KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PRAKTINĖS INFORMATIKOS KATEDRA

Audrius Gedgaudas

Tabu paieškos algoritmas ir programa kvadratinio paskirstymo uždaviniui

Magistro darbas

Darbo vadovas
doc. dr. A. Misevičius

Kaunas, 2004

Turinys

1. Pratarmė

Vienas iš plačiausiai taikomų euristinių algoritmų kombinatorinio optimizavimo uždaviniams spręsti yra tabu paieškos algoritmas. Šiame darbe pasiūlytas patobulintas tabu paieškos algoritmas gerai žinomam kombinatorinio optimizavimo uždaviniui, būtent, kvadratinio paskirstymo (KP) uždaviniui spręsti. Naujosios šio algoritmo savybės yra: dvigubas iteracijų skaičius, t.y. atliekamos dviejų lygių iteracijos, todėl gaunami tikslesni rezultatai, taip pat algoritmas patobulintas įdiegiant atskaitos taškus, kadangi vykdant sudėtingų uždavinių sprendimą algoritmo darbo laikas išauga. Šio patobulinimo dėka programos darbą galima stabdyti bet kuriuo laiko momentu, o vėliau jį toliau pratęsti. Naujasis algoritmas išbandytas panaudojant KP uždavinio testinius pavyzdžius iš testinių pavyzdžių bibliotekos – QAPLIB. Gauti eksperimentų rezultatai liudija, jog siūlomas algoritmas yra pranašesnis už ankstesnius tabu paieškos algoritmus didelei daliai KP uždavinio testinių pavyzdžių.

Darbas pradedamas įvadu. Trečiame sk. aptariama viena iš KP uždavinio taikymo sričių – elektroninės aparatūros projektavimas. Euristiniai optimizavimo metodai, orientuoti šiam uždaviniui, apžvelgiami 4 sk. Naujasis patobulintas algoritmas KP uždaviniui aprašomas 5 sk., 6 sk. pateikiami eksperimentinių tyrimų rezultatai bei jų analizė. Darbas baigiamas išvadomis. Papildomai pateikiami kai kurie priedai.

2. Įvadas

Kvadratinio paskirstymo (KP) uždavinys (angl. *quadratic assignment problem*) formuluojamas taip [35]: duotos matricos $\boldsymbol{A}=(a_{ij})_{n\times n}$ ir $\boldsymbol{B}=(b_{kl})_{n\times n}$ bei aibė Π, kurią sudaro visi galimi natūrinių skaičių 1, 2, …, n perstatymai. Reikia tarp visų perstatymų iš Π surasti tokį perstatymą $\pi=(\pi(1), \pi(2), ..., \pi(n))$, kuriam esant būtų minimizuota funkcija

$$z(\pi)=\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \ . \tag{1}$$

Jeigu matricos $\boldsymbol{A}$ ir $\boldsymbol{B}$ yra simetrinės, tai gaunamas simetrinio kvadratinio paskirstymo uždavinio atvejis.

Kvadratinio paskirstymo uždavinys pasižymi įvairiais praktiniais taikymais, tarp kurių paminėtini šie: pastatų išdėstymas, planavimas [17, 18]; įrengimų išdėstymas [35, 45]; vaizdų (pilkų atspalvių) formavimas [55]; elektroninės aparatūros projektavimas (panelių, klaviatūrų projektavimas; elektroninių komponentų išdėstymas spausdinto montažo plokštėse ar didelės integracijos mikroschemose) [28, 32, 36, 52]. Elektroninės aparatūros projektavimas komponentų išdėstymo kontekste plačiau aptariamas 2 sk. Kitos taikymo sritys aprašytos [6, 7, 8, 11, 19, 39] literatūroje.

Kvadratinio paskirstymo uždavinys priklauso NP-sunkių kombinatorinio optimizavimo uždavinių klasei [48]. Tokie uždaviniai tiksliai išsprendžiami tik esant labai nedidelėms jų apimtims ($n\leq 30$) [26]. Todėl vidutinės ir didelės apimties KP uždaviniams spręsti naudojami euristiniai algoritmai. Reikšmingą tokių algoritmų grupę sudaro atkaitinimo modeliavimo (angl. *simulated annealing*) [5, 10, 15, 57], tabu paieškos (angl. *tabu search*) [33, 49, 51, 54] ir genetiniai (angl. *genetic algorithms*) [3, 20, 41, 56] algoritmai. Jie apžvelgiami 3 sk.

Kadangi kvadratinio paskirstymo uždavinys yra vienas iš kombinatorinio optimizavimo uždavinių, trumpai apibrėšime pagrindines sąvokas, susijusias su tokiais uždaviniais.

Tarkime, kad S yra kombinatorinio (diskretinio) optimizavimo uždavinio sprendinių aibė, o f: $S\rightarrow R^1$ – tikslo funkcija. Be to, yra duota aplinkinių sprendinių (aplinkos) funkcija N: $S\rightarrow 2^S$, bet kuriam sprendiniui $s\in S$ apibrėžianti aibę $N(s)\subseteq S$ – sprendinio s aplinkinių sprendinių aibę. Bet kuris sprendinys $s'\in N(s)$ gali būti betarpiškai pasiektas iš sprendinio s, atlikus elementarią sprendinio s pertvarkymo operaciją, vadinamą perturbacija (perėjimu). Paprastai prieš darant perturbaciją apskaičiuojama tikslo funkcijos f reikšmė; sakoma, jog

atliekamas bandymas. Tikslas yra surasti tokį sprendinį $s^{\bullet} \in S$, kuris minimizuotų funkciją f, t. y. $s^{\bullet} = \arg\min_{s \in S} f(s)$.

Labai dažnai kvadratinio paskirstymo uždaviniui, kaip ir kitiems kombinatorinio optimizavimo uždaviniams, naudojama vadinamoji porinių sukeitimų funkcija, kitaip tariant, aplinka (žymima N_2). KP uždavinio atveju ši funkcija apibrėžiama taip: $N_2(\pi) = \{\pi' \mid \pi' \in \Pi,\ d(\pi, \pi') = 2\}$, čia π – bet kuris sprendinys (perstatymas) iš Π, $d(\pi, \pi')$ – atstumas tarp sprendinių π ir π', kuris gali būti išreikštas tokia formule $d(\pi, \pi') = \sum_{i=1}^{n} \operatorname{sgn} |\pi(i) - \pi'(i)|$ (n yra uždavinio apimtis). Taigi funkcija N_2 apibrėžia aibę, sudarytą iš tokių sprendinių-kaimynų, kurie gaunami, duotame sprendinyje atlikus 2-jų narių sukeitimą vietomis. Šiuo atveju priimta sakyti, kad atliekamas perėjimas iš esamo sprendinio į sprendinį-kaimyną. Formaliai perėjimas aprašomas panaudojant specialų operatorių – perturbaciją, kurios metu sukeičiama vietomis i-tasis ir j-tasis sprendinio nariai (žymima p_{ij} ($1 \le i,j \le n$, $i \ne j$)). (Tuomet perėjimas iš spredinio π į spredinį π' galėtų būti užrašytas, pvz., taip: $\pi' = \pi \oplus p_{ij}$.)

Pažymėsime, kad funkcijos N_2 generuojamos aibės dydis lygus $n(n-1)/2$. Jeigu atlikta $K = n(n-1)/2$ bandymų, tai sakoma, jog įvykdyta viena porinių sukeitimų iteracija.

Dažniausiai kaimyniniai sprendiniai nagrinėjami ne atsitiktinai („aklai“), o pagal tam tikrą griežtą, fiksuotą tvarką. Perturbacijų p_{ij} indeksai i, j, apibrėžiantys sprendinių nagrinėjimo eiliškumą, gali būti imami iš masyvo, kuriame indeksai kuriuo nors būdu „sumaišomi“ (atskiru atveju išdėstomi tvarkingai, pvz., didėjimo tvarka); svarbu tik, kad kiekvienas indeksas pasitaikytų po vieną kartą. Tuomet bendru atveju sprendinių nagrinėjimo tvarką galima aprašyti sudarant tokią seką $\left\{p_{\mathrm{ISM}(i^{(k)})\mathrm{ISM}(j^{(k)})}\right\}_{k=1,2,\ldots}$, čia ISM – indeksų „sumaišymo“ masyvas; indeksai $i^{(k)}$, $j^{(k)}$ apskaičiuojami pagal formulę $\begin{cases} i^{(k)} = \mathrm{if}(j^{(k-1)} < n, i^{(k-1)}, \mathrm{if}(i^{(k-1)} < n-1, i^{(k-1)}+1, 1)) \\ j^{(k)} = \mathrm{if}(j^{(k-1)} < n, j^{(k-1)}+1, i^{(k)}+1) \end{cases}$, kur k – bandymo eilės numeris, $i^{(k)}$, $j^{(k)}$ – nauji indeksai, $i^{(k-1)}$, $j^{(k-1)}$ – buvę prieš tai indeksai ($i^{(0)}=1$, $j^{(0)}=1$). (Funkcija if aprašoma taip: $\mathrm{if}(\mathrm{salyga}, x_1, x_2) = \begin{cases} x_1, \mathrm{salyga} = \mathrm{TRUE} \\ x_2, \mathrm{salyga} = \mathrm{FALSE} \end{cases}$.)

Pastebėsime, jog KP uždavinio atveju tikslo funkcijos pokytis $\Delta z(\pi, i, j)$ ($1 \le i, j \le n$, $i \ne j$), gautas, sprendinyje π sukeitus vietomis i-tąjį ir j-tąjį narius, t. y., atlikus perturbaciją p_{ij}, apskaičiuojamas, atlikus $O(n)$ operacijų:

$$\Delta z(\pi, i, j) = (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}) + \sum_{k=1, k\neq i, j}^{n} \left[(a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) + (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)}) \right] \quad (2)$$

jeigu a_{ii} (arba b_{ii})=const, i=1, 2, ..., n. (O tai reiškia, kad norint išnagrinėti visus aplinkos N_2 sprendinius, reikia atlikti $O(n^3)$ operacijų.)

Jeigu matricos ***A*** ir/arba ***B*** yra simetriškos, tai pateiktoji formulė žymiai supaprastėja. Pavyzdžiui, jeigu tik vienintelė matrica ***B*** yra simetriška, o matrica ***A*** – ne, tai galima transformuoti nesimetrišką matricą ***A*** į simetrišką matricą ***A′***, sudedant atitinkamus matricos ***A*** viršutinio ir apatinio „trikampio" elementus. Tuomet gaunama tokia kompaktiška formulė tikslo funkcijos pokyčiui apskaičiuoti:

$$\Delta z(\pi, i, j) = \sum_{k=1, k\neq i, j}^{n} (a'_{ik} - a'_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}), \quad (3)$$

kur $a'_{ik}=a_{ik}+ a_{ki}$, $\forall i,k \in \{1, 2, ..., n\}$, $i\neq k$.

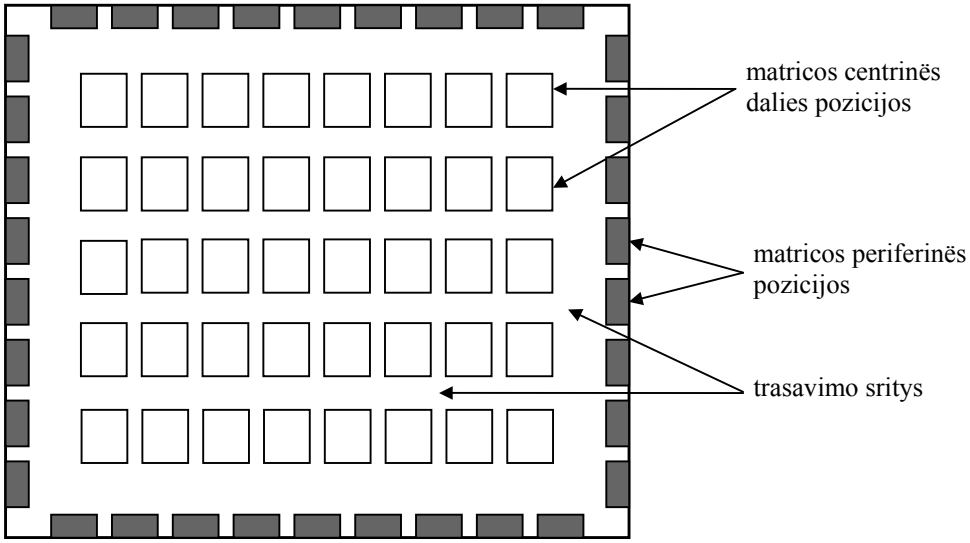
3. Kvadratinio paskirstymo uždavinio taikymai: elektroninės aparatūros projektavimas

Vienas iš aktualių kvadratinio paskirstymo uždavinio taikymų yra elektroninės apratūros projektavimas. Elektroninės aparatūros, tame tarpe integralinių mikroschemų, automatizuotojo projektavimo procesas apima daug projektavimo etapų [48, 49], iš kurių pagrindiniai yra:

- ♦ loginis-funkcinis projektavimas;
- ♦ schemotechninis projektavimas;
- ♦ topologijos projektavimas.

Vienas iš sudėtingiausių yra topologijos projektavimo etapas, apimantis funkcinių mazgų - elementų - komponavimo, išdėstymo ir sujungimų trasavimo bei kontrolės stadijas. Kuriant automatizuotojo projektavimo sistemas, topologijos projektavimo etapas išskaidomas į dvi santykinai savarankiškas stadijas: elementų išdėstymą ir sujungimų trasavimą. Tai daroma, siekiant maksimaliai palengvinti labai sudėtingo kompleksinio uždavinio sprendimą. Sujungimų trasavimo, taigi ir viso topologijos projektavimo etapo, atlikimo kokybė tiesiogiai priklauso nuo išdėstymo kokybės. Todėl elementų išdėstymo uždavinys yra ypatingai aktualus topologijos projektavime, galima sakyti, nulemiantis galutinius rezultatus ir projektavimo sėkmę.

Elektroninėje aparatūroje plačiai naudojamas didelės integracijos mikroschemas sudaro tūkstančiai topologinių vienetų (tranzistorių, rezistorių ir pan.), kurie apjungiami į stambesnius mazgus - funkcinius elementus (logines schemas, atminties ląsteles ir pan.). Funkciniai elementai (arba tiesiog elementai) - tai mažiausi nedalomi vienetai, kuriais operuojama išdėstymo uždavinyje.



1 pav. Bazinio matricinio kristalo struktūra

Didelės integracijos mikroschemos dažnai realizuojamos vadinamuose baziniuose matriciniuose kristaluose (BMK). Elementai dėstomi jiems skirtose matricinio kristalo pozicijose. Reguliarios struktūros kristaluose šių pozicijų padėtys yra griežtai apibrėžtos: pozicijos išdėstomos horizontaliose eilutėse ir vertikaliuose stulpeliuose. Pozicijų gabaritai (plotis, aukštis) ir forma yra vienodi. Matricinio kristalo konstrukcijoje paprastai išskiriamos centrinės dalies pozicijos bei periferinės pozicijos (žr. 1 pav.).

Aprašant komponentų išdėstymo uždavinį kaip kvadratinio paskirstymo uždavinį, t. y., išdėstymo uždavinį modeliuojant KP uždavinio pagalba matrica $\boldsymbol{A}=(a_{ij})$ interpretuojama kaip elektrinių ryšių tarp komponentų matrica, kur a_{ij} yra ryšių („grandinių"), jungiančių i-tąjį komponentą su j-tuoju komponentu, skaičius. Matrica $\boldsymbol{B}=(b_{kl})$ yra atstumų matrica, kur b_{kl} – atstumas (stačiakampėje arba Euklido metrikoje) tarp k-tosios ir l-tosios dėstymo pozicijos. Perstatymas $\pi=(\pi(1), \pi(2), ..., \pi(n))$ apibrėžia komponentų išdėstymą į pozicijas, t. y., išdėstymo konfigūraciją (čia $\pi(i)$ – pozicijos, į kurią paskirtas i-tasis komponentas, numeris). Tuomet $z(\pi)$ yra bendras ryšių (sujungimų) tarp komponentų ilgis, kurį reikia minimizuoti tam, kad sudaryti kuo palankesnes sąlygas tolimesniems projektavimo etapams.

4. Moderniųjų euristinių metodų kvadratinio paskirstymo uždaviniui apžvalga

4.1. Atkaitinimo modeliavimas

Atkaitinimo modeliavimo metodo ištakos slypi statistinėje mechanikoje, o jeigu tiksliau, sistemos, sudarytos iš didelio skaičiaus mažų diskrečiųjų dalelių, procesų modeliavime. Čia turima omenyje tokios sistemos energijų lygių modeliavimas „atkaitinant“ tą sistemą, t. y. iš pradžių sistemai suteikiant aukštą temperatūrą, o paskui temperatūrą palaipsniui mažinant tol, kol sistema „užsigrūdins“ (pereis į optimalią būseną). Šio modeliavimo pradininkai (1953 m.) buvo Metropolis, Rozenblutas ir kt. [42]. Jie pasinaudojo Bolcmano (eksponentiniu) pasiskirstymo dėsniu apibrėždami tikimybę, kad sistema, įvykus joje tam tikrai perturbacijai, pereis iš vieno energijos lygio (E_1) į kitą (E_2), kai temperatūra lygi t: $\mathrm{P} = \begin{cases} 1, \Delta E < 0, \\ \mathrm{e}^{-\Delta E / C_B t}, \Delta E \geq 0 \end{cases}$, čia $\Delta E = E_2 - E_1$, o C_B – Bolcmano konstanta. Cerný (1982 m. [12]) ir Kirkpatrikas su bendraautoriais (1983 m. [34]) buvo pirmieji, kurie panaudojo atkaitinimo modeliavimo metodą sprendžiant kombinatorinio optimizavimo uždavinius.

Konkrečios atkaitinimo modeliavimo algoritmų realizacijos skiriasi viena nuo kitos šiais veiksniais (faktoriais): aplinkos funkcija, atkaitinimo („atšaldymo“) schema ir baigimo sąlyga. Viena iš dažniausiai naudojamų aplinkos funkcijų yra vadinamoji porinių sukeitimų funkcija N_2

Yra dvi galimos alternatyvos peržiūrint „sprendinius-kaimynus“: pirma, pasirinkti einamojo sprendinio „kaimyną“ atsitiktinai; antra, nagrinėti einamojo sprendinio aplinką pagal tam tikrą fiksuotą, determinuotą tvarką, pvz., nuosekliai vykdant perturbacijas p_{ij}, kai i kinta nuo 1 iki $n-1$, o j – nuo $i+1$ iki n (norint išnagrinėti visus aplinkos N_2 sprendinius, reikia atlikti $O(n^3)$ operacijų).

Formuojant atkaitinimo schemą, yra svarbūs tokie veiksniais: a) pradinės (galutinės) temperatūros parinkimas, b) ekvilibriumo (pusiausvyros) testas ir c) temperatūros mažinimo formulė.

Atkaitinimo modeliavime temperatūra nėra nekintamas dydis, ji turi būti palaipsniui mažinama, pradedant nuo tam tikros pradinės reikšmės t_0. Ši reikšmė neturėtų būti nei per didelė, nei per maža. Iš tikrųjų, jei pradinė temperatūra būtų pernelyg aukšta, tai atkaitinimo procesas užsitęstų labai jau ilgai, bereikalingai nagrinėjant daug „blogų“ sprendinių. Kita vertus, per daug žema pradinė temperatūra galėtų lemti greitą „įkritimą“ į nebūtinai gero

lokaliojo optimumo „duobę“. Apibrėžiant konkrečią reikšmę t_0, ji galėtų būti prilyginama, pvz., Δf_{max}, čia Δf_{max} – didžiausias (teigiamas) tikslo funkcijos pokytis, gautas dviems „sprendiniams-kaimynams“ atlikus tam tikrą skaičių bandymų.

Atkaitinimo modeliavimo algoritmuose naudojamas vadinamasis ekvilibriumo testas tam, kad nustatyti, kuriuo būtent momentu turi būti mažinama temperatūra. Ekvilibriumas trumpai gali būti charakterizuojamas kaip tam tikra stabili, be žymesnių fliuktuacijų proceso būsena. Kombinatorinių uždavinių atveju ekvilibriumo kriterijumi galėtų būti, pvz., tikslo funkcijos reikšmių svyravimų apmlitudė (jeigu ji nedidelė, tai konstatuojama, jog ekvilibriumas pasiektas). Atsižvelgiant į tai, kaip realizuojamas ekvilibriumo testas, skiriami du atkaitinimo tipai: homogeninis ir nehomogeninis atkaitinimas. Pirmuoju atveju atliekama daug bandymų esant tai pačiai, fiksuotai temperatūrai; tai tęsiama tol, kol pasiekiamas ekvilibriumas – tada temperatūra sumažinama, ir procesas kartojamas. Antruoju atveju temperatūra mažinama po kiekvieno įvykdyto bandymo; galima sakyti, jog ekvilibriumo testas iš viso neatliekamas (šis atkaitinimo tipas kaip tik panaudotas toliau aprašomame algoritme).

Modeliuojant atkaitinimą svarbu parinkti tinkamą temperatūros mažinimo formulę. Praktiškai taikomuose atkaitinimo algoritmuose plačiausiai naudojamos yra: geometrinė formulė ($t_k = \alpha \cdot t_{k-1}$; tk – einamoji temperatūros reikšmė; k=1, 2, ...; $t0$=const; $0.8 \le \alpha \le 0.99$ [34]) ir Lundy-Mees formulė ($t_k = t_{k-1} / (1 + \beta t_{k-1})$; k=1, 2, ...; t_0=const; $\beta << t_0$ [38]).

Tiesa, naujausiose atkaitinimo modeliavimo algoritmų versijose temperatūra greičiau keičiama periodiškai nei monotoniškai mažinama. Tyrimų, atliktų dar 1989 m. [27], rezultatai liudija, kad vietoj tiesioginio atkaitinimo tikslingiau taikyti tam tikrą „atkaitinimų“ ir „kaitinimų“ seką, t. y., vadinamąjį reatkaitinimą (angl. reannealing), kuris suteikia papildomas, lankstesnes galimybes atkaitinimo modeliavimo algoritmams [5, 46].

Baigimo sąlyga. Atkaitinimo procesas teoriškai turėtų būti tęsiamas tol, kol galutinė temperatūra t_g tampa 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, arba, kai, atlikus pakamai daug bandymų, priimtų teigiamų sprendimų (t. y., perėjimų iš vieno sprendinio į kitą) skaičius tampa mažesnis už tam tikrą slenkstį. Dažnai kaip baigimo sąlyga tarnauja apriori fiksuotas atliekamų bandymų skaičius, t. y. atkaitinimo schemos „ilgis“. Išsamiai atkaitinimo modeliavimas įvairiems optimizavimo uždaviniams aprašomas [1, 2, 37] veikaluose.

4.2. Tabu paieška

Tabu paieškos metodas buvo pasiūlytas Hanseno, Jaumardo ir Gloverio 1987–1990 m. [22, 23, 29]. Šis metodas tapo labai populiarus ir plačiai taikomas įvairiems uždaviniams [24, 30].

Tabu paieška yra pagrįsta kaimyninių sprendinių paieška pereinant nuo vieno lokaliojo optimumo prie kito. Pagrindinė tabu paieškos idėja yra leidimas atlikti perėjimus net ir tais atvejais, kai nėra pagerinamas kaimyninis sprendinys. Vistik kai kurie perėjimai yra draudžiami, siekiant išvegti ciklinimų.

Tabu paieška pradedama nuo pradinio, galbūt atsitiktinai sugeneruoto, sprendinio s iš aibės S, po to iteraciniu būdu kartojamas perėjimų iš vieno sprendinio į kitą procesas. Kiekvienu procedūros žingsniu yra analizuojama einamojo sprendinio s kaimyninių sprendinių aibė $N(s)$, ir atliekamas tas perėjimas, kuris labiausiai pagerina (sumažina) tikslo funkcijos f reikšmę. Jeigu nėra pagerinančių perėjimų, tai pasirenkamas tas, kuris mažiausiai pablogina (padidina) tikslo funkcijos reikšmę. Trumpai tariant, perėjimas atliekamas į geriausiąjį kaimyninį sprendinį s' iš $N(s)$.

Siekiant išvengti grįžimo į neseniai nagrinėtą sprendinį, atvirkštinis perėjimas turi būti draudžiamas. Tai realizuojama įsimenant šį perėjimą (arba perėjimo požymį) tam tikroje atmintyje, vadinamoje tabu sąrašu (T). Šiame sąraše įsimenama $h=|T|$ paskutiniųjų perėjimų „žymės" („pėdsakai"). Dydis $h=|T|$ yra vadinamas tabu sąrašo ilgiu. Šis ilgis yra labai svarbus: jeigu jis yra labai mažas, tai gali būti gaunamas nepageidaujamas ciklinimas; tuo tarpu, jeigu – labai didelis, tai apribojama paieška srityse, „teikiančiose vilčių". Taigi perėjimas iš sprendinio s į sprendinį $s' \in N(s)$ yra traktuojamas kaip tabu, jeigu jis (ar jo požymis) yra sąraše T. Taip siekiama apsisaugoti nuo sugrįžimo į jau „aplankytus" sprendinius ir algoritmo „užsiciklinimo". Tačiau, po tam tikro laiko, grįžimas į anksčiau „aplankytus" sprendinius gali būti naudingas. Dėl tos priežasties, įvedamas vadinamasis aspiracijos kriterijus tam, kad leisti „tabu būsenai" būti atšauktai esant tam tikroms palankioms aplinkybėms; pavyzdžiui, tabu perėjimą iš s į s' galima leisti, jeigu $f(s')<f(s^*)$, kur s^* yra geriausias iki šiol rastas sprendinys). Tiek tabu sąrašas, tiek aspiracijos kriterijus yra dinamiškai atnaujinami algoritmo vykdymo eigoje. Paprastai, procesas yra baigiamas, kai tik aliekamas iš anksto užduotas bandymų skaičius.

4.3. Genetiniai algoritmai

Genetiniai algoritmai (GA) ir jų sudarymo principas buvo pasiūlytas XX-ojo amžiaus 70-aisiais metais Hollando [31]. Pradedant Hollando pirmuoju darbu, genetiniai algoritmai buvo sėkmingai išbandyti, sprendžiant įvairius optimizavimo uždavinius, pvz., tokius, kaip elektroninių komponentų išdėstymas [14], operacijų planavimas [47] ir kt. [16, 25, 43, 44]. Genetiniai algoritmai (GA), jų veikimas yra pagrįstas evoliucijos, vykstančios gyvojoje gamtoje, t. y., natūraliosios atrankos proceso imitavimu. Pagrindinės sąvokos, kurios naudojamos modeliuojant biologinės evoliucijos procesus, yra „individas“ ir „populiacija“. „Individas“ yra tam tikras elementarus, daugiau neskaidomas vienetas, objektas. Didesnė ar mažesnė „individų“ grupė sudaro „populiaciją“. Dar vienas svarbus dalykas yra vadinamasis „individo tinkamumas“ – savotiška „individo vertė“. „Individo vertę“ galima traktuoti kaip „individo“ sugebėjimo sėkmingai prisitaikyti (prie aplinkos), išlikti ir reprodukuotis laipsnį. Šia prasme „vertingesnis“ (grynai biologiškai) yra tas „individas“, kuris sugeba geriausiai prisitaikyti (būti „stipresnis“ už kitus) ir, gal būt, palikti didesnį palikuonių („vaikų“) skaičių.

Optimizavime vietoje sąvokų „individas“, „populiacija“, „individo vertė“ naudojamos tradicinės, įprastos sąvokos: „individui“ atitinka atskiras sprendinys, „populiacijai“ – sprendinių aibė (grupė, rinkinys), pagaliau, „individo vertė“ yra asocijuojama su tikslo funkcijos reikšme duotajam sprendiniui. Taip, kaip gyvosios gamtos evoliucijoje išlieka tik „vertingiausi“ („stipriausi“) „individai“, taip ir optimizavime siekiama gauti kuo „geresnį“ sprendinį, t. y., sprendinį su kuo mažesne (ar didesne) – nelygu, koks optimizavimo uždavinys (minimizavimo ar maksimizavimo) sprendžiamas – tikslo funkcijos reikšme.

Iš tikrųjų, terminas „genetiniai algoritmai“ reiškia ne kokį nors atskirą, specifinį euristinį optimizavimo algortimą. Greičiau šis terminas nusako tam tikrą bendrą, gana universalų metodą, strategiją – metaheuristiką, t. y., šeimą euristinių algoritmų su panašiais veikimo principais ir savybėmis. Taigi genetiniai algoritmai priklauso tai euristinių metodų klasei, kuriai būdingos sekančias savybės:

1. Operuojama su viena ar daugiau leistinų sprendinių „populiacijų“.
2. Atskiri sprendiniai iš vienos ar kelių „populiacijų“ parenkami (panaudojant kurią nors heuristinę taisyklę) tolimesniam nagrinėjimui, apdorojimui, teikiant pirmenybę tiems sprendiniams, kuri turi „geresnes“ tikslo funkcijos reikšmes.
3. Naudojamas tam tikras „mechanizmas“, kuris skirtas formuoti naujiems leistiniems sprendiniams, kombinuojant (derinant) anksčiau gautus sprendinius ar kurias nors jų savybes, charakteristikas.

4. Esant reikalui, panaudojamas papildomas „mechanizmas“ generuoti naujiems galimiems sprendiniams iš atskirų anksčiau gautų sprendinių (atliekant atsitiktinius tų sprendinių elementų perstatymus (perturbacijas)).

5. Atlikus (2)–(4) žingsnius, esama(os) „populiacija(os)“ „atnaujinama“, pašalinant iš jos (jų) tam tikrus sprendinius (paprastai, „blogesnius“) bei paliekant joje (jose) „geresniuosius“ sprendinius, tame tarpe, naujus suformuotus (sugeneruotus) (jeigu jie „pakankamai geri“).

Formalizuojant genetinių algoritmų aprašymą, aukščiau minėtos bendrosios savybės, „mechanizmai“ susiejami su atitinkamomis procedūromis. Taip (2) savybė susiejama su vadinamąja atrinkimo procedūra, (3) „mechanizmas“ tradiciškai vadinamas krosoveriu, (4) „mechanizmas“ – mutavimo procedūra („mutatoriumi“). (5) etape (žingsnyje) atliekamas „populiacijos“ atnaujinimas (savotiškas „filtravimas“, „gryninimas“).

Bendroji genetinių algoritmų bazinė struktūra (paradigma) pateikiama žemiau:

1) sukurti pradinę (atsitiktinių) leistinų sprendinių „populiaciją“ (arba kelias „populiacijas“);
2) kartoti {
   2a) parinkti „sprendinius-tėvus“;
   2b) vykdyti krosoverį parinktiems „sprendiniams-tėvams“, gaunant „sprendinį-palikuonį“, kuris įtraukiamas į „populiaciją“;
   2c) vykdyti mutavimo procedūrą tam tikriems duotosios „populiacijos“ sprendiniams;
   2d) atnaujinti esamąją „populiaciją“;
   }
3) baigti kai patenkinama baigimo sąlyga.

Egzistuoja daugybė įvairių būdų, kaip konkrečiai realizuoti „sprendinių-tėvų“ parinkimo, krosoverio, mutavimo bei „populiacijos“ sprendinių atnaujinimo procedūras [25]. Pavyzdžiui, mes galime operuoti su viena didele sprendinių „populiacija“, tačiau galime turėti ir kelias mažesnes („lygiagrečias“) „populiacijas“ („sub-populiacijas“). Toliau, mes galime pasirinkti „tėvus“, atsižvelgiant į jų vertingumą (tinkamumą), t. y., absoliutinę tikslo funkcijos reikšmę – arba į kokius nors kitus įvertinimus, charakteristikas ar kriterijus. Galima labai įvairiai realizuoti krosoverio bei mutavimo procedūras, be to, galima nuspręsti apjungti krosoverį ir mutavimą į vieną procedūrą arba tai daryti atskirų, nepriklausomų procedūrų pagalba. Pagaliau, mes galime apsispręsti pakeisti visus be išimties einamosios „populiacijos“ narius (sprendinius) naujai gautais „palikuonimis“ po kiekvieno algoritmo ciklo iteracijos įvykdymo. Dar reikia pastebėti, kad genetinio algoritmo realizacijos gali priklausyti nuo to, kaip yra vaizduojami, pateikiami sprendiniai, kitaip tariant, nuo sprendinio „kodavimo“ pobūdžio.

Pastebėsime, jog genetinius algoritmus kiek primena vadinamieji „skruzdėlių algoritmai”, besiremiantys skruzdėlių kolonijų elgsenos imitavimu [40, 53].

4.4. Apibendrinamosios pastabos

Išnagrinėjus aukščiau apžvelgtus metodus, galima suformuluoti tokias išvadas ir apibendrinimus:

- atkaitinimo modeliavimo algoritmai yra stochastinio pobūdžio; jų konvergavimas į lokaliai optimalų sprendinį gali būti pakankamai greitas, tačiau gautojo sprendinio kokybė nebūtinai visais atvejais gali būti patenkinama; gautojo sprendinio aplinkoje gali egzistuoti ir geresnių sprendinių;
- tabu paieškos algoritmai, skirtingai negu atkaitinimo modeliavimo algoritmai, gali būti traktuojami kaip determinuotieji algoritmai; parinkus tinkamą tabu sąrašą, jo nagrinėjimo strategiją, taip pat aspiracijos kriterijų, galima kai kuriems uždaviniams gauti geresnius rezultatus negu atkaitinimo modeliavimo atveju;
- genetiniai algoritmai gali leisti pasiekti irgi labai gerus rezultatus, tačiau šie algoritmai reikalauja žymiai daugiau skaičiavimų laiko, taip pat ir atminties; neretai juos gali būti gana sunku praktiškai realizuoti; be to, norint gauti kuo geresnius rezultatus, genetiniai algoritmai savo ruožtu turi naudoti kitus (lokaliosios paieškos) algoritmus, t. y., turi būti naudojami hibridiniai genetiniai algoritmai.

Ieškant kompromisinių variantų, turi būti numatomos tokios naujų metodų sudarymo kryptys:

- metodų ir algoritmų modifikavimas, tobulinimas („specializavimo” kryptis);
- universalios paskirties optimizavimo metodų adaptavimas sudarant konkrečius euristinius algoritmus, pritaikytus konkrečiam uždaviniui („adaptavimo” kryptis);
- esamų strategijų ir metodų kombinavimas siekiant surasti efektyvius algoritmus egzistuojančių metodų sandūroje („selekcijos” kryptis).

Galima ir dar viena kryptis - tai iš principo naujų metodų generavimas įvertinant geriausią sukauptą patirtį.

Atsižvelgiant į šias kryptis, ir buvo sukurtas patobulintas tabu paieškos algoritmas.

5. Tyrimo dalis: tabu paieškos algoritmo patobulinimas

5.1. Tabu paieška: klasikinė schema

Kaip jau minėta, tabu paieškos metodas remiasi išplėsta lokaliąja pieška. Skirtingai, negu lokaliosios paieškos procedūros, kurios apsiriboja lokaliai optimalaus sprendinio (lokaliojo optimumo) suradimu nagrinėjamo sprendinio aplinkoje (kaimyninių sprendinių aibėje), tabu paieška pagrįsti algoritmai tęsia paiešką ir tuo atveju, kai surandamas lokalinis optimumas, t. y., duoto sprendinio aplinkoje neįmanoma surasti geresnio sprendinio (sprendinio su mažesne – jei sprendžiamas minimizavimo uždavinys – tikslo funkcijos reikšme). Tabu paieška pagrįsta uždraudimų metodologija: draudimai yra būtini tam, kad neleistų grįžti į tas pačias situacijas, t. y., neseniai nagrinėtus sprendinius, ir tokiu būdu išvengti ciklinimų.

Tabu paieška pradedama nuo pradinio sprendinio $s^{(0)}$ iš galimų sprendinių aibės S. Pradinis sprendinys gali būti sugeneruotas tiesiog atsitiktiniu būdu. Algoritmo vykdymo eigoje analizuojama sprendinio $s \in S$ aplinkos (kaimyninių) sprendinių aibė $N(s)$. Baigus analizę, pereinama į tą sprendinį s' iš $N(s)$, kuriam tikslo funkcijos f reikšmė yra mažiausia. Perėjimas atliekamas ir tuo atveju, kai tikslo funkcijos pokytis yra teigiamas (t. y., tikslo funkcijos reikšmė „pablogėja“) – taip galima pereiti nuo vieno lokaliai optimalaus sprendinio prie kito. Grįžimas į anksčiau nagrinėtą sprendinį turi būti uždraudžiamas tam tikram laikotarpiui – kad būtų išvengta ciklo (paieškos kartojimo iš naujo nuo to pačio „taško“ (sprendinio)). Taigi nagrinėtieji sprendiniai tam tikrais momentais tampa „tabu“, t. y., jie įtraukiami į specialią atmintį – vadinamąjį tabu sąrašą (jį žymėsime T). Tokiu būdu perėjimas į sprendinį $s' \in N(s)$ yra draudžiamas, jeigu tas sprendinys (ar tam tikras požymis, susijęs su tuo sprendiniu) duotu metu yra sąraše T.

Tiesmukiškas sprendinių draudimas gali apriboti paiešką kai kuriose sprendinių aibės S srityse. Todėl kartu su tabu sąrašu naudojamas ir vadinamasis aspiracijos kriterijus. Šio kriterijaus pagalba galima anuliuoti, „nekreipti dėmesio” į esamą „tabu būseną“, esant tam tikroms, „palankioms“ sąlygoms (aplinkybėms). Vienas iš standartinių aspiracijos kriterijų yra toks: „perėjimas“ iš sprendinio s į sprendinį s' –nors jis ir yra „tabu“ – leidžiamas, jeigu tenkinama sąlyga $f(s') < f(s^*)$, kur s^* yra geriausias iki šiol paieškos eigoje surastas sprendinys. Tabu paieškos algoritmo lankstumui padidinti tabu sąrašas, tiksliau, tabu sąrašo ilgis (dydis) algoritmo vykdymo eigoje gali būti kartas nuo karto atnaujinamas. Tabu paieškos algoritmo vykdymas baigiamas, atlikus iš anksto nustatytą paieškos iteracijų (bandymų) skaičių.

Galimos ir kitos algoritmo baigimo sąlygos. Tabu paieškos algoritmo rezultatas yra geriausiasis surastas sprendinys – visai nebūtinai tas sprendinys, kuris gautas paskutinėje paieškos iteracijoje; geriausias rastas sprendinys turi būti „įsimenamas“ tuo momentu, kuriuo jis randamas.

Tabu paieškos algoritmai skiriasi vienas nuo kito, atsižvegiant į tabu sąrašo realizavimą, aspiracijos kriterijaus parinkimą bei kitus veiksnius, tokius kaip papildoma („ilgalaikė“) tabu atmintis, paieškos intensifikacijos, diversifikacijos mechanizmai ir kt. Paminėtinos šios tabu paieškos formos: deterministinė tabu paieška (fiksuotoji tabu paieška, reaktyvioji tabu paieška) ir stochastinė tabu paieška (tikimybinė tabu paieška, randomizuotoji tabu paieška). Plačiau tabu paieškos metodai ir algoritmai išdėstyti [24, 30] veikaluose.

Apibendrintas, formalizuotas tabu paieškos algoritmo aprašymas pateikimas 2 pav.

```
   function tabu_paieška(s);
   // pradiniai duomenys: s – pradinis sprendinys; rezultatai: s* – geriausias rastas sprendinys //
   s* := s;
   inicializuoti tabu sąrašą T;
   repeat // vykdyti tabu paieškos ciklą //
     rasti geriausią sprendinį s′ ∈ Θ(s) ⊆ Θ(s), čia Θ(s) – sprendinio s aplinkos
     poaibis, kurio sprendiniai (arba "perėjimo" iš s į Θ(s) „pėdsakai")
nepriklauso
     tabu sąrašui T arba tenkina aspiracijos sąlygą;
     s := s′; // pakeisti esamą sprendinį nauju ir naudoti kaip „išeities tašką“
tolimesnėse iteracijose //
     įsiminti sprendinį s (arba "perėjimo" iš s į s′ „pėdsaką") tabu sąraše T;
     if f(s) < f(s*) then s* := s; // įsiminti geriausią rastą sprendinį //
     jei reikia, atnaujinti tabu sąrašą T
   until patenkinta baigimo sąlyga;
   return s*
end.
```

2 pav. Tabu paieškos paradigma

Pažymėsime, jog šiek tiek adaptuotus TP algoritmus galima inkorporuoti į kitas meta-euristikas, pavyzdžiui, genetinius algoritmus (tiksliau, hibridinius genetinius (memetinius) algoritmus), kur TP algoritmai „tarnauja“ kaip pradinių aukštos kokybės sprendinių „populiacijų“ generatoriai ir(arba) labai efektyvios atskirų sprendinių gerinimo procedūros.

5.2. Klasikiniai tabu paieškos algoritmai KP uždaviniui

Tabu paieškos metodas sėkmingai išbandytas įvairiems kombinatorinio optimizavimo uždaviniams, tame tarpe ir kvadratinio paskirstymo uždaviniui. Pirmasis tabu paieškos algoritmas KP uždaviniui buvo pasiūlytas 1990 m. Skorin-Kapov [49]. Tai – fiksuotosios tabu paieškos algoritmas. Algoritme panaudotas fiksuoto ilgio (dydžio) h tabu sąrašas T (h=|T|). Algoritmo vykdymo eigoje tas sąrašas tvarkomas pagal taisyklę: „pirmasis įėjo – pirmasis

išėjo“ (angl. *FIFO* – „*first input – first output*“), t. y., sprendinys (ar to sprendinio požymis), kuris anksčiausiai įtraukiamas į sąrašą, anksčiausiai iš to sąrašo ir pašalinamas. Paieškos procedūra pagrįsta tokia strategija: jeigu duotojo sprendinio (perstatymo) sukeičiami (perstatomi) elementai, pvz., elementai u ir v, priklauso tabu sąrašui T (tai reiškia, jog tie elementai buvo sukeisti ne anksčiau kaip prieš h iteracijų), tai toks sukeitimas yra neleidžiamas („tabu“) – nebent tų elementų sukeitimo pasekoje būtų gautas geresnis sprendinys negu geriausias iki tol surastas. Perėjus prie naujo sprendinio (atlikus elementų sukeitimą), sukeistieji elementai įtraukiami į tabu sąrašą, tuo pačiu pašalinant iš to sąrašo anksčiausiai į jį įtrauktą elementų porą – tai porai anuliuojma „tabu būsena“.

Kitas tabu paieškos algoritmas – randomizuotosios tabu paieškos algoritmas – buvo pasiūlytas 1991 m. Taillardo [54]. Priešingai negu Skorin-Kapov algoritme, Taillardo algoritme buvo panaudotas dinamiškai (atsitiktiniu būdu) kintančio ilgio (dydžio) h tabu sąrašas. Šiame algoritme yra taip pat kiek modifikuota sprendinių elementų įtraukimo į tabu sąrašą ir jų nagrinėjimo tvarka. Be to, panaudotas papildomas aspiracijos kriterijus, būtent: jeigu kuri nors elementų pora nebuvo sukeista labai ilgą laiko tarpą, tai, įvykdžius tam tikrą iteracijų skaičių, tos poros elementai yra sukeičiami, ir pereinama į atitinkamą sprendinį – nepriklausomai nuo „tabu būsenos“ ir tikslo funkcijos reikšmės. Svarbi Taillard‘o algoritmo savybė yra ta, jog einamojo sprendinio (perstatymo) π aplinka $N(\pi)$ išnagrinėjama panaudojant $O(n^2)$ operacijų vietoje $O(n^3)$ operacijų, kaip yra Skorin-Kapov algoritme – o tai labai paspartina paiešką. Kaip tik dėl to, panaudojant šį algoritmą, buvo gauti labai geri rezultatai KP uždaviniui [54]. Taillardo algoritmas yra apskritai vienas iš efektyviausių euristinių algoritmų KP uždaviniui, be to, jis yra labai lengvai programuojamas ir įdiegiamas.

Sukurta ir kitų tabu paieška besiremiančių algoritmų. Vienas iš tokių yra Battiti ir Tecchiolli algoritmas [4] – reaktyviosios tabu paieškos algoritmas. Šis algoritmas kai kuriems KP uždavinio testiniams pavyzdžiams yra šiek tiek pranašesnis už Taillardo algoritmą, tačiau jį žymiai sunkiau suprogramuoti ir realizuoti. Kitų tabu paieškos algoritmų bei jų modifikacijų aprašymus galima rasti atitinkamuose straipsniuose, pvz., Chakrapani ir Skorin-Kapov, 1993 [13], Skorin-Kapov, 1994 [50], Kelly et al., 1994 [33], Sondergeld ir Voss, 1996 [51] ir kt.

5.3. Tabu paieškos algoritmo patobulinimas

Toliau pateikiamas patobulintas tabu paieškos algoritmas KP uždaviniui. Iteratyvioji tabu paieška (ITPA) yra viena iš naujausių euristinių algoritmų grupių. ITPA savo ruožtu remiasi iteratyviosios lokaliosios paieškos (ILP) koncepcija (Lourenco ir kt., 2002). ILP esmė yra ta, jog tradicinių LP algoritmų rezultatus galima pagerinti pritaikant tam tikrus

papildomus sprendinių pertvarkymus. Nesigilinant į detales, ILP gali būti lakoniškai įvardijama kaip „griauti ir atstatyti“ principu grįsta paieškos strategija. Tokios strategijos panaudojimo tikslas – „pereinant“ nuo vieno lokaliai optimalaus sprendinio prie kito, stengtis aprėpti kuo didesnį sprendinių „poerdvį“, taip sudarant palankesnes sąlygas artėti prie globaliojo optimumo.

Iteratyviajai tabu paieškai būdingi du pagrindiniai etapai (fazės): 1) sprendinio pagerinimas panaudojant tabu paieškos algoritmą, 2) sprendinio rekonstravimas (dar vadinamas mutavimu) panaudojant tam tikslui skirtą procedūrą. Tarp šių etapų paprastai įsiterpia ne toks svarbus sprendinio kandidato atrankos rekonstravimui žingsnis; be to, kai kuriais atvejais, kai ilgą laiką nepavyksta surasti geresnių sprendinių (t.y. pasireiškia savotiška paieškos „stagnacija“), vietoje rekonstravimo vykdomas grynai atsitiktinio sprendinio generavimas („šaltasis restartas“).

ITPA inicijuojama pradinio sprendinio pagerinimu, gaunant (pirmąjį) lokaliai optimalų sprendinį, tarkime $s^{\bullet}$. Gautasis lokaliai optimalus sprendinys yra tam tikru mastu „suardomas“, tiksliau, rekonstruojamas gaunant naują sprendinį, pavyzdžiui, $s^{\sim}$. Rekonstruojant sprendinį nesiekiama jo visiškai, absoliučiai „sugriauti“. Atvirkščiai, tikslinga, kad gautas naujas sprendinys „paveldėtų“ tam tikras geras ankstesnio sprendinio charakteristikas. Tačiau kartu turi būti užtikrinamas ir deramas diversifikavimo (įvairovės) lygis. Taigi rekonstravimas neturi būti nei per daug „stiprus“, nei per daug „silpnas“. Pirmuoju atveju paieška taptų labai panaši į daugkartinę paiešką startuojant tiesiog nuo atsitiktinių sprendinių, ir būtų prarandama naudinga paieškos metu sukaupta informacija; antruoju atveju rekonstravimas negarantuotų pakankamai „nutolusio“ (nuo duotojo) sprendinio generavimo – o tai lemtų „grįžimą“ atgal į ankstesnįjį LO po sprendinio pagerinimo etapo. Rekonstruotas sprendinys $s^{\sim}$ vaidina pradinio sprendinio iš naujo startuojančiai TP procedūrai vaidmenį. TP procedūra grąžina naują lokaliai optimalų sprendinį $s^{\bullet}$, šis vėl rekonstruojamas, ir t.t. Geriausias lokalusis optimumas (s^{*}) „įsimenamas“. Sprendinių pagerinimo (TP būdu) ir rekonstravimo etapai iteraciniu būdu kartojami kiek norima daug kartų; paprastai ITP vykdymo trukmė valdoma iteracijų skaičiumi. Apibendrinta ITP schema pateikiama 3 paveiksle.

```
function iteratyvioji_tabu_paieška(s);
   // pradiniai duomenys: s – pradinis sprendinys; rezultatai: s* – geriausias rastas sprendinys //
   s• := tabu_paieška(s);  // pagerinti pradinį sprendinį panaudojant TP procedūrą //
   s := s•; s* := s•;
   repeat // vykdyti iteratyviosios tabu paieškos ciklą //
     if ilgą laiko tarpą nerasta geresnio sprendinio then s~ := naujas
sprendinys
     else begin
        s := kandidato_nustatymas(s,s•); // parinkti sprendinį rekonstravimui //
        s~ := rekonstravimas(s)          // rekonstruoti sprendinį s gaunant sprendinį s~ //
     end;
     s• := tabu_paieška(s~); // pagerinti sprendinį s~ (TP pagalba) gaunant sprendinį s• //
     if f(s•) < f(s*) then s* := s• // įsiminti geriausią rastą sprendinį //
   until patenkinta baigimo sąlyga;
   return s*
end.
```

3 pav. Patobulintos (iteratyviosios) tabu paieškos algoritmo KP uždaviniui šablonas

5.4. Iteratyviosios tabu paieškos algoritmas KP uždaviniui

Šiame darbe pasiūlytas patobulintas tabu paieškos algoritmas, kuris pavadintas ITTPA – iteratyviosios tabu paieškos algoritmas. Pirmiausia, jis remiasi kai kuriais vadinamosios randomizuotos tabu paieškos (RTP), aprašytos Taillard'o straipsnyje [54], principais. Kaip ir Taillard'o algoritme, mūsų algoritme tabu sąrašas realizuotas dvimatės matricos $\boldsymbol{T}=(t_{ij})_{nxn,}$ sudarytos iš sveikų skaičių, pagalba. Pradžioje visi matricos $\boldsymbol{T}$ elementai yra lygūs 0. Algoritmo vykdymo eigoje matricos elementas t_{ij} saugo einamosios iteracijos numerį ir tabu sąrašo koeficientą h $(h>0)$; kitaip tariant, t_{ij} yra lygus numeriui iteracijos, kuria pradedant i–tajam ir j-tajam sprendinio elementams anuliuojama „tabu būsena", t.y., tie elementai vėl gali būti sukeičiami vietomis. Taigi „perėjimas" į sprendinį, sukeičiant elementus i ir j vietomis, yra „tabu", jeigu $t_{ij}>=q,$ kur q yra einamosios iteracijos numeris.

Labai svarbu parinkti prideramą koeficiento h reikšmę. Esant šiai reikšmei labai mažai, padidėja paieškos „ciklinimosi" tikimybė; tuo tarpu, esant reikšmei pernelyg didelei, gali būti apribojama paieška atskirose sprendinių „erdvės" srityse – o tarp šių gali būti ir tokių, kuriose „slypi" labai geri lokaliai optimalūs sprendiniai (ar net globaliai optimalus sprendinys). Eksperimentais nustatyta, kad vienas iš perspektyvių būdų yra naudoti kintamą tabu sąrašo koeficientą, užuot naudojus jį iš anksto fiksuotą. Geri rezultatai gaunami, kai reikšmė h yra kaitaliojama randomizacijos keliu, t.y., ji parenkama atsitiktiniu būdu ir taip, kad patektų į tam tikrą užduotą intervalą, pvz., $[h_{min}, h_{max}]$. Parametrų h_{min} ir h_{max} reikšmes galima tiesiog susieti su uždavinio apimtimi $n,$ pvz., $h_{min} = 0.1n,$ $h_{max} = 0.3n.$ Reikšmės h kaitaliojimo dažnumas prilyginamas $2h_{max,}$ t.y., h perskaičiuojama kiekvieną kartą, kai tik įvykdoma $2h_{max}$ tabu paieškos iteracijų.

Algoritme ITTPA panaudotas klasikinis aspiracijos kriterijus, t.y., „tabu statusas“ ignoruojamas, jeigu „perėjimas“ atliekamas į tokį sprendinį, kuris yra geresnis už bet kurį kitą iki tol rastą.

Algoritme panaudota efektyvi formulė tikslo funkcijos z pokyčiams Δz skaičiuoti [21, 54]:

$$\begin{aligned}\Delta z(\pi', i, j) = \Delta z(\pi, i, j) + (a_{iu} - a_{iv} + a_{jv} - a_{ju})(b_{\pi(i)\pi(u)} - b_{\pi(i)\pi(v)} + b_{\pi(j)\pi(v)} - b_{\pi(j)\pi(u)}) + \\ (a_{ui} - a_{vi} + a_{vj} - a_{uj})(b_{\pi(u)\pi(i)} - b_{\pi(v)\pi(i)} + b_{\pi(v)\pi(j)} - b_{\pi(u)\pi(j)})\end{aligned}, \quad (5)$$

čia π yra esamasis sprendinys (perstatymas); π' – naujas sprendinys iš esamo sprendinio aplinkos $N(\pi)$, gautas, sukeičiant sprendinyje π u-tąjį ir v-tąjį elementus vietomis (atliekant porinį sukeitimą); a_{ij} ir b_{kl} yra atitinkami duomenų matricų $\boldsymbol{A}$ ir $\boldsymbol{B}$ elementai. Duota formulė galioja visiems indeksams $i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., n\}$, išskyrus indeksus $i=u$, $i=v$, $j=u$ ir $j=v$. Pastariesiems indeksams turi būti taikoma formulė (2). Nors (2) formulės sudėtingumas yra $O(n)$, tačiau ji taikoma tik keturioms indeksų poroms, o tai leidžia sprendinių aplinką pilnai išnagrinėti (patikrinti), panaudojant tik $O(n^2)$ operacijų. Išimtį sudaro inicializacijos fazė, t. y., pati pirmoji iteracija, kurios metu vistiek reikia (bet tik vieną kartą) atlikti $O(n^3)$ operacijų tam, kad išnagrinėti visą pradinio sprendinio aplinką (aplinkos sprendinių aibę).

Tabu paieškos algoritmo vykdymas valdomas panaudojant iteracijų skaitiklį – Q_{TP}. Detalizuota modifikuoto tabu paieškos algoritmo struktūrinė schema (šablonas) pateikiama 4 pav.

```
procedure MRTPA { modifikuotas tabu paieškos algoritmas }
  { įėjimo duomenys: π – einamasis (pradinis) sprendinys, n – uždavinio apimtis }
  {                  Q_TP – iteracijų skaičius (Q_TP≥1) }
  {                  h_min, h_max – mažesnio ir didesnio tabu sąrašo dydžiai (h_min<h_max) }
  { išėjimo duomenys: π*, z* – geriausias sprendinys ir atitinkama tikslo funkcijos reikšmė }
  apskaičiuojama tikslo funkcijos reikšmė z perstatymui π
  π*:=π, z*:=z,  K := n(n−1)/2
  for i:=1 to n−1 do for j:=i+1 to n do paskaičiuoti DELTA(i,j)=Δz(π,i,j)
  for i:=1 to n−1 do for j:=i+1 to n do T(i,j):=0
  pasirenkamas atsitiktinis h iš intervalo hmin ir hmax
  i:=1, j:=1
  for q:=1 to Q_TP do begin { pagrindinis algoritmo ciklas }
    Δmin:=∞
    for k:=1 to K do begin { rasti geriausią perstatymą }
      i := iif(j < n, i, iif(i < n−1, i+1, 1)) ,  j := iif(j < n, j+1, i+1)
      Δ:=DELTA(i,j)
      forbidden:=(T(i,j)≥q)
      aspired:=(z+Δ<z*) and (forbidden=TRUE)
      if ((Δ<Δmin) and (forbidden=FALSE)) or (aspired=TRUE) then begin
        u:=i, v:=j
        if aspired=TRUE then Δmin:=−∞ else Δmin:=Δ
      end { if }
    end { for }
    π := π ⊕ p_uv  { atlikti perstatymą }
    z:=z+DELTA(u,v)
    if z<z* then begin
      π*:=π, z*:=z { įsiminti geriausią sprendinį ir tikslo funkcijos reikšmę }
    end { if }
    T(u,v):=q+h
    for u:=1 to n−1 do for v:=u+1 to n do atnaujinti DELTA(u,v)
    if q mod 2hmax = 0 then pasirinkti naują atsitiktinį h iš intervalo hmin ir hmax
  end { pagrindinio ciklo pabaiga }
end { MRTPA }
```

4 pav. Modifikuotas tabu paieškos algoritmas KP uždaviniui

Siekiant dar daugiau pagerinti TP efektyvumą išbandytas tabu paieškos ir „godžiąja“ strategija besiremiančios „greito nusileidimo“ procedūros kombinavimas. Idėja slypi tame, kad tam tikrais paieškos periodais tikslinga atlikti savotišką „relaksaciją“, t.y., tabu paiešką pakeisti „godžiąja“ paieška laikinai anuliuojant visus draudimus. Šiuo atveju „godžioji“ paieška gali būti traktuojama kaip paieškos intensifikavimas atskirose sprendinių „erdvės“ dalyse. Finišavus „greito nusileidimo“ procedūrai, atstatomi buvę draudimai ir vėl tęsiama įprasta TP procedūra. Taip kombinuojant paieškos režimus pavyksta pagerinti gaunamus rezultatus, lyginant juos su „grynosios“ TP rezultatais.

Be aukščiau apžvelgtų bruožų algoritmas ITTPA pasižymi dar viena labai svarbia nauja savybe, būtent tuo, kad yra realizuota vadinamosios iteratyviosios tabu paieškos strategija. Šios strategijos esmė glūdi tame, jog galima pagerinti įprastų TP algoritmų rezultatus pritaikant tam tikrus sprendinių pertvarkymus (mutavimus). Iteratyviajai tabu

paieškai būdingi du pagrindiniai etapai: 1) sprendinio pagerinimas (panaudojant įprasyą tabu paiešką) ir 2) sprendinio rekonstravimas (panaudojant mutacijos procedūrą). Beje, galimi du rekonstravimo tipai: a) „karštasis“ rekonstravimas ir b) „šaltasis“ rekonstravimas.

Pirmojo tipo rekonstravimas paprastai atliekamas po kiekvieno sprendinio pagerinimo. „Karštojo“ rekonstravimo metu nesiekiama duotojo sprendinio visiškai „suardyti“, o siekiama, kad naujai gautas sprendinys „paveldėtų“ tam tikras geras ankstesnio sprendinio charakteristikas, nors tuo pačiu būtų užtikrinamas ir tam tikras geras ankstesnio sprendinio charakteristikas, nors tuo pačiu turi būti užtikrinamas ir tam tikras diversifikavimo (įvairovės) lygis. „Karštasis“ rekonstravimas neturi būti nei per daug „stiprus“, nei per daug „silpnas“. Pirmuoju atveju paieška taptų labai panaši į daugkartinę (gana chaotišką) tabu paiešką startuojant nuo visai atsitiktinių sprendinių, ir būtų prarandama naudinga paieškos eigoje eigoje sukaupta informacija; antruoju atveju rekonstravimas negarantuotų pakankamai „nutolusio“ (nuo duotojo) sprendinio generavimo – o tai galėtų sąlygoti paieškos ciklinimąsi, t.y., „sugrįžimą“ į jau anksčiau nagrinėtus sprendinius. „Karštąjį“ rekonstravimą galima realizuoti kaip atsitiktinį „perėjimą“ į to sprendinio aukštesnės eilės aplinką N_λ, kur $\lambda>2$. Vienas iš galimų mechanizmų tokiam „perėjimui“ modeliuoti – tai panaudoti atsitiktiniu būdu generuojamą elementarių perturbacijų (tokių, pvz., kaip φ_{ij}) seką, pvz., $\{\varphi_{rs}\}_{\mu,}$ kur r ir s gaunami (pseudo-) atsitiktinių skaičių generatoriaus pagalba, o μ yra sekos „ilgis“.

Antrojo tipo rekonstravimo ypatybė yra ta, kad šiuo atveju galima sprendinį visiškai „sugriauti“. Tokio „sugriovimo“ tikslas yra stengtis aprėpti kiek įmanoma didesnius, įvairesnius sprendinių „poerdvius“, taip sudarant palankesnes sąlygas artėti prie globaliojo optimumo. Paprasčiausiu atveju, „šaltąjį“ rekonstravimą galime realizuoti tiesiog kaip naujo, visai atsitiktinio sprendinio generavimą.

Algoritmo ITTPA vykdymas valdomas panaudojant šiuos parametrus: Q_1 – išplėstinės tabu paieškos iteracijų skaičius; Q_2 – modifikuoto randomizuotos tabu paieškos iteracijų skaičius; h_{min}, h_{max} – tabu sąrašo ilgio rėžiai; $\mu_{min,}$ μ_{max} – atsitiktinių perturbacijų sekų ilgio (mutavimo lygio) rėžiai; r – “relaksacijos” koeficientas (parametras, apibrėžiantis kokiu dažnumu vykdyti “greitą nusileidimą”); ω - parametras, nurodantis kokiais periodais vykdyti “šaltąjį” rekonstravimą. ITTPA šablonas pateikiamas 5 pav. Algoritmas ITTPA savo ruožtu naudoja naudoja procedūras “MRTPA” (modifikuotas randomizuotos tabu paieškos algoritmas), jos šablonas parodytas 4 pav., “mutacija” ir “greitas_nusileidimas”. Paminėtina, kad algoritmas ITTPA gali būti vykdomas daug kartų, esant įvairioms valdymo parametrų reikšmių kombinacijoms – tie atskiri algoritmo įvykdymai vadinami restartais.

```
procedure ITTPA; (* iteratyviosios tabu paieškos algoritmas KP uždaviniui *)
  (* pradiniai duomenys: π - pradinis sprendinys; n – uždavinio apimtis *)
  (* Q1, Q2, μmin, μmax, hmin, hmax, ω - valdantieji parametrai *)
  (* rezultatai: π* - geriausias rastas sprendinys *)
  π0 := MRTPA(π,Q2 ); (* optimizuojamas pradinis sprendinys panaudojant algoritmą
MRTPA*)
  π• := π°; π* := π°; q• := 0; μ:=μmin – 1;
  for q := 1 to Q1 do begin (* pagrindinis išplėstinės tabu paieškos ciklas *)
      if q-q• > ω then begin q• := q; sugeneruoti naują atsitiktinį sprendinį π, π• := π end
        else begin
          if μ < μmax then μ := μ+1 else μ:=μmin ;
           π := mutacija(π°,μ)
        end;
      π0 := MRTPA(π,Q2 )
      if z(π°)<z(π•) then begin
        q• :=q; μ:=μmin – 1; π• := π°; if z(π•) < z(π* ) then π* := π•
       end
  end
end
```

5 pav. ITTPA algoritmo šablonas.

6. Eksperimentai ir jų rezultatai

Tiriant naujojo algoritmo ITTPA efektyvumą, atlikta daug eksperimentų. Eksperimentiniams tyrimams pasinaudota KP uždavinio testiniais pavyzdžiais iš bibliotekos QAPLIB [9]. Eksperimentai vykdyti su programa OPTIQAP (OPTImizer for the QAP) ir 500 MHz taktinio dažnio AMD kompiuteriu. Programos struktūra pateikta 6 pav.

```
program OPTIQAP { optimizatorius KP uždaviniui }
  įvedami duomenys ir parametrai
  einamasis_vykdymo_skaicius:=0
  repeat { pagrindinis ciklas }
    einamasis_vykdymu_skaicius:= einamasis_vykdymu_skaicius +1
    generuojamas einamasis (pradinis) sprendinys π, naudojant procedūrą APG
    vykdomas algoritmas ITTPA sprendiniui π
  until ((einamasis_vykdymu_skaicius >W) arba patenkinta baigimo sąlyga)
end { OPTIQAP }
```

6 pav. Optimizatoriaus KP uždaviniui schema

Pastabos. 1. Procedūros APG (Atsitiktinių Perstatymų Generatorius) šablonas pateiktas 7 pav.

2. *W* yra maksimalus kartotinių vykdymų skaičius

```
procedure APG { atsitiktinių perstatymų generatorius }
  { įėjimo duomenys: n – perstatymų skaičius;
    išėjimo duomenys: π – sugeneruotas perstatymas }
  for i:=1 to n do π(i):=i
  for i:=1 to n–1 do begin
    j:=i+RANDOM(n–i+1)
    atliekamas perstatymo π i-ojo ir j-ojo elementų sukeitimas vietomis
  end { for }
end { APG }
```

7 pav. Atsitiktinių perstatymų generatorius

Pastabos. 1. Ši procedūra naudoja atsistiktinių skaičių generatorių (funkcija RANDOM), realizuotą kompiliatoriuje Free Pascal (Funkcija RANDOM(*x*) grąžina atsitiktinį skaičių intervale [0, *x*).) 2. Prieš naudojant šią procedūrą, atsitiktinių skaičiaus generatoriaus inicializacijos parametras („šaknis“) turi būti prilygintas kuriai nors reikšmei, pvz., $RS+w-1$, kur $0 \le RS \le 2^{32}-1$ (yra naudojama $RS=10^9$), *w* – yra einamasis algoritmo kartotinio vykdymo numeris ($w=1, 2, ...$)

Algoritmo ITTPA rezultatai buvo lyginami su gerai žinomų, efektyvių TP algoritmų rezultatais. Tie algoritmai – tai Taillard‘o algoritmas [54], RATPA (randomizuotosios tabu paieškos algoritmas), kurio autorius - bei Battiti ir Tecchiolli algoritmas [4] (pavadintas RETPA – reaktyviosios tabu paieškos algoritmas).

Buvo pasirinkti tokie algoritmų darbo efektyvumo įvertinimo kriterijai (matai): a) vidutinis nuokrypis (nuo geriausio žinomo sprendinio) – $\bar{\delta}$ ($\bar{\delta} = 100(\bar{z} - z^{\sim})/z^{\sim}$ [%], čia $\bar{z}$ yra rastų tikslo funkcijos reikšmių z^* vidurkis, apskaičiuotas, atlikus W kartotinių vykdymų, o $z^{\sim}$ yra geriausia žinoma (tikslo funkcijos) reikšmė (GŽR)), (GŽR pateiktos [9] straipsnyje); b) sprendinių, esančių „1% optimalumo intervale“, skaičius – $C_{1\%}$; c) surastų geriausių žinomų sprendinių skaičius – $C_{ger.}$

Visiems trims lygintiems algoritmams sudarytos panašios sąlygos – identiški pradiniai sprendiniai, tas pats restartų skaičius W . Algoritmų palyginimo rezultatai atsitiktiniams ir „realaus pasaulio“ duomenims pateikti 1 ir 2 lentelėse. Geriausios vidutinio nuokrypio reikšmės atspausdintos paryškintai.

1 lentelė. Tabu paieškos algoritmų palyginimo rezultatai atsitiktiniams duomenims

Testinis pavyzdys	*n*	GŽR	RETPA		RATPA		ITTPA	
			$\bar{\delta}$	$C_{1\%}/C_{ger}$	$\bar{\delta}$	$C_{1\%}/C_{ger}$	$\bar{\delta}$	$C_{1\%}/C_{ger}$
TAI020A	20	703482	0.592	10/1	0.639	9/1	**0.472**	10/2
TAI025A	25	1167256	0.537	10/1	0.753	8/1	**0.402**	10/3
TAI030A	30	1818746	0.454	10/1	0.480	9/1	**0.177**	10/4
TAI035A	35	2422002	0.523	10/0	0.812	8/0	**0.380**	10/1
TAI040A	40	3152140	0.505	10/0	0.655	10/0	**0.570**	10/0
TAI050A	50	4962658	0.716	10/0	1.086	4/0	**0.629**	10/0
TAI060A	60	7228240	0.636	10/0	1.042	3/0	**0.570**	10/0
TAI080A	80	13563602	0.302	10/0	0.763	10/0	**0.220**	9/0
TAI100A	100	21115211	**0.194**	10/0	0.731	10/0	0.257	9/1

2 lentelė. Algoritmų palyginimo rezultatai „realaus pasaulio duomenims“

Testinis pavyzdys	n	GŽR	RETPA		RATPA		ITTPA	
			$\overline{\delta}$	$C_{1\%}/C_{ger}$	$\overline{\delta}$	$C_{1\%}/C_{ger}$	$\overline{\delta}$	$C_{1\%}/C_{ger}$
TAI020B	20	122455319	4.158	6/6	2.480	5/4	**0**	10/10
TAI025B	25	344355646	7.164	2/0	2.822	6/2	**0.007**	10/9
TAI030B	30	637808503	2.066	5/0	1.501	3/1	**0.026**	10/2
TAI035B	35	283315445	1.634	5/0	0.917	5/1	**0.107**	10/3
TAI040B	40	637250948	1.394	4/0	1.018	6/5	**0**	10/10
TAI050B	50	458821517	0.512	9/0	0.321	9/0	**0.176**	10/2
TAI060B	60	608215054	0.373	10/0	0.308	10/0	**0.057**	10/2
TAI080B	80	818561760	0.675	6/0	0.515	9/0	**0.053**	10/0
TAI100B	100	1185996137	**0.187**	10/0	0.294	10/0	0.243	9/4

Matyti, kad gauti algoritmų rezultatai priklauso nuo testinių duomenų prigimties. Esant tam pačiam vykdymo laikui, “realaus pasaulio” duomenims gaunami geresni rezultatai negu atsitiktiniams duomenims. Tačiau abiem atvejais rezultatai liudija, jog tirtų efektyvumo kriterijų atžvilgiu algoritmas ITTPA pranoksta tiek algoritmą RATPA, tiek algoritmą RETPA (išimtį sudaro vienas testinis pavyzdys, kuriam ITTPA nežymiai nusileidžia RETPA, bet aiškiai pralenkia RATPA). Labai akivaizdus ITTPA pranašumas prieš kitus du algoritmus (ypač RETPA) yra “realaus pasaulio” duomenims; tuo tarpu didelei daliai atsitiktinių duomenų ITTPA atotrūkis nuo RETPA nėra labai ryškus, bet šiuo atveju ITTPA demonstruoja žymiai geresnius rezultatus negu RATPA – tai ypač matyti didesnės apimties testiniams pavyzdžiams.

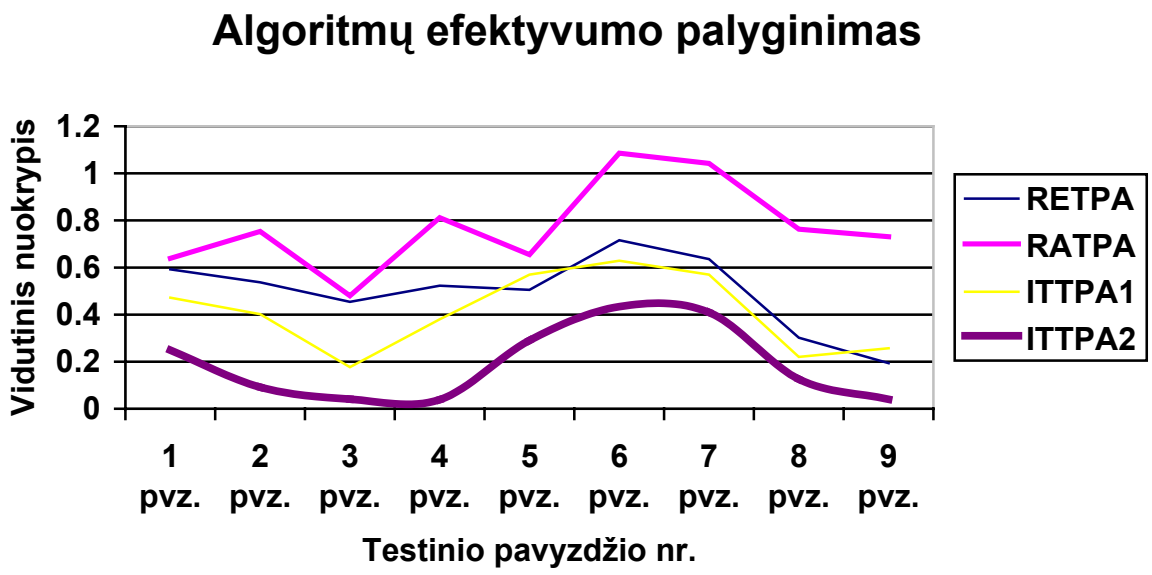
Algoritmo ITTPA pranašumas ypač išryškėja padidinus iteracijų skaičių reguliuojančių parametrų Q_1 ir Q_2 reikšmes. Žinoma, tuomet algoritmo veikimo laikas padidėja. 3 ir 4 lentelėse pateikiami algoritmo ITTPA paieškos rezultatai pakeitus parametrų Q_1 ir Q_2 reikšmes.

3 lentelė. Algoritmų palyginimas

Testinis pavyzdys	N	GŽR	$ITTPA_1$			$ITTPA_2$		
			$\overline{\delta}$	$C_{1\%}$	t	$\overline{\delta}$	$C_{1\%}$	t
TAI020A	20	703482	0.472	10	5	**0.249**	10	18
TAI025A	25	1167256	0.402	10	21	**0.092**	10	01:06
TAI030A	30	1818746	0.177	10	43	**0.041**	10	03:05
TAI035A	35	2422002	0.380	10	01:53	**0.039**	10	08:13
TAI040A	40	3152140	0.570	10	04:53	**0.290**	10	18:01
TAI050A	50	4962658	0.629	10	18:04	**0.434**	10	01:09:30
TAI060A	60	7228240	0.570	10	56:44	**0.409**	10	03:3554
TAI080A	80	13563602	0.220	9	02:16:58	**0.126**	**10**	10:01:48
TAI100A	100	21115211	0.257	9	04:15:42	**0.040**	**10**	22:36:41

4 lentelė. Algoritmų palyginimas

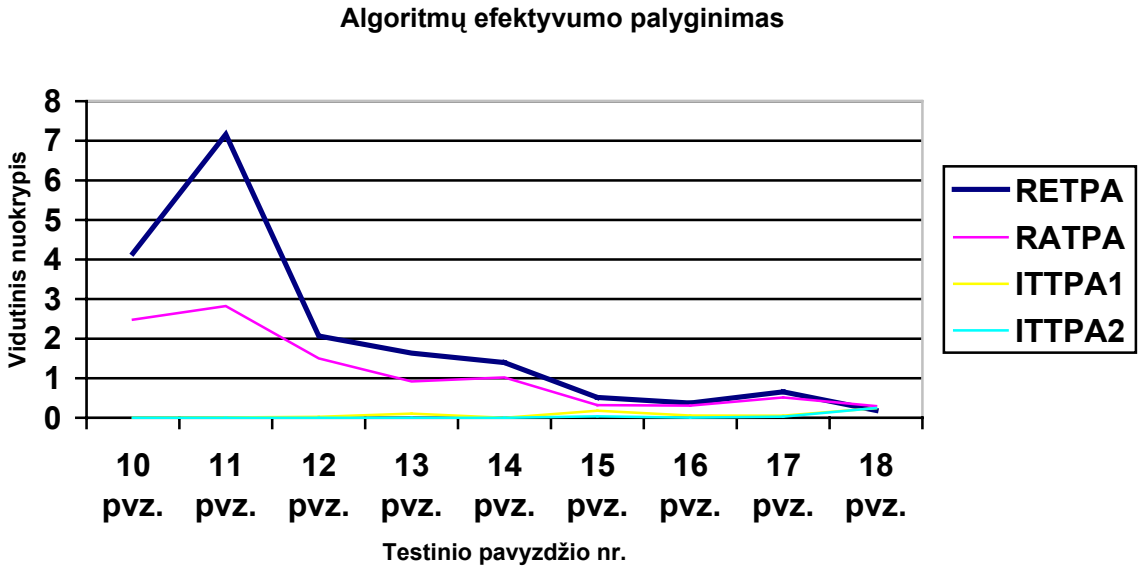
Testinis pavyzdys	*N*	GŽR	ITTPA$_1$			ITTPA$_2$		
			$\overline{\delta}$	$C_{1\%}$	*t*	$\overline{\delta}$	$C_{1\%}$	*t*
TAI020B	20	122455319	0	10	4	0	10	13
TAI025B	25	344355646	0.007	10	10	**0**	10	37
TAI030B	30	637117113	0.026	10	24	**0.001**	10	01:35
TAI035B	35	283315445	0.107	10	52	**0**	10	03:16
TAI040B	40	637250498	0	10	01:41	0	10	06:20
TAI050B	50	458821517	0.176	10	05:15	**0.036**	10	20:31
TAI060B	60	608215054	0.570	10	13:42	**0.004**	10	53:05
TAI080B	80	818561760	0.053	10	44:06	**0.029**	**10**	02:46:20
TAI100B	100	1185996137	**0.243**	9	01:44:19	0.245	**9**	06:45:46



8 pav. Algoritmų efektyvumo palyginimas

7 lentelė. Testinio pavyzdžio nr.

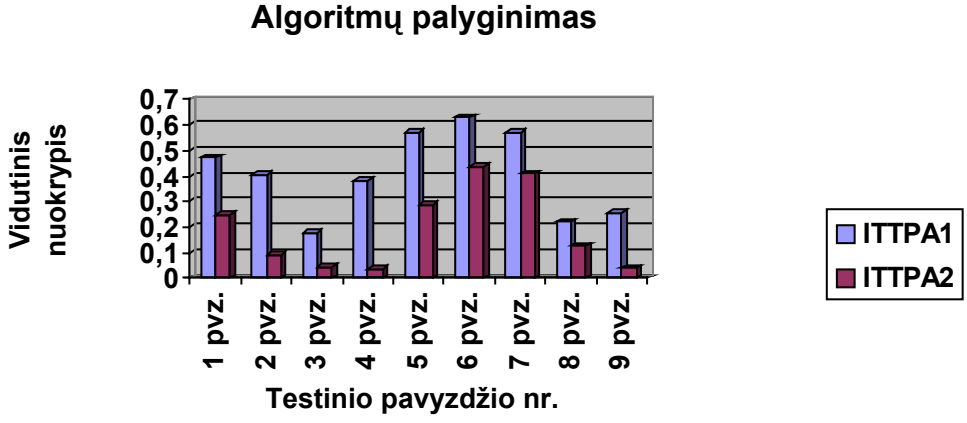
Nr.	Testinis pavyzdys	Nr.	Testinis pavyzdys	Nr.	Testinis pavyzdys
1	TAI020A	4	TAI035A	7	TAI060A
2	TAI025A	5	TAI040A	8	TAI080A
3	TAI030A	6	TAI050A	9	TAI100A



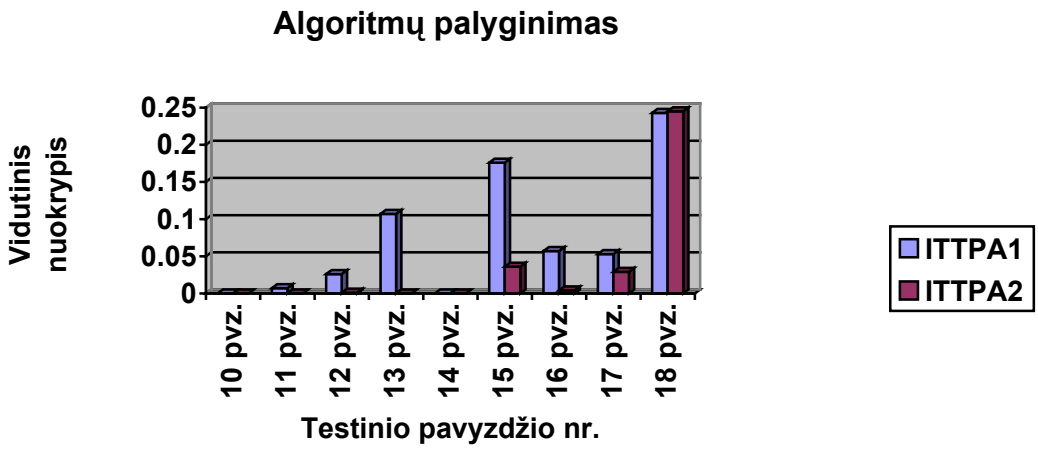
9 pav. Testinių pavyzdžių vidutinio nuokrypio palyginimas

8 lentelė. Testinio pavyzdžio nr.

Nr.	Testinis pavyzdys	Nr.	Testinis pavyzdys	Nr.	Testinis pavyzdys
10	TAI020B	13	TAI035B	16	TAI060B
11	TAI025B	14	TAI040B	17	TAI080B
12	TAI030B	15	TAI050B	18	TAI100B



10 pav. Algoritmų palyginimas



11 pav. Algoritmų palyginimas

Kaip matyti iš pateiktų lentelių ir 8, 9, 10, 11 paveikslų, algoritmo ITTPA rezultatai (t.y., vidutinio tikslo funkcijos reikšmių nuokrypis nuo (pseudo)optimalios reikšmės) yra įtakojami nuo algoritmo parametrų (Q_1, Q_2) konfigūracijos: parenkant tinkamesnes šių parametrų reikšmes, galima daugiau ar mažiau pagerinti rezultatų kokybę.

Kita vertus, algoritmo ITTPA rezultatų kokybę galima pagerinti padidėjusios skaičiavimų apimties kaina, būtent padidinant iteracijų (arba bandymų) skaičių.

Pastebėsime, kad didindami kartotinių vykdymų skaičių, kai iteracijų (bandymų) skaičius išlieka nepakitęs, mes galime tiktai tikėtis, jog didės skaičius (bet ne procentas) sprendinių, kurie patenka į „1% optimalumo intervalą“.

Kitas būdas gauti geresnės kokybės rezultatus – tai parinkti tinkamesnes, tikslesnes valdymo parametrų $h_{min,}$ $h_{max,}$ μ_{min}, $\mu_{max,}$ r ir ω reikšmes. Tam reikalingi papildomi išsamesni eksperimentai, ir tai galėtų būti kitų, naujų tyrimų objektas.

7. Išvados

I. Kvadratinio paskirstymo uždavinys, kaip ir daugelis kitų kombinatorinio optimizavimo uždavinių, dėl jo sudėtingumo išsprendžiamas tiksliai tik tuomet, kai jo apimtis labai ribota. Todėl šiam uždaviniui spręsti taikytini euristiniai algoritmai. Vienas iš tokių algoritmų – iteratyviosios tabu paieškos algoritmas ITTPA – pateiktas šiame darbe.

II. Pagrindinė naujojo tabu paieškos algoritmo idėja yra ta, kad yra siekiama pagerinti įprastinę tabu paieškos veikimo schemą panaudojant tam tikrus papildomus sprendinių pertvarkymus ir išplečiant tradicinę tabu paiešką į vadinamąją iteratyviąją tabu paiešką

III. Iteratyvioji tabu paieška pasižymi tuo, jog čia įprasta tabu paieška kombinuojama su lokaliai optimalių sprendinių rekonstravimu (mutavimu). Tam tikru mastu transformuojant lokaliai optimalius sprendinius - jų perdaug "nesugriaunant" – ir kartojant tabu paiešką nuo tų rekonstruotų sprendinių siekiama aprėpti kiek įmanoma didesnį sprendinių aibės "poerdvį", taip sudarant palankesnes sąlygas artėti prie globaliai optimalių sprendinių.

IV. Kaip parodė eksperimentų, atliktų su kvadratinio paskirstymo uždaviniu, rezultatai, tokia išplėtotos tabu paieškos strategija iš tikrųjų įgalina pasiekti sprendinius, mažiau nutolusius nuo globaliojo optimumo, lyginant su įprasta "grynąja" tabu paieška.

V. Atliktų eksperimentų rezultatai paliudijo, kad naujasis išplėstinės tabu paieškos algoritmas kvadratinio paskirstymo uždaviniui laikytinas vienu iš pačių efektyviausių euristinių algoritmų atsitiktiniu būdu generuojamiems duomenims. Kalbant apie "realaus pasaulio" duomenis, tai ir šiuo atveju galima sukurti išties gerą metodą. Šiuo atveju algoritmą ITTPA (jį atitinkamai adaptuojant) galima būtų "hibridizuoti" su genetiniais algoritmais, pvz., panaudoti jį aukštos kokybės ("elitinių") sprendinių "populiacijų" generavimu ir pan. Be to, ir pats išplėstinės tabu paieškos algoritmas gali būti toliau modifikuojamas bei tobulinamas.

VI. Naujasis algoritmas ITTPA išbandytas su įvairių tipų kvadratinio paskirstymo uždavinio testiniais pavyzdžiais (duomenimis) iš KP uždavinio duomenų bibliotekos QAPLIB.

VII. Atliktų eksperimentų rezultatai liudija, jog pasiūlytas algoritmas ITTPA yra labai efektyvus, daugeliu atvejų pranoksta ankstesnius tabu paieškos algoritmus didelei daliai testinių pavyzdžių iš bibliotekos QAPLIB. Be to, nežiūrint į tai, kad yra efektyvus, šis algoritmas yra lengvai programuojamas bei realizuojamas praktikoje.

Literatūros sąrašas

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

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

[3] **Ahuja R. K., Orlin J. B., Tiwari A.,** 2000. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research* 27, 917–934.

[4] **Battiti R., Tecchiolli G.** The reactive tabu search// *ORSA Journal on Computing*, 1994, *Vol*.6, p. 126–140.

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

[6] **Burkard R. E.** Locations with spatial interactions: the quadratic assignment problem. In P.B.Mirchandani, R.L.Francis (eds), *Discrete Location Theory*, Wiley, New York, 1991, p. 387–437.

[7] **Burkard R. E.** Quadratic assignment problems// *European Journal of Operational Research*, 1984, p. 15, 283–289.

[8] **Burkard R. E., Çela E., Pardalos P. M., Pitsoulis L.** The quadratic assignment problem. In D.Z.Du, P.M.Pardalos (eds), *Handbook of Combinatorial Optimization*, Kluwer, 1998, *Vol*. 3, p. 241–337.

[9] **Burkard R. E., Karisch S., Rendl F.** QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, p. 10, 391–403.

[10] **Burkard R. E., Rendl F.** A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 1984, p. 17, 169–174.

[11] **Çela E.** *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer, 1998.

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

[13] **Chakrapani J., Skorin-Kapov J.** Massively parallel tabu search for the quadratic assignment problem// *Annals of Operations Research*, 1993, *Vol*.41, p. 327–341.

[14] **Cohoon J. P., Paris W. D.,** 1987. Genetic placement. *IEEE Transactions on Computer-Aided Design* CAD-6, 956–964.

[15] **Connolly D. T.** An improved annealing scheme for the QAP. *European Journal of Operational Research*, 1990, p. 46, 93–100.

[16] **Davis L.,** 1991. *Handbook of Genetic Algorithms*, Van Nostrand, New York.

[17] **Dickey J. W., Hopkins J. W.** Campus building arrangement using TOPAZ. *Transportation Research*, 1972, *Vol*.6, p. 59–68.

[18] **Elshafei A. N.** Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, 1977, *Vol*.28, p. 167–179.

[19] **Finke G., Burkard R. E., Rendl F.** Quadratic assignment problems. *Annals of Discrete Mathematics*, 1987, *Vol*.31, p. 61–82.

[20] **Fleurent C., Ferland J.A.,** 1994. Genetic hybrids for the quadratic assignment problem. In: Pardalos, P.M., Wolkowicz, H., (Eds.), *Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol.16, AMS, Providence, pp. 173–188.

[21] **Frieze A. M., Yadegar J., El-Horbaty S., Parkinson D.** Algorithms for assignment problems on an array processor. *Parallel Computing*, 1989, *Vol*.11, p. 151–162.

[22] **Glover F.** Tabu search: part I. *ORSA Journal on Computing*, 1989, *Vol.*1, p. 190–206.

[23] **Glover F.** Tabu search: part II. *ORSA Journal on Computing*, 1990, *Vol.*2, p. 4–32.

[24] **Glover F., Laguna M.** *Tabu search*. Kluwer, 1997.

[25] **Goldberg D. E.,** 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

[26] **Hahn P. M., Hightower W. L., Johnson T. A., Guignard-Spielberg M., Roucairol C.** Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. Submitted for publication, 1999.

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

[28] **Hanan M., Kurtzberg J. M.** Placement techniques. M.A.Breuer (red.), *Design Automation of Digital Systems: Theory and Techniques*, 1972, 1, p. 213–282.

[29] **Hansen P., Jaumard B.** Algorithms for the maximum satisfiability problem. RUTCOR search Report 43–87, Rutgers University, 1987.

[30] **Hertz A., Taillard E., de Werra D.** Tabu search. In E.Aarts, J.K.Lenstra (eds), *Local Search in Combinatorial Optimization*. Wiley, Chichester, 1997, p. 121–136.

[31] **Holland J.H.,** 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[32] **Hu T. C., Kuh (ed.) E. S.** *VLSI Circuit Layout: Theory and Design*, IEEE Press, New York, 1985.

[33] **Kelly J. P., Laguna M., Glover F.** A study of diversification strategies for the quadratic assignment problem// *Computers and Operations Research*, 1994, *Vol.*21, p. 885–893.

[34] **Kirkpatrick S., Gelatt C. D., Vecchi M. P.** Optimization by simulated annealing. *Science*, 1983, p. 220, 671–680.

[35] **Koopmans T., Beckmann M.** Assignment problems and the location of economic activities. *Econometrica*, 1957, *Vol.*25, p. 53–76.

[36] **Krarup J., Pruzan P. M.** Computer-aided layout design. *Mathematical Programming Study*, 1978, *Vol.*9, p. 75–94.

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

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

[39] **Malucelli F.** Quadratic assignment problems: solution methods and applications. PhD Thesis, University of Pisa, Italy, 1993.

[40] **Maniezzo V., Colorni A., Dorigo M.** The ant system applied to the quadratic assignment problem. Tech. Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.

[41] **Merz P., Freisleben B.,** 1997a. A genetic local search approach to the quadratic assignment problem. *Proceedings of the Seventh International Conference on Genetic Algorithms* (*ICGA'97*), Morgan Kaufmann, 465–472.

[42] **Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E.** Equation of state calculation by fast computing machines// *Journal of Chemical Physics*, 1953, p. 21, 1087–1092.

[43] **Mühlenbein H.,** 1991. Evolution in time and space – the parallel genetic algorithm. In: Rawlins, G.J.E, (Ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, 316–337.

[44] **Mühlenbein H.,** 1997. Genetic algorithms. In: Aarts, E., Lenstra, J.K., (Eds.), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 137–171.

[45] **Nugent C. E., Vollmann T. E., Ruml J.** An experimental comparison of techniques for the assignment of facilities to locations// *Journal of Operations Research*, 1969, *Vol.*16, p. 150–173.

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

[47] **Reeves C.R.,** 1995. Genetic algorithms and combinatorial optimisation. In: Rayward-Smith, V.J. (Ed.), *Applications of Modern Heuristic Methods*, Alfred Waller, Henley-on-Thames, UK, pp. 111–126.

[48] **Sahni S., Gonzalez T.** P-complete approximation problems// *Journal of ACM*, 1976, p. 23, 555–565.

[49] **Skorin-Kapov J.** Tabu search applied to the quadratic assignment problem// *ORSA Journal on Computing*, 1990, p. 2, 33–45.

[50] **Skorin-Kapov J.** Extension of a tabu search adaptation to the quadratic assignment problem// *Computers and Operations Research*, 1994, *Vol*.21, p. 855–865.

[51] **Sondergeld L., Voss S.** A star-shaped diversification approach in tabu search. In I.H.Osman, J.P.Kelly (eds), *Meta-Heuristics: Theory and Applications*. Kluwer, 1996, p. 489–502.

[52] **Steinberg L.** The backboard wiring problem: a placement algorithm// *SIAM Review*, 1961, p. 3, 37–50.

[53] **Stuetzle T., Dorigo M.** ACO algorithms for the quadratic assignment problem. In D.Corne et al. (eds), *New Ideas in Optimization*. McGraw-Hill, 1999.

[54] **Taillard E.** Robust taboo search for the QAP// *Parallel Computing*, 1991, p. 17, 443–455.

[55] **Taillard E., Gambardella L. M.** Adaptive memories for the quadratic assignment problem. Tech. Report IDSIA-87-97, Lugano, Switzerland, 1997.

[56] **Tate D. M., Smith A. E.,** A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, 1995, *Vol*.1, 73–83.

[57] **Wilhelm M., Ward T.** Solving quadratic assignment problems by simulated annealing// *IIE Transactions*, 1987, p. 19, 107–119.

A TABU SEARCH ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

Summary

Tabu search based algorithms are among the widely used heuristic algorithms for combinatorial optimization problems. Here we propose an improved enhanced tabu search algorithm for the well-known combinatorial optimization problem, the quadratic assigment problem (QAP). The new algorithm was tested on a number of instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed algorithm appears to be superior to the earlier "pure" tabu search algorithms on many instances of the QAP.

Priedas A

Straipsnis, atspausdintas Kauno tecnologijos universiteto išleistame leidinyje "Informacinės technologijos 2004, konferencijos pranešimų medžiaga" Kaunas, Technologija, 2004, p. 48 – 55.

TABU PAIEŠKOS ALGORITMAS KVADRATINIO PASKIRSTYMO UŽDAVINIUI

Audrius Gedgaudas, Alfonsas Misevičius

Kauno technologijos universitetas, Praktinės informatikos katedra,
Studentų g. 50–400a/416a, LT–3031 Kaunas
Tel. 8 689 86851, 8 37 300372
El. paštas: audriusg@dokeda.lt, alfonsas.misevicius@ktu.lt

Santrauka

Vieni iš plačiausiai taikomų euristinių algoritmų kombinatorinio optimizavimo uždaviniams spręsti yra tabu paieška (TP) (angl. *tabu search*) besiremiantys algoritmai. Šiame straipsnyje pasiūlytas patobulintas išplėstinės tabu paieškos algoritmas gerai žinomam kombinatorinio optimizavimo uždaviniui, būtent, kvadratinio paskirstymo (KP) uždaviniui. Naujasis TP algoritmas išbandytas panaudojant KP uždavinio testinius pavyzdžius iš testinių pavyzdžių bibliotekos – QAPLIB. Gauti eksperimentų rezultatai liudija, jog siūlomas algoritmas yra pranašesnis už ankstesnius tabu paieškos algoritmus didelei daliai KP uždavinio testinių pavyzdžių.

1. Įvadas

Kvadratinio paskirstymo uždavinys formuluojamas taip: duota matricos $\boldsymbol{A}=(a_{ij})_{n\times n}$ ir $\boldsymbol{B}=(b_{kl})_{n\times n}$ bei aibė Π, kurią sudaro visi galimi perstatymai iš natūrinių skaičių nuo 1 iki n; reikia tarp perstatymų iš aibės Π surasti tokį perstatymą $\pi=(\pi(1), \pi(2), ..., \pi(n))$, kuriam esant būtų minimizuota funkcija

$$z(\pi)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{\pi(i)\pi(j)}\ , \tag{1}$$

čia z atlieka vadinamosios tikslo funkcijos, o π – sprendinio vaidmenį; n yra uždavinio apimtis.

Viena iš KP uždavinio taikymo sričių yra elektroninės aparatūros projektavimas, pvz., elektroninių komponentų išdėstymas mikroschemose ar spausdinto montažo plokštėse [10,14]. Kitos KP uždavinio taikymo sritys aprašytos [2,4] knygose.

Kvadratinio paskirstymo uždavinys priklauso NP-sunkių kombinatorinio optimizavimo uždavinių klasei [18]. Tokie uždaviniai išsprendžiami tiksliai tik esant labai nedidelėms jų apimtims ($n\leq 36$). Todėl vidutinės ir didelės apimties KP uždaviniams spręsti naudojami euristiniai algoritmai, tame tarpe, tabu paieškos algoritmai [1,5,15,19,20,21] (išsamų kitų euristinių algoritmų KP uždaviniui sąrašą galima rasti [2,4] knygose).

Aibės Π perstatymams (sprendiniams) galima sukonstruoti vadinamąją aplinkos funkciją N: $\Pi\rightarrow 2^{\Pi}$. Šios funkcijos pagalba bet kuriam perstatymui $\pi\in\Pi$ priskiriama aibė $N(\pi)\subseteq\Pi$ – perstatymo π aplinka („aplinkinių“ perstatymų aibė). Perstatymo π aplinkos $N(\pi)$ atskirą perstatymą π' galima gauti, atlikus atitinkamą perstatymo π transformaciją (perturbaciją) ϕ: $\Pi\rightarrow\Pi$ – šiuo atveju sakoma, jog atliekamas „perėjimas“ iš π į π'. Vienas iš konkrečių aplinkos funkcijų pavyzdžių yra vadinamoji porinių sukeitimų funkcija (žymima N_2), kuri apibrėžiama taip: $N_2(\pi)=\{\pi' \mid \pi'\in\Pi, \rho(\pi,\pi')=2\}$, čia $\rho(\pi,\pi')=\sum_{i=1}^{n}\operatorname{sgn}|\pi(i)-\pi'(i)|$. Esant aplinkai N_2 „perėjimus“ lengva sumodeliuoti elementarių perturbacijos operatorių ϕ_{ij} ($i,j\in\{1, 2, ..., n\}$), pagalba – čia ϕ_{ij} tiesiog atlieka dviejų perstatymo elementų (i-tojo ir j-tojo) sukeitimą vietomis. („Perėjimą“ iš π į π' panaudojant perturbaciją ϕ_{ij} formaliai galime užrašyti taip: $\pi'=\pi\oplus\phi_{ij}$.) Perstatymo aplinka N_2 gali būti pilnai išnagrinėta (patikrinta), duotam perstatymui nuosekliai atliekant perturbacijas ϕ_{ij}, kai i kinta nuo 1 iki n–1, o j – nuo i+1 iki n (t.y., ϕ_{12}, ϕ_{13}, ..., ϕ_{1n}, ϕ_{23}, ..., $\phi_{n-1,n}$).

Vykdant perturbaciją ϕ_{ij}, tikslo funkcijos z pokytis $\Delta z(\pi,i,j)$ ($\Delta z(\pi,i,j)= z(\pi \oplus \phi_{ij}) - z(\pi)$ ($1 \le i,j \le n$, $i \ne j$)) apskaičiuojamas panaudojant $O(n)$ operacijų:

$$\Delta z(\pi,i,j) = (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}) + \sum_{k=1,k\ne i,j}^{n} \left[(a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) + (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)})\right], \quad (2)$$

čia a_{ii}(arba b_{ii}) yra konstanta (paprastai 0) visiems i nuo 1iki n. Tikslo funkcijos pokytį dviems „perstatymams-kaimynams“ π ir $\pi' = \pi \oplus \phi_{uv}$ galima apskaičiuoti ir žymiau greičiau – su sąlyga, jog yra įsimintos (išsaugotos) visos ankstesnių pokyčių reikšmės $\Delta z(\pi,i,j)$ (i,j=1, 2, ..., n). Pokyčių saugojimui pakanka dvimatės matricos $\boldsymbol{D}=(d_{ij})_{n\times n}$. Tuomet, apskaičiuojant pokytį $\Delta z(\pi',i,j)$ ($i \ne u,v$ ir $j \ne u,v$), pakanka $O(1)$ operacijų [21]:

$$\Delta z(\pi',i,j) = \Delta z(\pi,i,j) + (a_{iu} - a_{iv} + a_{jv} - a_{ju})(b_{\pi(i)\pi(u)} - b_{\pi(i)\pi(v)} + b_{\pi(j)\pi(v)} - b_{\pi(j)\pi(u)}) + (a_{ui} - a_{vi} + a_{vj} - a_{uj})(b_{\pi(u)\pi(i)} - b_{\pi(v)\pi(i)} + b_{\pi(v)\pi(j)} - b_{\pi(u)\pi(j)}) . \quad (3)$$

(Pastaba. Jeigu $i=u(v)$ arba $j=u(v)$, tai turi būti taikoma (2) formulė; nors pastarosios formulės sudėtingumas yra $O(n)$, tačiau ji taikoma tik keturioms indeksų poroms, o tai įgalina esamo perstatymo π aplinką $N_2(\pi)$ pilnai išnagrinėti, panaudojant viso tik $O(n^2)$ operacijų – išimtį sudaro inicializacijos fazė, t.y., pirmoji iteracija, kurios metu vistiek reikia (bet tik vieną kartą) atlikti $O(n^3)$ operacijų tam, kad išnagrinėti pradinio sprendinio aplinką ir įsiminti visus galimus tikslo funkcijos pokyčius, reikalingus tolesnėse iteracijose.)

Toliau šiame straipsnyje aprašomas naujas pasiūlytas, išplėstinės tabu paieškos algoritmas KP uždaviniui. Prieš tai, 2 skyrelyje bendrais bruožais aptariama tabu paieška, jos veikimo principas, paradigma. Naujasis algoritmas nagrinėjamas 3 skyrelyje. 4 skyrelyje pateikiami eksperimentinių tyrimų rezultatai, gauti sprendžiant KP uždavinio testinius pavyzdžius. Straipsnis baigiamas išvadomis.

2. Tabu paieška: veikimo principas, paradigma

Tabu paieškos metodas buvo pasiūlytas Hanseno, Jaumardo [11] ir Gloverio 1987–1990 m. [6,7]. Šis metodas yra tapęs labai populiarus ir plačiai taikytas įvairiems optimizavimo uždaviniams (žr., pvz., [9,13,17,23,24]), tarp jų, kaip jau minėta, ir KP uždaviniui (žr. 1 sk.).

TP metodas remiasi išplėsta lokaliąja paieška. Skirtingai, negu įprasti lokaliosios paieškos algoritmai, kurie apsiriboja lokaliai optimalaus sprendinio (lokaliojo optimumo) suradimu, tabu paieška pagrįsti algoritmai tęsia paiešką ir tuo atveju, kai surandamas lokaliai optimalus sprendinys, t.y., duoto sprendinio aplinkoje neįmanoma surasti geresnio sprendinio (sprendinio su mažesne – kai sprendžiamas minimizavimo uždavinys – tikslo funkcijos reikšme). Pagrindinė TP idėja yra ta, kad leidžiama atlikti „perėjimus“ net ir tais atvejais, kai naujasis sprendinys iš esamo sprendinio aplinkos yra blogesnis už duotąjį sprendinį. Tokiu būdu tabu paieška įgalina daug kartų „pereiti“ nuo vieno lokaliojo optimumo prie kito įsimenant geriausiąjį iš surastų lokaliai optimalių sprendinių. Tačiau kai kurie „perėjimai“ turi būti uždraudžiami, t.y., tabu paieška yra pagrįsta draudimų metodologija: draudimai yra būtini tam, kad neleisti grįžti į tuos pačius, jau nagrinėtus sprendinius ir taip išvengti ciklinimų.

Taigi, tarkime, kad duota diskretinių sprendinių aibė S, tikslo funkcija f: $S \to R^1$, o taip pat aplinkos funkcija N: $S \to 2^S$. Tuomet TP veikimas galėtų būti glaustai aprašytas taip♣: paieška pradedama nuo tam tikro pradinio, galbūt atsitiktinai sugeneruoto, sprendinio s iš aibės S, po to „perėjimai“ iš vieno sprendinio į kitą vykdomi iteraciniu būdu. Algoritmo vykdymo eigoje analizuojama einamojo sprendinio s aplinkos sprendinių aibė $N(s)$, ir „pereinama“ į tą sprendinį s' iš $N(s)$, kuriam tikslo funkcijos f reikšmė yra mažiausia (geriausia). „Perėjimas“ gali būti atliekamas ir tuo atveju, kai jis pablogina einamąją tikslo

♣ nuodugnesnio ir platesnio tabu paieškos metodo aprašymo rekomenduojama ieškoti [8,12] veikaluose.

funkcijos reikšmę (t.y., tikslo funkcijos pokytis yra teigiamas); būtent, pasirenkamas tas „perėjimas“, kuris mažiausiai pablogina tikslo funkcijos reikšmę. Grįžimas į neseniai „aplankytus“ sprendinius turi būti uždraudžiamas tam tikram periodui – kad būtų išvengta „ciklinimosi“ (paieškos kartojimo iš naujo nuo to pačio „taško“). Tokiu būdu kai kurie sprendiniai tam tikrais momentais tampa „tabu“: jie (arba atitinkamų „perėjimų pėdsakai“) įtraukiami į specialų sąrašą – vadinamąjį tabu sąrašą T, kuriame įsimenama $h=|T|$ paskutiniųjų „perėjimų pėdsakų” („žymių”). Taigi „perėjimas“ į sprendinį $s' \in N(s)$ yra draudžiamas, traktuojamas kaip tabu, jeigu tas sprendinys arba atitinkamas „perėjimas“ duotu metu yra sąraše T.

```
procedure tabu_paieška;
  (* pradiniai duomenys: s° – pradinis sprendinys; rezultatai: s* – geriausias rastas sprendinys *)
  s ← s°; s* ← s°;
  inicializuoti tabu sąrašą T;
  repeat (* vykdyti tabu paieškos ciklą *)
    rasti geriausią sprendinį s' ∈ N(s) duotam tabu sąrašui T ir pasirinktam aspiracijos kriterijui;
    s ← s'; (* pakeisti einamąjį sprendinį nauju *)
    įtraukti naująjį sprendinį s (arba „perėjimo“ iš s į s' „pėdsaką“) į tabu sąrašą T;
    if f(s) < f(s*) then s* ← s; (* įsiminti geriausią surastą sprendinį *)
    jei reikia, atlikti tabu sąrašo T atnaujinimą
  until patenkinta baigimo sąlyga
end (* procedure *)
```

1 pav. Tabu paieškos paradigma

Vistik po tam tikro laiko grįžimas į anksčiau nagrinėtus sprendinius gali būti naudingas. Dėl tos priežasties įvedamas vadinamasis aspiracijos kriterijus tam, kad leisti „tabu būsenai“ būti atšauktai esant palankioms aplinkybėms; pavyzdžiui, tabu „perėjimą“ iš s į s' galima leisti, jeigu $f(s')<f(s^*)$, kur s^* yra geriausias iki duoto momento paieškos eigoje surastas sprendinys.

TP procesas paprastai yra baigiamas, kai aliekamas iš anksto užduotas iteracijų skaičius. TP veikimo schemos šablonas – paradigma – pateikta 1 pav.

3. Išplėstinės tabu paieškos algoritmas KP uždaviniui

Šiame darbe pasiūlytas patobulintas tabu paieškos algoritmas, kuris pavadintas ITPA – išplėstinės tabu paieškos algoritmas. Pirmiausiai, jis remiasis kai kuriais vadinamosios randomizuotos tabu paieškos (RTP), aprašytos Taillard'o straipsnyje [21], principais. Kaip ir Taillard'o algoritme, mūsų algoritme tabu sąrašas realizuotas dvimatės matricos $\boldsymbol{T}=(t_{ij})_{n\times n}$, sudarytos iš sveikų skaičių, pagalba. Pradžioje visi matricos $\boldsymbol{T}$ elementai yra lygūs 0. Algoritmo vykdymo eigoje matricos elementas t_{ij} saugo einamosios iteracijos numerį plius tabu sąrašo koeficientas h ($h>0$); kitaip tariant, t_{ij} yra lygus numeriui iteracijos, kuria pradedant i-tajam ir j-tajam sprendinio elementams anuliuojama „tabu būsena“, t.y., tie elementai vėl gali būti sukeičiami vietomis. Taigi „perėjimas“ į sprendinį, sukeičiant elementus i ir j vietomis, yra „tabu“, jeigu $t_{ij}>=q$, kur q yra einamosios iteracijos numeris.

Labai svarbu parinkti prideramą koeficiento h reikšmę. Esant šiai reikšmei labai mažai, padidėja paieškos „ciklinimosi“ tikimybė; tuo tarpu, esant tai reikšmei pernelyg didelei, gali būti labai apribojama paieška atskirose sprendinių „erdvės“ srityse – o tarp šių gali būti ir tokių, kuriose „slypi“ labai geri lokaliai optimalūs sprendiniai (ar net globaliai optimalus sprendinys). Eksperimentais nustatyta, kad vienas iš perspektyvių būdų yra naudoti kintamą tabu sąrašo koeficientą, užuot naudojus jį iš anksto fiksuotą. Geri rezultatai gaunami, kai reikšmė h yra kaitaliojama randomizacijos keliu, t.y., ji parenkama atsitiktiniu būdu ir taip, kad patektų į tam tikrą užduotą intervalą, pvz., $[h_{min},h_{max}]$. Parametrų h_{min} ir h_{max} reikšmes galima tiesiog susieti su uždavinio apimtimi n, pvz., $h_{min}=0.1n$, $h_{max}=0.3n$. Reikšmės h

kaitaliojimo dažnumas prilyginamas $2h_{max}$, t.y., h perskaičiuojama kiekvieną kartą, kai tik įvykdoma $2h_{max}$ tabu paieškos iteracijų.

Algoritme ITPA panaudotas klasikinis aspiracijos kriterijus, t. y., „tabu statusas“ ignoruojamas, jeigu „perėjimas“ atliekamas į tokį sprendinį, kuris yra geresnis už bet kurį kitą iki tol rastą. Be to, algoritme pasinaudota ypač efektyvia formule tikslo funkcijos pokyčiams skaičiuoti (žr. (3) formulę įvade). Tai leidžia sprendinių aplinką išnagrinėti atliekant $O(n^2)$ operacijų.

Siekiant dar daugiau pagerinti TP efektyvumą išbandytas tabu paieškos ir „godžiąja“ strategija besiremiančios „greito nusileidimo“ procedūros kombinavimas. Idėja slypi tame, kad tam tikrais paieškos periodais tikslinga atlikti savotišką „relaksaciją“, t.y., tabu paiešką pakeisti „godžiąja“ paieška laikinai anuliuojant visus draudimus. Šiuo atveju „godžioji“ paieška gali būti traktuojama kaip paieškos intensifikavimas atskirose sprendinių „erdvės“ dalyse. Finišavus „greito nusileidimo“ procedūrai, atstatomi buvę draudimai ir vėl tęsiama įprasta TP procedūra. Taip kombinuojant paieškos režimus pavyksta pagerinti gaunamus rezultatus, jeigu lyginti juos su „grynosios“ TP rezultatais.

Be aukščiau apžvelgtų bruožų algoritmas ITPA pasižymi dar viena labai svarbia nauja savybe, būtent tuo, kad yra realizuota vadinamosios iteratyviosios tabu paieškos strategija. Šios strategijos esmė glūdi tame, jog galima pagerinti įprastų TP algoritmų rezultatus pritaikant tam tikrus sprendinių pertvarkymus (mutavimus). Iteratyviajai tabu paieškai būdingi du pagrindiniai etapai: 1) sprendinio pagerinimas (panaudojant įprastą tabu paiešką) ir 2) sprendinio rekonstravimas (panaudojant mutacijos procedūrą). Beje, galimi du rekonstravimo tipai: a) „karštasis“ rekonstravimas ir b) „šaltasis“ rekonstravimas.

Pirmojo tipo rekonstravimas paprastai atliekamas po kiekvieno sprendinio pagerinimo. „Karštojo“ rekonstravimo metu nesiekiama duotojo sprendinio visiškai „suardyti“, o siekiama, kad naujai gautas sprendinys „paveldėtų“ tam tikras geras ankstesnio sprendinio charakteristikas, nors tuo pačiu turi būti užtikrinamas ir tam tikras diversifikavimo (įvairovės) lygis. „Karštasis“ rekonstravimas neturi būti nei per daug „stiprus“, nei per daug „silpnas“. Pirmuoju atveju paieška taptų labai panaši į daugkartinę (gana chaotišką) tabu paiešką startuojant nuo visai atsitiktinių sprendinių, ir būtų prarandama naudinga paieškos eigoje sukaupta informacija; antruoju atveju rekonstravimas negarantuotų pakankamai „nutolusio“ (nuo duotojo) sprendinio generavimo – o tai galėtų sąlygoti paieškos ciklinimąsi, t.y., „sugrįžimą“ į jau anksčiau nagrinėtus sprendinius. „Karštąjį“ rekonstravimą galima realizuoti kaip atsitiktinį „perėjimą“ į to sprendinio aukštesnės eilės aplinką N_λ, kur $\lambda>2$. Vienas iš galimų mechanizmų tokiam „perėjimui“ modeliuoti – tai panaudoti atsitiktiniu būdu generuojamą elementarių perturbacijų (tokių, pvz., kaip ϕ_{ij}) seką, pvz., $\{\phi_{rs}\}_\mu$, kur r ir s gaunami (pseudo-)atsitiktinių skaičių generatoriaus pagalba, o μ yra sekos „ilgis“, pvz., tarp μ_{min} ir μ_{max}, čia μ_{min}, μ_{max} savo ruožtu yra tarp 2 ir $\lfloor n/2 \rfloor$.

Antrojo tipo rekonstravimo ypatybė yra ta, kad šiuo atveju galima sprendinį visiškai „sugriauti“. Tokio „sugriovimo“ tikslas – stengtis aprėpti kiek įmanoma didesnius, įvairesnius sprendinių „poerdvius“, taip sudarant palankesnes sąlygas artėti prie globaliojo optimumo. Paprasčiausiu atveju, „šaltąjį“ rekonstravimą galime realizuoti tiesiog kaip naujo, visai atsitiktinio sprendinio generavimą.

Algoritmo ITPA vykdymas valdomas panaudojant šiuos parametrus: Q_1 – išplėstinės tabu paieškos iteracijų skaičius; Q_2 – modifikuoto randomizuotos tabu paieškos iteracijų skaičius; h_{min}, h_{max} – tabu sąrašo ilgio rėžiai; μ_{min}, μ_{max} – atsitiktinių perturbacijų sekų ilgio (mutavimo lygio) rėžiai; r – „relaksacijos“ koeficientas (parametras, apibrėžiantis kokiu dažnumu vykdyti „greitą nusileidimą“); ω – parametras, nurodantis kokias periodais vykdyti „šaltąjį“ rekonstravimą. Išplėstinės tabu paieškos algoritmo šablonas pateikiamas 2 pav. Algoritmas ITPA savo ruožtu naudoja procedūras „MRTPA“ (modifikuotas randomizuotos tabu paieškos algoritmas), „mutacija“; jų šablonai parodyti atitinkamai 3 ir 4 pav. Pagaliau, procedūros

„greitas_nusileidimas“, kurią naudoja algoritmas MRTPA, šablonas yra 5 pav. Pastebėsime, kad algoritmas ITPA gali būti vykdomas daug kartų, esant įvairioms valdymo parametrų reikšmių kombinacijoms – tie atskiri algoritmo įvykdymai vadinami restartais.

```
procedure ITPA; (* išplėstinės tabu paieškos algoritmas KP uždaviniui *)
  (* pradiniai duomenys:  π̂ – pradinis sprendinys; n – uždavinio apimtis *)
  (*                      Q1, Q2, μmin, μmax, hmin, hmax, ω – valdantieji parametrai *)
  (* rezultatai: π* – geriausias rastas sprendinys *)
  π°:=MRTPA(π̂ ,Q2); (* optimizuojamas pradinis sprendinys panaudojant algoritmą MRTPA *)
  π•:=π°; π*:=π°; q•:=0; μ:=μmin–1;
  for q:=1 to Q1 do begin (* pagrindinis išplėstinės tabu paieškos ciklas *)
    if q–q•>ω then begin q•:=q; sugeneruoti naują atsitiktinį sprendinį π˜; π•:=π˜ end
              else begin
                if μ<μmax then μ:=μ+1 else μ:=μmin;
                π˜:=mutacija(π°,μ) (* sprendinio π° mutavimas (rekonstravimas) gaunant sprendinį π˜ *)
              end;
    π°:=MRTPA(π˜,Q2); (* optimizuojamas mutuotas sprendinys panaudojant algoritmą MRTPA *)
    if z(π°)<z(π•) then begin (* rastas geresnis lokaliai optimalus sprendinys *)
      q•:=q; μ:=μmin–1; π•:=π°; if z(π•)<z(π*) then π*:=π• (* įsimenamas geriausias sprendinys *)
    end (* if *)
  end (* for *)
end (* procedure *)
```

2 pav. Išplėstinės tabu paieškos, naudojančios randomizuotą tabu paiešką, algoritmo KP uždaviniui šablonas

```
function MRTPA(π,Q); (* modifikuotas randomizuotos tabu paieškos algoritmas KP uždaviniui *)
  (* pradiniai duomenys: π – einamasis sprendinys, n – uždavinio apimtis; Q – iteracijų skaičius; *)
  (*                     hmin, hmax – tabu sąrašo ilgio rėžiai (hmin<hmax); r – „relaksacijos“ koeficientas *)
  (* rezultatai: π• – geriausias rastas sprendinys *)
  π•:=π, q′:=1; i:=1; j:=1; T:=0;
  for q:=1 to Q do begin (* pagrindinis randomizuotos tabu paieškos ciklas *)
    for k:=1 to n–1 do for l:=k+1 to n do D[k,l]=Δz(π,k,l);
    if q mod 2hmax = 1 then parinkti atsitiktinę tabu sąrašo ilgio reikšmę h iš intervalo [hmin,hmax];
    Δmin:=∞;
    for m:=1 to |N2| do begin
      i := iif ( j < n, i, iif (i < n – 1, i + 1,1)) ;  j := iif ( j < n, j + 1, i + 1) ; Δ:=D[i,j];
      uždrausta:=(T[i,j]≥q); aspiracija:=((z(π)+Δ<z(π•)) and (uždrausta=TRUE));
      if ((Δ<Δmin) and (uždrausta=FALSE)) or (aspiracija=TRUE) then begin
        u:=i; v:=j; if aspiracija=TRUE then Δmin:=–∞ else Δmin:=Δ
      end (* if *)
    end; (* for *)
    π := π ⊕ φuv ; (* esamas sprendinys pakeičiamas nauju *)
    if D[u,v]<0 and (q–q′≥h*r) then begin π:=greitas_nusileidimas(π); q′:=q end;
    if z(π)<z(π•) then π•:=π, (* įsimenamas geriausias sprendinys *)
    T[u,v]:=q+h (* perturbacija („perėjimas“) puv įtraukiama į tabu sąrašą, t.y. „uždraudžiama“ *)
  end; (* pagrindinio ciklo pabaiga *)
  return π•
end (* function *)
```

3 pav. Modifikuoto randomizuotos tabu paieškos algoritmo KP uždaviniui šablonas.
Pastabos. 1. Apskaičiuojant tikslo funkcijos pokyčių matricos *D* reikšmes pirmą kartą, pasinaudojama (2) formule, o atnaujinant (perskaičiuojant) šias reikšmes – (3) formule.
2. Funkcija „iif(x,y_1,y_2)“ grąžina y_1, jei loginė sąlyga x yra patenkinta, arba y_2 priešingu atveju

```
function mutacija(π,μ); (* sprendinių mutavimo procedūra KP uždaviniui *)
  (* pradiniai duomenys: π – einamasis sprendinys; μ – mutavimo lygis (atsitikt. perturbacijų sekos ilgis) *)
  (* rezultatai: π – mutuotas sprendinys *)
  for i:=1 to μ do begin
    sugeneruoti atsitiktinius teigiamus sveikus skaičius γ, η, ir tokius, kad: 1≤γ,η≤n, γ≠η;
    π := π ⊕ φγη ; (* sukeisti γ-tąjį ir η-tąjį sprendinio (perstatymo) π elementus vietomis *)
  end; (* for *)
```

```
  return π
end (* function *)
```

4 pav. Sprendinių mutavimo algoritmo KP uždaviniui šablonas

```
function greitas_nusileidimas(π); (* „greito nusileidimo“ procedūra KP uždaviniui *)
  (* pradiniai duomenys: π – einamasis sprendinys; bendri duomenys: D – tikslo f. skirtumų matrica *)
  (* rezultatai: π – (galbūt) pagerintas sprendinys *)
  i:=1; j:=1;
  repeat (* pagrindinis ciklas *)
    Δmin:=∞;
    for k:=1 to |N2| do begin
      i := iif(j < n, i, iif(i < n−1, i+1,1)) ;  j := iif(j < n, j+1, i+1) ;
      if D[i,j]<Δmin then begin Δmin:=D[i,j]; u:=i; v:=j end
    end; (* for *)
    if Δmin<0 then begin
      π := π ⊕ φuv ; (* esamas sprendinys pakeičiamas nauju *)
      for l:=1 to n−1 do for m:=l+1 to n do perskaičiuoti D[l,m]
    end (* if *)
  until Δmin≥0;
  return π
end (* function *)
```

5 pav. „Greito nusileidimo“ algoritmo KP uždaviniui šablonas.
Pastaba. Perskaičiuojant tikslo funkcijos pokyčių matricos *D* reikšmes pasinaudojama (3) formule

4. Eksperimentiniai tyrimai ir jų rezultatai

Algoritmo ITPA efektyvumo tyrimams pasinaudota KP uždavinio testiniais pavyzdžiais (duomenimis) iš bibliotekos QAPLIB [3]. Tarp daugelio pavyzdžių išskirti atsitiktiniu būdu sugeneruoti pavyzdžiai ir „realaus pasaulio“ duomenis imituojantys pavyzdžiai. Pirmajai grupei priklauso šie konkretūs pavyzdžiai: tai25a, tai30a, tai35a, tai40a, tai50a, tai60a, tai80a ir tai100a (tai∗a), o antrajai – tokie pavyzdžiai: tai25b, tai30b, tai35b, tai40b, tai50b, tai60b, tai80b ir tai100b (tai∗b).

Algoritmo ITPA rezultatai buvo lyginami su kitų gerai žinomų, efektyvių TP algoritmų rezultatais. Tie algoritmai – tai (jau minėtas) Taillard'o algoritmas [21] (pavadintas RTPA – randomizuotos tabu paieškos algoritmas) bei Battiti ir Tecchiolli algoritmas [1] (pavadintas ReTPA – reaktyviosios tabu paieškos algoritmas). Beje, pastarasis algoritmas kai kuriems KP uždavinio testiniams pavyzdžiams yra pranašesnis už Taillard'o algoritmą, tačiau jį žymiai sunkiau suprogramuoti ir realizuoti. Tiek algoritmas RTPA, tiek algoritmas ReTPA dar iki šių dienų laikytini vienais iš efektyviausių euristinių algoritmų KP uždaviniui, ypač tai pasakytina apie atsitiktiniu būdu sugeneruotus testinius pavyzdžius [22]. (Tiesa, „realaus pasaulio“ pavyzdžių atveju daug žadančių rezultatų yra pasiekta panaudojant genetinius algoritmus [16], kurie čia detaliau nenagrinėjami.)

Buvo pasirinkti tokie algoritmų darbo efektyvumo įvertinimo kriterijai: a) vidutinis nuokrypis nuo geriausio žinomo sprendinio – $\bar{\delta}$ ($\bar{\delta} = 100(\bar{z} - z_{ger})/z_{ger}$ [%], čia $\bar{z}$ yra gautų tikslo funkcijos reikšmių vidurkis, apskaičiuotas, atlikus W algoritmo restartų, o z_{ger} yra geriausia žinoma (tikslo funkcijos) reikšmė (GŽR) (GŽR pateiktos [3] straipsnyje)); b) sprendinių, esančių „1% optimalumo intervale“ ($\bar{\delta} \le 1$), skaičius (kai atlikta W „restartų“) – $C_{1\%}$; c) surastų geriausių žinomų (pseudo-optimalių) sprendinių skaičius – C_{ger}.

Visiems trims lygintiems algoritmams sudarytos panašios sąlygos – identiški pradiniai sprendiniai, kiek tai įmanoma, vienodi vykdymo laikai bei tas pats restartų skaičius W (visuose bandymuose W=10). Algoritmų palyginimo rezultatai pateikti 1 ir 2 lentelėse. Geriausios vidutinio nuokrypio reikšmės atspausdintos paryškintai.

Matyti, kad gauti algoritmų rezultatai priklauso nuo testinių duomenų prigimties. Esant tam pačiam vykdymo laikui, „realaus pasaulio“ duomenims gaunami geresni rezultatai negu atsitiktiniams duomenims. Tačiau abiem atvejais rezultatai liudija, jog tirtų efektyvumo kriterijų atžvilgiu algoritmas ITPA pranoksta tiek algoritmą RTPA, tiek algoritmą ReTPA (išimtį sudaro vienas testinis pavyzdys, kuriam ITPA nežymiai nusileidžia ReTPA, užtat aiškiai pralenkia RTPA). Labai akivaizdus ITPA pranašumas prieš kitus du algoritmus (ypač ReTPA) yra „realaus pasaulio“ duomenims; tuo tarpu didelei daliai atsitiktinių duomenų ITPA atotrūkis nuo ReTPA nėra labai ryškus, bet šiuo atveju ITPA gauna žymiai geresnius rezultatus negu RTPA – tai ypač matyti didesnės apimties testiniams pavyzdžiams.

1 lentelė. Algoritmų palyginimo rezultatai ($\overline{\delta}$, $C_{1\%}$, C_{ger}) ir vykdymo laikas, esant atsitiktiniams duomenų rinkiniams. Geriausios gautos vidutinio nuokrypio reikšmės atspausdintos paryškintai. Nurodytas vykdymo laikas, kuris reikalingas vienam „restartui“ (naudotas 900 MHz taktinio dažnio personalinis kompiuteris).

Testinis pavyzdys	n	GŽR	$\overline{\delta}$, $C_{1\%}/C_{ger}$						Laikas (sek.)
			RTPA		ReTPA		ITPA[a]		
tai20a	20	703482	0.249	10/4	0.183	10/4	**0.061**	10/8	*0.7*
tai25a	25	1167256	0.242	10/4	0.298	10/3	**0.088**	10/7	*2.5*
tai30a	30	1818146	0.250	10/1	0.082	10/6	**0.018**	10/9	*7.3*
tai35a	35	2422002	0.268	10/1	0.201	10/5	**0.104**	10/6	*18.5*
tai40a	40	3139370	0.536	10/0	0.360	10/0	**0.229**	10/0	*45*
tai50a	50	4941410	0.789	8/0	0.585	10/0	**0.430**	10/0	*180*
tai60a	60	7208572	0.839	9/0	0.472	10/0	**0.395**	10/0	*600*
tai80a	80	13557864	0.625	10/0	0.208	10/0	**0.122**	10/1	*1800*
tai100a	100	21125314	0.655	10/0	**0.040**	10/2	0.069	10/2	*3600*

[a] ITPA panaudotos tokios valdymo parametrų reikšmės: $Q_1=\frac{3}{4}n^2$; $Q_2=\frac{1}{4}n^2$; $h_{min}=0.2n$; $h_{max}=0.4n$; $\mu_{min}=0.3n$; $\mu_{max}=0.4n$; $r=2.5$; $\omega=2n$.

2 lentelė. Algoritmų palyginimo rezultatai ($\overline{\delta}$, $C_{1\%}$, C_{ger}) ir vykdymo laikas, esant „realaus pasaulio“ duomenų rinkiniams. Geriausios gautos vidutinio nuokrypio reikšmės atspausdintos paryškintai. Nurodytas vykdymo laikas, kuris reikalingas vienam „restartui“ (naudotas 900 MHz taktinio dažnio personalinis kompiuteris).

Testinis pavyzdys	n	GŽR	$\overline{\delta}$, $C_{1\%}/C_{ger}$						Laikas (sek.)
			RTPA		ReTPA		ITPA[a]		
tai20b	20	122455319	0.045	10/9	0.211	9/8	**0**		*0.3*
tai25b	25	344355646	0.044	10/8	1.384	8/3	**0**		*1.0*
tai30b	30	637117113	0.112	10/3	0.754	7/1	**0.000**	10/8	*2.7*
tai35b	35	283315445	0.192	9/7	0.469	9/1	**0.021**	10/8	*5.7*
tai40b	40	637250948	0.003	10/7	0.309	9/4	**0**		*14.0*
tai50b	50	458821517	0.229	10/0	0.330	10/0	**0.071**	10/8	*50*
tai60b	60	608215054	0.160	10/1	0.123	10/0	**0.008**	10/5	*130*
tai80b	80	818415043	0.263	10/0	0.170	10/0	**0.011**	10/3	*370*
tai100b	100	1185996137	0.046	10/1	0.108	10/0	**0.019**	10/3	*1300*

[a] ITPA panaudotos tokios valdymo parametrų reikšmės: $Q_1=1.5n^2$; $Q_2=n$; $h_{min}=0.1n$; $h_{max}=0.3n$; $\mu_{min}=0.35n$; $\mu_{max}=0.45n$; $r=2.5$; $\omega=2n$.

Algoritmo ITPA rezultatus galima dar pagerinti, pirmiausiai, padidinant iteracijų skaičių reguliuojančių parametrų Q_1 ir Q_2 reikšmes (aišku, šiuo atveju atitinkamai padidėtų

paieškos laikas). Kitas būdas gauti geresnės kokybės rezultatus – tai parinkti tinkamesnes, tikslesnes valdymo parametrų h_{min}, h_{max}, μ_{min}, μ_{max}, r ir ω reikšmes. Tam reikalingi papildomi išsamesni eksperimentai, ir tai galėtų būti kitų, naujų tyrimų objektas.

5. Išvados

Šiame straipsnyje pasiūlytas patobulintas, išplėstinės tabu paieškos algoritmas (ITPA) vienam iš sudėtingų kombinatorinio optimizavimo uždavinių – kvadratinio paskirstymo uždaviniui. Žinoma, kad KP uždavinys yra NP-sunkus, ir efektyvių euristinių algoritmų šiam uždaviniui kūrimas vis dar tebelieka aktualia ir daugelio tyrinėtojų nagrinėjama tema. Algoritmas ITPA ir yra bandymas prisidėti tobulinant esamus ir konstruojant naujus, modernius algoritmus KP uždaviniui.

Pagrindinė naujojo tabu paieškos algoritmo idėja yra ta, kad yra siekiama pagerinti įprastinę tabu paieškos veikimo schemą panaudojant tam tikrus papildomus sprendinių pertvarkymus ir išplečiant tradicinę tabu paiešką į vadinamąją iteratyviąją tabu paiešką. Iteratyvioji TP pasižymi tuo, jog čia įprasta TP kombinuojama su lokaliai optimalių sprendinių rekonstravimu (mutavimu). Tam tikru mastu transformuojant lokaliai optimalius sprendinius – jų perdaug „nesugriaunant“ – ir kartojant tabu paiešką nuo tų rekonstruotų sprendinių siekiama aprėpti kiek įmanoma didesnį sprendinių aibės „poerdvį“, taip sudarant palankesnes sąlygas artėti prie globaliai optimalių sprendinių. Kaip parodė eksperimentų, atliktų su kvadratinio paskirstymo uždaviniu, rezultatai, tokia išplėtotos tabu paieškos strategija iš tikrųjų įgalino pasiekti sprendinius, mažiau nutolusius nuo globaliojo optimumo, jeigu lyginti su įprasta „grynąja“ tabu paieška.

Atliktų eksperimentų rezultatai paliudijo, kad naujasis išplėstinės TP algoritmas KP uždaviniui laikytinas vienu iš pačių efektyviausių euristinių algoritmų atsitiktiniu būdu generuojamiems duomenims. Kas dėl „realaus pasaulio“ duomenų, tai ir duotu atveju galima sukonstruoti išties gerą metodą. Šiuo atveju algoritmą ITPA (jį atitinkamai adaptuojant) galima būtų „hibridizuoti“ su genetiniais algoritmais, pvz., panaudoti jį aukštos kokybės („elitinių“) sprendinių „populiacijų“ generavimui ir pan. Be to, ir pats išplėstinės TP algoritmas gali būti toliau modifikuojamas bei tobulinamas.

Literatūros sąrašas

[1] **R.Battiti, G.Tecchiolli.** The reactive tabu search. *ORSA Journal on Computing*, 1994, t.6, 126–140.

[2] **R.E.Burkard, E.Çela, P.M.Pardalos, L.Pitsoulis.** The quadratic assignment problem. D.Z.Du, P.M.Pardalos (red.), *Handbook of Combinatorial Optimization*, Kluwer, 1998, t.3, 241–337.

[3] **R.E.Burkard, S.Karisch, F.Rendl.** QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 1997, t.10, 391–403.

[4] **E.Çela.** *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer, 1998.

[5] **J.Chakrapani, J.Skorin-Kapov.** Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 1993, t.41, 327–341.

[6] **F.Glover.** Tabu search: part I. *ORSA Journal on Computing*, 1989, t.1, 190–206.

[7] **F.Glover.** Tabu search: part II. *ORSA Journal on Computing*, 1990, t.2, 4–32.

[8] **F.Glover, M.Laguna.** *Tabu search*, Kluwer, 1997.

[9] **S.Hanafi, A.Freville.** An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 1998, t.106, 93–100.

[10] **M.Hanan, J.M.Kurtzberg.** Placement techniques. M.A.Breuer (red.), *Design Automation of Digital Systems: Theory and Techniques*, Prentice-Hall, 1972, t.1, 213–282.

[11] **P.Hansen, B.Jaumard.** Algorithms for the maximum satisfiability problem. RUTCOR Search Report 43–87, Rutgers University, 1987.

[12] **A.Hertz, E.Taillard, D. de Werra.** Tabu search. E.Aarts, J.K.Lenstra (red.), *Local Search in Combinatorial Optimization*, Wiley, 1997, 121–136.

[13] **A.Hertz, D. de Werra.** Using tabu search techniques for graph coloring. *Computing*, 1987, t.39, 345–351.

[14] **T.C.Hu, E.S.Kuh (red.).** *VLSI Circuit Layout: Theory and Design*, IEEE Press, New York, 1985.

[15] **J.P.Kelly, M.Laguna, F.Glover.** A study of diversification strategies for the quadratic assignment problem. *Computers & Operations Research*, 1994, t.21, 885–893.

[16] **A.Misevicius.** Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. *Knowledge-Based Systems*, 2003, t.16, 261–268.

[17] **E.Rolland, H.Pirkul, F.Glover.** Tabu search for graph partitioning. *Annals of Operations Reserch*, 1996, t.63, 209–232.

[18] **S.Sahni, T.Gonzalez.** P-complete approximation problems. *Journal of ACM*, 1976, t.23, 555–565.

[19] **J.Skorin-Kapov.** Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 1990, t.2, 33–45.

[20] **J.Skorin-Kapov.** Extension of a tabu search adaptation to the quadratic assignment problem. *Computers & Operations Research*, 1994, t.21, 855–865.

[21] **E.Taillard.** Robust taboo search for the QAP. *Parallel Computing*, 1991, t.17, 443–455.

[22] **E.Taillard.** Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 1995, t.3, 87–105.

[23] **P.Thomas, S.Salhi.** A tabu search heuristic for the resource constrained project scheduling problem. *Journal of Heuristics*, 1998, t.4, 123–139.

[24] **N.A.Wassan, I.H.Osman.** Tabu search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*, 2002, t.53, 768–782.

A TABU SEARCH ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

Summary

Tabu search based algorithms are among the widely used heuristic algorithms for combinatorial optimization problems. In this paper, we propose an improved enhanced tabu search algorithm for the well-known combinatorial optimization problem, the quadratic assignment problem (QAP). The new algorithm was tested on a number of instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed algorithm appears to be superior to the earlier "pure" tabu search algorithms on many instances of the QAP.