Euristinių algoritmų klasifikavimas

Alfonsas Misevičius

Kauno technologijos universiteto
Multimedijos inžinerijos katedros
tech. m. dr., docentas

Kaunas University of Technology,
Department of Multimedia Engineering,
Dr., Assoc. Prof.

Studentų g. 50-400a/416a, 51368 Kaunas
Tel. (8 37) 300 372
El. paštas: alfonsas.misevicius@ktu.lt

Vytautas Bukšnaitis

Kauno technologijos universiteto
Multimedijos inžinerijos katedros lektorius

Kaunas University of Technology,
Department of Multimedia Engineering,
Lecturer

Studentų g. 50-400, 51368 Kaunas
Tel. (8 37) 300 370
El. paštas: vytautas.buksnaitis@ktu.lt

Jonas Blonskis

Kauno technologijos universiteto
Multimedijos inžinerijos katedros
tech. m. dr., docentas

Kaunas University of Technology,
Department of Multimedia Engineering,
Dr., Assoc. Prof.

Studentų g. 50-400, 51368 Kaunas
Tel. (8 37) 300 370
El. paštas: jonas.blonskis@ktu.lt

Straipsnis skiriamas euristinių optimizavimo algoritmų, kurie jau kelis dešimtmečius traukia kompiuterių mokslo specialistų dėmesį, klasifikavimo klausimų aptarčiai. Jame apibrėžiami euristinių algoritmų tikslai, paskirtis, jų principiniai skiriamieji faktoriai, savybės. Apžvelgiamos svarbesnių euristinių optimizavimo algoritmų (tokių kaip atkaitinimo modeliavimas, tabu paieška, genetiniai algoritmai ir pan.) klasifikavimo schemos (metodikos). Nagrinėjamas universalios algoritmų sudedamųjų komponentų matricos – substancinių konceptų sistemos – naudojimas klasifikuojant euristinius algoritmus. Pabaigoje pateikiamos apibendrinamosios išvados.

Reikšminiai žodžiai: *algoritmai, algoritmų klasės, euristiniai ir metaeuristiniai algoritmai, algoritmų klasifikavimas.*

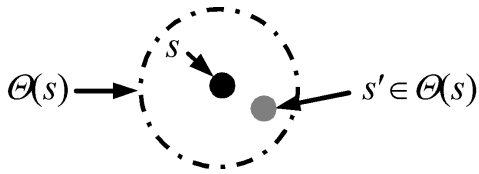
Viena svarbių informatikos krypčių yra optimizavimo algoritmai, metodai ir jų tyrimas. Didelę optimizavimo algoritmų klasę sudaro euristiniai algoritmai (EA) (angl. *heuristic algorithms*), kurie pastaraisiais metais yra labai išpopuliarėję ir tapę daugelio mokslininkų kolektyvų tyrimų objektu (Du, Pardalos, 1998; Floudas, Pardalos, 2001; Pardalos, Resende, 2002; Hoos, Stützle, 2004; Schneider, Kirkpatrick, 2006; Lee, El-Sharkawi, 2008; Siarry, Michalewicz, 2008; Yang, 2008; EU/ME European Chapter on Metaheuristics, 2009; Metaheuristics Network, 2009). Reikšmingas EA plėtotės stimulas yra intensyvėjanti tiek praktinė eksperimentinė, tiek mokslinė tiriamoji veikla kartu su šioje veikloje iškylančiais sudėtingais uždaviniais, kurie savo prigimtimi neretai yra kombinatorinio (diskretinio) pobūdžio. Formaliai tokius uždavinius galima aprašyti pora (S, f); čia S yra diskretinių sprendinių aibė, f – tikslo funkcija, kurios apibrėžimo sritis yra aibė S, o reikšmių aibė – realieji skaičiai. Tuomet išspręsti uždavinį (S, f) reiškia surasti sprendinį $s* \in S$ ir tokį, kad $s^* \in S^* = \left\{ s^\nabla \mid s^\nabla = \arg\min_{s \in S} f(s) \right\}$ (čia laikoma, kad tikslo funkcija turi būti minimizuojama; sprendinys $s*$ vadinamas (globaliai) optimaliu sprendiniu).

Euristiniai optimizavimo algoritmai

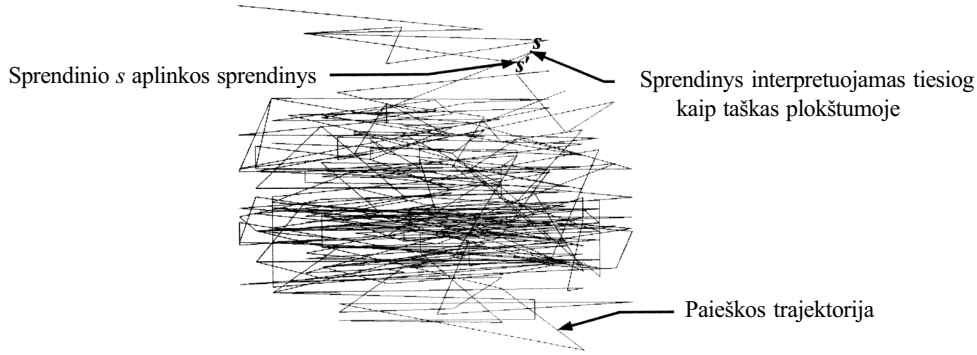
Terminas „euristinis algoritmas“ („euristika“) optimizavimo uždavinių sprendimo kontekste buvo pasiūlytas G. Polya dar XX amžiaus 5-ajame dešimtmetyje (Polya, 1945), tačiau ir šiuo metu ši sąvoka vis dar gana įvairiai traktuojama. Įdomumo dėlei vienas optimizavimo specialistų H. Streimas – tiesa, daugiau kaip prieš 30 metų – buvo nurodęs 11 euristinių algoritmų apibrėžimų. Visgi daugelis tyrinėtojų sutaria, kad euristiniai algoritmai yra tokios intelektualios optimizavimo uždavinių sprendimo priemonės, kuriomis siekiama rasti aukštos kokybės (bet nebūtinai optimalius) sprendinius per priimtiną laiką (Michalewicz, Fogel, 2000). EA negarantuoja, kad rasti sprendiniai bus optimalūs. Toks tikslas nekeliamas. Optimalumas paaukojamas paieškos laiko labui. Tuo EA skiriasi nuo tiksliųjų algoritmų, užtikrinančių optimalaus sprendinio radimą; deja, šiuo atveju laikas, reikalingas optimumui pasiekti, dažniausiai yra susietas su uždavinio apimtimi eksponentine priklausomybe.

Svarbų vaidmenį euristiniuose optimizavimo algoritmuose atlieka aplinkos funkcija $\Theta: S \to 2^S$. Ją naudojant bet kuriam sprendiniui s iš S galima priskirti poaibį $\Theta(s) \subseteq S$ – sprendinio s aplinką (sprendinių kaimynų aibę) (1 pav.). Aplinkų visuma nusako sprendinių erdvės topologiją. Sprendinys $s^\bullet$ yra lokaliai optimalus (lokalusis optimumas) aplinkos Θ atžvilgiu, jeigu kiekvienam sprendiniui s' iš aplinkos $\Theta(s^\bullet)$ galioja $f(s^\bullet) \le f(s')$.

Lokaliai optimalių sprendinių sąvoka yra viena fundamentalių euristinių optimizavimo algoritmų sąvokų. Ja iš esmės yra charakteri-



1 pav. ***Sprendinio* s *aplinkos* Θ(s) *iliustracija***



2 pav. ***Paieškos proceso trajektorijų grafinis interpretavimas***

zuojama euristinių algoritmų – kaip lokaliosios paieškos (LP) algoritmų – prigimtis ir paskirtis. Iš tiesų, EA veikimą galima su nedidelėmis išimtimis interpretuoti kaip tam tikrą efektyvų sprendinių erdvės nagrinėjimo procesą (perėjimų iš vienų sprendinių aplinkų į kitas trajektorijų seką) (Hoos, Stützle, 2004) (2 pav.).

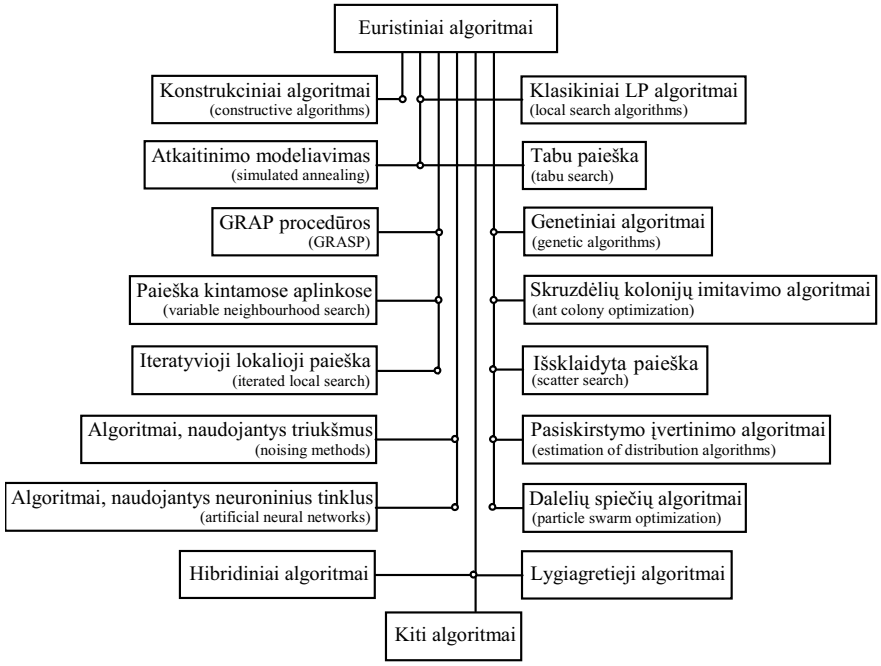
Euristinių algoritmų klasifikavimas

Nagrinėjant euristinius algoritmus, susiduriama su sunkiai aprėpiamu jų daugialypiškumu ir įvairove. Gali būti suskaičiuota iki kelioliko stambių EA grupių ir mažiausiai kelių šimtų atskirų algoritmų. Euristiniai algoritmai apima išties labai didelę optimizavimo būdų įvairovę: nuo paprastų konstrukcinių iki sudėtingų metaeuristinių algoritmų (angl. *metaheuristics*); nuo klasikinių vienetinių lokalių sprendinių paieškos algoritmų iki savireguliuojančių, apsimokančių sistemų modeliavimo metodų; nuo mus supančio pasaulio procesų imitacinių algoritmų (tokių kaip atkaitinimo modeliavimas, genetiniai (evoliuciniai) algoritmai, skruzdėlių kolonijų elgsenos imitavimo algoritmai ir pan.) iki labai jau dirbtinių, „sintetintų“ metodų (tokių kaip tabu paieška, godžiosios randomizuotos adaptyvios paieškos (GRAP) procedūros ir kt.) (Glover, Kochenberger, 2002) (3 pav.).

Euristinius algoritmus galime klasifikuoti atsižvelgdami į įvairius aspektus, pavyzdžiui, pagal tai, kokie procesai ir kokiu abstrahavimo lygiu imituojami (modeliuojami), koks algoritmo sudėtingumo (intelektualizavimo) laipsnis, kiek ir kokių yra algoritmo struktūrinių dalių, komponentų ir pan. (Blum, Roli, 2003; Calégari ir kt., 1999; Hertz, Kobler, 2000; Laguna, 2002; Osman, Laporte, 1996; Talbi, 2002; Vaessens ir kt., 1998; Voß, 2001).

Ch. Blumo ir A. Roli veikale (2003) išskiriamos tokios pagrindinės euristinių algoritmų kategorijos[1]:

[1] Reikia nepamiršti, kad, būdami santykinai vienalyčiai, euristiniai algoritmai gali būti tarpusavyje jungiami, derinami įvairiausiais būdais ir gaunami hibridiniai algoritmai. Natūralu, kad sudėtingesnės struktūros hibridinių algoritmų klasifikavimas tampa dar komplikuotesnis ir rafinuotesnis (Talbi, 2002).



3 pav. ***Pagrindinių euristinių algoritmų grupių diagrama***

- imituojantys gamtos procesus ir jų neimituojantys (angl. *nature-inspired and non-nature inspired*);
- naudojantys sprendinių rinkinius (populiacijas) ir jų nenaudojantys (angl. *population-based and single solution-based*);
- su statinėmis ir dinaminėmis tikslo funkcijomis (angl. *static and dynamic objective function*);
- pagrįsti viena ir keliomis (skirtingomis) aplinkos funkcijomis (angl. *one and various neighbourhood structures*);
- su atmintimi ir be atminties (angl. *memory and memory-less algorithms*).

Pateikti klasifikavimo pjūviai nėra vienodai reikšmingi, tikslūs ir griežti. Štai pirmojo nurodyto klasifikavimo pjūvio atveju ne taip lengva nuspręsti, ar, pavyzdžiui, modernieji sudėtiniai (hibridiniai) metodai yra dar mus supančio pasaulio procesų imitavimo algoritmai, ar tai jau dirbtinės, „atitolusios nuo gamtos“ euristinės procedūros. Negana to, atrodo, jog kai kurie algoritmai netenkina nė vieno tų klasių kriterijaus. Čia minėtina, pavyzdžiui, „Bolcmano mašinos“ (angl. *Boltzmann machines*) (Hinton, Sejnowski, 1987), „elastingieji tinklai“ (angl. *elastic nets*) (Durbin, Willshow, 1987), „hipereuristikos“ (angl. *hyper-heuristics*) (Burke ir kt., 2002), modifikuotų aplinkų naudojimas (Glover, 1992), nemonotoninė paieška (angl. *non-monotonic search*) (Hajek, Sasaki, 1989) ir kt.

Antrojo pjūvio atveju skiriamoji riba yra kur kas aiškesnė: čia klasifikavimo kriteri-

jus yra algoritmo naudojamų vienu metu sprendinių skaičius. Klasikiniai euristiniai algoritmai paprastai operuoja su vienu sprendiniu. Tačiau nemažai moderniųjų metodų (pvz., genetiniai algoritmai) yra grindžiami naudojamais sprendinių rinkiniais. Čia optimizavimo procesas asocijuojamas su individų poaibių, t. y. populiacijų (ar populiacijų „archipelagų“), tam tikru evoliucionavimu, o ne pavienių sprendinių kaip taškų judėjimu.

Kai dėl tikslo funkcijos, dažniausiai euristiniai algoritmai paieškos proceso metu išsaugo fiksuotą (pradinę) tikslo funkcijos formą. Tai nėra privaloma. Sprendžiant kai kuriuos optimizavimo uždavinius, pasiekta gerų rezultatų panaudojus specialiu būdu valdomos lokaliosios paieškos (angl. *guided local search*) metodiką, kurios esmė yra atitinkamos tikslo funkcijos korekcijos. Trumpai tariant, užuot transformavus sprendinius, koreguojama tikslo funkcija (Voudouris, Tsang, 2002). Kitas atvejis yra uždavinio duomenų korekcijos, t. y. tam tikrų triukšmų inkorporavimas į duomenis (angl. *noising methods*).

Didesnei daliai algoritmų būdingas vienos fiksuotos aplinkos funkcijos naudojimas. Būdinga išimtis yra paieška kintamose aplinkose (angl. *variable neighbourhood search*). (Tiesa, ir iteratyviojoje lokaliojoje paieškoje (angl. *iterated local search*) galimi perėjimai iš vienų aplinkų į kitas.)

Klasifikuoti algoritmus galima ir atsižvelgiant į tai, ar yra naudojama atmintis, ar ne. Nenaudojančius atminties algoritmus galima suvokti kaip Markovo procesus, kuriuose, kaip žinoma, tolesnę paieškos proceso kryptį lemia tik tam tikru momentu esanti būsena (tipinis pavyzdys yra atkaitinimo modeliavimo algoritmai). Priešingybė Markovo procesams yra tam tikrą atmintį naudojantys algoritmai (pavyzdys – tabu paieškos procedūros). Labai svarbi yra vadinamoji adaptyvioji atmintis (angl. *adaptive memory*), kurios paskirtis – akumuliuoti svarbius paieškos proceso parametrus, kad pagal juos būtų galima lanksčiai reguliuoti paieškos pobūdį.

M. Laguna (2002) pasiūlė paprastesnę klasifikavimo schemą – *x*/*y*/*z*-klasifikavimą. Tai klasifikavimas savotiškoje trimatėje algoritmų sudaromųjų (komponentų) erdvėje. Sudaromoji *x* skirta nurodyti, ar algoritmas disponuoja adaptyviąja atmintimi (paieškos proceso istorija), ar ne. Sudaromoji *y* apibrėžia paieškos proceso prigimtį (pvz., ar tai yra sistemiška paieška, naudojant fiksuotą sprendinių aplinkų struktūrą ir apibrėžtą aplinkos funkciją, ar tai grynai stochastinio (atsitiktinio) pobūdžio paieškos procesas). Pagaliau *z* charakterizuoja sprendinių bazės matiškumą, kuris gali kisti nuo 1 (tai reiškia, jog manipuliuojama su vienu sprendiniu) iki kurio nors baigtinio sveikojo skaičiaus *k* (šiuo atveju operuojama su sprendinių populiacija, sudaryta iš *k* sprendinių). Pavyzdžiui, klasikinės lokaliosios paieškos algoritmas galėtų būti identifikuotas taip: „*be adaptyvios atminties*“/„*determinuota paieška sprendinių aplinkose*“/„1“.

P. Calégari ir kt. (1999) savo straipsnyje yra pateikę klasifikavimo metodiką, orientuotą atskirai į euristinių algoritmų kategoriją – evoliucinius algoritmus. Evoliuciniai algoritmai klasifikuojami naudojant vadinamąją evoliucinių algoritmų lentelę (angl. *table of evolutionary algorithms – TEA*). Tokią lentelę sudaro aštuoni stulpeliai ir viena eilutė. Stulpelius atitinka pagrindiniai struk-

Populiacijos dydis	Populiacijos struktūruotumas	Paveldėjimo informacija	Populiacijos atnaujinimo pobūdis	Neleistini sprendiniai	Individo istorija	Atsitiktinumai	Pagerinimo algoritmas

4 pav. ***Evoliucinių algoritmų lentelės šablonas***

Populiacijos dydis	Populiacijos struktūruotumas	Paveldėjimo informacija	Populiacijos atnaujinimo pobūdis	Neleistini sprendiniai	Individo istorija	Atsitiktinumai	Pagerinimo algoritmas
fiksuotas	nėra	2[†]	pp[‡]	niekada	nėra	yra	nėra

[†] informacija paveldima iš **2** sprendinių („tėvų“);
[‡] pilnasis **p**akeitimas (visų populiacijos narių atnaujinimas)

5 pav. ***Standartinio genetinio algoritmo komponentų lentelė***

tūriniai algoritmų komponentai (ingredientai), o eilutėje nurodomos tam tikros komponentų reikšmės. Siūlomi tokie klasifikavimo ingredientai (taip pat žr. 4 pav.): 1) populiacijos, t. y. sprendinių (individų) rinkinio dydis; 2) populiacijos struktūruotumas; 3) paveldėjimo informacija, t. y. individų paveldimų savybių šaltinis; 4) populiacijos atnaujinimo pobūdis; 5) galimybė operuoti su neleistinais sprendiniais; 6) individo istorija (informacija, gauta evoliucionuojant procesui); 7) atsitiktinumai (mutacijos, triukšmai); 8) pagerinimo algoritmas, t. y. procedūra naujai gaunamiems sprendiniams optimizuoti.

Atskiros evoliucinio algoritmo lentelės su atitinkamomis algoritmo komponentų reikšmėmis pavyzdys pateikiamas 5 pav.

Universalizuoto klasifikavimo paradigma

Pirmiau aptarta evoliucinių algoritmų klasifikavimo schema galėtų būti pritaikyta ir platesnei euristinių algoritmų visumai. Siūloma mintis, kad galbūt yra galimà hipotetinė euristinių (ir ne tik) algoritmų matrica, savotiškas universumas – nelyginant Mendelejevo periodinė cheminių elementų lentelė (jei tik toks sugretinimas čia tinkamas). Tai ir yra pamatinė universalizuoto klasifikavimo paradigmos idėja. Universaliajai algoritmų matricai būtinas fundamentalių algoritmų sudedamųjų (substancinių) elementų (komponentų) – tam tikrų konceptų – radimas. Tai nėra paprastas dalykas. Visų pirma, išlieka svarbus atviras klausimas dėl tokios matricos, kaip substancinių konceptų

sistemos, baigtinumo. Antra, net ir tos matricos baigtinumo prielaidos atveju anaiptol nėra aišku, kiek ir kokių tų substancinių komponentų gali potencialiai egzistuoti. Šiaip ar taip, sudaryti baigtinę substancinių konceptų matricą yra neįmanoma be sisteminių, integruotų teorinių analitinių ir empirinių praktinių tyrimų.

Pateiktoje paradigmoje – skirtingai negu A. Misevičiaus (2003) ir E. Talbi (2002) darbuose, kuriuose daugiau dėmesio telkiama į algoritmų realizavimo, funkcionavimo būdus, – mes pabrėžiame struktūrinius algoritmų komponentus, taip sakant, algoritmų „projekciją“ substancinių konceptų lauke.

Yra siūloma tokia prototipinė pagrindinių algoritmų substancinių konceptų aibė[2]:

- sprendinio konstravimas (trumpai žymėsime S);
- paieškos proceso valdymo taisyklės (T)*;
- sprendinių transformacijos (ST)**;
- aplinkos funkcija (F);
- adaptyvioji atmintis (A);
- grįžtamasis ryšys (GR)***;
- triukšmai (Tr)****.

* Valdymo taisyklės galėtų apimti sprendimus apie sprendinių priėmimą / atmetimą, baigimo sąlygų (pvz., paieškos iteracijų skaičiaus) apibrėžimą ir t. t. (Savo ruožtu sprendimai gali būti priimami determinuotai ar naudojant tikimybines funkcijas.) ** Galimi skirtingi sprendinių transformacijų variantai, pavyzdžiui: perėjimas iš vieno sprendinio į kitą sprendinį (trumpai „vienas-vienas“ ($S \to S$)); sprendinio sudarymas naudojant porą ar daugiau sprendinių („du-vienas“ ($S \times S \to S$), „daugelis-vienas“ ($2^S \to S$)); operacijos su sprendinių populiacijomis („daugelis-daugelis“ ($2^S \to 2^S$)). *** Grįžtamasis ryšys leidžia reikiamu būdu veikti paieškos procesą, suintensyvinant jį vienose ar kitose sprendinių erdvės srityse, atsižvelgiant į aposteriorinę informaciją, sukauptą iki tol vykusio proceso metu. **** Terminu „triukšmai“ norima įvardyti uždavinio sprendinių (duomenų) ar tikslo funkcijos reikšmių įvairios prigimties deformacijas.

Prototipinių substancinių konceptų rinkinį formaliai aprašykime kaip aibę Ψ, t. y. $\Psi = \{S, T, ST, F, A, GR, Tr\}$. Tuomet hipotetinė algoritmų (klasių) erdvė galėtų būti apibrėžta aibe $\Omega = 2^{\Psi}$; čia Ω žymi aibę, sudarytą iš visų galimų aibės Ψ narių poaibių, t. y. $\Omega = \{\{S\}, \{T\}, \{ST\}, ..., \{Tr\}, \{S, T\}, \{S, ST\}, ..., \{S, Tr\}, \{T, ST\}, ..., \{GR, Tr\}, \{S, T, ST\}, \{S, T, F\}, \ldots, \{S, T, Tr\}, \{T, ST, F\}, ..., \{A, GR, Tr\}, \{S, T, ST, F\}, ..., \{F, A, Gr, Tr\}, ..., \{S, T, ST, F, A, GR, Tr\}\}$. Atskiros algoritmų klasės gali būti identifikuojamos taip: <skaičius><koncepto identifikatorius> [,<koncepto identifikatorius>] [, ...]; čia <koncepto identifikatorius> :: = S | T | ST | F | A | GR | Tr, o skaičius nurodo, kiek algoritmų klasė turi ją apibrėžiančių substancinių konceptų, pavyzdžiui, 1S, 2S, T; 3S, T, F ir pan.

Algoritmų su vienu sprendinio konstravimo konceptu klasei (1S) priklausytų konstrukciniai algoritmai, iš jų ir algoritmai, generuojantys vieną sprendinį randomizuotu būdu. (Kiti konceptai čia nėra aktualūs, nes formuojamas tik vienas pradinis sprendinys, toliau jo neoptimizuojant.) Daugiakarčio

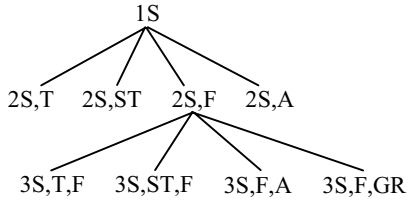
[2] Naujų konceptų analizės ir išplėsto (ar galimo pilno) substancinių konceptų rinkinio formavimo klausimai galėtų būti atskiro darbo tema.

sprendinių generavimo (angl. *multi-start*) atveju (2S, T) yra jau aktualu ir sprendinių priėmimo / atmetimo taisyklės (taip pat baigimo sąlyga). Lokaliosios paieškos algoritmų grupėje (3S, T, F), be to, turi būti apibrėžta aplinkos funkcija (kitaip tariant, sprendinių aplinkų struktūra – topologija). Genetinių (evoliucinių) algoritmų šeimoje (3S, T, ST) svarbų vaidmenį atlieka sprendinių transformacijos (šiuo atveju evoliuciniai operatoriai, tokie kaip atrinkimas (selekcija), kryžminimas, mutavimas). Tabu paieškos algoritmuose (4S, T, F, A) papildomai naudojama adaptyvioji atmintis. Hibridiniuose algoritmuose (4S, T, ST, F; 5S, T, ST, F, A) derinama lokalioji ir/arba tabu paieška bei sprendinių transformacijos. Panašiai galėtume identifikuoti ir kitas algoritmų klases.

Taip kaip cheminiai elementai yra surikiuoti periodinėje lentelėje pagal elementariųjų dalelių (elektronų) atomuose skaičių – taip ir algoritmų klasės galėtų būti rikiuojamos atsižvelgiant į substancinių konceptų skaičių (savotišką valentiškumo skaičiaus analogą). Taigi, algoritmų grupė $\langle$ 1S, 1T, 1ST, ... $\rangle$ priklausytų „vienvalenčių“ algoritmų klasei, $\langle$ 2S, T; 2S, ST; 2S, F; ... $\rangle$ – „dvivalenčių“ algoritmų klasei ir t. t. Atskiri konceptai surikiuotose algoritmų grupėse periodiškai atsikartotų.

Algoritmų klasių su tuo pačiu valentiškumo (t. y. konceptų) skaičiumi panašumas gali būti kiekybiškai apibrėžiamas kaip atitinkamų konceptų aibių skirtumo didumas (pvz., $\left|\Omega_{\Im_1} \setminus \Omega_{\Im_2}\right|$; čia $\Omega_{\Im_1}$, $\Omega_{\Im_2}$ žymi atitinkamai algoritmų $\Im_1$ ir $\Im_2$ konceptų aibes).

Algoritmų su didesniu konceptų (valentiškumo) skaičiumi klasės paveldi atitinkamas algoritmų su mažesniu konceptų skaičiumi klasių savybes, jeigu egzistuoja kokia nors netuščia tas klases charakterizuojančių konceptų poaibių sankirta. Pavyzdžiui, klasė 3S, ST, F paveldi klasės 2S, F savybes ($\{S, ST, F\} \cap \{S, F\} \neq \varnothing$) (6 pav.).



6 pav. ***Algoritmų klasių „genealoginio medžio“ fragmentas***

Naujoji paradigma leidžia lengvai formalizuoti naujų išvestinių, hibridinių algoritmų[3] klasių aprašymą naudojant aibių teorijos sąjungos operaciją. Pavyzdžiui, kombinuotų genetinės lokaliosios paieškos algoritmų klasė gali būti apibrėžta tiesiog kaip genetinių algoritmų klasės ir lokaliosios paieškos klasės konceptų aibių sąjunga, t. y. $\{S, T, ST\} \cup \{S, T, F\}$.

Išvados

Šiame straipsnyje išnagrinėti kai kurie klausimai, susiję su euristinių optimizavimo algoritmų klasifikavimu. Apžvelgtos svarbesnių euristinių algoritmų (EA) klasifikavimo schemos (metodikos), išskiriant esminius skiriamuosius algoritmų faktorius. Šie fak-

[3] Išvestinių, hibridinių algoritmų sudarymas yra eksplikatyvus (nepraplečia konceptų aibės), skirtingai nuo implikatyvaus algoritmų aibės išplėtimo įterpiant naujus fundamentalius algoritmų sudedamuosius komponentus, t. y. konceptus.

toriai yra svarbūs formuluojant optimizavimo uždavinių inovatyvių sprendimo metodų, būdų konstravimo principus ir metodologiją.

Aptarta ir universalizuoto EA klasifikavimo paradigma, pabrėžiant fundamentalius sudedamuosius algoritmų komponentus – substancinius konceptus. Pasiūlyta prototipinė pagrindinių algoritmų substancinių konceptų aibė (sistema). Ši sistema galėtų būti tuo metodologiniu pamatu, kuris suteiktų papildomų galimybių ne tik kuriant efektyvius algoritmus, ne tik numatant perspektyvias tolesnio EA tobulinimo kryptis, bet ir apskritai ieškant naujų konceptualių principų ir idėjų.

LITERATŪRA

BLUM, Christian; ROLI, Andrea (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, vol. 35, p. 268–308.

BURKE, Edmund; KENDALL, Graham; NEWALL, Jim; HART, Emma; ROSS, Peter; SCHULENBURG, Sonia (2002). Hyper-heuristics: an emerging direction in modern search technology. Iš Glover, F.; Kochenberger, G. (ed.). *Handbook of Metaheuristics*. Norwell: Kluwer, p. 457–474.

CALÉGARI, Patrice; CORAY, Giovanni; HERTZ, Alain; KOBLER, Daniel; KUONEN, Pierre (1999). A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics*, vol. 5, p. 145–158.

DU, Ding-Zhu; PARDALOS, Panos M. (1998). *Handbook of Combinatorial Optimization*. Vol. 1–3. Dordrecht: Kluwer. 2410 p.

DURBIN, Richard; WILLSHOW, David (1987). An analogue approach to the traveling salesman problem using an elastic net method. *Nature*, vol. 326, p. 689–691.

EU/ME European Chapter on Metaheuristics [interaktyvi nuorodų svetainė] [žiūrėta 2009-03-05]. Prieiga per internetą: <http://www.metaheuristics.eu/>.

FLOUDAS, Christodoulos A.; PARDALOS, Panos M. (ed.) (2001). *Encyclopedia of Optimization*. Dordrecht: Kluwer. 3200 p.

GLOVER, Fred (1992). New ejection chain and alternating path methods for traveling salesman problems. Iš Balci, O.; Shardi, R.; Zenios, S.A. (ed.). *Operations Research and Computer Science: New Developments in Their Interfaces*. Oxford: Pergamon Press, p. 449–509.

GLOVER, Fred; KOCHENBERGER, Gary (ed.) (2002). *Handbook of Metaheuristics*. Norwell: Kluwer. 570 p.

HAJEK, Bruce; SASAKI, Galen (1989). Simulated annealing – to cool or not. *Systems and Control Letters*, vol. 12, p. 443–447.

HERTZ, Alain; KOBLER, Daniel (2000). A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, vol. 126, p. 1–12.

HINTON, Geoffrey E.; SEJNOWSKI, Terence J. (1987). Learning and relearning in Boltzmann machines. Iš Rumelhart, D.E.; McLelland, J. L. (ed.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge: MIT Press, p. 282–317.

HOOS, Holger H.; STÜTZLE, Thomas (2004). *Stochastic Local Search*: *Foundations and Applications*. San Francisco: Morgan Kaufmann. 658 p.

LAGUNA, Manuel (2002). *Global optimization and meta-heuristics* [žiūrėta 2008 m. spalio 31 d.]. Prieiga per internetą: <http://leeds-faculty.colorado.edu/laguna/articles/elss.pdf>.

LEE, Kwang Y.; EL-SHARKAWI, Mohamed A. (ed.) (2008). *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems* [IEEE Press Series on Power Engineering]. Hoboken: Wiley-IEEE Press. 616 p.

Metaheuristics Network [interaktyvi nuorodų svetainė] [žiūrėta 2009-03-06]. Prieiga per internetą: <http://www.metaheuristics.net/>.

MICHALEWICZ, Zbigniew; FOGEL, David B. (2000). *How to Solve It: Modern Heuristics*. Berlin-Heidelberg: Springer. 467 p.

MISEVIČIUS, Alfonsas (2003). Intelektualieji optimizavimo metodai. *Informacijos mokslai* (Information Sciences), mokslo darbai, t. 26, p. 160–166.

OSMAN, Ibrahim H.; LAPORTE, Gilbert (1996). Metaheuristics: a bibliography. *Annals of Operations Research*, vol. 63, p. 513–623.

PARDALOS, Panos M.; RESENDE, Mauricio G.C. (ed.) (2002). *Handbook of Applied Optimization*. New York: Oxford University Press. 2026 p.

POLYA, George (1945). *How to Solve It*. Princeton: Princeton University Press. 224 p.

SCHNEIDER, Johannes J.; KIRKPATRICK, Scott (2006). *Stochastic Optimization*. New York: Springer. 565 p.

SIARRY, Patrick; MICHALEWICZ, Zbigniew (ed.) (2008). *Advances in Metaheuristics for Hard Optimization* [Natural Computing Series]. Berlin; Heidelberg; New York: Springer. 481 p.

TALBI, El-Ghazali (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, vol. 8, p. 541–564.

VAESSENS, Rob; AARTS, Emile H. L.; LENSTRA, Jan Karel (1998). A local search template. *Computers & Operations Research*, vol. 25, p. 969–979.

VOß, Stefan (2001). Meta-heuristics: the state of the art. Iš Nareyek, A. (ed.). *Local Search for Planning and Scheduling, Lecture Notes in Computer Science*, vol. 2148: Lecture Notes in Artificial Intelligence. Berlin; Heidelberg: Springer, p. 1–23.

VOUDOURIS, Chris; TSANG, Edward (2002). Guided local search. Iš Glover, F.; Kochenberger, G. (ed.). *Handbook of Metaheuristics*. Norwell: Kluwer, p. 185–218.

YANG, Xin-She (2008). *Nature-Inspired Metaheuristic Algorithms*. Frome: Luniver Press. 128 p.

ON THE CLASSIFICATION OF HEURISTIC ALGORITHMS

Alfonsas Misevičius, Jonas Blonskis, Vytautas Bukšnaitis

Summary

In this paper, the issues related to the classification (taxonomy) of heuristic optimization algorithms are discussed. Firstly, the main goals and features of heuristic techniques are introduced. Further, we outline some important classification schemes (templates) for the classical and modern heuristic algorithms such as (descent) local search, simulated annealing, tabu search, genetic (evolutionary) algorithms, ant colony optimization, etc. We also analyze the basic aspects of a universal classification template based on a set of so-called substantial concepts, i.e. the fundamental structural components of the algorithms. The paper is completed with concluding remarks.

Key words: algorithms, heuristic and metaheuristic algorithms, classification of algorithms.

Įteikta 2008 m. lapkričio 25 d.
Pataisyta 2009 m. kovo 18 d.