VYTAUTO DIDŽIOJO UNIVERSITETAS
VILNIAUS UNIVERSITETO MATEMATIKOS IR INFORMATIKOS INSTITUTAS

Jūratė SKŪPIENĖ

ALGORITMŲ IR JUOS REALIZUOJANČIŲ PROGRAMŲ VERTINIMAS INFORMATIKOS VARŽYBOSE

Daktaro disertacijos santrauka

Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

Vilnius, 2010

Distertacija rengta 2006–2010 m. Vilniaus universiteto Matematikos ir informatikos institute.
Doktorantūros teisė suteikta kartu su Vytauto Didžiojo universitetu
2004 m. gruodžio 13 d. Lietuvos Respublikos Vyriausybės nutarimu nr. 1285.

Mokslinis vadovas
prof. habil. dr. Antanas Žilinskas (Vilniaus universiteto Matematikos ir informatikos institutas, fiziniai mokslai, informatika - 09).

Disertacija ginama Vytauto Didžiojo universiteto Informatikos mokslo krypties taryboje:

Pirmininkas
prof. habil. dr. Vytautas Kaminskas (Vytauto Didžiojo universitetas, fiziniai mokslai, informatika - 09 P).

Nariai:
prof. dr. Eduardas Bareiša (Kauno technologijos universitetas, technologijos mokslai, informatikos inžierija - 07 T),
prof. dr. Valentina Dagienė (Vilniaus universiteto Matematikos ir informatikos institutas, fiziniai mokslai, informatika - 09 P),
dr. Olga Kurasova (Vilniaus universiteto Matematikos ir informatikos institutas, fiziniai mokslai, informatika - 09 P),
prof. habil. dr. Henrikas Pranevičius (Kauno technologijos universitetas, fiziniai mokslai, informatika - 09 P).

Oponentai:
prof. habil. dr. Artūras Kaklauskas (Vilniaus Gedimino technikos universitetas, fiziniai mokslai, informatika - 09 P),
doc. dr. Rimantas Vaicekauskas (Vilniaus universitetas, fiziniai mokslai, informatika - 09 P).

Disertacija bus ginama viešame Informatikos mokslo krypties tarybos posėdyje 2010 m. lapkričio 10 d. 13 val. Vilniaus universiteto Matematikos ir informatikos institute, 203 auditorijoje. Adresas: Akademijos g. 4, LT-08663 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta iki 2010 m. spalio 8 d.
Disertaciją galima peržiūrėti M. Mažvydo nacionalinėje bibliotekoje, Vytauto Didžiojo universiteto ir Vilniaus universiteto Matematikos ir informatikos instituto bibliotekose.

VYTAUTAS MAGNUS UNIVERSITY
INSTITUTE OF MATHEMATICS AND INFORMATICS OF
VILNIUS UNIVERSITY

Jūratė SKŪPIENĖ

EVALUATION OF ALGORITHM-CODE COMPLEXES IN INFORMATICS CONTESTS

Summary of Doctoral Dissertation

Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

Vilnius, 2010

This research was accomplished in the period from 2006 to 2010 at the Institute of Mathematics and Informatics of Vilnius University.
The right for the doctoral studies in informatics was granted to the Institute of Mathematics and Informatics together with Vytautas Magnus University by the Government of the Republic of Lithuania, Decree No.1285, issued on the 13th of December 2004.

Scientific Advisor

Prof. Habil. Dr. Antanas Žilinskas (Institute of Mathematics and Informatics of Vilnius University, Physical Sciences, Informatics - 09 P).

The dissertation is being defended at the Council of Scientific Field of Informatics at Vytautas Magnus University:

Chairman

Prof. Habil. Dr. Vytautas Kaminskas (Vytautas Magnus University, Physical Sciences, Informatics - 09 P).

Members:

Prof. Dr. Eduardas Bareiša (Kaunas University of Technology, Technological Sciences, Informatics Engineering - 07 T),

Prof. Dr. Valentina Dagienė (Institute of Mathematics and Informatics of Vilnius University, Physical Sciences, Informatics - 09 P),

Dr. Olga Kurasova (Institute of Mathematics and Informatics of Vilnius University, Physical Sciences, Informatics - 09 P),

Prof. Habil. Dr. Henrikas Pranevičius (Kaunas University of Technology, Physical Sciences, Informatics - 09 P).

Opponents:

Prof. Habil. Dr. Artūras Kaklauskas (Vilnius Gediminas Technical University, Physical Sciences, Informatics - 09 P),

Doc. Dr. Rimantas Vaicekauskas (Vilnius University, Physical Sciences, Informatics - 09 P).

The dissertation will be defended at the public session of the Scientific Council of Scientific Field of Informatics in the auditorium No 203 of the Institute of Mathematics and Informatics of Vilnius University at 1 p. m. on the 10^{th} of November 2010. Address: Akademijos str. 4, LT-08663 Vilnius, Lithuania.

The summary of the dissertation was distributed before the 8^{th} of October, 2010. A copy of the dissertation is available at Martynas Mažvydas National Library of Lithuania, the Library of the Institute of Mathematics and Informatics of Vilnius University, and the Library of Vytautas Magnus University.

TIRIAMA PROBLEMA IR JOS AKTUALUMAS

Algoritmų ir juos realizuojančių programų kokybės vertinimas yra aktuali problema ir ją tenka spręsti PĮ industrijoje, aukštosiose mokyklose, o taip pat ir informatikos varžybose.

Informatikos varžybų dalyviams pateikiami algoritmų kūrimo uždaviniai, dalyviai turi juos išanalizuoti, sukurti algoritmą, ir jį realizuoti leidžiama programavimo kalba. Tačiau neprašoma pateikti algoritmų teisingumo įrodymų, nes dalyviai yra mokiniai ir tokie reikalavimai viršytų jų gebėjimus. Vertinimui pateikiama programa, kurioje realizuotas nenustatytas duotąjį uždavinį sprendžiantis algoritmas, t. y. algoritmo bei programos kompleksas (žymėsime APK). Vertinant APK dominuoja *juodosios dėžės principu vykdomas testavimas* (toliau sutrumpintai vadinsime *testavimu*).

Užduočių rengėjai konkretiems uždaviniams sudaro galimų sprendimų taksonomiją, ją susieja su įverčiais (taškais). Testai kuriami taip, kad kokybės įverčiai, gauti remiantis testavimo rezultatais, atitiktų numatytuosius. Tačiau nėra žinoma, ar APK kokybės įverčių, gautų remiantis testavimo rezultatais, tikslumas yra pakankamas, o jei ne – tai koks neatitikimo mastas. Taigi APK kokybės vertinimas yra mokslinė problema.

Disertacijoje nagrinėjama vertinimo shema, taikoma *Lietuvos mokinių informatikos olimpiadose* (LMIO). Tačiau pati disertacija yra aktuali platesniame kontekste. Informatikos varžybos yra plačiai paplitusios. Jei dalyvavimą akivaizdinėse varžybose riboja organizaciniai ištekliai, tai yra atvejų, kai populiarios nuotolinės varžybos pritraukia iki šimto tūkstančių dalyvių. Kadangi LMIO organizuojama panašiais principais, kaip ir daugelis šių varžybų, tai disertacijos rezultatai gali būti pritaikyti ir kitose informatikos varžybose.

DARBO TIKSLAI IR UŽDAVINIAI

Darbo tikslai: ištirti APK vertinimo schemas, taikomus kriterijus; pasiūlyti vertinimo schemą, pagrįstą daugiakriterių sprendimų teorijos metodais, tinkamą naudoti Lietuvos mokinių infomatikos olimpiadose.

Siekiant įgyvendinti tikslus, disertacijoje sprendžiami šie uždaviniai:

1. Pateikti vertinimo informatikos varžybose patirties ir problemų apžvalgą. Ištirti programavimo užduočių, pateikiamų studentams aukštosiose mokyklose, sprendimų vertinimo praktiką. Įvertinti, ar ši patirtis gali būti perkeliama į informatikos varžybas.

2. Ištirti pusiau automatinio vertinimo įtraukimo į informatikos olimpiadas galimybes taikant APK vizualizaciją (uždavinių su grafais pavyzdžiu).

3. Išanalizuoti pasirinktą kiekį APK, nustatyti, koks yra APK kokybės įverčių, gautų pasiremiant testavimo rezultatais, tikslumas. Nustatyti, ar dalyvio pasirinkto algoritmo realizavimo kokybė susijusi su APK, kuriame realizuotas tas algoritmas, programavimo stiliaus kokybe.

4. Apibrėžti vertinimą informatikos varžybose kaip daugiakriterių sprendimų teorijos uždavinį. Išanalizuoti daugiakriterių uždavinių sprendimo procesą bei metodus, pasiūlyti tinkamus vertinimo uždavinio sprendimui.

5. Remiantis pasiūlytais metodais, sukurti vertinimo LMIO schemą, kurią sudarytų vertinimo paketo (angl. *submission*) sudėtis, jo vertinamų atributų sąrašas, kiekvieną atributą atitinkančių vertinimo kriterijų sąrašas bei rezultatų agregavimui skirta funkcija.

GINAMIEJI TEIGINIAI

1. Šiuo metu informatikos varžybose taikomą APK kokybės vertinimo schemą tikslinga tobulinti.

2. Pusiau-automatinio vertinimo taikymas pagreitina vertinimo procesą ir tokiu būdų sudaro galimybes į vertinimo schemą įtraukti papildomus vertinimo kriterijus.

3. Daugiakriterių sprendimų teorijos taikymas padeda sudaryti labiau pagrįstas vertinimo schemas.

TYRIMO METODIKA

Rengiant analitinę apžvalgą buvo taikomas informacijos sisteminimas bei lyginamoji analizė.

LMIO dalyvių sukurti APK buvo tiriami taikant algoritmų analizę. Rezultatai apdoroti ir pateikti taikant statistinį SPSS paketą ir aprašomąją statistiką. Analizuojant LMIO taikomų vertinimo kriterijų pagrįstumą buvo remiamasi programinės įrangos kūrimo metodologija.

Kuriant vertinimo schemą buvo naudojimas modeliavimas, Tikslų/ Klausimų/ Metrikų (angl. *Goal/Question/Metrics*) metodas, ekspertinio vertinimo metodai, daugiakriterių sprendimų teorijoje taikomi metodai, tarp jų vertės matavimo teorija (angl. *value measurement theory*), neraiškioji logika (angl. *fuzzy logic*).

DISERTACIJOS REZULTATAI

1. Išanalizuota vertinimo informatikos varžybose patirtis, ištirtos ir suklasifikuotos problemos testavimu susijusios problemos. Išanalizuota testavimo naudojimo programų vertinimui aukštosiose mokyklose patirtis.

2. Įvertintos grafų, realizuotų APK, vizualizavimo galimybės, parinkta vizualizavimo paradigma. Ištirti ir suklasifikuoti grafo vizualizavimo 191-iuose APK, sprendžiančiuose užduotis su grafais, būdai. Sukurtas įrankis, skirtas pusiau automatiniam, grafų, realizuotų, APK, vizualizavimui.

3. LMIO taikomi APK vertinimo kriterijai išanalizuoti kokybės modelio ISO-9126-1 atžvilgiu, įvertintas jų taikymo vertinimui pagrįstumas. Išanalizuota 290 APK. Nustatyta, kad šių APK kokybės įverčių nukrypimas nuo vertintojų nustatytų kokybės įverčių intervalų taikant dvi rezultatų agregavimo funkcijas: *dalinio agregavimo funkciją* lygus 20,2%, o taikant *grupinę viskas arba nieko rezultatų agregavimo funkciją* – 8,4%. Apskaičiuota koreliacija analizuotiems APK tarp programavimo stiliaus kokybės ir algoritmų realizavimo kokybės: 0,468.

4. *Vertinimo informatikos varžybose uždavinys* apibrėžtas kaip daugiakriterių sprendimų teorijos (DST) uždavinys, priklausantis *rangavimo*, *grupinių sprendimų*, bei *pasikartojančiųjų* uždavinių klasėms.

   Išnagrinėti DST metodai ir pasirinkti tinkami šiam uždaviniui spręsti: Tikslų/Klausimų/Metrikų metodas, bei *grupinių sprendimų palaikymo algoritmas* sujungtas su *S. J. Chen* pasiūlytu metodu.

5. Taikant minėtus DST metodus sukurta vertinimo schema, siūloma taikyti LMIO.

6. Pasiūlytoji vertinimo schema išbandyta praktiškai. Parinkti keturi uždaviniai, ir pateikti grupei mokinių nedidelėse varžybose. Apibendrinti rezultatai, pateikta praktinių pasiūlymų susijusių su vertinimo schemos adaptavimu konkretiems uždaviniams.

MOKSLINĖS PUBLIKACIJOS DISERTACIJOS TEMA

Straipsniai recenzuojamuosiuose periodiniuose mokslo leidiniuose:

1. **Skūpienė, J.** (2010). Score Calculation in Informatics Contests using Multiple Criteria Decision Method. *Informatics in Education*, accepted for publication.

2. **Skūpienė, J.** (2010). Improving the Evaluation Model for the Lithuanian Informatics Olympiads. *Informatics in Education*, ISSN 1648-5832, 9(1):141-158.

3. **Skūpienė, J.** (2009). Lietuvos informatikos olimpiados darbų vertinimas programinės įrangos kokybės modelio požiūriu. *Informacijos mokslai*, ISSN 1392-0561, 50: 153–159.

4. **Skūpienė, J.** (2009). Credibility of Automated Assessment in Lithuanian Informatics Olympiads: One Task Analysis. *Pedagogika*, ISSN 1392-0340, 96: 143–151.

5. **Skūpienė, J.** (2007). Prielaidos automatiniam programavimo stiliaus vertinimui informatikos olimpiadose. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 47: 273–278.

6. **Skūpienė, J.** (2007). Automatinio sprendimų vertinimo informatikos olimpiadose raida ir perspektyvos. *Informacijos mokslai*, ISSN 1392-0561, 42–43: 43–49.

7. **Skūpienė, J.** and Žilinskas, A. (2006). Evaluation in Informatics Contests: Aids for Tasks Involving Graphs. *The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology*, ISSN: 0773-182, 23(1–4): 39–46.

Straipsniai kituose recenzuojamuosiuose leidiniuose:

1. **Skūpienė, J.**, and Žilinskas, A. (2007). Automated Grading of Programming Tasks Fulfilled by Students: Evolution and Perspectives. In *Proceedings of the 4th E-learning'07 Conference.* Istanbul, Turkey.

2. **Skūpienė, J.** (2006). Programming style – Part of Grading Scheme in Informatics Olympiads: Lithuanian Experience. In *Information Technologies at School, Proceedings of the Second International Conference "Informatics in Secondary Schools: Evolution and Perspectives".* Vilnius, Lithuania, 2006, pages 545 – 552, (ISI Proceedings List).

3. **Skūpienė, J.** and Žilinskas, A. (2006). Evaluation of programs in Informatics Contests: Case of Implementation of Graph Algorithms. In *Proceedings of The Third E-learning Conference "Computer Science Education"*. Coimbra, Portugal, 2006.

4. **Skūpienė, J.** (2004). Automated testing in Lithuanian Informatics Olympiads. In *Informacinės technologijos 2004, Konferencijos pranešimų medžiaga*. Kaunas, Technologija, pages 37–41.

Disertacijos rezultatai buvo pristatyti dešimtyje nacionalinių bei tarptautinių konferencijų.

DISERTACIJOS STRUKTŪRA

Disertacija parašyta anglų kalba. Ją sudaro 6 skyriai, 2 priedai, literatūros sąrašas. Disertacijos apimtis 176 puslapiai, joje yra 45 iliustracijos, 38 lentelės, nuorodos į 194 literatūros šaltinius.

1. ĮVADAS

Pirmasis skyrius yra įvadinis. Jame pristatoma tiriama problema ir jos aktualumas, tyrimo tikslai bei uždaviniai, tyrime naudoti metodai, gauti rezultatai, darbo mokslinis naujumas, mokslinių publikacijų disertacijos tema sąrašas.

2. APK VERTINIMO INFORMATIKOS VARŽYBOSE APŽVALGA

Šiame skyriuje apžvelgiami su vertinimu informatikos varžybose susiję aspektai. Įvedama APK sąvoka. Tai terminas, apibūdinantis programą, kurioje realizuotas neatskleistas duotą uždavinį sprendžiantis algoritmas. Iškeliamos dvi svarbiausios problemos, susijusios su APK vertinimu. Pirmoji yra ta, kad sujungus algoritmą bei jo realizaciją į APK, tampa nebeaišku, kaip atskirai įvertinti sukurtą algoritmą ir kaip jo realizaciją. Antroji problema yra ta, kad informatikos varžybose dominuoja testavimas, kaip pagrindinis (ar vienintelis) vertinimo metodas. Tačiau testavimas negali nei apskritai nustatyti kokį algoritmą mokinys sukūrė, nei įvertinti APK vidinės struktūros, taip pat nėra aišku, ar testavimo pagrindu gauti APK kokybės įverčiai patikimi.

Toliau skyriuje sukonkretinama *informatikos varžybų*, taip kaip jos bus suprantamos disertacijoje, sąvoka, apžvelgiami LMIO tikslai ir struktūra, šiuo metu LMIO naudojama vertinimo schema, taip pat populiariausios kitose varžybose naudojamos vertinimo schemos.

Dalyvis pateikią vertinimo paketą, kurį sudaro APK ir papildomi komponentai, jeigu jų reikalauja sąlyga, pvz., testų rinkinys. Vertintojai turi pateikti kokybės įvertį. Skyriuje išnagrinėta kokybės samprata. Parodyta, kad neegzistuoja absoliuti kokybės samprata, ir mokslininkai kokybę apibrėžia arba kaip griežtą atitikimą iš anksto apibrėžtoms specifikacijoms, arba kaip atitikimą (nuolat kintantiems) vartotojų poreikiams. Informatikos varžybose šios dvi sampratos susilieja į vieną, nes vertinimo komisija yra ir specifikacijų sudarytoja, ir vienintelė vartotoja.

Išanalizavę programų, sukurtų aukštųjų mokyklų studentų, vertinimo patirtį, nustatėme, kad mokymosi ir varžybų kontekstai yra pakankamai skirtingi, akcentuojami skirtingi dalykai, sprendžiamos kitokios problemos. Dėl to netikslinga bandyti taikyti informatikos varžybose programavimo užduočių vertinimo patirtį. Įvardijome tik vieną sritį, kur universitetų patirtis galėtų būti bandoma taikyti informatikos varžybose. Tai automatinis programavimo stiliaus vertinimas. Tačiau norint tai padaryti reiktų atskiro išsamaus tyrimo.

Išanalizavome ir suskirstėme į dvi dalis mokslininkų keliamus, su

testavimu informatikos varžybose, susijusius klausimus. Tai klausimai, susiję su testavimo dominavimu vertinimo schemose, t. y., su kitų vertinimo formų nebuvimu. Kiti klausimai susiję su automatizuoto vertinimo patikimumu. Vertintojai remdamiesi uždavinio sąlygas ir patirtimi suskirsto būsimus APK į kategorijas (numatydami ir klaidingus APK) ir jas susieja su priimtinais taškų intervalais. Testus stengiamasi konstruoti taip, kad testavimo pagrindu gauti įverčiai patekų į šiuos intervalus. Įvairiuose straipsniuose keliamas klausimas, ar automatizuoto vertinimo pagrindu gautų APK kokybės įverčių tikslumas pakankamas, o jei ne – tai kokio masto yra neatitikimas.

3. DAUGIAKRITERIŲ SPRENDIMŲ ANALIZĖS PROCESO IR METODŲ APŽVALGA

Skyriuje pristatoma daugiakriterių sprendimų analizės (DSA) samprata bei parodoma, kad uždavinys, kurio tikslas sukurti moksliškai pagrįstą vertinimo LMIO schemą, atitinka DSA uždaviniams keliamus reikalavimus ir gali būti sprendžiamas DSA teorijos metodais.

Svarbiausias šios skyriaus tikslas: išanalizuoti DSA procesą, metodus ir parinkti tinkamus nagrinėjamam uždaviniui. Yra įvairių požiūrių kaip DSA procesą išskaidyti į etapus. Išanalizavę DSA literatūrą ir nagrinėjamos problemos specifiką, pasirinkome variantą, kai DSA procesas yra išskaidomas į tris etapus: problemos sisteminimas, problemos modeliavimas ir modelio analizavimas.

Galutinė pirmojo etapo, problemos sisteminimo, išdava turi būti konkretus alternatyvų, vertinimo kriterijų ir juos atitinkančių metrikų sąrašas, kurį turint jau galima kurti problemos modelį, t. y., ieškoti tinkamų DSA algoritmų. Įšanalizavę įvairius problemų sisteminimo būdus, nagrinėjamojo uždavinio sisteminimui parinkome Tikslų/ Klausimų/ Metrikų metodą. Šio metodo esmė yra ta, kad pirmiausia apibrėžiamas tikslas, po to jis transformuojamas į klausimų aibę. Tuomet nustatomos metrikos, kurios skirtos pateikti atsakymus į minėtus klausimus. Šis metodas numato hierarchinę tikslų, klausimų, metrikų struktūrą ir yra orientuotas į programinės įrangos kokybės vertinimą.

Problemos modeliavimo metu parenkamas konkretus DSA matematinis metodas. Kuriamai LMIO vertinimo schemai yra iškelti papildomi reikalavimai, į kuriuos būtina atsižvelgti parenkant matematinį DSA metodą. Papildomi reikalavimai sako, kad neužtenka išrikiuoti alternatyvas (t. y. pateiktus sprendimus), būtina pateikti kiekvienos alternatyvos, kiekvieno viešai skelbiamo dalinio vertinimo skaitinį įvertį. Turi būti galimybė naudoti lingvistinius alternatyvų įverčius (tais atvejais, kai taikomas holistinis, ne automatinis vertinimas). Taip pat itin svarbu,

kad tos vertinimo schemos dalys, kurios yra skelbiamos viešai, būtų kuo paprastesnės, kuo skaidresnės ir lengvai suprantamos varžybų dalyviams.

Todėl iš galimų DSA algoritmų pasirinkome *vertės matavimo teoriją*, kurios algoritmai numato, kad sukonstruojama funkcija, susiejanti kiekvieną alternatyvą su jos įverčiu (realiu skaičiumi). Alternatyvos ranguojamos remiantis šiais įverčiais.

Remiantis išvardintais reikalavimais, kaip priimtiniausią pasirinkome *svorinio sumavimo metodą* (SVSM). Tačiau tiesiogiai nagrinėjamam uždaviniui SVSM taikyti negalima, nes SVSM nenumatytas grupinis sprendimų priėmimas. Taip pat nenumatytas lingvistinių kintamųjų naudojimas.

Ieškojome svorinio sumavimo metodo praplėtimo, kuris būtų tinkamas minėtiems uždavinio ribojimams. Tinkamą algoritmą gavome sujungę *grupinių sprendimų palaikymo algoritmą* su *S. J. Chen* pasiūlytu metodu.

Prieš pristatydami šį algoritmą įvesime *neraiškiojo skaičiaus* sąvoką, nes šie skaičiai naudojami šiame algoritme.

Neraiškioji aibė apibrėžiama kaip aibė, kurios elementai turi skirtingus narystės aibėje laipsnius intervale $[0, 1]$, t. y., kiekvienam universaliosios aibės X poaibiui $\widetilde{Z}$ galima apibrėžti jo narystės neraiškiojoje aibėje funkciją: $\mu_{\widetilde{Z}} : X \to [0, 1]$.

Iškili, normalizuota, tolydi, apibrėžta aibėje $\mathbb{R}$ neraiškioji aibė vadinama *neraiškiuoju skaičiumi*.

Trapeciniai neraiškieji skaičiai yra neraiškieji skaičiai, aprašomi keturiais taškais $\widetilde{Z} = (z_1, z_2, z_3, z_4)$ tokiu būdu:

$$\mu_{\widetilde{Z}}(x) = \begin{cases} 0, & x < z_1 \\ \frac{x - z_1}{z_2 - z_1}, & z_1 \leq x \leq z_2 \\ 1, & z_2 \leq x \leq z_3 \\ \frac{a_4 - x}{z_4 - z_3}, & z_3 \leq x \leq z_4 \\ 0, & x > z_4 \end{cases} \tag{1}$$

Jei $z_2 = z_3$, tuomet neraiškusis skaičius vadinamas *trikampiu neraiškiuoju skaičiumi*.

Toliau trumpai pristatysime aukščiau minėtą algoritmą, gautą sujungus *grupinių sprendimų palaikymo algoritmą* su *S. J. Chen* pasiūlytu metodu.

Visur, kur algoritme naudosime terminą *lingvistiniai įverčiai* laikysime, kad prieš tai parenkamos tinkamos lingvistinės skalės iš DSA literatūroje pateiktų skalių ir lingvistiniai įverčiai imami iš tų skalių. Tokios skalės pavyzdys pateiktas 1 lentelėje.

Tegul $A = \{A_1, A_2, \cdots A_n\}$ yra alternatyvų aibė, $C = \{C_1, C_2, \cdots C_m\}$ yra kriterijų aibė, $D = \{D_1, D_2, \cdots, D_t\}$, $t \geq 2$ yra sprendimų priėmime dalyvaujančių asmenų (toliau tekste *asmenų*) aibė, n yra alternatyvų skaičius, m yra kriterijų skaičius. Kiekvienam asmeniui priskiriamas jo svarbumo lingvistinis įvertis $\widetilde{p} = \{\widetilde{p_1}, \widetilde{p_2}, \cdots, \widetilde{p_t}\}$.

Kiekvienas asmuo kiekvienam kriterijui priskiria lingvistinį jo svorio įvertį: $\widetilde{w}^k = \{\widetilde{w}_1^k, \widetilde{w}_2^k, \cdots, \widetilde{w}_m^k\}$, $k = 1, 2, \cdots, t$.

Pažymėkime $v_j^k(A_i)$ dalinę funkciją, kurios argumentas yra alternatyvos A_i įvertis kriterijaus C_j atžvilgiu, nustatytas asmens D_k. Čia $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$, $k = 1, 2, \cdots, t$. Jei kriterijų metrikos yra objektyvios, tuomet dalinių funkcijų argumentai yra realieji skaičiai ir yra vienodi visiems k, t. y. asmenys neturi įtakos šiems įverčiams. Jei kriterijų metrikos subjektyvios, tuomet šie argumentai yra lingvistiniai.

Naudojantis DSA literatūroje pateiktomis lentelėmis, pasirinktosios lingvistinės skalės konvertuojamos į trapecinius arba trikampius neraiškiuosius skaičius. Konvertavimo lentelės pavyzdys pateiktas 1 lentelėje ir iliustruotas 1 paveiksle.

Skalės elementas	**Neraiškusis skaičius**
Vidutinio svarbumo	$\big($0,3; 0,5; 0,7$\big)$
Svarbus	$\big($0,5; 0,7; 0,9$\big)$
Svarbesnis	$\big($0,7; 0,9; 1$\big)$
Svarbiausias	$\big($0,9; 1; 1$\big)$

1 lentelė. *Lingvistinės skalės konvertavimo į trikampius neraiškiuosius skaičius pavyzdys*

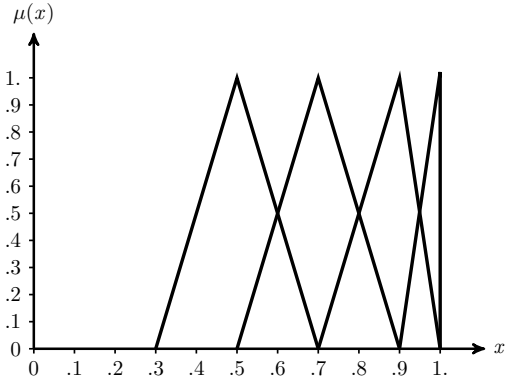
Tuomet kiekvienas neraiškusis skaičius konvertuojamas į realųjį toliau pateiktu būdu. Sakykime, turime trikampį neraiškųjį skaičių $\widetilde{Z} = (z_1, z_2, z_3)$. Apibrėžiamos dvi funkcijos: $\mu_{max}(x)$ ir $\mu_{min}(x)$:

$$\mu_{max}(x) = \begin{cases} x, & 0 \leq x \leq 1 \\ 0, & \text{kitu atveju} \end{cases} \tag{2}$$

$$\mu_{min}(x) = \begin{cases} 1 - x, & 0 \leq x \leq 1 \\ 0, & \text{kitu atveju} \end{cases} \tag{3}$$

Kairioji ir *dešinioji* $\widetilde{Z}$ reikšmės apibrėžiamos taip:

$$\mu_L(\widetilde{Z}) = \underset{x}{Sup}[\mu_{\widetilde{Z}}(x) \cap \mu_{min}(x)] \tag{4}$$



1 pav. *1 lentelėje pateiktos skalės iliustracija*

$$\mu_R(\widetilde{Z}) = \underset{x}{Sup}[\mu_{\widetilde{Z}}(x) \cap \mu_{max}(x)] \tag{5}$$

Realioji neraiškiojo skaičiaus $\widetilde{Z}$ reikšmė apibrėžiama taip:

$$\mu_T(\widetilde{Z}) = (\mu_R(\widetilde{Z}) + 1 - \mu_L(\widetilde{Z}))/2 \tag{6}$$

Tokiu būdu visi lingvistiniai įverčiai pakeičiami juos atitinkančiomis realiomis reikšmėmis: $\widetilde{w}_j^k$, $\widetilde{p}_t$, pakeičiami į atitinkamai w_j^k, p_t. Tuomet suskaičiuojami agreguoti kiekvieno kriterijaus svoriai:

$$w_j = \frac{\sum_{k=1}^{t} w_j^k p_k}{\sum_{k=1}^{t} p_k}, j = 1, 2, \cdots, m \tag{7}$$

Analogiškai suskaičiuojamos agreguotos dalinių funkcijų reikšmės kiekvienai alternatyvai kiekvieno kriterijaus atžvilgiu:

$$v_j(A_i) = \frac{\sum_{k=1}^{t} v_j^k(A_i) p_k}{\sum_{k=1}^{t} p_k} \tag{8}$$

Agreguoti galutiniai kiekvienos akternatyvos įverčiai apskaičiuojami remiantis tokia formule:

$$v(A_i) = \frac{\sum_{j=1}^{m} v_j(A_i) w_j}{\sum_{j=1}^{m} w_j} \tag{9}$$

Remiantis šiais įverčiais alternatyvos ranguojamos.

Paskutinysis DSA proceso etapas yra modelio analizavimas. Nagrinėjamo uždavinio atveju jis atliekamas išbandant vertinimo schemą su konkrečiais uždaviniais varžybų sąlygomis. Analizuojant modelį, skaičiuojamas jautrumas, t. y. apskaičiuojama, kiek mažiausiai turi pasikeisti svoriai, kad pasikeistų alternatyvų tarpusavio išrikiavimas. Apskaičiuojami *kritiškiausias kriterijus* bei *kritiškiausias sprendimų matricos įvertis*.

4. PUSIAU ATOMATINIS APK, SPRENDŽIANČIŲ UŽDAVINIUS SU GRAFAIS, VIZUALIZAVIMAS

Informatikos varžybose nuolat keliamas klausimas dėl alternatyvių testavimui vertinimo formų, t. y. jų nebuvimo ar labai nedidelio naudojimo. Dažnai kitokios nei testavimas vertinimo formos nėra taikomos informatikos varžybose, nes jų praktinis realizavimas užimtų per daug laiko.

Šio skyriaus tikslas išanalizuoti APK, sprendžiančių uždavinius su grafas, vizualizavimo galimybes. Tokio vizualizavimo tikslas – supaprastinti ir pagreitinti APK analizavimą, tokiu būdu sudarant galimybes taikyti pusiau automatinį vertinimą ir į vertinimo schemą įtraukti papildomus kriterijus.

Uždavinius su grafais pasirinkome todėl, kad grafas savaime yra vizuali duomenų struktūra ir šių struktūrų vizualizavimas tiesiogiai palengvintų APK realizuotų algoritmų suvokimą. Be to, uždaviniai su grafais sudaro didelį procentą (apie 20%) LMIO ir kitose mūsų tirtose informatikos varžybose.

Egzistuoja dvi algoritmų vizualizavimo paradigmos. Pirmoji paremta *įdomių įvykių* (angl. *interesting event*) paieška. Taikant šią paradigmą, reikia gerai išmanyti vizualizuojamo APK sandarą, identifikavus įdomius įvykius, į APK tekstą reikia įterpti kreipinius į vizualizavimo procedūras. Antroji paradigma, pavadinta *būsenų žymėjimas* (angl. *state mapping*) nereikalauja nei gerai išmanyti vizualizuojamo APK struktūrą, nei modifikuoti APK tekstą. Vizualizacija sukuriama automatiškai remiantis APK kintamųjų reikšmėmis.

Vertintojai, kuriems tenka analizuoti APK, nėra jų autoriai, t. y. įdomių įvykių ieškojimas užtruktų nemažai laiko, o taip pat informatikos varžybose laikomasi požiūrio, kad dalyvių sukurti APK neturi būti modifikuojami. Taigi įvertinome, kad APK su grafais vizualizavimui labiau tinka *būsenų žymėjimo* paradigma.

Yra sukurta daug mokymosi įrankių, skirtų grafų vizualizavimui, tačiau jie remiasi pirmąja paradigma. Varžybose dalyvis gali pasirinkti bet kokią jam priimtiną (nebūtinai racionalią ar logišką) duomenų struktūrą

grafui vaizduoti. Vizualizavimas būtų galimas tuomet, jei paaiškėtų, kad dalyviai grafo vaizdavimui naudoja nedidelį skaičių skirtingų būdų.

Norėdami tai nustatyti, pasirinkome tris skirtingais metais LMIO finale pateiktus uždavinius su grafais ir išanalizavome visus 191 APK, kurie buvo pateikti olimpiadų metu kaip tų trijų uždavinių sprendimai.

Kiekvienas pateiktas APK buvo išanalizuotas ir surinkti duomenys apie tai, kokiais būdais jame realizuoti grafai. Grafų realizavimo programose būdus suskirstėme į penkias kategorijas (gretutinumo sąrašai, gretutinumo matricos, briaunų sąrašai, aibėmis paremtos realizacijos, dvidaliai grafai), kurias padalinome į dar smulkesnes kategorijas. Nors dalyvių pateiktuose sprendimuose, radome keletą gana neįprastų grafo realizavimo būdų, tačiau nustatėme apie dešimt populiariausių grafo realizacijos būdų. Daugumoje pateiktų sprendimų buvo panaudoti vienas ar keli iš šių dešimties būdų.

Pastebėjome įdomią tendenciją. Sudėtingesnius grafus dalyviai linkę surinkti iš komponentų (atskirų dalių) vietoje to, kad programuoti vieną sudėtingą grafą realizuojančią duomenų struktūrą. Išskyrėme trijų tipų komponentus. Pirmasis tipas tai – *grafo tipo komponentas*. Toks komponentas gali būti traktuojamas kaip atskiras grafas, arba gali būti prijungiamas prie kito jau egzistuojančio grafo. Antrasis tipas tai *viršūnių tipo komponentas*. Jis naudojamas, kai reikia saugoti tam tikrus duomenis apie viršūnes, t.y. kiekvienai viršūnei priskiriama tam tikra vertė. Šie komponentai gali būti prijungiami prie grafo tipo komponentų. Trečiojo tipo komponentai yra *viršūnių sąrašo tipo komponentai*. Šie komponentai taip pat gali būti prijungiami prie grafo tipo komponentų. Tačiau skirtingai nuo viršūnių tipo komponentų viršūnėms nėra saugomos papildomos reikšmės, o įtraukimas į viršūnių sąrašo tipo komponentą jau reiškia, kad viršūnė pasižymi tam tikra savybe.

Norėdami išbandyti vizualizavimo galimybes praktiškai, pasirinkome tris populiariausius grafo realizavimo būdus (gretutinumo sąrašai realizuoti įrašų masyvu, gretutinumo sąrašai realizuoti dvimačiu masyvu, gretutinumo matrica realizuota dvimačiu masyvu) ir juos realizavome sukurdami demonstracinį vizualizavimo įrankį. Šiame įrankyje taip pat realizavome galimybę konstruoti grafus iš minėtų trijų tipų komponentų. Įrankis veikia kaip derintuvės aplinka, kurioje grafų struktūros nurodomos kaip vizualizuojami stebimi reiškiniai. Stebimų reiškinių sąrašas pasirenkamas iš galimų (t.y. realizuotų) grafų realizacijos tipų.

Irankį išbandėme praktiškai ir įvertinome, kad tokį (t. y. sukurtą remiantis šiame tyrime pateiktomis idėjomis) įrankį labai paprasta naudoti ir jis tikrai atvertų papildomas galimybes pusiau-automatiniam vertinimui, o tuo pačiu ir papildomų vertinimo kriterijų įtraukimui.

Šiame skyriuje aprašytas tyrimas buvo atliktas pačioje doktorantūros studijų pradžioje, kai didžioji dauguma LMIO dalyvių (apie 90%) programavo Paskalio kalba. Tad visas tyrimas atliktas analizuojant Paskalio kalba parašytas programas. Sukurtasis demonstracinis įrankis taip pat pritaikytas Paskalio kalbai. Per tuos metus situacija pasikeitė, LMIO dalyvių tarpe ėmė dominuoti C/C++ kalbos. Tad norint praktiškai naudoti įrankį, jį reiktų pritaikyti šioms programavimo kalboms.

Tuo tarpu šiame skyriuje aprašyto tyrimo rezultatą pateikiame pirmiausia kaip teorinį, parodantį, kad pusiau-automatinis APK, sprendžiančių uždavinius su grafais, vizualizavimas yra galimas ir jis sudaro galimybes informatikos varžybose taikyti pusiau automatinį vertinimą.

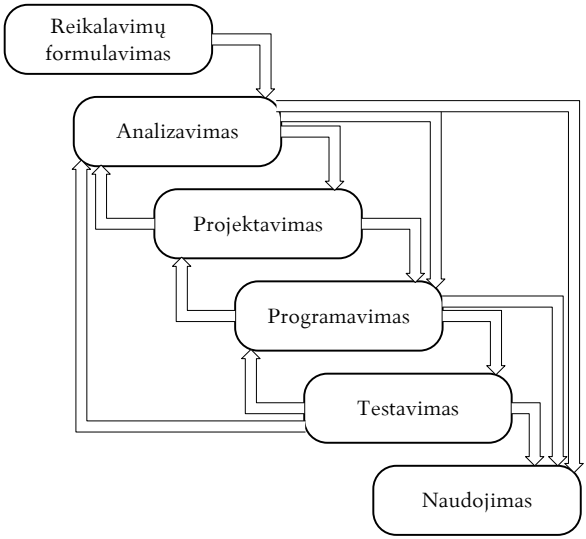
5. VERTINIMAS PROGRAMINĖS ĮRANGOS KOKYBĖS STANDARTŲ ATŽVILGIU

Svarbiausias šio skyriaus tikslas išnagrinėti šiuo metu LMIO taikoms vertinimo kriterijus programinės įrangos kokybės standartų (konkrečiai ISO–9126–1) atžvilgiu.

Pirmiausia palyginome APK ir programinės įrangos gyvavimo ciklus. Lyginimui pasirinkome krioklio gyvavimo ciklo modelį, nes laikoma, kad šis modelis yra itin tinkamas tokiais atvejais, kai yra gerai apibrėžta vartotojo sąsaja. Pateikiami sprendimai kaip tik pasižymi šia savybe. APK krioklio gyvavimo ciklo modelis pateiktas 2 paveiksle. Atkreipiame dėmesį, kad yra itin trumpa naudojimo fazė, todėl nėra poreikio palaikymui.

Kokybės modelis ISO-9126-1 numato šešias programinės įrangos kokybės charakteristikas: funkcionalumą, patikimumą, panaudojamumą, efektyvumą, pataisomumą ir perkeliamumą (angl. *functionality, reliability, usability, efficiency, maintainability, portability*). Nustatėme, kad į dabartinę LMIO vertinimo schemą įtrauktos trys charakteristikos: funkcionalumas, efektyvumas ir pataisomumas. Išnalizavę varžybų užduočių specifiką, nustatėme, kad netikslinga į vertinimo schemą įtraukti likusias tris charakteristikas.

LMIO vertinimo schemoje numatyta, kad funkcionalumo ir efektyvumo įvertinimai gaunami remiantis testavimo rezultatais. Testus ruošiantys uždavinių autoriai apgalvoja galimus klaidingus ir teisingus, mažiau ar daugiau efektyvius uždavinio sprendimus, juos suskirsto į kategorijas ir kiekvienai kategorijai priskiria norimą taškų kiekio intervalą. Tuomet konstruojami testai ir taškai už juos paskirstomi taip, kad pateikiami sprendimai, priklausantys atitinkamai kategorijai, gautų tą kategoriją atitinkantį taškų skaičių. Tačiau yra žinoma, kad bendru atveju testavimas nėra patikimas įrankis programos funkcionalumui įvertinti. Todėl



2 pav. *APK gyvavimo ciklo modelis*

ir yra keliamas klausimas, kiek patikimi testavimo rezultatais paremti APK kokybės įverčiai.

Norėdami tai įvertinti, parinkome vieną LMIO'2008 baigiamojo etapo uždavinį, kuriam sprendimus pateikė 160 dalyvių. Išanalizavus visus APK, galimi sprendimai buvo suklasifikuoti į keturias kategorijas: neteisingi algoritmai, dalinį uždavinį sprendžiantys teisingi algoritmai, teisingi, bet neefektyvūs algoritmai, teisingi ir efektyvūs algoritmai. Neteisingi algoritmai buvo papildomai suskirstyti į keturias kategorijas: nebaigti sprendimai, sprendimai, generuojantys atsitiktinius sprendinius, euristiniai sprendimai ir kiti sprendimai. Atkreipiame dėmesį, kad kategorijos sudarytos atsižvelgiant į dalyvio pasirinkto algoritmo tinkamumą uždaviniui spręsti, tačiau nekreipiant dėmesio į tai, ar sėkmingai pavyko realizuoti algoritmą.

Išanalizavus visuose APK realizuotus algoritmus ir atsižvelgiant į realizacijos kokybę, kiekvienam APK buvo priskirtas priimtinas įvertinimas, t. y. taškų intervalas.

Kiekvienai kategorijai priklausantys pateikti sprendimai buvo lyginami atskirai ir analizuojama, ar įvertinimas, gautas remiantis testavimo rezultatais priklauso priimtinam intervalui, o jei ne – kiek nuo jo nukrypsta.

Paties testavimo rezultatai yra loginiai, t. y. testas arba įveiktas, arba ne. Šie loginiai rezultatai transformuojami į taškus naudojant testavimo rezultatų agregavimo funkcijas. Taigi, galutinis rezultatas (už funkcionalumą ir efektyvumą skiriamas taškų skaičius) priklauso ne tik nuo testavimo rezultatų, bet ir nuo agregavimo funkcijos. LMIO iki šiol būdavo taikomas *dalinio agregavimo funkcija* (angl. *partial scoring*), tačiau IOI pereita prie *viskas-arba-nieko grupinio agregavimo funkcijos* (angl. *all-or-nothing batch scoring*). Tyrimo metu testavimo rezultatams taikėme abi šias funkcijas ir lyginome, kuri funkcija pateikia rezultatus artimesnius priimtinam intervalui. Priimtinus intervalus uždavinio kūrimo metu parenka vertinimo komisija, atsižvelgiant į juos kuriami testai bei siūlomi testų įverčiai. Duomenis apie vertinimo rezultatus, nepatekusius į priimtinus intervalus, taikant šias dvi funkcijas pateikti 2 lentelėje.

	Dalinis agregavimas	*Viskas arba nieko grupinis agregavimas*
Įvertinimas nepatenka į priimtiną intervalą	20,2%	9,1%
Įvertinimas nutolęs nuo priimtino intervalo per 20 ar daugiau taškų	8,4%	5,8%

2 lentelė. *Įvertinimų, gautų naudojant dalinio vertinimo bei viskas-arba-nieko grupinio agregavimo schemas, nukrypimas nuo priimtinų rezultatų intervalų.*

Tyrimas parodė, kad taikant ir vieną, ir kitą dalinės vertės funkcijas, gauti įverčiai (taškai) nukrypsta nuo priimtino intervalo. Tačiau taikant *viskas-arba-nieko grupinę agregavimo funkciją* nukrypimas yra mažesnis, ir, mūsų nuomone, testavimas kartu su šia dalinės vertės funkcija gali būti taikomas LMIO vertinime.

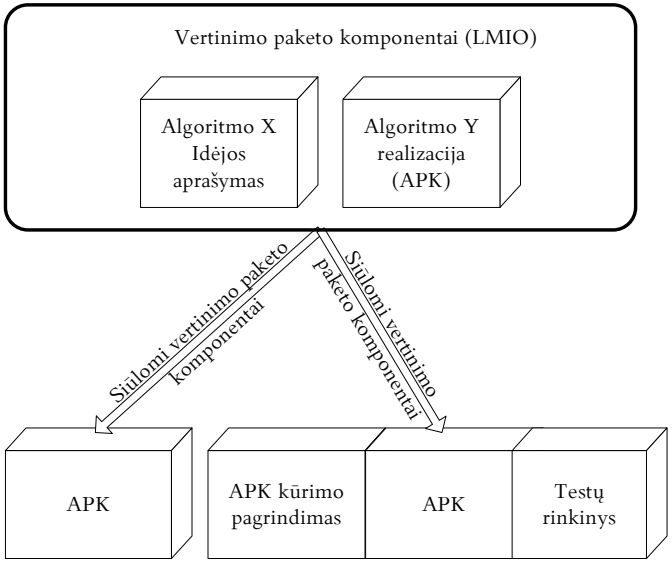
Pataisomumą informatikos varžybose atitinka *programavimo stilius*. LMIO programavimo stilių holistiškai vertina vertinimo komisijos nariai. Tyrimui pasirinkome vieną LMIO uždavinį, kuriam dalyviai pateikė 130 sprendimų. Kiekvieną APK bei kartu pateiktus sprendimo idėjos aprašymus išanalizavome, t. y. nustatėme, kokį algoritmą dalyvis bandė realizuoti savo APK. Taip pat įvertinome kiekvieno APK programavimo stilių. Apskaičiavę koreliaciją tarp algoritmo realizavimo APK kokybės ir APK programavimo stiliaus kokybės, gavome, kad koreliacija lygi 0.468. Tai rodo, kad geresnį programavimo stilių praktikuojantys daly-

viai sėkmingiau realizuoja savo idėjas. Kadangi vienas iš LMIO tikslų yra geros programavimo praktikos skleidimas mokinių ir mokytojų tarpe, todėl, mūsų manymu, yra tikslinga programavimo stiliaus vertinimą įtraukti į LMIO vertinimo schemą.

6. LMIO VERTINIMO SCHEMOS KŪRIMAS TAIKANT DAUGIAKRITERIŲ SPRENDIMŲ ANALIZĘ

Pirmasis DSA proceso etapas yra problemos sisteminimas, kuris apima įvairių su sprendžiama problema susijusių aspektų analizavimą. Todėl šiam etapui priskiriame antrą bei penktą disertacijos skyrius.

Galutinis šio etapo rezultatas turi būti alternatyvų (jos jau žinomos – dalyvių pateikti sprendimai) bei kriterijų sąrašas. Sisteminimo etapo rezultatui pasiekti pasirinkome Tikslų/Klausimų/Metrikų metodą. Taikydami metodą, pakvietėme dalyvauti dešimt informatikos varžybų ekspertų. Sudarėme klausimyną, kuriame nurodėme LMIO resursus bei ribojimus, remdamiesi Tikslų/Klausimų/Metrikų metodu sudarėme sprendinio-atributų-metrikų medį ir paprašėme ekspertų jį užpildyti, t. y. pasiūlyti vertinimo paketo sudėtį, jo matuojamus atributus, bei konkrečias metrikas, matuojančias kiekvieną atributą.



3 pav. *Ekspertų siūloma vertinimo paketo sudėtis*

Pirmasis hierarchijos lygmuo yra vertinimo paketo sudėtis. Ekspertų nuomonės šiuo klausimų išsiskaidė į dvi vienodo dydžio grupes (žr. 3 pav.). Pirmoji grupė akcentavo APK ir greitą automatizuotą vertinimą, todėl siūlė į vertinimo paketą įtraukti tik APK. Antroji ekspertų grupė akcentavo uždavinio sprendimą, kaip algoritmą bei uždavinio sprendimą kaip programinę įrangą, todėl siūlė į vertinimo paketą traukti ne tik APK, bet ir *APK kūrimo pagrindimą* bei testų rinkinį. Ekspertų nuomonių pasidalijimas atsipindi vyraujančius skirtingus požiūrius informatikos varžybų bendruomenėse, o tuo pačiu ir mokslinėse publikacijose.

LMIO tikslas skatinti gerą programavimo praktiką bei akcentuoti abu – algoritmų bei programų kūrimo – aspektus nulėmė, kad pasirinkome platesnę pateikiamo sprendimo sampratą.

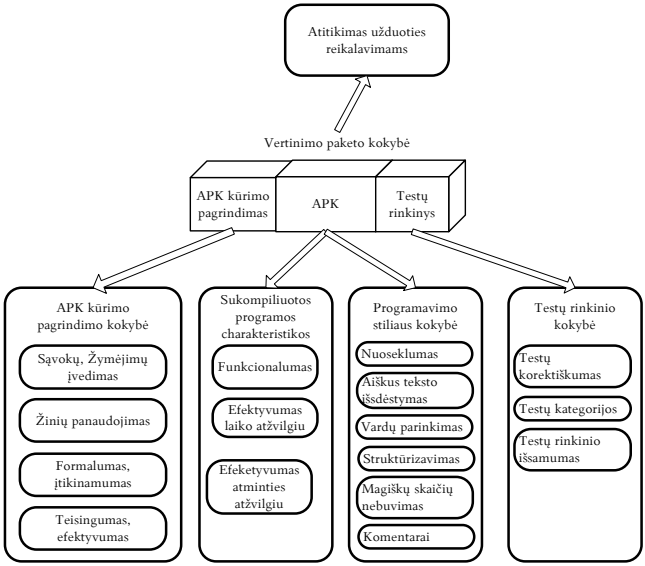
Remdamiesi pasirinkta vertinimo paketo sudėtimi bei ekspertų atsakymais, sudarėme antrąjį hierarchinio medžio lygmenį – matuojamų vertinimo paketo atributų sąrašą (žr. 4 pav.). Palyginus su iki šiol naudota LMIO vertinimo schema, atsirado du nauji atributai: testų rinkinio kokybė bei atitikimas užduoties reikalavimams.

Sprendimo idėjos aprašymas pakeistas *APK kūrimo pagrindimu*. Jei sprendimo idėjos aprašymas galėjo būti visiškai nesusijęs su APK, tai APK kūrimo pagrindimas privalo atspindėti konkretų APK. Taip pat numatyta vertinti ne tik aprašomo algoritmo teisingumą ir efektyvumą, bet ir formalumą, įtikinamumą, žinių panaudojimą.

Trečiasis hierarchinio medžio lygmuo yra kriterijų lygmuo, kuriuos irgi parinkome išanalizavę ekspertų pasiūlymus. Skyriuje išanalizuota, kaip ir kodėl kiekvienam atributui parinkti konkretūs kriterijai, nurodytas kiekvieno kriterijaus matavimo būdas (automatinis ar ne automatinis), bei matavimui naudojamos skalės tipas (loginė, ranginė, santykių).

Pasiūlytasis sprendimo/atributų/kriterijų medis yra abstraktus. Norint jį naudoti konkrečiam uždaviniui, reikia atsižvelgti į uždavinio specifiką, uždaviniui keliamus tikslus, visą dalyviams pateikiamą užduočių rinkinį ir adaptuoti konkrečiai situacijai, t. y., nuspręsti kokia bus vertinimo paketo sudėtis (ar dalyviai turės pateikti visus komponentus, ar tik kai kuriuos), kurie atributai bus vertinami, kurie konkrečiai kriterijai bus taikomi kiekvieno atributo vertinimui.

Antrasis DSA proceso etapas yra problemos modeliavimas, t. y., tinkamo metodo parinkimas kai jau žinomas alternatyvų bei kriterijų sąrašas. Šis etapas buvo atliktas trečiame skyriuje, kuriame buvo pasirinktas *svorinio sumavimo metodo* išplėtimas, gautas sujungus *grupinių sprendimų palaikymo algoritmą* su *S. J. Chen* pasiūlytu metodu.



4 pav. *Siūlomas vertinimo paketo atributų bei juos atitinkančių kriterijų sąrašas*

Trečiasis DSA proceso etapas yra modelio analizė. Nagrinėjamo uždavinio atveju tai atitinka pasiūlytos vertinimo schemos išbandymą varžybų sąlygomis su konkrečiu uždaviniu bei jautrumo, t. y. kritiškiausio kriterijaus ir kritiškiausio sprendimų matricos įverčio apskaičiavimą. Tam tikslui parinkome vieną užduotį ir pateikėme grupei mokinių, nedidelėse varžybose.

Vertinimo schemos adaptavimo konkrečiam uždaviniui procesas susidėjo iš keleto žingsnių:

Lingvistinių skalių parinkimas. Reikia parinkti tris lingvistines skales: vertinimo komisijos narių svarbumo skalę, atributų svorių parinkimo skalę, skalę, naudojama ne automatiniam vertinimui. Pats algoritmas nenumato konkrečių skalių ir jų pasirinkimas yra intuityvus, todėl jas parinkome šiame DSA etape darydami prielaidą, kad vertinimo komisija ateityje gali pasiūlyti ir kitokias lingvistines skales. Remdamiesi 3 skyriuje aprašytu metodu kiekvieną skalę konvertavome į neraiškiuosius skaičius, o neraiškiuosius skaičius – į juos atitinkančius realiuosius skaičius.

Svorių parinkimas vertinimo komisijos nariams. Lingvistinius vertinimo komisijos narių svarbumo svorius parinkome konkretiems asmenims priklausomai nuo jų patirties ir dalyvavimo aktyvumo informatikos varžybose. Kiekvienam lingvistiniam svoriui nurodėme jį atitinkantį realųjį skaičių.

Lingvistinių svorių siūlymas pateikiamo sprendimo atributams. Kiekvienas vertinimo komisijos narys pasiūlė lingvistinius svorius pateikiamo sprendimo atributams. Kiekvienam lingvistiniam svoriui nurodėme jį atitinkantį realųjį skaičių.

Vertinimo šablono sudarymas. LMIO priimta vertinimo schemoje naudoti šablonus. Šabloną sudaro vertinimo paketo sudėtis, vertinamų vertinimo paketo atributų sąrašas bei kiekvieno atributo svoris. Šablonus sudarėme atsirinkę priimtinas vertinamų atributų kombinacijas, naudodami pasiūlytus svorius atributams bei 3 skyriuje aprašytu metodu. Gauti šablonai pateikti 3 lentelėje. Siekdami vertinimo schemą padaryti suprantamesnę dalyviams, gautus įverčius suapvalinome.

Vertinimo šablono parinkimas. Siekdami geriau išbandyti vertinimo schemą pasirinkome pirmąjį šabloną, kuriame numatytas visų atributų vertinimas.

Kriterijų parinkimas konkrečiam šablonui. Atsižvelgdami į uždavinio specifiką bei vertinimui turimus resursus, iš vertinimo schemoje numatytų kriterijų parinkome konkretų vertinimo kriterijų rinkinį.

Dalinės vertės funkcijų parinkimas. Kiekvienam kriterijui parinkome dalinės vertės funkciją, konvertuojančią pateikiamo sprendimo matavimo rezultatus kriterijaus atžvilgiu į taškus. Dalinės vertės funkcijos parinktos remiantis šios disertacijos ankstesnių skyrių medžiaga.

Tarpkriterinių dalinės vertės funkcijų parinkimas. Parinkome reikiamas tarpkriterines dalinės vertės funkcijas. Svarbiausia jų – programavimo stiliaus vertinimo susiejimas su kitų kriterijų vertinimo rezultatais. Tam apklausėme visus vertinimo komisijos narius. Gavome trijų tipų siūlymus. Tuomet funkcijos parinkimą sumodeliavome kaip nedidelį nepriklausomą DSA uždavinį ir jį išsprendėme taikydami DSA metodą vadinamą *analitiniu hierarchiniu procesu*. Suskaičiavę trijų pasiūlytų alternatyvų įverčius,

gavome, kad priimtiniausia funkcija tokia:

$$S_{galutinis\_stilius} = \begin{cases} S_{stilius}, & \text{jei } S_{smc} = S_{max\_smc}, \\ 0, & \text{if } S_{smc} < S_{max\_smc}. \end{cases} \quad (10)$$

Čia $S_{galutinis\_stilius}$ reiškia galutinį įvertį už programavimo stilių, $S_{stilius}$ reiškia APK programavimo stiliaus įvertį, S_{smc} reiškia kriterijaus *nedidelių teisingumo testų, tikrinančių paprasčiausius atvejus* įvertį, S_{max_smc} reiškia to paties kriterijaus maksimalų galimą įvertį. Kitaip sakant, programavimo stilius vertinamas tik tuo atveju, jei APK pilnai įveikia nedidelių teisingumo testų, tikrinančių paprasčiausius atvejus, rinkinį.

Šabl. nr.	**APK kūrimo pagrindimas**	**Testavimas**	**Programavimo stilius**	**Testų rinkinys**
1	0,23 (0,25)	0,39 (0,40)	0,17 (0,15)	0,21 (0,20)
2	—	0,50 (0,50)	0,22 (0,20)	0,28 (0,30)
3	0,28 (0,30)	0,47 (0,45)	—	0,25 (0,25)
4	0,29 (0,30)	0,49 (0,50)	0,22 (0,20)	—
5	—	0,65 (0,65)	—	0,35 (0,35)
6	0,38 (0,40)	0,62 (0,60)	—	—
7	—	0,70 (0,70)	0,30 (0,30)	—
8	—	1	—	—

3 lentelė. *Galimi vertinimo paketo šablonai. Suapvalinti svoriai pateikti skliausteliuose*

Visus išvardintus žingsnius atlikome adaptuodami vertinimo schemą konkrečiam uždaviniui, tačiau bendru atveju kai kurių skaičiavimų rezultatai gali būti panaudoti kelis kartus ir jų nereikia perskaičiuoti kiekvienam uždaviniui.

Bandomosiose varžybose dalyvavo 14 mokinių, 3 vertinimo komisijos nariai, buvo pateikti 4 uždaviniai, kurių vertinimui pasirinkti 3, 7 ir 8 šablonai (žr. 3 lentelę). Dalyvių pateikti vertinimo paketai buvo įvertinti naudojant pasiūlytąją vertinimo schemą. Apibendrinus rezultatus įvertinta, kad pasiūlytoji vertinimo schema tinkama naudojimui LMIO. Taip pat buvo pateikti konkretūs siūlymai kaip geriau adaptuoti vertinimo schemą konkretiems uždaviniams, pvz. pasiūlyta pakeisti vieną iš trijų skalių.

IŠVADOS

1. Šiuo metu informatikos varžybose taikoma vertinimo schema yra tobulintina.

   Programavimo užduočių pateikiamų sprendimų vertinimo aukštosiose mokyklose patirtis daugeliu atveju nėra pritaikoma informatikos varžybose, nes skiriasi užduočių parengimo bei vertinimo tikslai, metodika, sudėtingumas.

2. Užduotys su grafais tinkamos pusiau automatiniam vertinimui taikant vizualizaciją.

3. Vertinimo paketo kokybę nusakantys atributai, įtraukti į LMIO vertinimo schemą, yra pagrįsti ISO-9126-1 požiūriu. Juodosios dėžės testavimo pagrindu gaunamų APK kokybės įverčių tikslumas siekia 80% ar daugiau priklausomai nuo taikomos rezultatų agregavimo funkcijos.

4. Daugiakriterių sprendimų teorijos metodais sukurta vertinimo schema yra tinkama naudoti LMIO.

INFORMACIJA APIE DISERTACIJOS AUTORĘ

Jūratė Skūpienė gimė 1970 m. lapkričio 17 d. Vilniuje.

1988 m. baigė Vilniaus m. 22-ąją vidurinę mokyklą. 1988–1993 m. studijavo Vilniaus universiteto Matematikos fakultete taikomąją matematiką ir 1993 m. įgijo matematiko kvalifikaciją.

2006–2010 m. studijavo jungtinėje Vilniaus universiteto Matematikos ir informatikos instituto bei Vytauto Didžiojo universiteto doktorantūroje.

SUMMARY

Statement of the Problem and its Relevance

Informatics contests for high school students is a fast growing extra-curricula activity in many countries. In such contests, the contestants are given an algorithmic problem and have to design an algorithm, to implement it, and submit as a working program, but are not required to submit the proof of algorithm correctness.

Thus we get the concept of *an algorithm-code complex*, i.e., the program which contains the implementation of an unknown algorithm designed to solve the given task. The current practice of automated evaluation in informatics contests is questionable, because it does not reveal the implemented algorithm, and it is not known how much the testing results conform with the expectations of the task designers.

The main topic of investigation in the thesis is evaluation in informatics contests, in particular in Lithuanian Informatics Olympiads (LitIO).

Research Objectives and Tasks

The objectives of this dissertation are: to investigate the evaluation criteria and the evaluation schemes applied in the evaluation of algorithm-code complexes; to develop the evaluation scheme based on the multiple criteria decision analysis methods, suitable to be applied in LitIO.

The tasks of the dissertation are as follows:

1. To present a survey of evaluation practice and problems in informatics contests. To analyse the evaluation of submissions to programming assignments in undergraduate studies. To determine whether this experience could be transferred to informatics contests.

2. To analyse the possibilities of including a semi-automated evaluation in informatics contests using visualisation of algorithm-code complexes (the case of tasks with graphs).

3. To analyse a chosen set of algorithm-code complexes, to determine the precision of measurements of the quality of algorithm-code complexes based on the testing results. To establish whether the the quality of the algorithm implementation in an algorithm-code complex is related to the quality of the programming style.

4. To define the evaluation in informatics contests as a multiple criteria decision problem. To analyse the multiple criteria decision process and methods, and to propose methods suitable for solving the evaluation problem.

5. Using the MCDA methods, to construct the evaluation scheme for LitIO, consisting of the list of components of a submission, the list of its measurable attributes, the list of evaluation criteria for each attribute, and the score aggregation function.

Research Approbation and Publications The main results of the research were presented in 10 Lithuanian and international conferences. The results were published in 10 scientific publications: 6 papers in the periodical reviewed journals, 4 papers in the proceedings of the conferences.

Research Scope and Structure The thesis is published in English. It consists of 6 chapters, glossary, the list of references, and appendix. The scope of thesis is 176 pages, 38 tables, 45 figures, 194 sources are quoted.

The first chapter is the introduction. It contains the problem statement and its relevance, research objectives and tasks, methods applied in the research, the findings and results, scientific novelty of the research, as well as the list of publications.

The second chapter presents a survey of the problematics of evaluation of algorithm-code complexes, the current LitIO evaluation scheme, and an overview of scoring schemes applied in other informatics contests. The concerns regarding black-box testing are identified and structured. The development and trends of the evaluation of solutions to programming assignments are presented and compared with that in informatics contests.

In the *third chapter*, we describe the problem of evaluation in LitIO as a multiple criteria decision analysis (MCDA) problem. We survey the stages of the MCDA process and various MCDA approaches for solving MCDA problems, among them the methods that apply the fuzzy logic and group decision making methods. We elicit the methods that are most suitable for solving the evaluation in LitIO problem.

The fourth chapter investigates the possibilities for a semi-automated evaluation by applying visualisation of graphs implemented in the algorithm-code complexes, designed by the contestants.

In *the fifth chapter*, the life cycles of a submission and of software are compared using the waterfall life cycle model. The current LitIO

evaluation scheme is analysed from the point of view of existing quality standards, in particular, the ISO-9126-1 software quality model.

The *sixth chapter* presents all the three required phases of the MCDA process: problem structuring, modelling, and model analysis. The outcome of this chapter is an evaluation scheme suggested to be used in LitIO.

CONCLUSIONS

1. The evalutation scheme currently applied in LitIO should be improved.

   The practice of evaluating the solutions to programming assignments in undergraduate courses in many cases is not applicable in informatics contests, because of different goals, different task complexity, and different evaluation methods.

2. The tasks with graphs are suitable for a semi-automated evaluation conducted by applying visualisation of the graphs, implemented in the algorithm-code complexes.

3. The attributes that describe the quality of an algorithm-code complex and are included into the current evaluation scheme are valid in terms of the ISO-9126-1 quality model. The precision of measurements of the quality of algorithm-code complexes, obtained by using black-box testing, is 80% or more depending upon the scoring function.

4. The evaluation scheme, developed applying the MCDA methods is suitable for LitIO.

INFORMATION ON THE AUTHOR OF THE DISSERTATION

Jūratė Skūpienė was born on 17 November 1970 in Vilnius, Lithuania.

In 1988 she graduated from Vilnius Secondary School No. 22. From 1988 to 1993 she has studied applied mathematics at the Faculty of Mathematics and graduated cum laude.

2006–2010 PhD studies at the Institute of Mathematics and Informatics of Vilnius University.

Jūratė SKŪPIENĖ

EVALUATION OF ALGORITHM-CODE COMPLEXES IN INFORMATICS CONTESTS

Summary of Doctoral Dissertation
Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

Jūratė SKŪPIENĖ

ALGORITMŲ IR JUOS REALIZUOJANČIŲ PROGRAMŲ VERTINIMAS INFORMATIKOS VARŽYBOSE

Daktaro disertacijos santrauka
Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

2010 09 27, 2 sp. l. Tiražas 60 egz.
Parengė spaudai ir išleido
Vilniaus universiteto Matematikos ir informatikos institutas
Akademijos g. 4, LT-08663 Vilnius
Interneto svetainė: http://www.mii.lt
Spausdino „Kauno technologijos universiteto spaustuvė“
Studentų g. 54, LT-51424 Kaunas