surasti aukštos kokybės, bet nebūtinai optimalius sprendinius per priimtiną laiką.

Vieni iš plačiausiai taikomų euristinių algoritmų kombinatorinio optimizavimo uždaviniams spręsti yra atkaitinimo modeliavimo (AM) algoritmai. Pats atkaitinimo modeliavimo metodas yra kilęs iš statistinės mechanikos ir sėkmingai taikomas sprendžiant optimizavimo uždavinius informatikoje. Šiame darbe tiriama AM algoritmo realizacija gerai žinomam kombinatorinio optimizavimo uždaviniui — komivojažieriaus uždaviniui (KU), priklausančiam NP-sunkių kombinatorių uždavinių klasei.

Darbe pateikiama eksperimentų su realizuotu algoritmu, panaudojant KU testinius pavyzdžius (duomenis) iš viešos elektroninės KU testinių pavyzdžių bibliotekos TSPLIB, rezultatų analizė. Buvo siekiama išsiaiškinti įvairių KU vykdymo parametrų įtaką algoritmo vykdymo laikui bei algoritmo rezultatų kokybei, taip palyginti AM algoritmo parametrus ir kitus KU parametrus tarpusavyje. Konstatuojama, jog algoritmo iteracijų skaičiaus (atkaitinimo trukmės) bei atkaitinimo proceso temperatūros valdymo parametrai turi lemiamos svarbos rezultatų kokybei.

Toliau darbe siūlomas patobulintas AM algoritmas komivojažieriaus uždaviniui, nagrinėjamos modifikacijos, kurios remiasi perėjimo sąlygos prie sekančio sprendinio AM algoritme patobulinimu. Atliekami išsamūs eksperimentiniai tyrimai, įrodantys modifikacijų pranašumą prieš iki šiol pasiūlytas uždavinio realizacijas. Identifikuotos modifikacijos, neatnešančios rezultatų pagerėjimo.

Darbas pradedamas įvadu, kuriame suformuluojami darbo tikslai, 2-ame skyriuje aprašomas komivojažieriaus uždavinys; 3-iame skyriuje apžvelgiamas atkaitinimo modeliavimo metodas ir algoritmas, jo taikymai; 4-tame skyriuje pristatomas darbo metu sudarytas atkaitinimo modeliavimo algoritmas ir jo modifikacijos; eksperimentai su atkaitinimo modeliavimo algoritmu komivojažieriaus uždaviniui, jų rezultatai ir analizė pateikiami 5-ame skyriuje; darbas baigiamas išvadomis.

1. Įvadas

Jau XX amžiaus viduryje buvo pradėta intensyviai ieškoti efektyvių būdų spręsti sudėtingus kombinatorinius uždavinius. Šį laikotarpį galima laikyti euristinių algoritmų (EA), skirtų kombinatorinio optimizavimo (KO) uždaviniams spręsti, kūrimo ir tyrimų pradžia. EA vystimąsi labai skatino kompiuterinės (skaičiavimo) technikos progresas. Šiuo metu yra sukurta daug įvairių euristinių algoritmų ir metodų. Ypač sparčiai EA plėtojami pastaraisiais dešimtmečiais. Dabar galima priskaičiuoti iki keliolikos atskirų iteracinių EA grupių, tai: atkaitinimo modeliavimas (angl. simulated annealing), genetiniai algoritmai (angl. genetic algorithms), godžiosios randomizuotos adaptyvios paieškos procedūros (angl. greedy randomized adaptive search procedures (GRASP)), „išsklaidyta paieška“ (angl. scatter search), iteratyvioji lokalioji paieška (angl. iterated local search), (klasikinė) lokalioji paieška (angl. local search), koreguojama lokalioji paieška (tikslo funkcijos korekcijos algoritmai) (angl. guided local search), paieška kintamose aplinkose (angl. variable neighbourhood search), skruzdėlių kolonijų elgsenos imitavimo algoritmai (angl. ant colony optimization), tabu paieška (angl. tabu search) ir kt.. Kuriant ir tiriant šiuos bei kitus euristinius metodus kombinatorinio optimizavimo uždaviniams, svarų indėlį įnešė daug optimizavimo algoritmų specialistų, tarp jų, tokie mokslininkai, kaip E. Aarts, J. Beasley, R. Burkard, M. Dorigo, Z. Drezner, D. Fogel, F. Glover, D. Goldberg, B. Golden, P. Hansen, D. Johnson, M. Laguna, E. Lawler, J. Lenstra, Z. Michalewicz, P. Moscato, P. Pardalos, M. Resende, T. Stützle, E. Taillard, S. Voss ir daugelis kitų. [1,2,3,4,5,6,7,8,9,10].

Šiame magistriniame darbe buvo koncentruojamasi į atkaitinimo modeliavimo algoritmo sudarymą ir jo taikymą kombinatorinio optimizavimo uždaviniui – komivojažieriaus uždaviniui.

Pagrindiniai darbo tikslai būtų sudaryti ir ištirti atkaitinimo modeliavimo realizaciją gerai žinomam komivojažieriaus uždaviniui. Taip pat atlikti išsamius atkaitinimo modeliavimo algoritmo modifikacijų eksperimentinius tyrimus, palyginti ir išanalizuoti eksperimentų rezultatus ir galiausiai suformuluoti išvadas.

2. Komivojažieriaus uždavinys

Šiame skyriuje pateikiama Komivojažieriaus (arba kitaip keliaujančio pirklio) uždavinio (angl. *traveling salesman problem*) [11, 12, 13] formuluotė, taikymai, sprendimo būdų apžvalga.

2.1. Komivojažieriaus uždavinio apibrėžimas

Duota atstumų tarp tam tikrų objektų (KU juos priimta vadinti miestais) matrica $\boldsymbol{D} = (d_{ij})_{n\times n}$ bei aibė Π, kurią sudaro visi galimi natūrinių skaičių nuo 1 iki n perstatymai. Tikslas yra surasti perstatymą $p_{\text{opt}} \in \Pi$, ir tokį, kad $p_{\text{opt}} = \arg\min_{p\in\Pi} z(p)$, čia z yra KU tikslo funkcija:

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

čia p atlieka KU sprendinio vaidmenį; n yra uždavinio apimtis. Perstatymui $p = (p(1), p(2), ..., p(n))$ ($p(i) \in \{1, 2, ..., n\}$) komivojažieriaus uždavinyje atitinka n miestų seka, kitaip tariant, maršrutas. Šiuo atveju $j = p(i)$ tiesiog žymi i-tąjį aplankytą maršruto miestą j. Perstatymo narių poros $(p(1), p(2))$, ..., $(p(i), p(i + 1))$, ..., $(p(n), p(1))$ paprastai vadinamos maršruto „atkarpomis“. Taigi KU formuluotę galima perfrazuoti ir taip: surasti trumpiausią galimą uždarą maršrutą, kuriame kiekvienas miestas aplankomas vieną ir tik vieną kartą.

2.2. Komivojažieriaus uždavinio taikymai

Uždavinys yra ypatingai aktualus, kadangi dažnai yra taikomas praktikoje. Iš taikymų labiausiai paminėtini yra šie:

krovinių gabenimas,

siuntinių pristatymas (laiškanešių maršrutai),

spausdintų montažo plokščių (angl. *printed circuit board*) projektavimas,

kristalų struktūros tyrimas,

staklių darbo grafikas,

eksponatų išdėstymas parodoje,

sandėliavimas,

garo turbinos rekonstravimas,

transporto priemonių sustatymo aikštelėje tvarka,

kompiuterių (telekomunikacijos, elektros) tinklų sudarymas.

2.3. Komivojažieriaus uždavinio sprendimo būdai

Komivojažieriaus uždavinys yra gana nesudėtingai apibūdinamas, tačiau sunkiai sprendžiamas, ypač kai susiduriama su didelėmis uždavinio apimtimis.

KU priklauso NP-sunkių uždavinių klasei [14]. Norint tiksliai išspręsti uždavinį, reikėtų įvertinti kiekvieną maršrutą, kas reikštų beveik *n!* galimybių. Tokie uždaviniai tiksliai išsprendžiami tik esant ribotoms jų apimtims. KU uždaviniui spręsti yra pasiūlyta visa eilė apytikslių ir euristinių algoritmų [15].

Apytiksliai algoritmai ir jų principas garantuoja kad rastas KU uždavinio sprendinys nebus blogesnis už kažkokią užsibrėžtą reikšmę. Paprastai nusakoma kaip *c* kartų blogesnę nei optimali reikšmė. Iki šiolei geriausias pasiūlytas algoritmas [16] garantuoja *(1+1/c)* aproksimaciją kiekvienam $c > 0$.

Maršruto konstravimo algoritmai taip pat dažnai naudojami šiai problemai spręsti. Jie visi turi bendrą savybę – radę sprendinį sustoja ir nebando jo gerinti, dėl to galima teigti, kad šie algoritmai yra tik 10-15% optimalūs. Pagrindinai šios algoritmų grupės atstovai: artimiausio kaimyno paieška, godžioji paieška, Cristofido algoritmai. Išsamesnė maršruto konstravimo euristikos apžvalga [17] veikale.

Vidutinės ir didelės apimties KU uždaviniams spręsti yra sėkmingai naudojami euristiniai algoritmai [18]. Reikšmingą tokių algoritmų grupę sudaro tabu paieškos [19], genetiniai [20], iteracinės lokaliosios paieškos (ILP), skruzdėlių kolonijos [21] ir atkaitinimo modeliavimo [22, 23] algoritmai.

3. Apžvalginė dalis

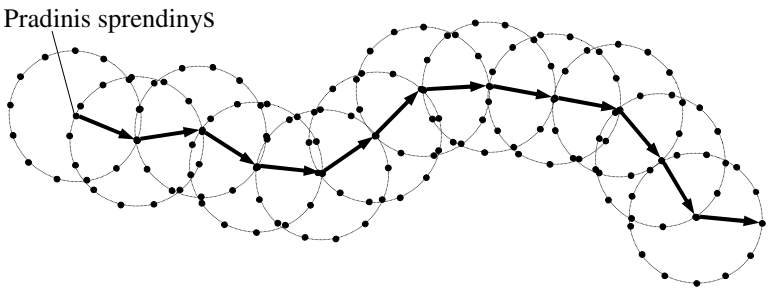
3.1. Euristiniai algoritmai

Euristinių algoritmų (EA) (angl. heuristic algorithms) tikslas - sprendžiamiems optimizavimo uždaviniams surasti aukštos kokybės, nors nebūtinai optimalius sprendinius.

Rasti sprendiniai paprastai yra tik lokaliai optimalūs kurios nors aplinkos atžvilgiu, todėl galima teigti, kad euristiniai algoritmai negarantuoja gautų sprendinių optimalumo. Todėl neretai euristiniai algoritmai įvardijami tiesiog lokaliosios paieškos (LP) (angl. local search) algoritmais. EA iš esmės skiriasi tiek nuo tiksliųjų algoritmų (angl. exact algorithms), kurie garantuoja optimalaus sprendinio suradimą, tiek nuo apytikslių (aproksimacinių) algoritmų (angl. approximate algorithms), kurie užtikrina, kad gauto sprendinio kokybė skirsis nuo optimalaus sprendinio kokybės ne daugiau kaip iš anksto duotas dydis (paklaida) $\delta > 0$. Kita vertus, euristiniai algoritmai pasižymi ta gera savybe, jog skaičiavimų trukmė yra susieta su uždavinio apimtimi dažniausiai polinomine priklausomybe.

Terminai „euristika“, „euristinis algoritmas“ naudojami jau keli dešimtmečiai. Šių sąvokų suformulavimo optimizavimo uždavinių sprendimo kontekste ištakų reikėtų ieškoti Polya veikale „Kaip spręsti?“ (angl „How to solve it“), išleistame dar 1945 m. [24]. Tačiau ir šiuo metu tos sąvokos įvairių autorių vis dar gana skirtingai suprantamos ir apibrėžiamos [25]. Įdomumo dėlei, Streim‘as — tiesa, daugiau kaip prieš 30 metų — buvo nurodęs 11 euristinių algoritmų apibrėžimų [26].

Euristika yra taisyklių, besiremiančių žmogaus intuicija, sveiku protu, rinkinys, turint tikslą surasti vieną ar daugiau pakankamai gerų, artimų savo kokybe globaliajam optimumui sprendinių per priimtiną skaičiavimų laiką [27]. Kitaip tariant, euristiniai metodai gali būti traktuojami kaip tam tikri formalūs, pakankamai aukšto abstrakcijos lygio nurodymai, skirti kurio nors tipo uždavinių bendrai sprendimo idėjai, principui aprašyti [28].



3.1.1 pav. Sprendinių nagrinėjimo proceso grafinė iliustracija

Euristinis algoritmas, jo veikimas gali būti interpretuojamas kaip iteracinis, kryptingas, konverguojantis paieškos procesas. Tuomet EA veikimo principas, t.y. paradigma galėtų būti trumpai aprašyta taip. Paieška pradedama nuo tam tikru būdu sukonstruoto pradinio sprendinio. Toliau paieškos procesas vykdomas nagrinėjant sprendinius bei jų aplinkas ir nuosekliai transformuojant sprendinius, tiksliau, pereinant nuo vienų sprendinių prie kitų (žr. 3.1.1 pav.). Procesas baigiamas, kai patenkinama baigimo sąlyga, pvz. įvykdomas a priori užduotas paieškos iteracijų skaičius. Štai klasikiniuose LP („nusileidimo“) algoritmuose perėjimų nuo vienų (blogesnių) sprendinių prie kitų (geresnių) sprendinių procesas tęsiamas tol, kol esamas sprendinys nėra lokaliai optimalus. Paieškos „topologiją“ apibrėžia pasirinkta aplinkos funkcija, o sprendimai dėl perėjimų priimami atsižvelgiant į sprendinių kokybės įverčius (tikslo funkcijos reikšmes). Neretai remiamasi tikimybinėmis, randomizuotomis sprendimų priėmimo taisyklėmis. Bet tai nereiškia, kad tais atvejais euristinis algoritmas redukuojamas į visiškai nedeterminuotą paieškos procesą, „aklą klaidžiojimą nežinioje“. Atvirkščiai, randomizuoti sprendimai euristikose tik padeda diversifikuoti paieškos eigą, nukreipiant procesą viena ar kita kryptimi. Ne atsitiktinumai valdo paieškos procesą, o mes kurdami algoritmus, tinkamai pasinaudojame atsitiktinumais, siekdami, kad paieška būtų lankstesnė, efektyvesnė [29].

Šiuolaikiniai, modernieji euristiniai algoritmai orientuojami kelių (daugiau nei vieno) lokaliai optimalių sprendinių paieškai, kas yra visiškai priešinga ankstyviesiems klasikiniams algoritmams, kuriuose apsiribojama tik vieno (lokaliai optimalaus) sprendinio suradimu. Taigi atsiranda didesnės potencialios galimybės surasti aukštesnės kokybės sprendinius. Juk galima samprotauti: didėjant surandamų lokaliųjų optimumų skaičiui, didėja ir tikimybė, jog kuris nors surastas sprendinys bus geresnis, negu pirmasis surastas lokaliai optimalus sprendinys. Aišku, paieškos laikas atitinkamai pailgėja, tačiau geriau gauti aukštos kokybės rezultatą per ilgesnį laiką, negu tenkintis blogu rezultatu, gautu per labai trumpą laiką. Net ir pakankamai ilgas (bet tenkinantis vartotoją) paieškos laikas yra pateisinamas, jeigu tik tai suteikia galimybes apčiuopiamai pagerinti sprendinių kokybę. Žinoma, nuolatos turi būti stengiamasi, kad ir aukštos kokybės sprendiniai būtų pasiekiami per kuo trumpesnį laiką.

Vienas iš euristinių algoritmų, sėkmingai taikomas sprendžiant daugelį optimizavimo uždavinių yra atkaitinimo modeliavimo algoritmas. Šiame darbe atkaitinimo modeliavimo algoritmas buvo taikomas komivojažieriaus uždaviniui spręsti.

3.2. Atkaitinimo modeliavimo metodas, savybės, taikymai

Atkaitinimo modeliavimo (AM) metodo ištakos slypi statistinėje mechanikoje, tiksliau, energetinių procesų, vykstančių sistemose, sudarytose iš didelio skaičiaus dalelių (atomų), imitavime. Šio imitavimo idėja yra ta, jog iš pradžių sistema „pervedama“ į didelės energijos būseną, o po to, palengva mažinant energiją, stengiamasi pasiekti būseną, atitinkančią žemiausią (optimalų) sistemos energijos lygį — tarsi (kietas) fizikinis kūnas būtų įkaitintas iki pakankamai aukštos temperatūros, o paskui, jį atkaitinant (atšaldant), t.y. mažinant temperatūrą, jis būtų savotiškai užgrūdinamas. Atkaitinimo modeliavimo pradininkai buvo Metropolis, Rozenbluth‘as ir kt. [30], kurie pasinaudojo eksponentinio pasiskirstymo funkcija apibrėždami tikimybę, kad sistema, įvykus joje tam tikram pasikeitimui, pereis iš vieno energijos lygio (E_1) į kitą lygį (E_2), kai sistemos temperatūra lygi t. Ta tikimybė apskaičiuojama pagal tokią formulę:

$$\mathrm{P}(\Delta E) = \begin{cases} 1, & \Delta E < 0 \\ e^{-\Delta E/Ct}, & \Delta E \geq 0 \end{cases}; \qquad (2)$$

čia $\Delta E = E_2 - E_1$, o C — konstanta. Cerný [31] ir Kirkpatrick‘as su bendraautoriais [32] buvo pirmieji, pritaikę atkaitinimo modeliavimo idėją sprendžiant kombinatorinio (ir ne tik) optimizavimo uždavinius.

Bendroji AM algoritmų funkcionavimo schema yra gana paprasta [32]. Sakykime, kad (kombinatorinio) optimizavimo uždavinys aprašytas poros (S, f) pagalba; čia S yra sprendinių aibė, o f: $\mathrm{S} \to \mathrm{R1}$ — tikslo funkcija, kuri turi būti minimizuota. Tuomet algoritmo vykdymas pradedamas nuo atsitiktiniu (ar kuriuo nors kitu) būdu sugeneruoto sprendinio s iš aibės S. Esamo sprendinio s aplinkoje (sprendinio s aplinka („kaimyninių“ sprendinių aibe) vadinamas poaibis $\mathrm{S}' \subseteq \mathrm{S}$, sudarytas iš sprendinių, tam tikru mastu „artimų“ sprendiniui s. Formaliai aplinka apibrėžiama, panaudojant specialią funkciją Θ: $\mathrm{S} \to 2\mathrm{S}$, kur 2S žymi aibės S visų galimų poaibių aibę) parenkamas sprendinys s' ir apskaičiuojamas tikslo funkcijos pokytis $\Delta \mathrm{f} = \mathrm{f}(\mathrm{s}') - \mathrm{f}(\mathrm{s})$. Proceso eiga priklauso nuo šio pokyčio: jeigu $\Delta \mathrm{f} < 0$, t.y. tikslo funkcijos reikšmė pagerėja, tai esamas sprendinys s pakeičiamas nauju sprendiniu s' ir naudojamas kaip išeities „taškas“ tolesniuose bandymuose; priešingu atveju, sprendinio pakeitimas daromas su tikimybe $\mathrm{P}(\Delta f) = e^{-\Delta f/t}$; čia t yra einamoji temperatūra (konstanta C nenaudojama). Temperatūros parametras t palaipsniui mažinamas algoritmo vykdymo metu. Procesas tęsiamas tol, kol patenkinama iš anksto užduota baigimo sąlyga (pvz. temperatūra pasiekia duotą apatinę ribą arba įvykdomas nurodytas iteracijų skaičius). AM algoritmo rezultatas yra geriausias algoritmo vykdymo eigoje surastas sprendinys.

Konkrečios atkaitinimo modeliavimo algoritmų realizacijos skiriasi viena nuo kitos šiais pagrindiniais veiksniais: atkaitinimo („atšaldymo“) schema ir baigimo sąlyga. Formuojant atkaitinimo schemą, savo ruožtu, yra svarbūs tokie faktoriai: a) pradinės ir galutinės temperatūros parinkimas, b) vadinamasis ekvilibriumo (pusiausvyros) testas ir c) temperatūros mažinimo formulė.

Parenkant temperatūrą turi būti vadovaujamasi tokiais samprotavimais: jei pradinė temperatūra pernelyg aukšta, tai atkaitinimo procesas gali užsitęsti labai jau ilgai, bereikalingai nagrinėjant daug „blogų“ sprendinių. Kita vertus, per daug žema pradinė temperatūra gali sąlygoti greitą „įkritimą“ į nelabai gero lokaliojo optimumo „duobę“.

Ekvilibriumo testas naudojamas tam, kad nustatyti, kuriuo būtent momentu turi būti mažinama temperatūra. Galimi du variantai: a) temperatūra mažinama, tik atlikus pakankamai daug bandymų ir pasiekus didesnės ar mažesnės pusiausvyros būseną – šis variantas vadinamas homogeniniu atkaitinimu; b) temperatūra mažinama po kiekvieno įvykdyto bandymo (galima sakyti, jog ekvilibriumo testas iš viso neatliekamas) – tai nehomogeninio atkaitinimo variantas.

Praktikoje taikomuose atkaitinimo algoritmuose plačiausiai yra naudojamos šios temperatūros mažinimo formulės: geometrinė formulė ($t_k = \alpha \cdot t_{k-1}$; t_k – einamoji temperatūros reikšmė; k=1, 2, ...; t_0 =const; $0.8 \le \alpha \le 0.99$ [32]) ir Lundy-Mees formulė ($t_k = t_{k-1} / (1 + \beta t_{k-1})$; k=1, 2, ...; t_0 =const; $\beta << t_0$ [33]). (Pirmoji formulė naudojama homogeninio atkaitinimo atveju, antroji – nehomogeninio.)

Reikia pastebėti, kad naujausiose atkaitinimo modeliavimo algoritmų versijose temperatūra greičiau keičiama periodiškai, nei monotoniškai mažinama. Eksperimentinių tyrimų rezultatai [34, 35, 36] liudija, kad vietoje „grynojo“ atkaitinimo tikslingiau yra taikyti „atkaitinimų“ ir „kaitinimų“ seką, t.y., vadinamąjį re-atkaitinimą (angl. reannealing), kuris suteikia papildomas, lankstesnes galimybes atkaitinimo modeliavimo algoritmams.

Svarbu tinkamai parinkti algoritmo pabaigos sąlygą, atkaitinimo procesas teoriškai turėtų būti tęsiamas tol, kol galutinė temperatūra t_g taptų lygi 0. Tačiau praktiškai atkaitinimą galima baigti ir anksčiau – kai tampa mažai tikėtina, jog bus pasiektas pagerinimas, pvz., kai tikslo funkcijos reikšmė nemažėja pakankamai ilgą laiko tarpą. Dažnai baigimo sąlygos vaidmenį atlieka apriori fiksuotas iteracijų skaičius „atkaitinimo schemos ilgis“.

Daugiau informacijos apie atkaitinimo modeliavimo algoritmo tyrimus galima rasti [22, 37, 38] darbuose. Atkaitinimo modeliavimo algoritmas pseudo kalboje pateikiamas 3.2.1 paveiksle.

```
procedure AtkaitinimoModeliavimas
   // pradiniai duomenys: p(0) – einamasis sprendinys;
     // rezultatas: p* – geriausias surastas sprendinys
 begin
   p := p(0), p* := p;
   nustatyti pradinę temperatūrą t0, t := t0
   repeat // atkaitinimo modeliavimo ciklas
     parinkti naują sprendinį p′ iš esamo sprendinio p aplinkos,
      apskaičiuoti Δz=z(p′)−z(p)
     if Δz < 0 then begin
        p := p′; // pereiti į kitą sprendinį
        if z(p) < z(p*) then p* := p; // geriausio sprendinio fiksavimas atmintyje
     end
     else begin
        sugeneruoti atsitiktinį skaičių r iš intervalo [0,1]
        if r < e^(−Δz/t) then p := p′;
     end
     jei reikia, pakeisti esamą temperatūrą t, atsižvelgiant į
      ekvilibriumo testą ir atkaitinimo schemą
   until patenkinta baigimo sąlyga
 end
```

3.2.1 pav. Atkaitinimo modeliavimo algoritmas pseudo kalboje.

Atkaitinimo modeliavimo algoritmas praktiškai taikomas sprendžiant sudėtingas optimizavimo problemas įvairiose gyvenimiškose srityse [39]. Vienas dažniausių atkaitinimo modeliavimo algoritmo taikymų – tvarkaraščių ir grafikų sudarymo uždavinių sprendime. Tokie uždaviniai reikalauja efektyvaus turimų resursų pozicionavimo laike ir erdvėje, kitaip tariant siekia optimizuoti tikslo funkciją. Vienas iš daugelio tvarkaraščių sudarymo pavyzdžių taikant atkaitinimo modeliavimą aprašomas [40] veikale. Jame bandoma spręsti oro kompanijos įgulos tvarkaraščio problemą, įgulą paskirstant tarp skrydžių taip, kad būtų aptarnaujami visi kompanijos skrydžiai minimizuojant įgulos kaštus. Pasiūlyti būdai, kaip galima pagerinti tvarkaraščių sudarymo rezultatus, naudojant AM principus.

Atkaitinimo modeliavimo principas taikomas vaizdų atkūrime elektrinės varžos tomografijoje (angl. electrical impedance tomography), biomedicinoje. AM algoritmas naudojamas statistinėms rekonstrukcijoms sudaryti, ir taip spręsti statinę elektrinės varžos tomografijos atvirkštinės tvarkos problemą [41].

Be šių atkaitinimo modeliavimo algoritmo taikymų dar galima paminėti ir grafų dalijimo bei dažymo uždavinius, įvairias problemas finansų, biologijos, geofizikos, fizikos, matematikos ir kombinatorikos, duomenų analizės, neuroninių tinklų, vaizdų apdorojimo srityse.

3.3. Atkaitinimo modeliavimo algoritmo variantai komivojažieriaus uždaviniui

Atkaitinimo modeliavimo principai gali būti sėkmingai taikomi spręsti komivojažieriaus uždaviniui, tai buvo pastebėta ir aprašyta dar 1983 metais S. Kirkpatrick, C. D. Gelatt, Jr. ir M. P. Vecchi veikale „Optimization by Simulated Annealing“ [32]. Šiame straipsnyje buvo koncentruojamasi kaip optimizuoti elektronines schemas kompiuterio mikroschemose ir kartu spręsti komivojažieriaus uždavinį. Straipsnio autoriai rėmėsi viena pirmųjų ir gerai žinoma 1971 metais S. Lin ir B. W. Kernighan pasiūlyta algoritmo KU variacija [42], šalia perstatymo/mutavimo naudodami ir atkaitinimo modeliavimo principus. Atkaitimo grafikas buvo nustatomas empiriškai, aprašomos temperatūrų parinkimo schemos, pateikiama algoritmo rezultatų analizė.

Vėliau buvo gana nemažai bandymų patobulinti atkaitinimo modeliavimo schemą KU. Vienas iš jų 1993 metais Peng Tian ir Zihou Yang [43] pasiūlytas pagerinimas. Jie aprašė naują algoritmą kombinatorinio optimizavimo problemoms spręsti, kurio pagrindas atkaitinimo modeliavimo principas. Šalia paprasto AM algoritmų privalumų atsirado galimybė išvengti užstrigimo lokaliuose optimumuose, nes naujas algoritmas mokėjo startuoti iš kelių pradinių būsenų. Taip pat integruotas genetiniais principais pagrįstas mechanizmas, kuris remiantis natūralios atrankos modeliu garantuoja konvergavimą iki globalaus optimumo. Galimybė turėti daugialypes būsenas atveria kelius uždavinius spręsti remiantis lygiagretaus programavimo principais.

2002 metais Joshua W. Pepper pateikė originalų KU sprendimo algoritmą [54], kuris taip pat remiasi atkaitinimo modeliavimo metodais. Naujasis algoritmas buvo pavadintas Demono algoritmu. Norint suvokti Demono algoritmą reiktų įsivaizduoti, kad sistema turi pradinį energijos kiekį, arba demono kiekį. Kai sistema patiria energijos praradimą, imamas energijos „kreditas“ iš demono kiekio. Sistemos energijos padidėjimas galimas tik tada, jei energijos pokytis yra mažesnis už demono kiekį. Energijos pokytis pridedamas prie demono reikšmės. Taigi demono kiekis nuolatos kinta, o pati procedūra nekonverguoja. Kad pasiekti konvergavimą yra pasiūlyti du metodai: ribojantis demono kiekį ir kaitinantis demono reikšmę (abu algoritmai apsaugo nuo energijos padidėjimo vėlyvosiose algoritmo vykdymo stadijose). Keli algoritmo variantai, kaip juos pasiūlė autorius aprašyti 3.3.1 ir 3.3.2 paveiksluose:

Kaitinamas Demono algoritmas

1. *Pasirinkti pradinį turą S*
2. *Pasirinkti pradinį demono kiekį D>0*
3. *Pasirinkti kaitinimo grafiką e*
4. *Kartoti:*
   - *4.1. Pasirinkti naują maršrutą S‘*
   - *4.2. Tegu dE = E(S‘) – E(S), kur E(S) yra maršruto ilgis*
   - *4.3. jei dE<=0, priimti naują maršrutą, t.y, S = S‘*
   - *4.4. jei ne, jei dE<D , priimti naują maršrutą, t.y, S = S‘*
   - *4.5. jei ne atmesti naują maršrutą*
   - *4.6. D = D – dE*
   - *4.7. D = e * D*
5. *Kol tenkinama pabaigos sąlyga*

3.3.1 pav. Kintamas demono algoritmas

Apribotas demono algoritmas

1. *Pasirinkti pradinį turą S*
2. *Pasirinkti pradinį demono kiekį D = DB>0*
3. *Kartoti:*
   - *3.1. Pasirinkti naują maršrutą S‘*
   - *3.2. Tegu dE = E(S‘) – E(S), kur E(S) yra maršruto ilgis*
   - *3.3. jei dE<=0, priimti naują maršrutą, t.y, S = S‘*
   - *3.4. jei ne, jei dE<D , priimti naują maršrutą, t.y, S = S‘*
   - *3.5. jei ne atmesti naują maršrutą*
   - *3.6. D = min(D – dE, DB)*
4. *Kol tenkinama pabaigos sąlyga*

3.3.2 pav. Apribotas demono algoritmas

Išvadose buvo pastebėta, kad Demono algoritmas rezultatais nenusileidžia egzistuojančiam AM algoritmui, o kai kuriais atvejais jį nežymiai lenkia.

Kiek vėliau A. Misevičius straipsnyje apie AM algoritmo taikymą KU [45] pastebėjo, kad pradinės ir galinės temperatūrų skaičiavimo formulės bei atkaitinimo grafiko kartu su osciliacijos komponentu patobulinimas gali padėti pagerinti rezultatus, aprašė eksperimentinių tyrimų rezultatus ir identifikavo kitas „jautriąsias“ algoritmo vietas, kurias tobulinant galima tikėtis pagerėjimo AM algoritmo veikime.

Žvelgiant į KU, yra pastebima, kad yra gausybė bandymų rasti geriausiai veikiantį algoritmą uždaviniui spręsti, o AM algoritmas yra vienas tarp duodančių geriausius rezultatus. Būta nemažai bandymų jį pagerinti ir ne retai jie buvo sėkmingi, tačiau ir iki šiol tokį tikslą keliasi nauji tyrinėtojai, nes panašu, kad galimybių naujoms modifikacijoms dar yra.

4. Tyrimo dalis

4.1. Atkaitinimo modeliavimo algoritmo realizacija komivojažieriaus uždaviniui

Komivojažieriaus uždaviniui galima „sukonstruoti“ įvairių aplinkos funkcijų. Dažnai naudojama vadinamoji „porinių sukeitimų“ funkcija Θ_2, kuri apibrėžiama tokia formule

$$\Theta_2(p) = \{\ p' \mid p' \in \Pi,\ \rho(p,p') \le 2\ \}; \tag{3}$$

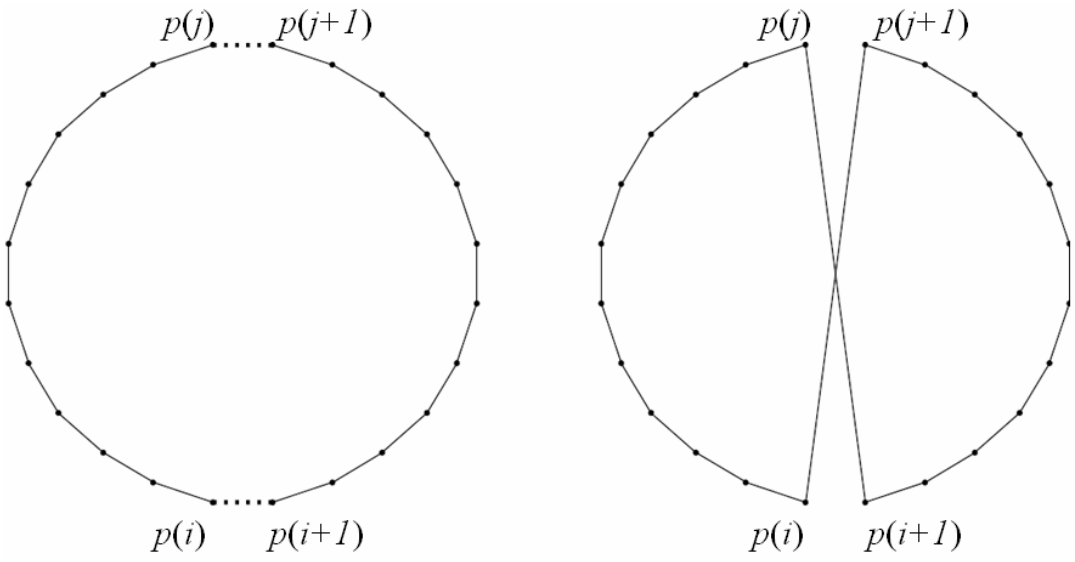
čia p — bet kuris perstatymas iš Π, o $\rho(p,p')$ — atstumas tarp perstatymų p ir p'. Savaime suprantama, kad perstatymams netinka tolydžioms erdvėms įprastas Euklido atstumo tarp taškų apibrėžimas. Perstatymų atveju remiamasi vadinamojo Hamming‘o atstumo samprata. Natūralus būdas formaliai aprašyti šį atstumą tarp perstatymų p ir p' galėtų būti suskaičiavimas elementų, kurie yra skirtingose perstatymų p ir p' pozicijose, t.y. $\rho(p, p') = \left|\{i \mid p(i) \neq p'(i)\}\right|$. Tačiau reikia pasakyti, kad KU atveju operuojama ne su atskirais perstatymų elementais ($j = p(i)$), o su perstatymų elementų poromis, t.y. „atkarpomis“ ($j_1 = p(i)$, $j_2 = p(i+1)$). Turint tai omenyje, atstumo apibrėžimą tikslinga pakoreguoti: atstumas tarp dviejų perstatymų p ir p' lygus skaičiui elementų porų, esančių viename iš perstatymų (pvz. p), bet nesančių kitame (pvz. p') [20]. Matematinis atstumo tarp perstatymų p ir p' aprašymas šiuo atveju yra toks:

$$\rho(p, p') = |\Omega|; \tag{4}$$

čia aibė Ω yra sudaryta iš visų galimų elementų porų (p(i), p((i mod n)+1)) (i $\in$ {1, 2, ..., n}), ir tokių, kad $\exists$ j, ir toks, jog

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

Sprendinio p aplinkos $\Theta_2(p)$ atskirą sprendinį p' galima pasiekti, atlikus mažiau ar daugiau sudėtingą sprendinio p transformaciją (pertvarkymą). KU sprendinių transformacijas, tiksliau „perėjimus“ iš duoto sprendinio į „sprendinį-kaimyną“ aprašysime vadinamojo dviejų „atkarpų“ sukeitimo operatoriaus $\phi(p,i,j)$ pagalba. Operatorius $\phi(p,i,j)$: $\Pi \times \mathrm{N} \times \mathrm{N} \to \Pi$ (čia N yra natūrinių skaičių aibė) duotą perstatymą p transformuoja į kitą perstatymą p', ir taip, kad $\rho(p, p') = 2$; be to, perstatyme p yra pašalinamos elementų poros, esančios būtent i-oje ir j-oje pozicijose, t.y. poros ($p(i)$, $p(i+1)$) ir ($p(j)$, $p((j \bmod n)+1)$); vietoje šių pridedamos (įterpiamos) naujos poros ($p(i)$, $p(j)$) ir ($p(i+1)$, $p((j \bmod n)+1)$) (žr. 4.1 pav.).



4.1.1 pav. Perstatymo iliustracija

Jeigu dar tiksliau, operatorius $\phi(p,i,j)$ suformuoja perstatymą p', ir tokį, jog $p'(i) = p(i)$, $p'(i+1) = p(j)$, $p'(j) = p(i+1)$, $p'((j \bmod n)+1) = p((j \bmod n)+1)$, kai $1 \le i, j \le n \wedge 1 < j-i < n-1$; be to, jeigu $j-i-2 \ge 1$, tai $p'(i+k+1) = p(j-k)$ kiekvienam $k \in \{1, ..., j-i-2\}$ (tai būtina sąlygos $\rho(p, p') = 2$ galiojimo užtikrinimui). Toliau taip pat bus naudojama kompaktiška operatoriaus ϕ forma ϕ_{ij} perėjimui iš nesvarbu kurio perstatymo į perstatymą $\phi(\cdot,i,j)$. Tokiu būdu, pvz. užrašas $p' = p \oplus \phi_{ij}$ reikštų, kad p' yra gautas iš p panaudojant $\phi(p,i,j)$.

Dviejų „atkarpų“ sukeitimo atveju labai nesudėtinga apskaičiuoti tikslo funkcijos (maršruto ilgio) z pokytį Δz, kuris gaunamas iš naujo maršruto p' ($p' = \phi(p,i,j)$) ilgio $z(p')$ atėmus prieš tai buvusio (esamo) maršruto p ilgį $z(p)$, t.y.

$$\Delta z = d_{p(i),p(j)} + d_{p(i+1),p((j \bmod n)+1)} - d_{p(i),p(i+1)} - d_{p(j),p((j \bmod n)+1)}. \quad (5)$$

Šiame darbe pateikiamame AM algoritme pradinė ir galutinė atkaitinimo temperatūra nustatoma pagal A. Misevičiaus pasiūlytą ir sėkmingai išbandytą kvadratinio paskirstymo uždaviniui formulę [46]:

$$\begin{cases} t_0 = (1-\alpha_1)\Delta z^+_{\min} + \alpha_1 \Delta z^+_{\text{vid}} \\ t_g = (1-\alpha_2)\Delta z^+_{\min} + \alpha_2 \Delta z^+_{\text{vid}} \end{cases}; \quad (6)$$

čia t_0, t_g yra atitinkamai pradinė ir galutinė temperatūra, $\Delta z^+_{\min}$ — minimalus teigiamas tikslo funkcijos reikšmių pokytis, o Δz^+_{vid} — teigiamų tikslo funkcijos pokyčių vidurkis ($\Delta z^+_{\min}$, Δz^+_{vid} gaunami, atlikus tam tikrą skaičių bandomųjų sprendinių perėjimų panaudojant tą pačią aplinką Θ_2); α_1, α_2 yra realieji skaičiai ($\alpha_1 \in (0,1]$, $\alpha_2 \in [0,1)$, $\alpha_1 > \alpha_2$).

Pasirinkta glotnaus temperatūros mažinimo formulė (Lundy-Mees (LM) formulė) [33]:

$$t_q = t_{q-1}/(1+\beta t_{q-1})\ ;\ q = 1, 2, ...;\ t_0 = \text{const};\ \beta \ll t_0. \tag{7}$$

Koeficiento β (temperatūros kitimo greičio) reikšmę lengva apskaičiuoti analitiškai, jeigu žinoma pradinė ir galutinė temperatūra bei atkaitinimo proceso iteracijų skaičius — Q (žr. toliau). Šiuo atveju:

$$\beta = (t_0 - t_g)/Qt_0t_g\ . \tag{8}$$

Šioje realizacijoje temperatūra mažinama, atlikus tam tikrą kiekį bandymų — M, t.y. išnagrinėjus ne kurią aplinkos dalį (optimizavimo teorijoje tokia atkaitinimo schema priskiriama vadinamojo homogeninio (subalansuoto) atkaitinimo tipui). Natūralu bandymų skaičių M susieti su aplinkos Θ_2 dydžiu $W = |\ \Theta_2(\cdot)\ |$, t.y. $M = \max\{1, \lambda W\}$, čia λ yra sprendinių aplinkos nagrinėjimo bandymų skaičiaus (paieškos gylio) valdymo parametras (koeficientas) ($\lambda > 0$).

Atkaitinimo procesas, jeigu laikantis teorinių nuostatų, turėtų būti tęsiamas tol, kol galutinė temperatūra taptų lygi 0. Kadangi tai pareikalautų didelių laiko sąnaudų, tai praktinėse AM realizacijose naudojami kiti kriterijai. Mūsų algoritme baigimo sąlygos vaidmenį atlieka iš anksto fiksuotas atkaitinimo proceso iteracijų skaičius (atkaitinimo trukmė) — Q.

AM algoritmo efektyvumui padidinti aukščiau aprašyta nuoseklaus temperatūros mažinimo schema papildyta periodiniais temperatūros svyravimais — osciliacija [34]. Taigi, vietoje monotoninio (statinio) atkaitinimo turime dinaminį atkaitinimą (periodiškai kartojamus laipsniškus atšaldymus ir staigius įkaitinimus) [36].

Dinaminio atkaitinimo schema yra tokia. Parenkamas atkaitinimo iteracijų skaičius Q. Atkaitinimas pradedamas, esant pradinei ir galutinei temperatūrai, apskaičiuotai pagal aukščiau pateiktą formulę. Temperatūra mažinama pagal Lundy-Mees formulę. Kai pasiekiama „stagnacijos“ būsena (nėra perėjimų nuo vieno sprendinio prie kito), atkaitinimas sustabdomas. Procesas tęsiamas su naujais atkaitinimo schemos parametrais: nauja pradine ir galutine temperatūra bei perskaičiuotu koeficientu β. Parametrai perskaičiuojami, atsižvelgiant į stabdymo momentu esančią temperatūrą t^*:

$$t_0 = \alpha_3 t^*,\ t_g = \alpha_4 t^*,\ \beta = (t_0 - t_g)/\min\{q^*, Q-q\}\cdot t_0 \cdot t_g\ ; \tag{9}$$

čia α_3, α_4 yra realieji skaičiai, tenkinantys sąlygas: $\alpha_3 > 1$, $\alpha_4 < 1$; q^* žymi iteracijos, kurios metu pirmą kartą aktyvuota osciliacija, numerį; q yra duotu momentu esančios iteracijos numeris. Tokiu būdu, tuoj po atkaitinimo sustabdymo temperatūra staigiai padidinama, ir vėl pradedamas atšaldymas — tik su naujais parametrais (atšaldant vėl naudojama LM formulė), ir t.t. Algoritmo vykdymas tęsiamas tol, kol patenkinama baigimo

sąlyga, t.y. eilinis atkaitinimo iteracijos numeris q tampa didesnis už maksimalų iteracijų skaičių Q. Dinaminio atkaitinimo pobūdį (osciliacijos aktyvavimo momentą) galima reguliuoti, turint kokią nors stagnacijos būsenos fiksavimo taisyklę. Viena iš paprastų taisyklių yra fiksuoti stagnaciją tuo atveju, kai nėra perėjimų tarp sprendinių tam tikrą laiko tarpą τ (τ gali būti susietas, pvz. su aplinkos dydžiu W, t.y. $\tau = \max\{1, \omega W\}$, čia ω yra osciliacijos valdymo parametro (koeficiento) vaidmenyje ($\omega > 0$)).

Detalizuotas atkaitinimo modeliavimo algoritmo komivojažieriaus uždaviniui šablonas pateiktas 4.1.2 pav.

```
procedure AtkaitinimoModeliavimoAlgoritmas;
// pradiniai duomenys: n – uždavinio dydis (miestų skaičius)
// parametrai: Q – iteracijų skaičius, α1, α2, α3, α4 – temperatūros valdymo parametrai (koeficientai),
//             λ – paieškos gylio valdymo parametras, ω – osciliacijos valdymo parametras
// rezultatai: p* – geriausias surastas sprendinys (maršrutas)
begin
  W := (n(n−1))/2 − n ;  // W – aplinkos Θ2(·) dydis
  sugeneruoti pradinį sprendinį (maršrutą) p atsitiktiniu būdu; p* := p;
  apskaičiuoti Δz+min , Δz+vid vykdant W atsitiktinių sprendinio p perėjimų;
  inicializuoti atkaitinimo parametrus t0, tg, β;  t := t0;
  M := MAX(1,λW); τ := MAX(1,ωW);
  i := 1; j := 2; nėra_perėjimų := 0; osciliacija := FALSE;
  for q := 1 to Q do begin  // pagrindinis atkaitinimo modeliavimo ciklas
    Δz := ∞;
    for m := 1 to M do begin  // atliekama M duoto sprendinio aplinkos nagrinėjimo bandymų
      i := IF(j<n−1+SIGN(i−1), i, IF(i<n−2, i+1, 1)); j := IF(j<n−1+SIGN(i−1), j+1, i+2);
      Δz′ := z( p ⊕ φij ) − z(p);
      if Δz′ < Δz then begin Δz := Δz′; k := i; l := j end
    end;  // for m
    if Δz < 0          then pereiti_į_kitą_sprendinį := TRUE
                       else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
    if pereiti_į_kitą_sprendinį = TRUE then begin
       p := p ⊕ φkl ; if z(p) < z(p*) then p* := p; if Δz ≠ 0 then nėra_perėjimų := 0
    end
    else nėra_perėjimų := nėra_perėjimų + 1;
    if (nėra_perėjimų ≥ τ) or (temperatūra t yra žemiausiame taške) then begin
      if osciliacija = FALSE then begin q* := q; t* := t; osciliacija := TRUE end
      perskaičiuoti atkaitinimo parametrus t0, tg, β (atsižvelgiant į q*, t*);
      t := t0
    end
    else t := t/(1 + β t)  // sumažinama einamoji temperatūra
  end  // for q
end.
```

4.4.2 pav. Atkaitinimo modeliavimo algoritmo komivojažieriaus uždaviniui šablonas.
Pastabos. 1. Funkcijos z reikšmės apskaičiuojamos pagal (1) formulę (žr. 1 sk.).
2. Funkcija IF(x,y_1,y_2) grąžina y_1, jei x = TRUE, arba y_2, jei x = FALSE.
3. Funkcija RANDOM(0,1) generuoja pseudo-atsitiktinį skaičių iš intervalo [0,1)

4.2. Atkaitinimo modeliavimo algoritmo modifikacijos komivojažieriaus uždaviniui

Kaip pagrindinis modifikacijų objektas atkaitinimo modeliavimo algoritme buvo pasirinktas perėjimo tarp sprendinių sąlyga, pagrindiniame atkaitinimo modeliavimo cikle.

Metropolis ir Rozenbluth'as [30] dar pradinėse atkaitinimo modeliavimo algoritmo tyrinėjimo stadijose pasinaudojo eksponentinio pasiskirstymo funkcija apibrėždami tikimybę, kad sistema, įvykus joje tam tikram pasikeitimui, pereis iš vieno energijos lygio (E_1) į kitą lygį (E_2), kai sistemos temperatūra lygi t. Ta tikimybė apskaičiuojama pagal tokią formulę:

$$\mathrm{P}(\Delta E) = \begin{cases} 1, & \Delta E < 0 \\ e^{-\Delta E/Ct}, & \Delta E \geq 0 \end{cases}; \qquad (10)$$

čia ΔE = E2 − E1, o C — konstanta.

Būtent šios tikimybės skaičiavimas, vienas iš ašinių skaičiavimų visame AM algoritme, man pasirodė galintis atnešti daugiausiai teigiamų pokyčių, įvedus atitinkamas modifikacijas jame.

Bendru AM algoritmo atveju šios tikimybinės sąlygos pseudo kodas pateikiamas 4.2.1 paveiksle.

```
if Δz < 0 then begin  // Δz - tikslo funkcijos pokytis
        p := p′; // pereiti į kitą sprendinį
end
else begin
        sugeneruoti atsitiktinį skaičių r iš intervalo [0,1]
        if r < e^(−Δz/t)  // Δz - tikslo funkcijos pokytis, t – sistemos temperatūra
                then  p := p′; // vykdomas perėjimas prie naujo sprendinio
end
```

4.2.1 pav. AM algoritmo tikimybinės sąlygos pseudo kodas

Tyrimo metu realizuotame AM algoritme naudojamas loginis kintamasis, nusakantis ar reikalingas perėjimas prie kito sprendinio prie tam tikros temperatūros atitinkamu laiko momentu, pateikiamas 4.2.2 paveiksle:

```
if  Δz < 0 // Δz - tikslo funkcijos pokytis
    then  pereiti_į_kitą_sprendinį := TRUE
    else  pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
    // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
```

4.2.2 pav. AM algoritmo perėjimo sąlygos loginis kintamasis

Kaip matome, viena iš pagrindinių AM algoritmo sėkmės priežasčių yra jo dinamiškas nenuspėjamumas organizuojant perėjimą prie geresnių sprendinių arba tokių, kurie potencialiai galėtų atvesti prie geresnių rezultatų. Nors ir yra įvestas atsitiktinio skaičiaus generavimo mechanizmas, lemiantis sprendinių pasirinkimus, vis tik jis yra apribotas eksponentine funkcija, kuri garantuoja, kad kuo ilgiau bus vykdomas algoritmas, tuo perėjimo prie sekančio sprendinio tikimybė didės. Tokiu būdu išvengiama „užstrigimo“ lokaliuose optimumuose vėlesniuose algoritmo vykdymo etapuose.

Modifikacijose buvo įvestas naujas, baziniame algoritmo variante nenaudotas kintamasis, apibūdinantis algoritmo vykdymo metu rastų tikslo funkcijų pokyčių vidurkius (TFV). Tolesnėse modifikacijose šis kintamasis yra taip pat modifikuojamas, didinant teigiamų tikslo funkcijų pokyčių įtaką jo skaitinei reikšmei, galiausiai jis apribojimas tik teigiamų tikslo funkcijų vidurkių saugojimu.

Indeksinis temperatūros koeficientas (ITK) – taip pat naujas kintamasis, įvedamas būtent naujai pasiūlytose modifikacijose, jo esmė yra apibūdinti tikslo funkcijų vidurkių įtaką sprendžiant ar reikalingas perėjimas prie kito sprendinio tam tikrais laiko momentais. Kitaip tariant, šis koeficientas yra tiesiogiai įtakojamas esamos temperatūros atkaitimo procese. Kuo temperatūra mažesnė, tuo mažesnis ir pats koeficientas

Tikslo funkcijų pokyčių vidurkių parametro (TFV) ir indeksinio temperatūrų koeficiento (ITK) sandauga – sudaro tolerancijos koeficientą einamos iteracijos metu rastam tikslo funkcijos pokyčiui. Patogumo dėlei įvedamas naujas parametras – šios tolerancijos valdymo parametras (TV). Tolerancijos valdymo parametras tuo didesnis, kuo mažiau apribojimų užduodama TFV kintamajam. Jei, pavyzdžiui, didinama tam tikrų tikslo funkcijų pokyčių įtaką bendram TFV kintamajam, ši tolerancija mažėja, nes koncentruojamasi į kažkokią specifinę tikslo funkcijų pokyčių grupę. Tolerancijos mažėjimo kryptį galima nusakyti ta savybe, kurios įtaka TFV kintamajam yra didinama.

Toliau pateikiamos atliktos bazinio atkaitinimo modeliavimo algoritmo modifikacijos ir jų variantai.

Modifikacija 01_B_R. Indeksinis temperatūros koeficientas kinta nuo 1 iki 0 nuosekliai mažėdamas, Δzv fiksuoja bendrą tikslo funkcijų pokyčių vidurkį, tolerancijos valdymo parametras neaktyvuotas. Perėjimo sąlygos fragmentas pseudo kode 4.2.3 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
   ...
   fiksuojame visas rastas tikslo funkcijų pokyčių reikšmes Δzv kintamajame
   itk = (bendras iteracijų sk – esamos iteracijos numeris)/ bendras iteracijų sk
   if Δz < 0 // Δz - tikslo funkcijos pokytis
      then pereiti_į_kitą_sprendinį := TRUE
      else if Δz< Δzv * itk
            // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
             then pereiti_į_kitą_sprendinį := TRUE
             else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( –Δz/t ),TRUE,FALSE);
                  // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
end.
```

4.2.3 pav. 01_B_R modifikacija

Modifikacija 02_BT_R. Indeksinis temperatūros koeficientas kinta nuo 1 iki 0 nuosekliai mažėdamas, Δzv fiksuoja bendrą tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo parametras teigiamiems tikslo funkcijų pokyčiams suteikia didesnį svorį. Perėjimo sąlygos fragmentas pseudo kode 4.2.4 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
   ...
   fiksuojame visas rastas tikslo funkcijų pokyčių reikšmes Δzv kintamajame;
   if Δz < 0 then Δz įtraukiama į Δzv pakartotinai
   itk = (bendras iteracijų sk – esamos iteracijos numeris)/ bendras iteracijų sk
   if Δz < 0 // Δz - tikslo funkcijos pokytis
      then pereiti_į_kitą_sprendinį := TRUE
      else if Δz< Δzv * itk
            // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
             then pereiti_į_kitą_sprendinį := TRUE
             else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( –Δz/t ),TRUE,FALSE);
                  // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
end.
```

4.2.4 pav. Modifikacija 02_BT_R

Modifikacija 03_T_R. Indeksinis temperatūros koeficientas kinta nuo 1 iki 0 nuosekliai mažėdamas, Δzv fiksuoja tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo parametras griežtai apriboja Δzv tik teigiamų tikslo funkcijų pokyčių fiksavimu. Šiuo atveju teigiamomis tikslo funkcijomis laikomos tos, kurios yra geresnės už tos iteracijos metu esančią Δzv reikšmę, padaugintą iš ITK koeficiento. 03_T_R, nuo prieš tai pasiūlytų

modifikacijų dar skiriasi ir tuo, kad leidžia perėjimus ne tais atvejais, kai Δz yra mažiau už nulį, tačiau reikalauja, kad Δz būtų mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį, padaugintą iš ITK koeficiento. Perėjimo sąlygos fragmentas pseudo kode 4.2.5 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
   …
  itk = (bendras iteracijų sk – esamos iteracijos numeris)/ bendras iteracijų sk
  if Δz< Δzv * itk
          // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
           then begin
                    pereiti_į_kitą_sprendinį := TRUE
                      fiksuojame Δz< Δzv * itk rastas tikslo funkcijų pokyčių
                            reikšmes Δzv kintamajame.
           end
           else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
               // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
end.
```

4.2.5 pav. Modifikacija 03_T_R

Modifikacija 04_B_R. Indeksinis temperatūros koeficientas kinta nuo 1 iki 0 nuosekliai mažėdamas, Δzv fiksuoja tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo parametras neaktyvuotas. Leidžiami tik tie perėjimai tarp sprendinių, kai Δz yra mažesnis už tikslo funkcijų pokyčių vidurkį, padaugintą iš ITK koeficiento. Perėjimo sąlygos fragmentas pseudo kode 4.2.6 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
   …
  fiksuojame visas rastas tikslo funkcijų pokyčių reikšmes Δzv kintamajame;
  itk = (bendras iteracijų sk – esamos iteracijos numeris)/ bendras iteracijų sk
  if Δz< Δzv * itk
          // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
           then pereiti_į_kitą_sprendinį := TRUE
           else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
               // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
end.
```

4.2.6 pav. Modifikacija 04_B_R

Modifikacija 05_T_R. Indeksinis temperatūros koeficientas kinta nuo 1 iki 0 nuosekliai mažėdamas, Δzv fiksuoja tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo

parametras griežtai apriboja Δzv tik teigiamų tikslo funkcijų pokyčių fiksavimu. Šiuo atveju teigiamomis tikslo funkcijomis laikomos tos, kurios yra geresnės už tos iteracijos metu esančią Δzv reikšmę. Leidžiami perėjimus ne tais atvejais, kai Δz yra mažiau už nulį, tačiau reikalaujama, kad Δz būtų mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį. Šios modifikacijos esmė – pašalintas ITK koeficientas. Perėjimo sąlygos fragmentas pseudo kode 4.2.7 pav.

```
for q := 1 to Q do begin  // pagrindinis atkaitinimo modeliavimo ciklas
  ...
  if Δz< Δzv
        // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
        then begin
              pereiti_į_kitą_sprendinį := TRUE
                fiksuojame Δz< Δzv rastas tikslo funkcijų pokyčių
                      reikšmes Δzv kintamajame.
        end
        else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
            // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
end.
```

4.2.7 pav. Modifikacija 05_T_R

Modifikacija 06_T2S_R. Indeksinis temperatūros koeficientas kinta nuo 0,5 iki 0 nuosekliai greitai mažėdamas, koeficientas į algoritmą įvedamas tik įpusėjus algoritmo vykdymą. Δzv fiksuoja tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo parametras griežtai apriboja Δzv tik teigiamų tikslo funkcijų pokyčių fiksavimu. Šiuo atveju teigiamomis tikslo funkcijomis laikomos tos, kurios yra geresnės už tos iteracijos metu esančią Δzv reikšmę pirmame algoritmo etape ir tos, kurios yra geresnės už tos iteracijos metu esančią Δzv, padaugintą iš ITK koeficiento, reikšmę antrame algoritmo etape. Leidžiami perėjimai, kai Δz mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį pirmame algoritmo vykdymo etape ir , kai Δz mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį, padaugintą iš ITK koeficiento, antrame algoritmo vykdymo etape . Perėjimo sąlygos fragmentas pseudo kode 4.2.8 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
  ...
 if dar neįpusėtas algoritmo vykdymas then begin
   if Δz< Δzv
        // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
         then begin
                  pereiti_į_kitą_sprendinį := TRUE
                        fiksuojame Δz< Δzv rastas tikslo funkcijų pokyčių
                        reikšmes Δzv kintamajame.
         end
         else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
 else begin
   itk = (bendras iteracijų sk – esamos iteracijos numeris)/ bendras iteracijų sk
   if Δz< Δzv * itk
        // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
         then begin
                  pereiti_į_kitą_sprendinį := TRUE
                        fiksuojame Δz< Δzv * itk rastas tikslo funkcijų pokyčių
                        reikšmes Δzv kintamajame.
         end
         else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
             // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
 end
end.
```

4.2.8 pav. Modifikacija 06_T2S_R

Modifikacija 07_T2N_R. Indeksinis temperatūros koeficientas kinta nuo 0,5 iki 0 nuosekliai mažėdamas (tačiau mažėdamas lėčiau, nei 06_T2S_R modifikacijoje), koeficientas į algoritmą įvedamas tik įpusėjus algoritmo vykdymą. Δzv fiksuoja tikslo funkcijų pokyčių vidurkį, kai tolerancijos valdymo parametras griežtai apriboja Δzv tik teigiamų tikslo funkcijų pokyčių fiksavimu. Šiuo atveju teigiamomis tikslo funkcijomis laikomos tos, kurios yra geresnės už tos iteracijos metu esančią Δzv reikšmę pirmame algoritmo etape ir tos, kurios yra geresnės už tos iteracijos metu esančią Δzv, padaugintą iš ITK koeficiento, reikšmę antrame algoritmo etape. Leidžiami perėjimai, kai Δz mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį pirmame algoritmo vykdymo etape ir kai Δz mažesnis už teigiamų tikslo funkcijų pokyčių vidurkį, padaugintą iš ITK koeficiento, antrame algoritmo vykdymo etape . Perėjimo sąlygos fragmentas pseudo kode 4.2.9 pav.

```
for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
  ...
 if dar neįpusėtas algoritmo vykdymas then begin
   if Δz< Δzv
        // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
         then begin
                pereiti_į_kitą_sprendinį := TRUE
                      fiksuojame Δz< Δzv rastas tikslo funkcijų pokyčių
                      reikšmes Δzv kintamajame.
         end
         else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
 else begin
   itk = ((bendras iter. sk/2) – (esamos iter. numeris/2))/ (bendras iter. sk/2)
   if Δz< Δzv * itk
        // Δzv - tikslo funkcijų pokyčių vidurkis; Δz – einamos iteracijos tikslo funkcijos pokytis
         then begin
                pereiti_į_kitą_sprendinį := TRUE
                      fiksuojame Δz< Δzv * itk rastas tikslo funkcijų pokyčių
                      reikšmes Δzv kintamajame.
         end
         else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
            // Funkcija IF(x,y1,y2) grąžina y1, jei x = TRUE, arba y2, jei x = FALSE
 end
end.
```

4.2.9 pav. Modifikacija 07_T2N_R

Tyrimo metu buvo atlikta ir daugiau modifikacijų, tame tarpe ir bandymai atisakyti RANDOM funkcijos atkaitinimo modeliavimo procese, tačiau rezultatuose buvo pastebimas pablogėjimas iki kelių dešimčių kartų, dėl to tęsti tyrimus šia kryptimi buvo atsisakyta. Šios ir kitos modifikacijos čia nėra pateikiamos, nes jos visos daugiau mažiau atsispindi aukščiau aprašytose modifikacijose, arba vienaip ar kitaip vedė prie jų.

5. Eksperimentinė dalis.

5.1. Eksperimentų su atkaitinimo modeliavimo algoritmu metodika bei tikslai

Tiriant sudaryto atkaitinimo modeliavimo algoritmo efektyvumą, atlikti išsamūs eksperimentiniai tyrimai su komivojažieriaus uždavinio testiniais pavyzdžiais (duomenimis) iš viešos elektroninės KU testinių pavyzdžių bibliotekos — TSPLIB [47]. Prad=ioje eksperimentuota su šešiolika skirtingų testinių pavyzdžių. Kaip kriterijus algoritmo efektyvumo laipsniui įvertinti buvo pasirinktas santykinės gaunamų sprendinių kokybės rodiklis, tiksliau, gaunamų tikslo funkcijos reikšmių, t.y. maršrutų ilgių vidutinis santykinis nuokrypis nuo tikėtinos optimalios tikslo funkcijos reikšmės (minimalaus galimo maršruto ilgio). (Aišku, kad kuo šis nuokrypis mažesnis, tuo efektyvumo laipsnis didesnis.) Vidutinis santykinis nuokrypis (jį žymėsime $\bar{\delta}$) apibrėžiamas pagal formulę:

$$\bar{\delta} = 100(\bar{z} - z_{\text{opt}})/z_{\text{opt}}\ [\%]; \tag{15}$$

čia $\bar{z}$ yra gautų tikslo funkcijos reikšmių (maršrutų ilgių) vidurkis, kuris apskaičiuojamas, atlikus 10 algoritmo pakartotinių vykdymų (kiekvieną kartą — su vis kitu pradiniu atsitiktiniu sprendiniu); z_{opt} yra tikėtina optimali tikslo funkcijos reikšmė (šios reikšmės pateikiamos bibliotekoje TSPLIB [47]).

Vienas iš pagrindinių eksperimentinių tyrimų tikslas — nustatyti geriausias empirines AM algoritmo valdymo parametrų (koeficientų) (žr. 5.1.1 lentelę) reikšmes. Tuo pačiu siekta išsiaiškinti, kurie būtent algoritmo valdymo parametrai turi didžiausią įtaką algoritmo efektyvumui (t.y. gaunamų sprendinių kokybei, įvertinamai kriterijumi $\bar{\delta}$). Optimalaus valdymo parametrų reikšmių rinkinio apskaičiavimas yra atskiras sudėtingas uždavinys. Šių eksperimentų metu pasinaudota supaprastinta parametrų tyrimo metodika. Metodikos esmė — suformuoti kiek tai įmanoma priimtinesnį (naudojamo efektyvumo kriterijaus atžvilgiu) parametrų reikšmių poaibį, vykdant nuoseklius savarankiškus (mini-)eksperimentus su atskirais parametrais.

5.1.1 lentelė. Eksperimentuose tirti AM algoritmo valdymo parametrai

Nr.	Parametras		Galimos reikšmės
	Parametro pavadinimas	**Žymėjimas**	
1	Iteracijų skaičius	Q	≥ 1
2	Pradinės temperatūros valdymo parametras (koeficientas)	α_1	(0,1]
3	Galutinės temperatūros valdymo parametras (koeficientas)	α_2	[0,1)
4	Temperatūros didinimo (osciliacijos metu) koeficientas	α_3	>1
5	Temperatūros mažinimo (osciliacijos metu) koeficientas	α_4	[0,1)
6	Paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras (koeficientas)	λ	(0,1]
7	Osciliacijos valdymo parametras (koeficientas)	ω	>0

Reikia pabrėžti, kad eksperimentavimas buvo pradedamas, turint jau iš anksto sudarytą tam tikrą preliminarią pradinę parametrų reikšmių konfigūraciją. Pradinis parametrų reikšmių rinkinys gali žymiai įtakoti eksperimentų metu gaunamas reikšmes, todėl yra labai gerai, jeigu preliminarus apriorinių parametrų reikšmių rinkinys sudaromas optimizavimo srities specialisto ar kito eksperto, remiantis kuriomis nors išankstinėmis teorinėmis prielaidomis. Mūsų AM algoritmo apriorinių valdymo parametrų reikšmių rinkinys pateiktas 5.1.2 lentelėje.

5.1.2 lentelė. Apriorinės parametrų reikšmės

Nr	Parametras		Reikšmė
	Parametro pavadinimas	**Žymėjimas**	
1	Iteracijų skaičius	Q	10^6
2	Pradinės temperatūros valdymo parametras (koeficientas)	α_1	0,3
3	Galutinės temperatūros valdymo parametras (koeficientas)	α_2	0,003
4	Temperatūros didinimo (osciliacijos metu) koeficientas	α_3	5
5	Temperatūros mažinimo (osciliacijos metu) koeficientas	α_4	0,4
6	Paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras (koeficientas)	λ	0,03
7	Osciliacijos valdymo parametras (koeficientas)	ω	0,27

Vykdant eksperimentus su duotu parametru, visų kitų rinkinio parametrų reikšmės yra fiksuotos (nekintančios). Eksperimentuodami su atskiromis parametrų reikšmėmis, mes lyg eksperimentuojame su skirtingomis algoritmo modifikacijomis. Palyginę duotos modifikacijos rezultatus (sprendinių kokybę) su kitų modifikacijų rezultatais, galime spręsti apie tos modifikacijos (taigi, ir atitinkamos parametro reikšmės) gerumą. Tokiu būdu galima arba atmesti duotą modifikaciją (parametro reikšmę) kaip neefektyvią, arba laikyti ją „atskaitos tašku“ tolimesniems eksperimentams. Tęsiant tokia „bandymų ir klaidų“ metodika besiremiantį eksperimentavimą, išsiaiškinama geriausia modifikacija (t.y. tokia tiriamo parametro reikšmė, kuri įgalina gauti geriausius sprendinius — mažiausią vidutinį nuokrypį

$\bar{\delta}$). Panašiai elgiamasi su likusiais parametrais. Atlikus tokius eksperimentus su visais parametrais, gaunamas daugiau mažiau geras (arba algoritmo tyrėją tenkinantis) parametrų reikšmių rinkinys.

Eksperimentams su algoritmo modifikacijomis, kaip bazinis ir pagrindinis parametrų rinkinys buvo naudotas optimizuotas parametrų sąrašas pateikiamas 5.2.5 lentelėje. Kad patvirtinti gautų rezultatų, naudojant modifikacijas, tendencingumą, papildomi eksperimentai buvo atliekami, pakeitus paieškos gylio parametro reikšmę (λ) iš optimalios 0,01 į atnešusią prastesnius rezultatus 0,003. Šis (pasitikrinimo) parametras buvo pasirinktas todėl, kad jis turi tiesioginės įtakos tikslo funkcijos Δz pokyčio nustatyme, o tikslo funkcijos pokyčiai siūlomose modifikacijose vaidina ypač svarbų vaidmenį.

Antroje eksperimentavimo dalyje eksperimentų testiniai pavyzdžiai buvo suskirstyti į dvi skirtingas grupes. Pirmoji grupė buvo naudojama tokia pati kaip ir eksperimentuose su parametrais (žr. 5.1.3 lentelę). Antroje grupėje testiniai pavyzdžiai parinkti didesnės apimties miestų skaičiaus prasme, kad būtų įmanoma patikrinti modifikacijos efektyvumą ir didesniems KU variantams (žr. 5.1.4 lentelę).

5.1.3 lentelė. Pirmoji testinių pavyzdžių grupė

Pirmoji testinių pavyzdžių grupė		
Nr	Pavadinimas	Miestų skaičius
1	bier127	127
2	ch130	130
3	eil101	101
4	gr96	96
5	gr120	120
6	kroa100	100
7	krob100	100
8	kroc100	100
9	lin105	105
10	pr107	107
11	rat99	99
12	rd100	100
13	st70	70
14	ts225	225
15	tsp225	225
16	u159	159
Vidurkis:		**123**

5.1.4 lentelė. Antroji testinių pavyzdžių grupė

Antroji testinių pavyzdžių grupė		
Nr	Pavadinimas	Miestų skaičius
1	d657	625
2	fl417	417
3	gr666	666
4	lin318	318
5	si535	535
6	u574	574
7	ts225	225
8	tsp225	225
Vidurkis:		**448**

Eksperimentams su algoritmo modifikacijomis, kaip kriterijus algoritmo efektyvumo laipsniui įvertinti buvo pasirinktas ne tik santykinės gaunamų sprendinių kokybės rodiklis, bet ir algoritmo vykdymo laikas T.

Pagrindinis eksperimentinių su algoritmo modifikacijomis tyrimų tikslas — nustatyti pasiūlytų modifikacijų efektyvumą, ištiriant, kurios iš jų duoda geresnius rezultatus, ir kurios turi neigiamos įtakos algoritmo rezultatų kokybei. Šiais eksperimentais buvo remiamasi sprendžiant ar pavyko pagerinti bazinį algoritmo variantą, įvedant atitinkamas mutacijas naudojamame atkaitimo modeliavimo procese.

5.2. Eksperimentai su baziniu atkaitinimo modeliavimo algoritmu

Eksperimentavimo su bazinio atkaitinimo modeliavimo algoritmu ir jo parametrais pagal aukščiau aprašytą metodiką eigoje gauti rezultatai aprašomi šiame skyriuje. Čia pateikiamos optimizuotos parametrų reikšmės, pateikiami konkretūs rezultatai (santykiniai sprendinių nuokrypiai), gauti su tirtomis parametrų reikšmėmis, (palyginimui yra parodyti ir rezultatai, gauti su neoptimizuotomis parametrų reikšmėmis), paaiškinami eksperimentų rezultatai.

5.2.1 lentelė. Eksperimentų rezultatai, gauti su pradinėmis ir optimizuotomis parametrų reikšmėmis

Nr	**Testinio pavyzdžio pavadinimas**[‡]	**Optimali tikslo funkcijos reikšmė (minimalus maršruto ilgis) (z_{opt})**	**Vidutinis santykinis nuokrypis nuo optimumo ($\bar{\delta}$)**			
			Pradinis (neoptimizuotas) parametrų rinkinys ($Q = 10^6$)	**Pradinis (neoptimizuotas) parametrų rinkinys ($Q = 10^7$)**	**Optimizuotas parametrų rinkinys ($Q = 10^6$)**	**Optimizuotas parametrų rinkinys ($Q = 10^7$)**
1	bier127	118282	0,826	0,245	0,800	0,080
2	ch130	6110	0,586	0,239	0,496	0,240
3	eil101	629	0,223	0,016	0,111	0,000
4	gr96	55209	0,725	0,137	0,190	0,130
5	gr120	6942	0,808	0,311	0,347	0,258
6	kroa100	21282	0,346	0,067	0,061	0,044
7	krob100	22141	0,786	0,145	0,268	0,002
8	kroc100	20749	0,438	0,075	0,224	0,002
9	lin105	14379	0,403	0,064	0,125	0,000
10	pr107	44303	0,018	0,000	0,107	0,126
11	rat99	1211	0,140	0,000	0,041	0,000
12	rd100	7910	0,474	0,044	0,248	0,000
13	st70	675	0,430	0,000	0,252	0,000
14	ts225	126643	1,260	0,059	0,030	0,006
15	tsp225	3916	1,034	0,889	1,021	0,988
16	u159	42080	0,640	0,603	0,662	0,679
Vidurkis:			0,571	0,181	0,311	0,160

[‡ numeris, esantis testinio pavyzdžio pavadinime, nurodo miestų skaičių]

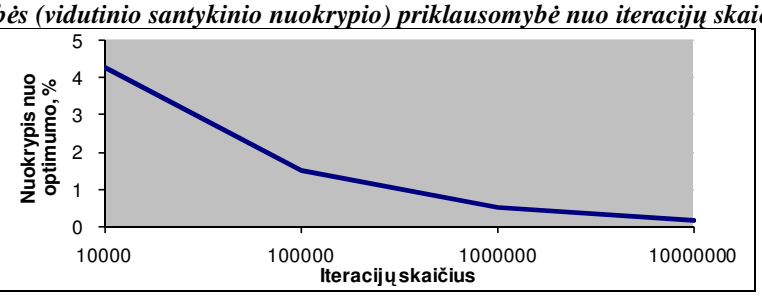
5.2.1 lentelėje pateikiami iteracijų skaičiaus (Q) parametro įtaka algoritmo vykdymo rezultatams. Atkreipkite dėmesį, kad rezultatai su neoptimizuotais parametrais naudoja

apriorines parametrų reikšmes, o, vaizdumo dėlei, šalia yra pateikiami rezultatai ir iteracijų skaičiaus įtaka galutiniam rezultatui, naudojant optimizuotas parametrų reikšmes.

Kaip ir buvo galima iš anksto tikėtis, didžiausią įtaką gaunamų sprendinių kokybei turi iteracijų skaičiaus parametras Q, nusakantis atkaitinimo proceso trukmę. Natūralu, jog kuo daugiau atliekama iteracijų, tuo tikslesni gaunami rezultatai. Tačiau būtina turėti omenyje, kad didinant iteracijų skaičių, atitinkamai pailgėja algoritmo vykdymo laikas (ypač esant didesnėms KU duomenų apimtims). Čia svarbu, kokie yra konkretūs prioritetai ir kiek išaugęs vykdymo laikas „atperkamas“ pagerėjusia rezultatų kokybe. Bendrai, net ir pakankamai ilgas, bet tenkinantis tyrėją (vartotoją) vykdymo laikas gali būti pateisinamas, jeigu tai leidžia apčiuopiamai pagerinti sprendinių kokybės įverčius. (Geriau yra gauti aukštos kokybės rezultatus per ilgesnį, bet priimtiną laiką, negu tenkintis blogais rezultatais, gautais per labai trumpą laiką.) Gaunamų sprendinių kokybės priklausomybė nuo iteracijų skaičiaus iliustruojama 5.2.2 lentelėje.

5.2.2 lentelė. Sprendinių kokybės (vidutinio santykinio nuokrypio) priklausomybė nuo iteracijų skaičiaus

Iteracijų skaičius	$\bar{\delta}$
10^4	4,272
10^5	1,500
10^6	0,541
10^7	0,181



Panašią įtaką kaip ir iteracijų skaičius turi paieškos gylio (sprendinių aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras λ. Galima pastebėti aiškią koreliaciją tarp šio parametro reikšmių ir vidutinio santykinio sprendinių kokybės nuokrypio reikšmių (žr. 5.2.3 lentelę). Iš tiesų, didinant nagrinėjamų aplinkos sprendinių skaičių, kitaip tariant, atidžiau išnagrinėjant sprendinių aplinką (didinant paieškos tikslumą) kiekvienos iteracijos metu, ženkliai didėja ir tikimybė aptikti geresnės kokybės sprendinius. Vėlgi reikia akcentuoti, jog paieškos gylio didinimas implikuoja išaugusią skaičiavimų apimtį, taigi, ir algoritmo vykdymo trukmę. Rezultatų pagerėjimas yra padidėjusių skaičiavimų resursų kaina.

5.2.3 lentelė. Paieškos gylio valdymo koeficiento įtaka rezultatų kokybei

Nr.	Testinio pavyzdžio pavadinimas	Vidutinis santykinis nuokrypis ($\bar{\delta}$)		
		λ = 0,0005	λ = 0,003	λ = 0,01
1	bier127	1,150	0,826	0,656
2	ch130	1,030	0,586	0,524
3	eil101	1,962	0,223	0,060
4	gr96	1,301	0,725	0,341
5	gr120	1,071	0,808	0,556
6	kroa100	0,358	0,346	0,233
7	krob100	0,782	0,786	0,256
8	kroc100	0,567	0,438	0,112
9	lin105	0,245	0,403	0,173
10	pr107	0,111	0,018	0,000
11	rat99	0,930	0,140	0,080
12	rd100	0,711	0,474	0,280
13	st70	0,759	0,430	0,311
14	ts225	2,471	1,260	0,132
15	tsp225	1,478	1,034	1,178
16	u159	1,467	0,640	0,678
	Vidurkis:	1,025	0,571	0,350

Labai svarbūs AM algoritmo parametrai yra atkaitinimo temperatūros dinamikos valdymo parametrai, ypač tai pasakytina apie pradinės ir galutinės temperatūros valdymo parametrus (koeficientus α1, α2), nulemiančius atkaitinimo proceso charakterį, tuo pačiu, gaunamus galutinius rezultatus. Eksperimentų rezultatai iš esmės patvirtina hipotetines prielaidas apie pradinės ir galutinės temperatūros reikšmių suderinimo įtaką gaunamų sprendinių kokybei. Kaip ir spėta, jei pradinė temperatūra per daug aukšta (o/arba galutinė temperatūra per žema), tai konvergavimo į aukštos kokybės sprendinius procesas užsitęsia pernelyg ilgai. Kita vertus, per daug žema pradinė (arba per aukšta galutinė) temperatūra neužtikrina pakankamai geros kokybės sprendinių suradimo. Abiem atvejais algoritmo efektyvumo laipsnis yra mažesnis (žr. 5.2.4 lentelę). Vis tik, kaip matyti iš 5.2.4 lentelėje pateiktų rezultatų, parinkti optimalias pradinės ir galutinės temperatūros reikšmes nėra jau taip trivialu. Galima iš tų rezultatų įžvelgti, kad algoritmas gana „jautriai“ reaguoja į temperatūros valdymo koeficientų reikšmių pakitimus — tai patvirtina šių faktorių reikšmę algoritmo efektyvumui. Net ir nežymūs temperatūros valdymo koeficientų α1, α2 pokyčiai gali iššaukti ryškius sprendinių kokybės įvertinimų svyravimus (žr. 5.2.4 lentelės 7–11 stulpelius); kai kuriais atvejais tie svyravimai yra gana nenuspėjami (žr. 5.2.4 lentelės 2–4 stulpelius). (Kas liečia temperatūros valdymo osciliacijos metu parametrus α3, α4, kaip ir

pačios osciliacijos valdymo parametrą ω, tai šių parametrų reikšmių pokyčiai turi tik minimalų poveikį galutiniams rezultatams.)

5.2.4 lentelė. AM algoritmo rezultatų priklausomybė nuo temperatūros parametrų

Testinio pavyzdžio pavadinimas	**Vidutinis santykinis nuokrypis ($\bar{\delta}$)**									
	α_1 = 0,1	**α_1 = 0,3**	**α_1 = 0,5**	**α_1 = 0,7**	**α_1 = 0,9**	**α_1 = 0,7**				
	α_2 = 0,006					**α_2 = 0,1**	**α_2 = 0,01**	**α_2 = 0,003**	**α_2 = 0,006**	**α_2 = 0,0001**
bier127	0,545	0,826	1,140	0,742	0,917	3,771	0,817	0,826	0,321	1,839
ch130	0,949	0,586	0,901	0,624	0,638	5,448	0,419	0,586	0,656	0,712
eil101	0,095	0,223	0,273	0,191	0,413	7,075	0,827	0,223	0,478	0,175
gr96	0,568	0,725	0,606	0,574	0,588	4,284	0,449	0,725	0,333	1,691
gr120	0,749	0,808	0,691	0,703	0,763	5,774	0,575	0,808	0,620	1,165
kroa100	0,234	0,346	0,356	0,302	0,290	3,681	0,165	0,346	0,281	0,731
krob100	0,326	0,786	0,360	0,526	0,509	4,651	0,454	0,786	0,351	0,657
kroc100	0,464	0,438	0,510	0,288	0,322	4,015	0,231	0,438	0,222	0,901
lin105	0,631	0,403	0,367	0,170	0,309	3,586	0,233	0,403	0,110	0,156
pr107	0,008	0,018	0,031	0,038	0,000	6,439	0,270	0,018	0,102	0,010
rat99	0,173	0,140	0,256	0,264	0,256	7,027	0,223	0,140	0,145	0,149
rd100	0,375	0,474	0,567	0,574	0,784	5,202	0,159	0,474	0,356	0,799
st70	0,311	0,430	0,133	0,415	0,430	3,230	0,341	0,430	0,280	0,326
ts225	1,140	1,260	0,254	0,149	0,927	6,194	0,197	1,260	0,142	4,062
tsp225	0,769	1,034	1,253	1,315	1,264	6,992	1,652	1,034	1,094	1,864
u159	0,935	0,640	0,790	0,823	0,687	6,043	0,681	0,640	0,629	2,164
Vidurkis:	0,517	0,571	0,531	0,481	0,569	5,213	0,481	0,571	0,383	1,088

Galiausiai pateikiamos optimalios skaitinės reikšmės (žr. 5.2.5 lentelę), aktualiausių tyrime naudotų parametrų.

5.2.5 lentelė. Optimizuotos parametrų reikšmės

Nr.	**Parametras**		**Reikšmė**
	Parametro pavadinimas	**Žymėjimas**	
1	Iteracijų skaičius	Q	10^7
2	Pradinės temperatūros valdymo parametras (koeficientas)	α_1	0,7
3	Galutinės temperatūros valdymo parametras (koeficientas)	α_2	0,006
4	Temperatūros didinimo (osciliacijos metu) koeficientas	α_3	10
5	Temperatūros mažinimo (osciliacijos metu) koeficientas	α_4	0,8
6	Paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras (koeficientas)	λ	0,01
7	Osciliacijos valdymo parametras (koeficientas)	ω	0,15

5.3. Eksperimentai su bazinio atkaitinimo modeliavimo algoritmo modifikacijomis

Atliktų eksperimentų rezultatai pateikiami 5.3.(1-8) lentelėse bei 5.3.(1-3) pav.

5.3.1 lentelė. Eksperimentų su 01_B_R modifikacija rezultatai

Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 01_B_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	2,062	0,854
2	ch130	0,514	0,496	1,25	0,689
3	eil101	0,477	0,111	2,226	0,175
4	gr96	0,461	0,19	2,963	0,846
5	gr120	0,536	0,347	1,746	0,775
6	kroa100	0,394	0,061	0,858	0,84
7	krob100	0,272	0,268	1,674	0,915
8	kroc100	0,339	0,224	1,255	0,405
9	lin105	0,136	0,125	0,393	0,209
10	pr107	0,251	0,107	4,883	0,13
11	rat99	0,297	0,041	0,884	0,124
12	rd100	0,37	0,248	1,381	1,149
13	st70	0,385	0,252	1,156	0,474
14	ts225	0,224	0,03	0,411	0,137
15	tsp225	1,249	1,021	1,987	0,97
16	u159	0,716	0,662	1,444	0,8
	Vidurkis:	0,4564375	0,311438	1,660813	0,59325

5.3.2 lentelė. Eksperimentų su 02_BT_R modifikacija rezultatai

Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 02_BT_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	0,957	0,731
2	ch130	0,514	0,496	1,216	0,54
3	eil101	0,477	0,111	1,176	1,431
4	gr96	0,461	0,19	2,696	0,453
5	gr120	0,536	0,347	1,704	0,536
6	kroa100	0,394	0,061	0,251	0,371
7	krob100	0,272	0,268	2,118	0,593
8	kroc100	0,339	0,224	1,873	0,157
9	lin105	0,136	0,125	0,349	0,273
10	pr107	0,251	0,107	0,159	1,243
11	rat99	0,297	0,041	1,098	0,462
12	rd100	0,37	0,248	0,822	0,268
13	st70	0,385	0,252	0,148	1,304
14	ts225	0,224	0,03	2,025	0,114
15	tsp225	1,249	1,021	1,869	1,067
16	u159	0,716	0,662	5,614	0,685
	Vidurkis:	0,4564375	0,311438	1,5047	0,63925

5.3.3 lentelė. Eksperimentų su 03_T_R modifikacija rezultatai

Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 03_T_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	0,451	0,841
2	ch130	0,514	0,496	0,501	0,661
3	eil101	0,477	0,111	0,397	0,064
4	gr96	0,461	0,19	0,327	0,148
5	gr120	0,536	0,347	0,556	0,431
6	kroa100	0,394	0,061	0,204	0,06
7	krob100	0,272	0,268	0,213	0,336
8	kroc100	0,339	0,224	0,194	0,131
9	lin105	0,136	0,125	0,305	0,046
10	pr107	0,251	0,107	0,291	0,13
11	rat99	0,297	0,041	0,107	0,008
12	rd100	0,37	0,248	0,488	0,01
13	st70	0,385	0,252	0,207	0,193
14	ts225	0,224	0,03	0,48	0,038
15	tsp225	1,249	1,021	1,553	0,894
16	u159	0,716	0,662	0,559	0,606
	Vidurkis:	0,4564375	0,311438	0,4271	0,28731

5.3.4 lentelė. Eksperimentų su 04_B_R modifikacija rezultatai

Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 04_B_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	1,095	0,854
2	ch130	0,514	0,496	1,039	0,774
3	eil101	0,477	0,111	0,843	0,159
4	gr96	0,461	0,19	1,157	0,862
5	gr120	0,536	0,347	1,606	0,931
6	kroa100	0,394	0,061	0,742	0,84
7	krob100	0,272	0,268	1,769	0,952
8	kroc100	0,339	0,224	0,683	0,292
9	lin105	0,136	0,125	0,496	0,166
10	pr107	0,251	0,107	0,32	0,125
11	rat99	0,297	0,041	0,553	0,107
12	rd100	0,37	0,248	0,861	1,149
13	st70	0,385	0,252	0,578	0,415
14	ts225	0,224	0,03	0,65	0,137
15	tsp225	1,249	1,021	1,619	0,809
16	u159	0,716	0,662	1,283	0,679
	Vidurkis:	0,4564375	0,311438	0,9559	0,57819

5.3.5 lentelė. Eksperimentų su 05_T_R modifikacija rezultatai

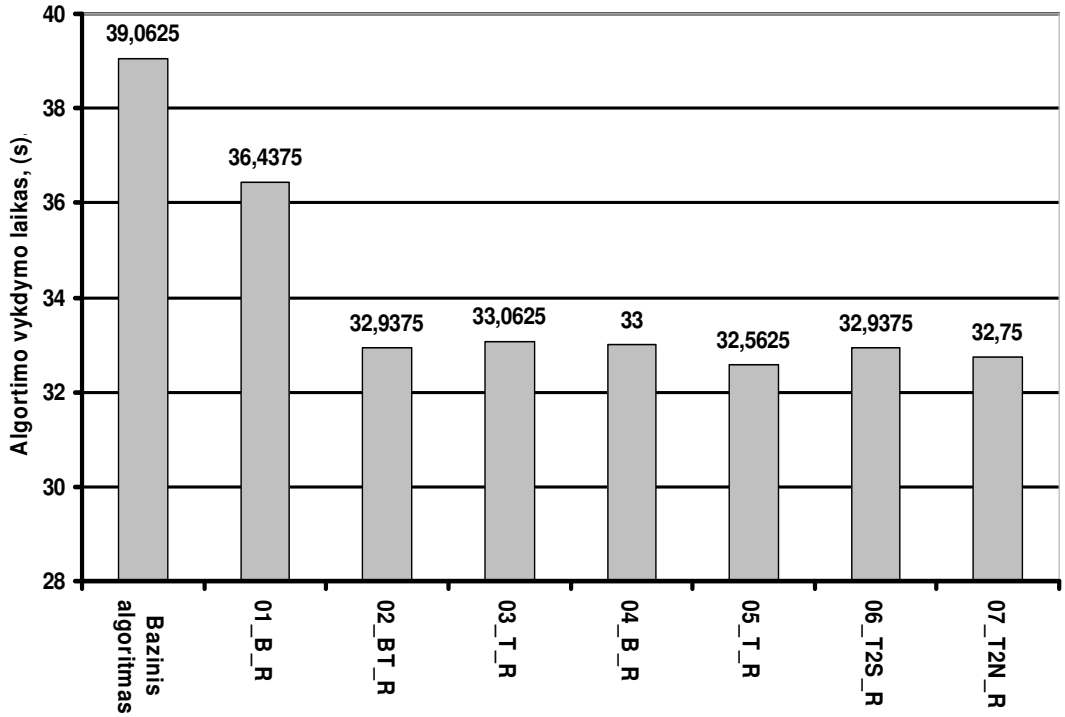
Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 05_T_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	0,554	0,58
2	ch130	0,514	0,496	0,537	0,619
3	eil101	0,477	0,111	0,62	0,175
4	gr96	0,461	0,19	0,475	0,344
5	gr120	0,536	0,347	0,493	0,55
6	kroa100	0,394	0,061	0,162	0,096
7	krob100	0,272	0,268	0,296	0,183
8	kroc100	0,339	0,224	0,281	0,021
9	lin105	0,136	0,125	0,237	0,115
10	pr107	0,251	0,107	0,369	0,198
11	rat99	0,297	0,041	0,231	0,041
12	rd100	0,37	0,248	0,364	0,322
13	st70	0,385	0,252	0,341	0,089
14	ts225	0,224	0,03	0,197	0,02
15	tsp225	1,249	1,021	1,496	0,945
16	u159	0,716	0,662	0,785	0,523
	Vidurkis:	0,4564375	0,311438	0,46488	0,30131

5.3.6 lentelė. Eksperimentų su 06_T2S_R modifikacija rezultatai

Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 06_T2S_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	0,554	0,58
2	ch130	0,514	0,496	0,552	0,62
3	eil101	0,477	0,111	0,509	0,127
4	gr96	0,461	0,19	0,475	0,344
5	gr120	0,536	0,347	0,493	0,569
6	kroa100	0,394	0,061	0,162	0,096
7	krob100	0,272	0,268	0,296	0,183
8	kroc100	0,339	0,224	0,281	0,021
9	lin105	0,136	0,125	0,237	0,115
10	pr107	0,251	0,107	0,289	0,214
11	rat99	0,297	0,041	0,157	0,058
12	rd100	0,37	0,248	0,364	0,322
13	st70	0,385	0,252	0,385	0,163
14	ts225	0,224	0,03	0,149	0,017
15	tsp225	1,249	1,021	1,509	0,924
16	u159	0,716	0,662	0,776	0,521
	Vidurkis:	0,4564375	0,311438	0,44925	0,30463

5.3.7 lentelė. Eksperimentų su 07_T2N_R modifikacija rezultatai

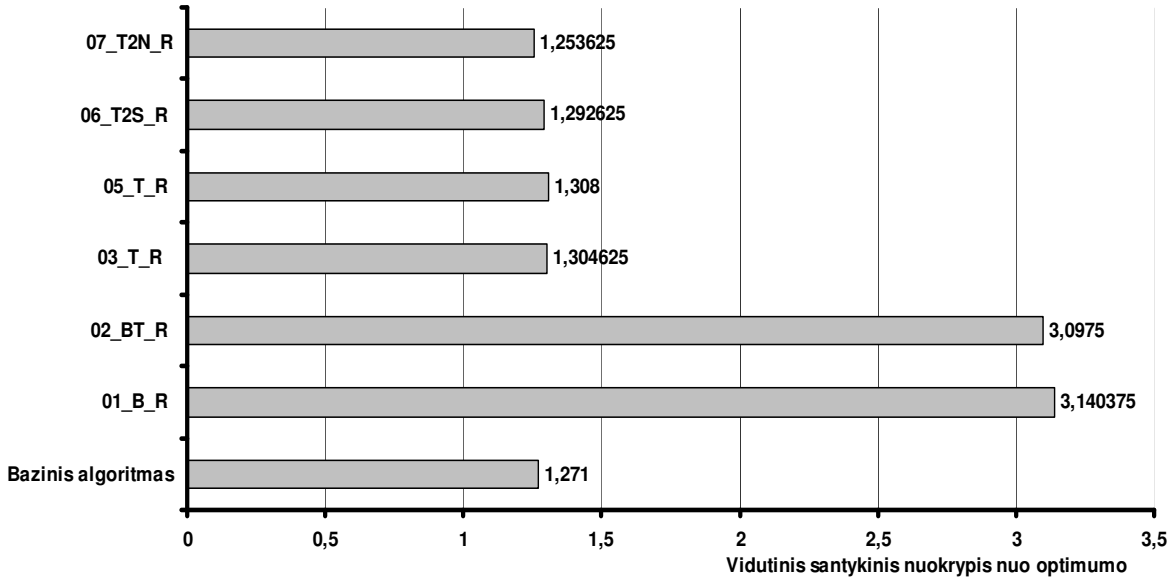
Nr	Testinio pavyzdžio pavadinimas	$\bar{\delta}$, kai naudojamas bazinis algoritmo variantas		$\bar{\delta}$, kai naudojama 07_T2N_R modifikacija	
		$\lambda = 0{,}003$	$\lambda = 0{,}01$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	0,682	0,8	0,554	0,58
2	ch130	0,514	0,496	0,552	0,62
3	eil101	0,477	0,111	0,509	0,127
4	gr96	0,461	0,19	0,475	0,344
5	gr120	0,536	0,347	0,493	0,569
6	kroa100	0,394	0,061	0,162	0,096
7	krob100	0,272	0,268	0,296	0,183
8	kroc100	0,339	0,224	0,281	0,021
9	lin105	0,136	0,125	0,237	0,115
10	pr107	0,251	0,107	0,289	0,214
11	rat99	0,297	0,041	0,157	0,058
12	rd100	0,37	0,248	0,364	0,322
13	st70	0,385	0,252	0,385	0,163
14	ts225	0,224	0,03	0,149	0,017
15	tsp225	1,249	1,021	1,509	0,924
16	u159	0,716	0,662	0,776	0,521
	Vidurkis:	0,4564375	0,311438	0,44925	0,30463



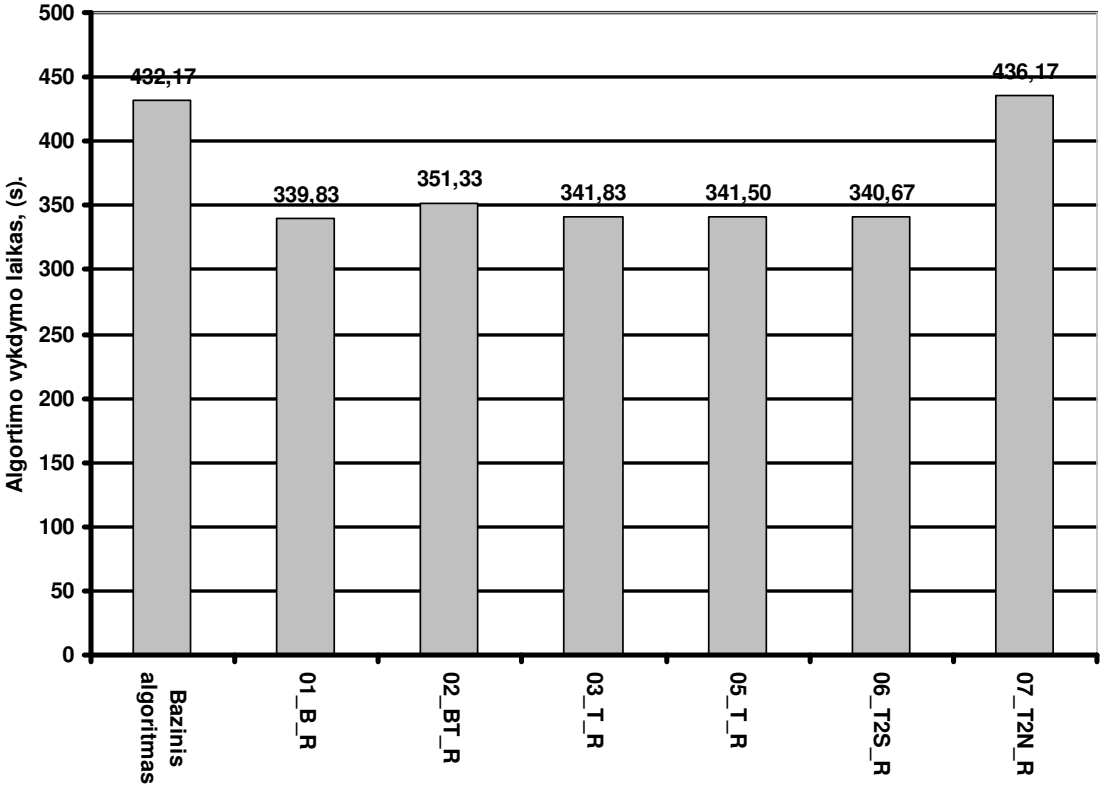
5.3.1 pav. Eksperimentų atliktoms modifikacijoms vykdymo laikas.

5.3.8 lentelė. Eksperimentų su modifikacijomis, esant didelėm KU apimtimis, rezultatai

Nr	Testinio pavyzdžio pavad.	Vidutinis santykinis nuokrypis nuo optimumo ($\bar{\delta}$)					
		Bazinis algoritmas	03_T_R	01_B_R	02_BT_R	05_T_R	07_T2N_R
1	d657	1068	524	526	545	524	1055
2	fl417	187	188	188	194	189	188
3	gr666	527	531	523	528	529	545
4	lin318	95	97	94	108	94	100
5	si535	335	332	329	337	332	342
6	u574	381	379	379	396	381	387
7	ts225	102	93	92	93	93	91
8	tsp225	102	97	92	93	93	93



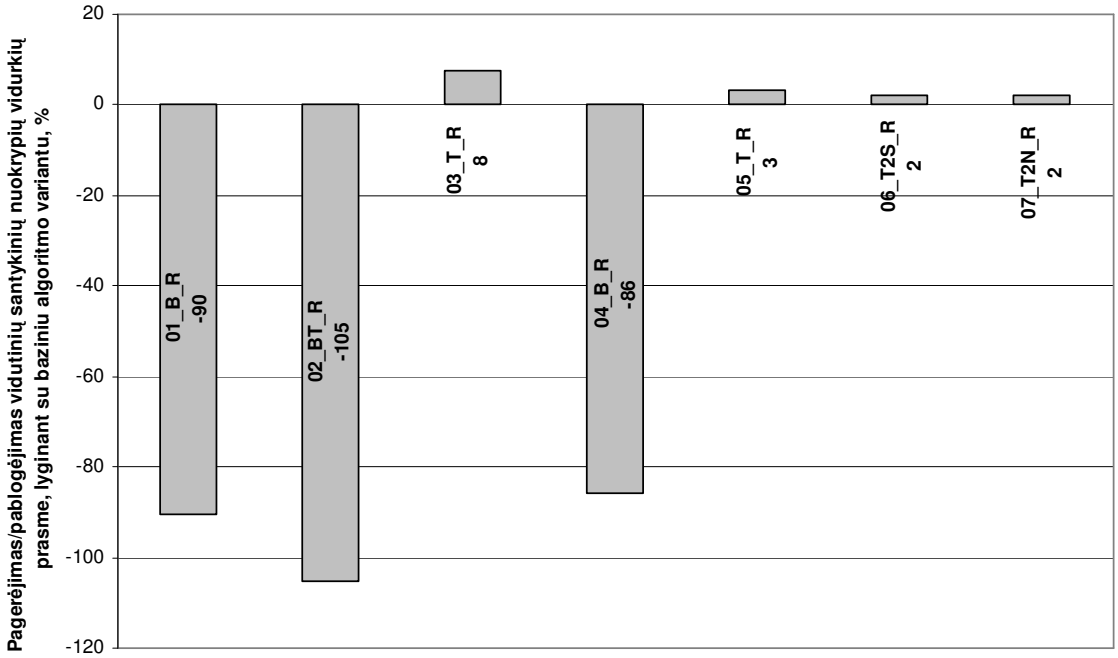
5.3.2 pav. Vidutinių santykinių nuokrypių vidurkiai atliktoms modifikacijoms prie didelių KU apimčių.



5.3.3 pav. Eksperimentų, atliktoms modifikacijoms, vykdymo laikas prie didelių KU apimčių.

5.4. Eksperimentų rezultatų analizė

Atlikus eksperimentus su tyrimo metu pasiūlytomis modifikacijomis, paaiškėjo, kad tam tikroms modifikacijoms pavyko aplenkti bazinį atkaitinimo modeliavimo algoritmą komivojažieriaus uždaviniui vidutinių santykinių nuokrypių arba algoritmo vykdymo laiko prasmėmis. Kitos neatnešė teigiamų rezultatų. 5.4.1 paveikslėlyje matome apibendrintą modifikacijų procentinį pagerėjimą/pablogėjimą vidutinių santykinių nuokrypių vidurkių prasme, lyginant su baziniu algoritmo variantu, kai KU vidutinė apimtis yra 122 miestai (žr. 5.1.3 lentelę).



5.4.1 pav. Apibendrintas modifikacijų procentinis pagerėjimas/pablogėjimas vidutinių santykinių nuokrypių vidurkių prasme, kai KU vidutinė apimtis yra 122 miestai.

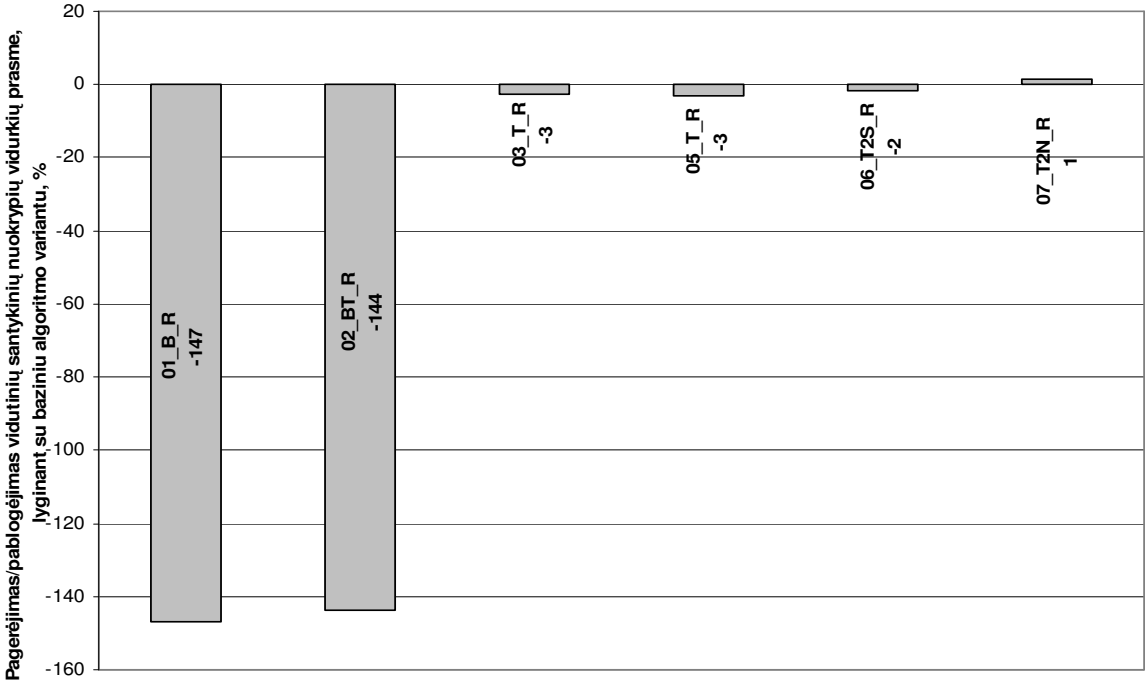
Modifikacijos 01_B_R (žr. 5.3.1 lentelę), 02_BT_R (žr. 5.3.2 lentelę) yra iš tų, kurios neatnešė norimo pagerėjimo, vidutinių santykinių nuokrypių vidurkių kokybė smuktelėjo atitinkamai 90 ir 105 procentais lyginant su baziniu algoritmo variantu. Iš to galime daryti išvadą, kad sąlygų $(\Delta z < 0)$ ir $(\Delta z < \Delta zv * itk)$ kombinacija (žr. 4.2.3, 4.2.4 paveikslus) nėra naudinga optimizuojant algoritmo veikimą.

Modifikacijose 03_T_R, 04_B_R (rezultatai 5.3.3, 5.3.4 lentelėse) buvo atsisakyta sąlygos ($\Delta z < 0$) ir kaip filtras perėjimo prie sekančio sprendinio atrinkime naudojama tik ($\Delta z < \Delta zv * itk$) sąlyga (žr. 4.2.5, 4.2.6 paveikslus). Šioje modifikacijų grupėje buvo pasiektas ir geriausias algoritmo veikimo pagerinimas lyginant su bazine algoritmo versija, kai KU

vidutinė apimtis yra 122 miestai. Modifikacija 03_T_R beveik 10 procentų pagerino bazinio algoritmo rezultatus.

Trečioji modifikacijų grupė, koncentravosi į mutacijas su *itk* koeficientu (rezultatai 5.3.5, 5.3.6, 5.3.7 lentelėse). 05_T_R, 06_T2S_R, 07_T2N_R modifikacijos (žr. 4.2.7, 4.2.8, 4.2.9 paveikslus) – visos atnešė iki 3 procentų pagerėjimą lyginant su bazinio algoritmo rezultatais.

Galiausiai buvo pabandyta padidinti KU apimtis, miestų skaičiaus vidurkį padauginant apytiksliai keturis kartus. 5.4.2 paveikslėlyje matome apibendrintą modifikacijų procentinį pagerėjimą/pablogėjimą vidutinių santykinių nuokrypių vidurkių prasme, lyginant su baziniu algoritmo variantu, kai KU vidutinė apimtis yra 448 miestai (žr. 5.1.4 lentelę)



5.4.2 pav. Apibendrintas modifikacijų procentinis pagerėjimas/pablogėjimas vidutinių santykinių nuokrypių vidurkių prasme, kai KU vidutinė apimtis yra 448 miestai.

Iš rezultatų matome, kad esant didesnėms KU apimtims, bazinį algoritmo variantą aplenkti sugebėjo tiktai modifikacija 07_T2N_R su nuožulniu *itk* koeficiento mažinimu antrojoje algoritmo proceso dalyje.

Į algoritmų modifikacijų rezultatus, žiūrint vykdymo laiko pagerinimo prasme, galime pastebėti, kad praktiškai su visomis modifikacijomis pavyko paspartinti algoritmo veikimą. Kai uždavinio apimtis yra apytiksliai 122 miestai, modifikuotų algoritmų vykdymo vidurkis siekia iki 5s pagreitėjimo (žr. 5.3.1 pav.) nei bazinis jų variantas. Kai uždavinio apimtis yra apytiksliai 448 miestai, modifikuotų algoritmų vykdymo vidurkis siekia iki 90s pagreitėjimo (žr. 5.3.3 pav.) nei bazinis jų variantas, išskyrus modifikacijai 07_T2N_R.

Išvados

1. Statistinės mechanikos principais besiremiančius kūnų atkaitinimo procesus galima sėkmingai imituoti, sprendžiant įvairius kombinatorinio optimizavimo uždavinius, tarp jų gerai žinomą komivojažieriaus uždavinį (KU). Realizuojant atkaitinimo modeliavimo (AM) algoritmus, yra labai svarbu tinkamai nustatyti atkaitinimo proceso valdymo parametrus. Išnagrinėti eksperimentinių tyrimų su KU rezultatai, gauti panaudojant sudarytą AM algoritmo realizaciją ir KU testinius pavyzdžius iš viešos elektroninės testinių pavyzdžių bibliotekos TSPLIB.
2. Atlikus eksperimentus su įvairiais AM algoritmo parametrais nustatyta, kad bene didžiausią įtaką gaunamų rezultatų kokybei turi atkaitinimo proceso trukmės (iteracijų skaičiaus) parametras. Panaši yra ir paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) įtaka. Svarbų vaidmenį taip pat vaidina atkaitinimo temperatūros valdymo parametrai, įgalinantys lanksčiai reguliuoti atkaitinimo proceso eigos sąlygas ir pobūdį. Kai kuriais atvejais koreliacija tarp atskirų temperatūros parametrų reikšmių ir gaunamų sprendinių kokybės yra visiškai akivaizdi; vis tik, įžvelgti aiškią bendrą tendenciją ir dėsningumą visiems atvejams nėra paprasta.
3. Buvo pasiūlytos bazinio AM algoritmo KU modifikacijos, ir ne viena iš jų, nors ir nedaug, tačiau leido pagerinti algoritmo veikimą tiek vidutinių nuokrypių nuo optimumo, tiek algoritmo vykdymo laiko prasmėmis.
4. Identifikuotos modifikacijos neatnešusios AM algoritmo veikimo pagerėjimo.
5. Pastebėta, kad didinant KU apimtis AM algoritmas ir jo modifikacijos keičia savo veikimo tendencijas. Prie didesnių KU miestų skaičiaus, modifikacijos davė kitus rezultatus, paaiškėjo nauja modifikacija „favoritė“, aplenkusi bazinį algoritmo variantą ir kitas jo modifikacijas.
6. Realizuotas AM algoritmas, bei jo modifikacijos leido pasiekti daug žadančius rezultatus šiame darbe panaudotiems KU testiniams pavyzdžiams. Esant tinkamai parinktoms parametrų reikšmėms, optimalūs sprendiniai (maršrutai) surandami per labai trumpą algoritmo vykdymo laiką. O įvestos algoritmo modifikacijos padėjo pasiekti ir dar didesnio rezultatų pagerinimo.

Literatūra

[1] **E.H.L.Aarts, J.K.Lenstra (eds.).** *Local Search in Combinatorial Optimization*, Chichester: Wiley, 1997.

[2] **D.Z.Du, P.M.Pardalos.** *Handbook of Combinatorial Optimization*, Dordrecht: Kluwer, 1998.

[3] **F.Glover, G.Kochenberger (eds.).** *Handbook of Metaheuristics*, Norwell: Kluwer, 2002.

[4] **Z.Michalewicz, D.B.Fogel.** *How to Solve It: Modern Heuristics*, Berlin-Heidelberg: Springer, 2000.

[5] **A.Misevičius, T.Blažauskas, J.Blonskis, J.Smolinskas.** An overview of some heuristic algorithms for combinatorial optimization problems. *Information Technology and Control*, 2004, no.1(30), 21-31.

[6] **P.M.Pardalos, M.G.C.Resende (eds.).** *Handbook of Applied Optimization*, New York: Oxford University Press, 2002.

[7] **D.T.Pham, D.Karaboga.** *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, London: Springer, 2000.

[8] **V.J.Rayward-Smith, I.H.Osman, C.R.Reeves, G.D.Smith (eds.).** *Modern Heuristic Search Methods*, Chichester: Wiley, 1996.

[9] **C.R.Reeves (ed.).** *Modern Heuristic Techniques for Combinatorial Problems*, Halsted: Blackwell, 1993.

[10] **S.Salhi.** Heuristic search methods. In G.Marcoulides (ed.), *Modern Methods for Business Research*, Mahwah: Lawrence Erlbaum, 1998, 147-175.

[11] **D.S.Johnson.** Local optimization and the traveling salesman problem. *In Proceedings of the 17th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science*, vol.443, Berlin: Springer, 1990, 446-461.

[12] **D.S.Johnson, L.A.McGeoch.** The traveling salesman problem: a case study. In E.Aarts, J.K.Lenstra (eds.), *Local Search in Combinatorial Optimization*, Chichester: Wiley, 1997, 215-310.

[13] **G.Reinelt.** The traveling salesman: computational solutions for TSP applications. *Lecture Notes in Computer Science*, 1994, vol.840, Berlin: Springer.

[14] **M.R.Garey, D.S.Johnson.** *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco: Freeman, 1979

[15] **Christian Nilsson**. Heuristics for the traveling salesman problem. *Technical report*, Linkoping University, 2003, 4-5.

[16] **S. Arora**, Polynomial Time Approximation Schemes for Euclidian Traveling Salesman and Other Geometric Problems, *Journal of the ACM*, Vol. 45, No. 5, September 1998, pp. 753-782.

[17] **D.S. Johnson and L.A. McGeoch**, Experimental Analysis of Heuristics for the STSP, *The Traveling Salesman Problem and its Variations*, Gutin and Punnen (eds), Kluwer Academic Publishers, 2002, pp. 369-443.

[18] **A. Misevičius.** Modernieji euristiniai optimizavimo algoritmai. Mokslinio tiriamojo darbo ataskaita, KTU, Kaunas, 2004, 28p.

[19] **A.Misevičius, J.Blonskis, V.Bukšnaitis**. Tabu paieška – efektyvus metodas kombinatorinio optimizavimo uždaviniams spręsti. *Informacijos mokslai (Information Sciences)*, mokslo darbai, 2004, t.29, 124-131.

[20] **B.Freisleben, P.Merz.** A genetic local search algorithm for solving symmetric and a symmetric traveling salesman problems. Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (Nagoya, Japan), 1996, 616-621.

[21] **M. Dorigo, L.M. Gambardella**, Ant Colonies for the Traveling Salesman Problem, Universit Libre de Bruxelles, Belgium, 1996.

[22] **E.H.L.Aarts, J.H.M.Korst, P.J.M. van Laarhoven.** Simulated annealing. E.Aarts, J.K.Lenstra (red.), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 1997, 91–120.

[23] **J.Pepper, B.Golden, E.Wasil.** Solving the traveling salesman problem with annealing-based heuristics: a computa-tional study. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 2002, vol.32, 72-77.

[24] **G.Polya.** *How to solve it*, Princeton: Princeton University Press, 1945.

[25] **S.Voss.** Meta-heuristics: the state of the art. In A.Nareyek (ed.), *Local Search for Planning and Scheduling*, *Lecture Notes in Artificial Intelligence*, vol.2148, Berlin-Heidelberg: Springer, 2001, 1-23.

[26] **H.Streim.** Heuristische Lösungsverfahren – Versuch einer Begriffsklärung. *Zeitschrift für Operations Research*, 1975, vol.19, 143-162.

[27] **E.A.Silver, R.V.V.Vidal, D. de Werra.** A tutorial on heuristic methods. *European Journal of Operational Research*, 1980, vol.5, 153-162.

[28] **H.Müller-Mehrbach.** Heuristics and their design: a survey. *European Journal of Operational Research*, 1981, vol.8, 1-23.

[29] **H.H.Hoos, T.Stützle.** *Stochastic Local Search – Foundations and Applications*, San Francisco: Morgan Kaufmann, 2004.

[30] **Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.** Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 1953, vol.21, 1087-1092.

[31] **Cerný V.** A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Tech. Report, Comenius University, Bratislava, CSSR, 1982.

[32] **Kirkpatrick S., Gelatt C.D., Jr., Vecchi M.P.** Optimization by simulated annealing. *Science*, 1983, vol. 220, 671-680.

[33] **M.Lundy, A.Mees.** Convergence of an annealing algorithm. *Mathematical Programming*, 1986, t.34, 111–124.

[34] **A.Boelte, U.W.Thonemann.** Optimizing simulated annealing schedules with genetic programming. *European Journal of Operational Research*, 1996, t.92, 402–416.

[35] **P.J.M. van Laarhoven, E.H.L.Aarts.** *Simulated Annealing: Theory and Applications*, Reidel, 1987.

[36] **B.Hajek, G.Sasaki.** Simulated annealing: to cool it or not. *Systems and Control Letters*, 1989, t.12, 443–447.

[37] **I.H.Osman.** Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 1993, t.41, 421–451.

[38] **E.H.L.Aarts, J.H.M.Korst.** *Simulated Annealing and Boltzmann Machines*, Wiley, 1989.

[39] **A. Lester Ingber**. Simulated annealing: Practice versus theory, *J. Mathl. Comput. Modelling*, 1993, 29-57.

[40] **T. Emden-Weinert, M. Proksch**. Best practice simulated annealing or the airline crew Scheduling Problem, *Journal of Heuristics*, 1999, 5, 419-436

[41] **Ho-Chan Kim, Chang-Jin Boo, and Yoon-Joon Lee.** Image Reconstruction using Simulated Annealing Algorithm in EIT, *International Journal of Control, Automation, and Systems*, 2005, vol. 3, nr. 2, 211-216

[42] **S. Lin ir B. W. Kernighan.** An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Bell Telephone Labaratories,* 1971.

[43] **Peng Tian ir Zihou Yang**. An Improved Simulated Annealing Algorithm with Genetic Characteristics and the Traveling Salesman Problem, *Journal of Information and Optimization sciences,*1993, vol 14, nr 3, 241-255

[44] **Joshua W. Pepper, Bruce L. Golden, Edward A. Wasil**. Solving the Traveling salesman Problem with Annealing based Heuristics: A Computational Study, *Systems and Humans*, 2002, vol 32, nr 1

[45] **Misevičius, A.** Simulated Annealing Algorithm for the Solution of the Traveling Salesman Problem, *Informacinės Technologijos 2008*, 4-10

[46] **Misevičius A.** A modified simulated annealing algorithm for the quadratic assignment problem. *Informatica*, 2003, vol.14, 497-514.

[47] **Reinelt G.** TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 1991, vol.3-4, 376-385. [Žr. taip pat http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.]

Summary

Simulated annealing (SA) algorithms are often used for solving various combinatorial optimization (CO) problems. In this work, an original simulated annealing algoritm for the well-known combinatorial optimization problem, the traveling salesman problem (TSP) is introduced. The analysis of the results of the experiments with the proposed algorithm by using the TSP instances from the publicly available electronic library TSPLIB is given. The influence of the algorithm control parameters on the results is discussed as well. Based on the results obtained, it is concluded that that the simulated annealing schedule length and the temperature control factors have a crucial effect on the quality of solutions.

Besides the basic simulated annealing algorithm for the traveling salesman problem and its experiments, the modifications to improve the functionality of the algorithm are introduced. Detailed experiments have been carried out to prove the utility of the modifications. The experiments are also based on the TSP instances from the publicly available electronic library TSPLIB, and have been divided into two categories regarding to the size of the problem. It has been discovered that four modifications managed to improve the results of the basic SA algorithm for the smaller travelling salesman problems, and one of them is giving better results for the TSPs with larger scopes.

Priedai

Straipsnis atspausdintas Kauno technologijos universiteto išleistame leidinyje „Information Technologies 2008“, Kaunas, Technologija, 2008, 17-24.

EKSPERIMENTAI SU ATKAITINIMO MODELIAVIMO ALGORITMU SPRENDŽIANT KOMIVOJAŽIERIAUS UŽDAVINĮ

Alfonsas Misevičius, Šarūnas Putrius

Kauno technologijos universitetas, Multimedijos inžinerijos katedra,
Studentų g. 50–400a/416a, LT–51368 Kaunas
Tel. 8 37 300372
El. paštas: alfonsas.misevicius@ktu.lt, sarunas.putrius@gmail.com

Santrauka. Vieni iš plačiausiai taikomų euristinių algoritmų kombinatorinio optimizavimo uždaviniams spręsti yra atkaitinimo modeliavimo (AM) algoritmai. Šiame straipsnyje tiriama AM algoritmo realizacija gerai žinomam kombinatorinio optimizavimo uždaviniui — komivojažieriaus uždaviniui (KU). Pateikiama eksperimentų su realizuotu algoritmu, panaudojant KU testinius pavyzdžius (duomenis) iš viešos elektroninės KU testinių pavyzdžių bibliotekos TSPLIB, rezultatų analizė. Nagrinėjama algoritmo valdymo parametrų įtaka gaunamiems rezultatams. Konstatuojama, jog algoritmo iteracijų skaičiaus (atkaitinimo trukmės) bei atkaitinimo proceso temperatūros valdymo parametrai turi lemiamos svarbos rezultatų kokybei.

1. Įvadas

Komivojažieriaus uždavinys (KU) [6] apibrėžiamas taip: duota atstumų tarp miestų matrica $\boldsymbol{D} = (d_{ij})_{n\times n}$ ir aibė Π, sudaryta iš visų galimų natūrinių skaičių nuo 1 iki n perstatymų; reikia surasti perstatymą $p \in \Pi$ ir tokį, jog būtų minimizuota tikslo funkcija:

$$z(p) = \sum_{i=1}^{n-1} d_{p(i)\,p(i+1)} + d_{p(n)\,p(1)}\,. \tag{1}$$

Perstatymui $p = (p(1), p(2), ..., p(n))$ ($p(i) \in \{1, 2, ..., n\}$, $i = 1, 2, ..., n$) atitinka n miestų seka, t.y. maršrutas. Atskiras perstatymo elementas $p(i)$ žymi i-tąjį maršruto miestą, o perstatymo poras $(p(1), p(2))$, ..., $(p(i), p(i+1))$, ..., $(p(n), p(1))$ galima interpretuoti kaip maršruto atkarpas (kelius tarp miestų); tuomet $z(p)$ yra maršruto p ilgis. Taigi, KU gali būti trumpai formuluojamas kaip tikslas surasti trumpiausią uždarą maršrutą, kuriame kiekvienas miestas aplankomas vieną kartą.

Komivojažieriaus uždavinys priklauso NP-sunkių kombinatorinio optimizavimo uždavinių kategorijai. Jis jau daugelį metų traukia dėmesį tų mokslininkų, kurie kuria ir tiria įvairius optimizavimo metodus, tame tarpe euristinius algoritmus [5,9,13,15,16]. Nors euristiniai algoritmai negarantuoja surandamų sprendinių optimalumo, tačiau įgalina surasti pakankamai aukštos kokybės sprendinius per priimtiną skaičiavimų laiką. Svarbią euristinių algoritmų grupę sudaro atkaitinimo modeliavimo (AM) (angl. *simulated annealing*) algoritmai [1,8].

Šiame straipsnyje pateikiama originali AM algoritmo realizacija, orientuota būtent komivojažieriaus uždaviniui, ir aprašomi eksperimentiniai tyrimai, atlikti su šia realizacija. Pradžioje trumpai charakterizuojamas bendrasis atkaitinimo modeliavimo principas. AM algoritmo realizacija komivojažieriaus uždaviniui detaliau aptariama 3 skyrelyje. Eksperimentinių tyrimų rezultatai, gauti sprendžiant KU testinius pavyzdžius iš testinių pavyzdžių bibliotekos TSPLIB, taip pat svarbiausi faktoriai (parametrai), lemiantys AM algoritmo efektyvumą, išsamiau analizuojami 4 skyrelyje. Straipsnis baigiamas apibendrinamosiomis pastabomis.

2. Atkaitinimo modeliavimo principas

Atkaitinimo modeliavimo (AM) metodo ištakos slypi statistinėje mechanikoje, tiksliau, energetinių procesų, vykstančių sistemose, sudarytose iš didelio skaičiaus dalelių (atomų), imitavime. Šio imitavimo idėja yra ta, jog iš pradžių sistema „pervedama“ į didelės energijos būseną, o po to, palengva mažinant energiją, stengiamasi pasiekti būseną, atitinkančią žemiausią (optimalų) sistemos energijos lygį — tarsi (kietas) fizikinis kūnas būtų įkaitintas iki pakankamai aukštos temperatūros, o paskui, jį atkaitinant (atšaldant), t.y. mažinant temperatūrą, jis būtų savotiškai užgrūdinamas. Atkaitinimo modeliavimo pradininkai buvo Metropolis, Rozenbluth‘as ir kt. [11], kurie pasinaudojo eksponentinio pasiskirstymo funkcija apibrėždami tikimybę, kad sistema, įvykus joje tam tikram pasikeitimui, pereis iš vieno energijos lygio (E_1) į kitą lygį (E_2), kai sistemos temperatūra lygi t. Ta tikimybė apskaičiuojama pagal tokią formulę:

$$\mathrm{P}(\Delta E) = \begin{cases} 1, & \Delta E < 0 \\ e^{-\Delta E/Ct}, & \Delta E \geq 0 \end{cases}; \tag{2}$$

čia $\Delta E = E_2 - E_1$, o C — konstanta. Cerný [3] ir Kirkpatrick'as su bendraautoriais [7] buvo pirmieji, pritaikę atkaitinimo modeliavimo idėją sprendžiant kombinatorinio (ir ne tik) optimizavimo uždavinius.

Bendroji AM algoritmų funkcionavimo schema yra gana paprasta [7]. Sakykime, kad (kombinatorinio) optimizavimo uždavinys aprašytas poros (S, f) pagalba; čia S yra sprendinių aibė, o f: $S \to R^1$ — tikslo funkcija, kuri turi būti minimizuota. Tuomet algoritmo vykdymas pradedamas nuo atsitiktiniu (ar kuriuo nors kitu) būdu sugeneruoto sprendinio s iš aibės S. Esamo sprendinio s aplinkoje♣ parenkamas sprendinys s' ir apskaičiuojamas tikslo funkcijos pokytis $\Delta f = f(s') - f(s)$. Proceso eiga priklauso nuo šio pokyčio: jeigu $\Delta f < 0$, t.y. tikslo funkcijos reikšmė pagerėja, tai esamas sprendinys s pakeičiamas nauju sprendiniu s' ir naudojamas kaip išeities „taškas" tolesniuose bandymuose; priešingu atveju, sprendinio pakeitimas daromas su tikimybe $\mathrm{P}(\Delta f) = e^{-\Delta f / t}$; čia t yra einamoji temperatūra (konstanta C nenaudojama). Temperatūros parametras t palaipsniui mažinamas algoritmo vykdymo metu. Procesas tęsiamas tol, kol patenkinama iš anksto užduota baigimo sąlyga (pvz. temperatūra pasiekia duotą apatinę ribą arba įvykdomas nurodytas iteracijų skaičius). AM algoritmo rezultatas yra geriausias algoritmo vykdymo eigoje surastas sprendinys.

AM algoritmai skiriasi vienas nuo kito įvairiais faktoriais, tarp jų: sprendinių aplinka, atkaitinimo schema (pradinės bei galutinės temperatūros parinkimu, temperatūros keitimo formule), baigimo sąlyga. AM algoritmo realizacija komivojažieriaus uždaviniui trumpai aprašoma sekančiame skyrelyje.

3. Atkaitinimo modeliavimo algoritmo realizacija komivojažieriaus uždaviniui

Atkaitinimo modeliavimo algoritmas pradedamas pradinio atsitiktinio sprendinio — kaip „atspirties taško" tolesniam atkaitinimo procesui — generavimu. Mūsų algoritmo realizacijoje pasinaudota vadinamąja dviejų atkarpų sukeitimo aplinkos funkcija Θ_2, kuri komivojažieriaus uždavinio atveju formaliai aprašoma taip: $\Theta_2(p) = \{p' \mid p' \in \Pi,\ \rho(p, p') = 2\}$, čia $p \in \Pi$, o ρ yra Hamming'o atstumas; šis gali būti matematiškai apibrėžtas tokiu būdu: $\rho(p,p') = n - |\Omega|$, čia $p,p' \in \Pi$, o aibė Ω yra sudaryta iš visų galimų elementų porų (atkarpų) $(p(i),p((i \bmod n) + 1))$ $(i \in \{1, 2, ..., n\})$ ir tokių, kad $\exists j$: $(p(i), p((i \bmod n)+1)) = \begin{cases} (p'(j), p'(j+1)), 1 \le j < n \\ (p'(n), p'(1)),\ j = n \end{cases}$ arba $(p(i), p((i \bmod n)+1)) = \begin{cases} (p'(j), p'(j-1)), 1 < j \le n \\ (p'(1), p'(n)),\ j = 1 \end{cases}$.

Perėjimą iš sprendinio (maršruto) p į aplinkos $\Theta_2(p)$ sprendinį p' galima formalizuoti, panaudojant specialų operatorių $\phi(p, i, j)$ $(i = 1, ..., n-2,\ j = i+2, ..., n-1+\mathrm{sign}(i-1))$, kuris transformuoja p į p' ir taip, kad $p'(i) = p(i)$, $p'(i+1) = p(j)$, $p'(j) = p(i+1)$, $p'((j \bmod n)+1) = p((j \bmod n)+1)$; be to, jeigu $j - i - 2 \ge 1$, tai $p'(i+k+1) += p(j-k)$, $k = 1, ..., j-i-2$. Pasinaudodami lakoniškesne operatoriaus $\phi(p, i, j)$ forma ϕ_{ij}, perėjimą iš p į p' galime užrašyti taip: $p' = p \oplus \phi_{ij}$. Aplinkos sprendiniai nagrinėjami pagal fiksuotą tvarką, kuri gali būti vienareikšmiškai apibrėžta indeksų porų (i,j) seka (1,3), (1,4), …, $(1,n-1)$, (2,4), …, $(2,n)$, …, $(n-2,n)$, … Aplinka $\Theta_2(p)$ išnagrinėjama, panaudojant $O(n^2)$ operacijų. Šiuo atveju pasiekiamas geras kompromisas tarp skaičiavimų (aplinkos nagrinėjimo) laiko ir gaunamų sprendinių kokybės; be to, aplinkos Θ_2 atveju algoritmo programinis realizavimas yra gana paprastas.

Mūsų AM algoritme pradinė ir galutinė atkaitinimo temperatūra nustatoma pagal A.Misevičiaus pasiūlytą ir sėkmingai išbandytą kvadratinio paskirstymo uždaviniui formulę [12]: $\begin{cases} t_0 = (1-\alpha_1)\Delta z^+_{\min} + \alpha_1 \Delta z^+_{\mathrm{vid}} \\ t_g = (1-\alpha_2)\Delta z^+_{\min} + \alpha_2 \Delta z^+_{\mathrm{vid}} \end{cases}$; čia t_0, t_g yra atitinkamai pradinė ir galutinė temperatūra, $\Delta z^+_{\min}$ — minimalus teigiamas tikslo funkcijos reikšmių pokytis, o $\Delta z^+_{\mathrm{vid}}$ — teigiamų tikslo funkcijos pokyčių vidurkis ($\Delta z^+_{\min}$, $\Delta z^+_{\mathrm{vid}}$ gaunami, atlikus tam tikrą skaičių bandomųjų sprendinių perėjimų panaudojant tą pačią aplinką Θ_2); α_1, α_2 yra realieji skaičiai ($\alpha_1 \in (0,1]$, $\alpha_2 \in [0,1)$, $\alpha_1 > \alpha_2$).

Pasirinkta glotnaus temperatūros mažinimo formulė (Lundy-Mees (LM) formulė) [10]: $t_q = t_{q-1} / (1 + \beta t_{q-1})$; $q = 1, 2, ...$; $t_0 = \mathrm{const}$; $\beta \ll t_0$. Koeficiento β (temperatūros kitimo greičio) reikšmę lengva apskaičiuoti analitiškai, jeigu žinoma pradinė ir galutinė temperatūra bei atkaitinimo proceso iteracijų skaičius — Q (žr. toliau). Šiuo atveju $\beta = (t_0 - t_g) / Q t_0 t_g$. Mūsų realizacijoje temperatūra mažinama, atlikus tam tikrą kiekį bandymų — M, t.y. išnagrinėjus ne kurią aplinkos dalį (optimizavimo teorijoje tokia atkaitinimo schema priskiriama vadinamojo homogeninio (subalansuoto) atkaitinimo tipui). Natūralu bandymų skaičių M susieti su aplinkos Θ_2 dydžiu $W = |\Theta_2(\cdot)|$, t.y. $M = \max\{1, \lambda W\}$, čia λ yra sprendinių aplinkos nagrinėjimo bandymų skaičiaus (paieškos gylio) valdymo parametras (koeficientas) ($\lambda > 0$).

♣ Sprendinio s aplinka („kaimyninių" sprendinių aibe) vadinamas poaibis $S' \subseteq S$, sudarytas iš sprendinių, tam tikru mastu „artimų" sprendiniui s. Formaliai aplinka apibrėžiama, panaudojant specialią funkciją Θ: $S \to 2^S$ (2^S žymi aibės S visų galimų poaibių aibę).

Atkaitinimo procesas, jeigu prisilaikyti teorinių nuostatų, turėtų būti tęsiamas tol, kol galutinė temperatūra taptų lygi 0. Kadangi tai pareikalautų didelių laiko sąnaudų, tai praktinėse AM realizacijose naudojami kiti kriterijai. Mūsų algoritme baigimo sąlygos vaidmenį atlieka iš anksto fiksuotas atkaitinimo proceso iteracijų skaičius (atkaitinimo trukmė) — Q.

AM algoritmo efektyvumui padidinti aukščiau aprašyta nuoseklaus temperatūros mažinimo schema papildyta periodiniais temperatūros svyravimais — osciliacija [2]. Taigi, vietoje monotoninio (statinio) atkaitinimo turime dinaminį atkaitinimą (periodiškai kartojamus laipsniškus atšaldymus ir staigius įkaitinimus) [4].

Dinaminio atkaitinimo schema yra tokia. Parenkamas atkaitinimo iteracijų skaičius Q. Atkaitinimas pradedamas, esant pradinei ir galutinei temperatūrai, apskaičiuotai pagal aukščiau pateiktą formulę. Temperatūra mažinama pagal Lundy-Mees formulę. Kai pasiekiama „stagnacijos“ būsena (nėra perėjimų nuo vieno sprendinio prie kito), atkaitinimas sustabdomas. Procesas tęsiamas su naujais atkaitinimo schemos parametrais: nauja pradine ir galutine temperatūra bei perskaičiuotu koeficientu β. Parametrai perskaičiuojami, atsižvelgiant į stabdymo momentu esančią temperatūrą t^*: $t_0 = \alpha_3 t^*$, $t_g = \alpha_4 t^*$, $\beta = (t_0 - t_g)\big/\min\{q^*, Q-q\}\cdot t_0 \cdot t_g$; čia α_3, α_4 yra realieji skaičiai, tenkinantys sąlygas: $\alpha_3 > 1$, $\alpha_4 < 1$; q^* žymi iteracijos, kurios metu pirmą kartą aktyvizuota osciliacija, numerį; q yra duotu momentu esančios iteracijos numeris. Tokiu būdu, tuoj po atkaitinimo sustabdymo temperatūra staigiai padidinama, ir vėl pradedamas atšaldymas — tik su naujais parametrais (atšaldant vėl naudojama LM formulė), ir t.t. Algoritmo vykdymas tęsiamas tol, kol patenkinama baigimo sąlyga, t.y. eilinis atkaitinimo iteracijos numeris q tampa didesnis už maksimalų iteracijų skaičių Q. Dinaminio atkaitinimo pobūdį (osciliacijos aktyvizavimo momentą) galima reguliuoti, turint kokią nors stagnacijos būsenos fiksavimo taisyklę. Viena iš paprastų taisyklių yra fiksuoti stagnaciją tuo atveju, kai nėra perėjimų tarp sprendinių tam tikrą laiko tarpą τ (τ gali būti susietas, pvz. su aplinkos dydžiu W, t.y. $\tau = \max\{1, \omega W\}$, čia ω yra osciliacijos valdymo parametro (koeficiento) vaidmenyje ($\omega > 0$)).

Detalizuotas atkaitinimo modeliavimo algoritmo komivojažieriaus uždaviniui šablonas pateiktas 1 pav.

```
procedure AtkaitinimoModeliavimoAlgoritmas;
// pradiniai duomenys: n – uždavinio dydis (miestų skaičius)
// parametrai: Q – iteracijų skaičius, α1, α2, α3, α4 – temperatūros valdymo parametrai (koeficientai),
//             λ – paieškos gylio valdymo parametras, ω – osciliacijos valdymo parametras
// rezultatai: p* – geriausias surastas sprendinys (maršrutas)
begin
  W := (n(n−1))/2 − n ;  // W – aplinkos Θ2(·) dydis
  sugeneruoti pradinį sprendinį (maršrutą) p atsitiktiniu būdu; p* := p;
  apskaičiuoti Δz+min , Δz+vid vykdant W atsitiktinių sprendinio p perėjimų;
  inicializuoti atkaitinimo parametrus t0, tg, β; t := t0;
  M := MAX(1,λW); τ := MAX(1,ωW);
  i := 1; j := 2; nėra_perėjimų := 0; osciliacija := FALSE;
  for q := 1 to Q do begin // pagrindinis atkaitinimo modeliavimo ciklas
    Δz := ∞;
    for m := 1 to M do begin // atliekama M duoto sprendinio aplinkos nagrinėjimo bandymų
      i := IF(j<n−1+SIGN(i−1), i, IF(i<n−2, i+1, 1)); j := IF(j<n−1+SIGN(i−1), j+1, i+2);
      Δz′ := z( p ⊕ φij ) − z(p);
      if Δz′ < Δz then begin Δz := Δz′; k := i; l := j end
    end; // for m
    if Δz < 0   then pereiti_į_kitą_sprendinį := TRUE
                else pereiti_į_kitą_sprendinį := IF(RANDOM(0,1)<EXP( −Δz/t ),TRUE,FALSE);
    if pereiti_į_kitą_sprendinį = TRUE then begin
      p := p ⊕ φkl ; if z(p) < z(p*) then p* := p; if Δz ≠ 0 then nėra_perėjimų := 0
    end
    else nėra_perėjimų := nėra_perėjimų + 1;
    if (nėra_perėjimų ≥ τ) or (temperatūra t yra žemiausiame taške) then begin
      if osciliacija = FALSE then begin q* := q; t* := t; osciliacija := TRUE end
      perskaičiuoti atkaitinimo parametrus t0, tg, β (atsižvelgiant į q*, t*);
      t := t0
    end
    else t := t/(1 + β t)  // sumažinama einamoji temperatūra
  end // for q
end.
```

1 pav. Atkaitinimo modeliavimo algoritmo komivojažieriaus uždaviniui šablonas.
Pastabos. 1. Funkcijos z reikšmės apskaičiuojamos pagal (1) formulę (žr. 1 sk.).
2. Funkcija IF(x,y_1,y_2) grąžina y_1, jei x = TRUE, arba y_2, jei x = FALSE.
3. Funkcija RANDOM(0,1) generuoja pseudo-atsitiktinį skaičių iš intervalo [0,1)

4. Eksperimentiniai tyrimai

Tiriant sudaryto atkaitinimo modeliavimo algoritmo efektyvumą, atlikti išsamūs eksperimentiniai tyrimai su komivojažieriaus uždavinio testiniais pavyzdžiais (duomenimis) iš viešos elektroninės KU testinių pavyzdžių bibliotekos — TSPLIB [14]. Eksperimentuota su šešiolika skirtingų testinių pavyzdžių. Kaip kriterijus algoritmo efektyvumo laipsniui įvertinti buvo pasirinktas santykinės gaunamų sprendinių kokybės rodiklis, tiksliau, gaunamų tikslo funkcijos reikšmių, t.y. maršrutų ilgių vidutinis santykinis nuokrypis nuo tikėtinos optimalios tikslo funkcijos reikšmės (minimalaus galimo maršruto ilgio). (Aišku, kad kuo šis nuokrypis mažesnis, tuo efektyvumo laipsnis didesnis.) Vidutinis santykinis nuokrypis (jį žymėsime $\overline{\delta}$) apibrėžiamas pagal formulę:

$$\overline{\delta} = 100(\overline{z} - z_{\text{opt}})/z_{\text{opt}}\ [\%]; \tag{3}$$

čia $\overline{z}$ yra gautų tikslo funkcijos reikšmių (maršrutų ilgių) vidurkis, kuris apskaičiuojamas, atlikus 10 algoritmo pakartotinių vykdymų (kiekvieną kartą — su vis kitu pradiniu atsitiktiniu sprendiniu); z_{opt} yra tikėtina optimali tikslo funkcijos reikšmė (šios reikšmės pateikiamos bibliotekoje TSPLIB [14]).

Pagrindinis eksperimentinių tyrimų tikslas — nustatyti geriausias empirines AM algoritmo valdymo parametrų (koeficientų) (žr. 1 lentelę) reikšmes. Tuo pačiu siekta išsiaiškinti, kurie būtent algoritmo valdymo parametrai turi didžiausią įtaką algoritmo efektyvumui (t.y. gaunamų sprendinių kokybei, įvertinamai kriterijumi $\overline{\delta}$). Optimalaus valdymo parametrų reikšmių rinkinio apskaičiavimas yra atskiras sudėtingas uždavinys. Šių eksperimentų metu pasinaudota supaprastinta parametrų tyrimo metodika. Metodikos esmė — suformuoti kiek tai įmanoma priimtinesnį (naudojamo efektyvumo kriterijaus atžvilgiu) parametrų reikšmių poaibį, vykdant nuoseklius savarankiškus (mini-)eksperimentus su atskirais parametrais.

Reikia pabrėžti, kad eksperimentavimas turi būti pradedamas, turint jau iš anksto sudarytą tam tikrą preliminarią pradinę parametrų reikšmių konfigūraciją. Pradinis parametrų reikšmių rinkinys gali žymiai įtakoti eksperimentų metu gaunamas reikšmes, todėl yra labai gerai, jeigu preliminarus apriorinių parametrų reikšmių rinkinys sudaromas optimizavimo srities specialisto ar kito eksperto, remiantis kuriomis nors išankstinėmis teorinėmis prielaidomis. Mūsų AM algoritmo apriorinių valdymo parametrų reikšmių rinkinys pateiktas 2 lentelėje.

Vykdant (mini-)eksperimentus su duotu parametru, visų kitų rinkinio parametrų reikšmės yra fiksuotos (nekintančios). Eksperimentuodami su atskiromis parametrų reikšmėmis, mes lyg eksperimentuojame su skirtingomis algoritmo modifikacijomis. Palyginę duotos modifikacijos rezultatus (sprendinių kokybę) su kitų modifikacijų rezultatais, galime spręsti apie tos modifikacijos (taigi, ir atitinkamos parametro reikšmės) gerumą. Tokiu būdu galima arba atmesti duotą modifikaciją (parametro reikšmę) kaip neefektyvią, arba laikyti ją „atskaitos tašku“ tolimesniems eksperimentams. Tęsiant tokia „bandymų ir klaidų“ metodika besiremiantį eksperimentavimą, išsiaiškinama geriausia modifikacija (t.y. tokia tiriamo parametro reikšmė, kuri įgalina gauti geriausius sprendinius — mažiausią vidutinį nuokrypį $\overline{\delta}$). Panašiai elgiamasi su likusiais parametrais. Atlikus tokius eksperimentus su visais parametrais, gaunamas daugiau mažiau geras (arba algoritmo tyrėją tenkinantis) parametrų reikšmių rinkinys.

1 lentelė. Eksperimentuose tirti AM algoritmo valdymo parametrai

Nr	Parametras		Galimos reikšmės
	Parametro pavadinimas	**Žymėjimas**	
1	Iteracijų skaičius	Q	≥ 1
2	Pradinės temperatūros valdymo parametras (koeficientas)	α_1	(0,1]
3	Galutinės temperatūros valdymo parametras (koeficientas)	α_2	[0,1)
4	Temperatūros didinimo (osciliacijos metu) koeficientas	α_3	>1
5	Temperatūros mažinimo (osciliacijos metu) koeficientas	α_4	[0,1)
6	Paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras (koeficientas)	λ	(0,1]
7	Osciliacijos valdymo parametras (koeficientas)	ω	>0

2 lentelė. Apriorinės parametrų reikšmės

Parametras	Reikšmė
Iteracijų skaičius	10^6
Pradinės temperatūros valdymo parametras	0,3
Galutinės temperatūros valdymo parametras	0,003
Temperatūros didinimo koeficientas	5
Temperatūros mažinimo koeficientas	0,4
Paieškos gylio valdymo parametras	0,03
Osciliacijos valdymo parametras	0,27

3 lentelė. Optimizuotos parametrų reikšmės

Parametras	Reikšmė
Iteracijų skaičius	10^7
Pradinės temperatūros valdymo parametras	0,7
Galutinės temperatūros valdymo parametras	0,006
Temperatūros didinimo koeficientas	10
Temperatūros mažinimo koeficientas	0,8
Paieškos gylio valdymo parametras	0,01
Osciliacijos valdymo parametras	0,15

Eksperimentavimo pagal aukščiau aprašytą metodiką eigoje gautos optimizuotos parametrų rinkinio reikšmės parodytos 3 lentelėje. Konkretūs rezultatai (santykiniai sprendinių nuokrypiai), gauti su šiomis parametrų reikšmėmis, pateikiami 4 lentelėje (palyginimui yra parodyti ir rezultatai, gauti su neoptimizuotomis parametrų reikšmėmis). (Tarpiniai ir ne taip svarbūs eksperimentavimo eigos rezultatai nėra pateikti dėl ribotos šio straipsnio apimties.)

4 lentelė. Eksperimentų rezultatai, gauti su pradinėmis ir optimizuotomis parametrų reikšmėmis

Nr.	Testinio pavyzdžio pavadinimas[‡]	Optimali tikslo funkcijos reikšmė (minimalus maršruto ilgis) (z_{opt})	Vidutinis santykinis nuokrypis nuo optimumo ($\bar{\delta}$)			
			Pradinis (neoptimizuotas) parametrų rinkinys ($Q = 10^6$)	Pradinis (neoptimizuotas) parametrų rinkinys (be Q) ($Q = 10^7$)	Optimizuotas parametrų rinkinys (be Q) ($Q = 10^6$)	Optimizuotas parametrų rinkinys ($Q = 10^7$)
1	bier127	118282	0,826	0,245	0,800	0,080
2	ch130	6110	0,586	0,239	0,496	0,240
3	eil101	629	0,223	0,016	0,111	0,000
4	gr96	55209	0,725	0,137	0,190	0,130
5	gr120	6942	0,808	0,311	0,347	0,258
6	kroa100	21282	0,346	0,067	0,061	0,044
7	krob100	22141	0,786	0,145	0,268	0,002
8	kroc100	20749	0,438	0,075	0,224	0,002
9	lin105	14379	0,403	0,064	0,125	0,000
10	pr107	44303	0,018	0,000	0,107	0,126
11	rat99	1211	0,140	0,000	0,041	0,000
12	rd100	7910	0,474	0,044	0,248	0,000
13	st70	675	0,430	0,000	0,252	0,000
14	ts225	126643	1,260	0,059	0,030	0,006
15	tsp225	3916	1,034	0,889	1,021	0,988
16	u159	42080	0,640	0,603	0,662	0,679
		Vidurkis:	**0,571**	**0,181**	**0,311**	**0,160**

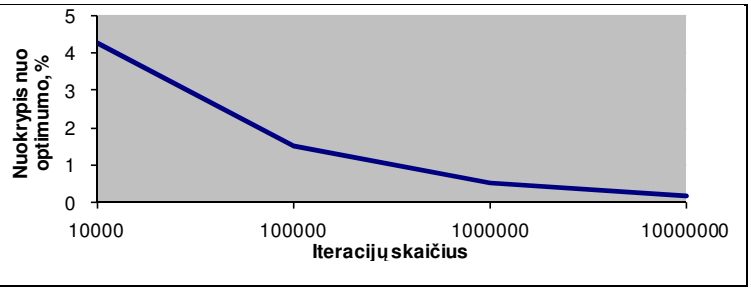
[‡ numeris, esantis testinio pavyzdžio pavadinime, nurodo miestų skaičių]

Kaip ir buvo galima iš anksto tikėtis, didžiausią įtaką gaunamų sprendinių kokybei turi iteracijų skaičiaus parametras Q, nusakantis atkaitinimo proceso trukmę. Natūralu, jog kuo daugiau atliekama iteracijų, tuo tikslesni gaunami rezultatai. Tačiau būtina turėti omenyje, kad didinant iteracijų skaičių, atitinkamai pailgėja algoritmo vykdymo laikas (ypač esant didesnėms KU duomenų apimtims). Čia svarbu, kokie yra konkretūs prioritetai ir kiek išaugęs vykdymo laikas „atperkamas“ pagerėjusia rezultatų kokybe. Aplamai, net ir pakankamai ilgas, bet tenkinantis tyrėją (vartotoją) vykdymo laikas gali būti pateisinamas, jeigu tai leidžia apčiuopiamai pagerinti sprendinių kokybės įverčius. (Geriau yra gauti aukštos kokybės rezultatus per ilgesnį, bet priimtiną laiką, negu tenkintis blogais rezultatais, gautais per labai trumpą laiką.) Gaunamų sprendinių kokybės priklausomybė nuo iteracijų skaičiaus iliustruojama 5 lentelėje.

Panašią įtaką kaip ir iteracijų skaičius turi paieškos gylio (sprendinių aplinkos nagrinėjimo bandymų skaičiaus) valdymo parametras λ. Galima pastebėti aiškią koreliaciją tarp šio parametro reikšmių ir vidutinio santykinio sprendinių kokybės nuokrypio reikšmių (žr. 6 lentelę). Iš tiesų, didinant nagrinėjamų aplinkos sprendinių skaičių, kitaip tariant, atidžiau išnagrinėjant sprendinių aplinką (didinant paieškos tikslumą) kiekvienos iteracijos metu, ženkliai didėja ir tikimybė aptikti geresnės kokybės sprendinius. Vėlgi reikia akcentuoti, jog paieškos gylio didinimas implikuoja išaugusią skaičiavimų apimtį, taigi, ir algoritmo vykdymo trukmę. Rezultatų pagerėjimas yra padidėjusių skaičiavimų resursų kaina.

5 lentelė. Sprendinių kokybės (vidutinio santykinio nuokrypio) priklausomybė nuo iteracijų skaičiaus

Iteracijų skaičius	$\bar{\delta}$
10^4	4,272
10^5	1,500
10^6	0,541
10^7	0,181



Labai svarbūs AM algoritmo parametrai yra atkaitinimo temperatūros dinamikos valdymo parametrai, ypač tai pasakytina apie pradinės ir galutinės temperatūros valdymo parametrus (koeficientus α_1, α_2), nulemiančius atkaitinimo proceso charakterį, tuo pačiu, gaunamus galutinius rezultatus. Eksperimentų rezultatai iš esmės patvirtina hipotetines prielaidas apie pradinės ir galutinės temperatūros reikšmių suderinimo įtaką gaunamų sprendinių kokybei. Kaip ir spėta, jei pradinė temperatūra per daug aukšta (o/arba galutinė temperatūra per žema), tai konvergavimo į aukštos kokybės sprendinius procesas užsitęsia pernelyg ilgai. Kita vertus, per daug žema pradinė (arba per aukšta galutinė) temperatūra neužtikrina pakankamai geros kokybės sprendinių

suradimo. Abiem atvejais algoritmo efektyvumo laipsnis yra mažesnis (žr. 7 lentelę). Vis tik, kaip matyti iš 7 lentelėje pateiktų rezultatų, parinkti optimalias pradinės ir galutinės temperatūros reikšmes nėra jau taip trivialu. Galima iš tų rezultatų įžvelgti, kad algoritmas gana „jautriai“ reaguoja į temperatūros valdymo koeficientų reikšmių pakitimus — tai patvirtina šių faktorių reikšmę algoritmo efektyvumui. Net ir nežymūs temperatūros valdymo koeficientų α_1, α_2 pokyčiai gali iššaukti ryškius sprendinių kokybės įvertinimų svyravimus (žr. 7 lentelės 7–11 stulpelius); kai kuriais atvejais tie svyravimai yra gana nenuspėjami (žr. 7 lentelės 2–4 stulpelius). (Kas liečia temperatūros valdymo osciliacijos metu parametrus α_3, α_4, kaip ir pačios osciliacijos valdymo parametrą ω, tai šių parametrų reikšmių pokyčiai turi tik minimalų poveikį galutiniams rezultatams.)

6 lentelė. Paieškos gylio valdymo koeficiento įtaka rezultatų kokybei

Nr.	Testinio pavyzdžio pavadinimas	Vidutinis santykinis nuokrypis ($\bar{\delta}$)		
		$\lambda = 0{,}0005$	$\lambda = 0{,}003$	$\lambda = 0{,}01$
1	bier127	1,150	0,826	0,656
2	ch130	1,030	0,586	0,524
3	eil101	1,962	0,223	0,060
4	gr96	1,301	0,725	0,341
5	gr120	1,071	0,808	0,556
6	kroa100	0,358	0,346	0,233
7	krob100	0,782	0,786	0,256
8	kroc100	0,567	0,438	0,112
9	lin105	0,245	0,403	0,173
10	pr107	0,111	0,018	0,000
11	rat99	0,930	0,140	0,080
12	rd100	0,711	0,474	0,280
13	st70	0,759	0,430	0,311
14	ts225	2,471	1,260	0,132
15	tsp225	1,478	1,034	1,178
16	u159	1,467	0,640	0,678
	Vidurkis:	**1,025**	**0,571**	**0,350**

7 lentelė. AM algoritmo rezultatų priklausomybė nuo temperatūros parametrų

Testinio pavyzdžio pavadinimas	Vidutinis santykinis nuokrypis ($\bar{\delta}$)									
	$\alpha_1 = 0{,}1$	$\alpha_1 = 0{,}3$	$\alpha_1 = 0{,}5$	$\alpha_1 = 0{,}7$	$\alpha_1 = 0{,}9$	$\alpha_1 = 0{,}7$				
	$\alpha_2 = 0{,}006$					$\alpha_2 = 0{,}1$	$\alpha_2 = 0{,}01$	$\alpha_2 = 0{,}003$	$\alpha_2 = 0{,}006$	$\alpha_2 = 0{,}0001$
bier127	0,545	0,826	1,140	0,742	0,917	3,771	0,817	0,826	0,321	1,839
ch130	0,949	0,586	0,901	0,624	0,638	5,448	0,419	0,586	0,656	0,712
eil101	0,095	0,223	0,273	0,191	0,413	7,075	0,827	0,223	0,478	0,175
gr96	0,568	0,725	0,606	0,574	0,588	4,284	0,449	0,725	0,333	1,691
gr120	0,749	0,808	0,691	0,703	0,763	5,774	0,575	0,808	0,620	1,165
kroa100	0,234	0,346	0,356	0,302	0,290	3,681	0,165	0,346	0,281	0,731
krob100	0,326	0,786	0,360	0,526	0,509	4,651	0,454	0,786	0,351	0,657
kroc100	0,464	0,438	0,510	0,288	0,322	4,015	0,231	0,438	0,222	0,901
lin105	0,631	0,403	0,367	0,170	0,309	3,586	0,233	0,403	0,110	0,156
pr107	0,008	0,018	0,031	0,038	0,000	6,439	0,270	0,018	0,102	0,010
rat99	0,173	0,140	0,256	0,264	0,256	7,027	0,223	0,140	0,145	0,149
rd100	0,375	0,474	0,567	0,574	0,784	5,202	0,159	0,474	0,356	0,799
st70	0,311	0,430	0,133	0,415	0,430	3,230	0,341	0,430	0,280	0,326
ts225	1,140	1,260	0,254	0,149	0,927	6,194	0,197	1,260	0,142	4,062
tsp225	0,769	1,034	1,253	1,315	1,264	6,992	1,652	1,034	1,094	1,864
u159	0,935	0,640	0,790	0,823	0,687	6,043	0,681	0,640	0,629	2,164
Vidurkis:	**0,517**	**0,571**	**0,531**	**0,481**	**0,569**	**5,213**	**0,481**	**0,571**	**0,383**	**1,088**

5. Apibendrinamosios pastabos

Statistinės mechanikos principais besiremiančius kūnų atkaitinimo procesus galima sėkmingai imituoti, sprendžiant įvairius kombinatorinio optimizavimo uždavinius, tarp jų gerai žinomą komivojažieriaus uždavinį (KU). Realizuojant atkaitinimo modeliavimo (AM) algoritmus, yra labai svarbu tinkamai nustatyti atkaitinimo proceso valdymo parametrus. Šiame straipsnyje kaip tik ir išnagrinėti eksperimentinių tyrimų su KU rezultatai, gauti panaudojant autorių sudarytą AM algoritmo realizaciją ir KU testinius pavyzdžius iš viešos elektroninės testinių pavyzdžių bibliotekos TSPLIB.

Pagrindinis šio darbo tikslas buvo identifikuoti svarbiausius AM algoritmo efektyvumą nulemiančius faktorius, nustatyti geriausias (empirines) algoritmo valdymo parametrų (koeficientų) reikšmes ir išsiaiškinti tirtų parametrų įtaką gaunamų rezultatų kokybei.

Pagal atitinkamą metodiką eksperimentuota su įvairiais AM algoritmo parametrais: atkaitinimo trukme (iteracijų skaičiumi), atkaitinimo temperatūros valdymo parametrais (koeficientais), paieškos gyliu (sprendinių aplinkos nagrinėjimo bandymų skaičiumi), osciliacijos valdymo koeficientu. Nustatyta, kad bene didžiausią įtaką gaunamų rezultatų kokybei turi atkaitinimo proceso trukmės (iteracijų skaičiaus) parametras. Panaši yra ir paieškos gylio (aplinkos nagrinėjimo bandymų skaičiaus) įtaka. Svarbų vaidmenį taip pat vaidina atkaitinimo temperatūros valdymo parametrai, įgalinantys lanksčiai reguliuoti atkaitinimo proceso eigos sąlygas ir pobūdį. Kai kuriais atvejais koreliacija tarp atskirų temperatūros parametrų reikšmių ir gaunamų sprendinių kokybės yra visiškai akivaizdi; vis tik, įžvelgti aiškią bendrą tendenciją ir dėsningumą visiems atvejams nėra paprasta. Todėl, be kita ko, būtų tikslingi nuodugnesni eksperimentai. Papildomi eksperimentai būtų naudingi ir išsamiau tiriant osciliacijos mechanizmą bei jo valdymą. Bet kuriuo atveju AM algoritmo sėkmė priklauso nuo aposteriorinių (eksperimentų metu nustatytų) atkaitinimo proceso eigą valdančių parametrų reikšmių.

Aplamai, mūsų realizuotas AM algoritmas leido pasiekti daug žadančius rezultatus šiame darbe panaudotiems KU testiniams pavyzdžiams. Esant tinkamai parinktoms parametrų reikšmėms, optimalūs sprendiniai (maršrutai) surandami per labai trumpą algoritmo vykdymo laiką. Nežiūrint į tai, įdomu būtų pratęsti tyrimus, siekiant dar padidinti algoritmo efektyvumo laipsnį. Be to, būtų galima šį algoritmą išbandyti su didesnės apimties KU testiniais pavyzdžiais, taip pat su kitais optimizavimo uždaviniais.

Literatūros sąrašas

[1] **Aarts E.H.L., Korst J.H.M., van Laarhoven P.J.M.** Simulated annealing. In E.Aarts, J.K.Lenstra (eds.), *Local Search in Combinatorial Optimization*, *Wiley*, 1997, 91-120.

[2] **Bölte A., Thonemann U.W.** Optimizing simulated annealing schedules with genetic programming. *European Journal of Operational Research*, 1996, vol.92, 402-416.

[3] **Cerný V.** A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Tech. Report, Comenius University, Bratislava, CSSR, 1982.

[4] **Hajek B., Sasaki G.** Simulated annealing: to cool it or not. *Systems and Control Letters*, 1989, vol.12, 443-447.

[5] **Johnson D.S.** Local optimization and the traveling salesman problem. *Proceedings of the 17th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science*, vol.443, *Springer*, 1990, 446-461.

[6] **Johnson D.S., McGeoch L.A.** The traveling salesman problem: a case study. In E.Aarts, J.K.Lenstra (eds.), *Local Search in Combinatorial Optimization*, *Wiley*, 1997, 215-310.

[7] **Kirkpatrick S., Gelatt C.D., Jr., Vecchi M.P.** Optimization by simulated annealing. *Science*, 1983, vol.220, 671-680.

[8] **van Laarhoven P.J.M., Aarts E.H.L.** Simulated Annealing: Theory and Applications, *Reidel*, 1987.

[9] **Lin S., Kernighan B.W.** An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 1973, vol.21, 498-516.

[10] **Lundy M., Mees A.** Convergence of an annealing algorithm. *Mathematical Programming*, 1986, vol.34, 111-124.

[11] **Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.** Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 1953, vol.21, 1087-1092.

[12] **Misevičius A.** A modified simulated annealing algorithm for the quadratic assignment problem. *Informatica*, 2003, vol.14, 497-514.

[13] **Nilsson C.** Heuristics for the traveling salesman problem. Tech. Report, Linköping University, Sweden, 2003. [Žr. taip pat `http://www.ida.liu.se/~TDDB19/reports_2003/htsp.pdf`.]

[14] **Reinelt G.** TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 1991, vol.3-4, 376-385. [Žr. taip pat http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.]

[15] **Reinelt G.** The traveling salesman: computational solutions for TSP applications. *Lecture Notes in Computer Science*, vol.840, *Springer*, 1994.

[16] **Rosenkrantz D.E., Stearns R.E., Lewis P.M.** An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 1977, vol.6, 563-581.

EXPERIMENTS WITH SIMULATED ANNEALING ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM

Summary. Simulated annealing (SA) algorithms are often used for solving various combinatorial optimization (CO) problems. In this paper, an original simulated annealing algoritm for the well-known combinatorial optimization problem, the traveling salesman problem (TSP) is introduced. The analysis of the results of the experiments with the proposed algorithm by using the TSP instances from the publicly available electronic library TSPLIB is given. The influence of the algorithm control parameters on the results is discussed as well. Based on the results obtained, it is concluded that that the simulated annealing schedule length and the temperature control factors have a crucial effect on the quality of solutions.