KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
SISTEMINĖS ANALIZĖS KATEDRA

Algimantas Ūsas

MAKSIMALAUS SRAUTO TINKLE RADIMO ALGORITMŲ ANALIZĖ

Magistro darbas

Vadovas
doc. dr. K. Plukas

KAUNAS, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
SISTEMINĖS ANALIZĖS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil. dr. R. Barauskas
2005-01-10

MAKSIMALAUS SRAUTO TINKLE RADIMO ALGORITMŲ ANALIZĖ

Informatikos magistro baigiamasis darbas

Vadovas
doc. dr. K. Plukas
2005-01-10

Recenzentas
doc. dr. R. Gaidys
2005-01-10

Atliko
IFN-2 gr. stud.
A. Ūsas
2005-01-10

KAUNAS, 2005

SUMMARY

Graphs are a pervasive data structure in computer science, and algorithms for working with them are fundamental to the field. There are hundreds of interesting computational problems defined in terms of graphs. This time we'll touch one of them – the problem of maximum net flow.

One of the main problems in the theory of graphs is the problem of maximum flow. The purpose is to calculate the biggest amount of matter, which can be relayed from the source to the flow. This is one of the easier tasks, which can be solved with the help of algorithms. Beside that the basic algorithms of maximum flow can be used for solving other problems about net flows.

The results of this work are:

- created the program equipment, realizing four classic methods of calculating maximum flow – Ford-Falkerson, Edmonds-Karp, Dinic and Goldberg;
- with the help of this program equipment is collected the statistic of these methods efficiency. Results showed, that every method can be realized within shorter time as was proved earlier;

TURINYS

ĮŽANGA

Grafai yra plačiai kompiuterių moksle paplitusi duomenų struktūra, ir darbo su grafais algoritmai yra užima svarbią vietą šioje sferoje. Daugybė įdomių skaičiavimo uždavinių apibrėžiama grafų teorijos sąvokomis. Šiame darbe bus paliestas vienas iš jų – maksimalaus srauto tinkle uždavinys.

Orientuotas grafas gali būti interpretuojamas kaip "tinklas, kuriuo teka tam tikras srautas" ir panaudotas rasti atsakymus apie materialius srautus. Laikykime, kad tam tikra materija juda sistema nuo šaltinio, kur ji atsiranda, į nuotakį, kur ji išnyksta. Šaltinis pastoviu kiekiu kuria materiją, o nuotakis tuo pačiu pastoviu kiekiu ją suvartoja. Materijos "srautas" bet kurioje sistemos vietoje intuityviai suprantamas kaip pratekėjusios materijos kiekis. Tinklai gali būti panaudojami modeliuoti skysčių tekėjimą vamzdžiais, detalių judėjimą surinkimo konvejeriu, srovę elektros grandine, informaciją ryšių kanalais ir taip toliau.

Kiekvienas tinklo lankas gali būti interpretuojamas kaip kanalas materijai tekėti. Kiekvienas kanalas turi tam tikrą fiksuotą pralaidumą, reiškiantį pratekančios materijos kiekį, pavyzdžiui 200 l skysčio per valandą arba 20 A srovės elektros laidu. Viršūnės yra kanalų susijungimo vietos ir, kitaip nei šaltinis ir nuotakis, materija prateka pro juos nesikaupdama. Kitaip tariant, į viršūnę atitekėjusios materijos kiekis turi būti lygus iš viršūnės ištekėjusios materijos kiekiui. Ši savybė vadinama srauto tvermės dėsniu (angl. flow conservation) ir atitinka Kirchhofo taisyklę elektros grandinėms.

Vienas svarbiausių grafų teorijos uždavinių yra uždavinys apie maksimalų srautą (arba maksimalaus srauto radimo tinkle uždavinys). Jo tikslas yra apskaičiuoti didžiausią kiekį materijos, kurį galima būtų perduoti iš šaltinio į nuotakį laikantis lankų pralaidumo apribojimų. Tai yra vienas paprastesnių uždavinių, liečiančių srautus tinkluose ir yra sprendžiamas įvairių algoritmų pagalba. Be to, baziniai maksimalaus srauto radimo algoritmai gali būti pritaikomi ir sprendžiant kitus uždavinius apie srautus tinkluose.

Pirmą kartą griežtai matematiškai suformuluotas Fordo ir Falkersono, maksimalaus srauto radimo uždavinys štai jau pusę amžiaus yra tyrinėjamas daugelio mokslininkų. Pagrindiniai šių tyrinėjimų tikslai yra šie:

1. efektyvesnių uždavinio sprendimo metodų sukūrimas;

2. "srautinio" požiūrio panaudojimas kitų kombinatorikos uždavinių sprendimui.

Maksimalaus srauto radimo algoritmų yra sukurta nemažai – pradedant klasikiniais šešiasdešimtųjų metų Danzigo, Fordo-Falkersono ir baigiant pastarųjų metų Diskretinės matematikos ir teorinio kompiuterių mokslo centro (DIMACS), vienijančio eilę Jungtinių Amerikos Valstijų universitetų bei laboratorijų, darbais.

Nežiūrint į tai, kad per šį gana ilgą laiko tarpą minėtas uždavinys buvo gerai išnagrinėtas ir jo sprendimui yra sukurta daug algoritmų, tyrimai šioje srityje vykdomi ir dabar. Reikia pažymėti, kad tinklams su sveikaskaitiniais ir ypač vienetiniais lankų pralaidumais pastaraisiais metais buvo gauti rezultatai, ženkliai pranokstantys anksčiau sukurtų algoritmų efektyvumą. Detali šių rezultatų apžvalga pateikta [10],[11]. Tuo tarpu tinklams su realiais lankų pralaidumais tokių rezultatų kol kas nėra.

Maksimalaus srauto radimo algoritmų efektyvumas vertinamas laiku, per kurį randamas uždavinio sprendinys. Jis išreiškiamas O-notacijų pagalba.

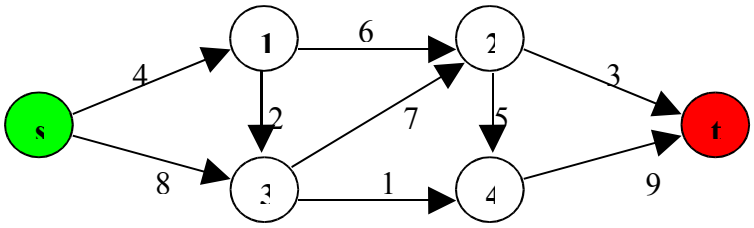
Šio darbo tikslas - sukurti programinę įrangą :

1. įgalinančią statistiškai įvertinti klasikinių maksimalaus srauto apskaičiavimo metodų – Fordo-Falkersono, Edmondso-Karpo, Dinico, Goldbergo – efektyvumą;
2. iliustruojančią šių algoritmų veikimą ir tinkamą naudoti mokymo procese.

1. MAKSIMALAUS SRAUTO PROBLEMA

1.1. Pagrindinės sąvokos

Tinklu (rus. сеть, angl. flow network) vadinamas orientuotas grafas $G=(V,E)$, kurio kiekvienam lankui $(u,v)\in E$ priskirtas skaičius $c(u,v)\geq 0$, vadinamas lanko **pralaidumu** (rus. пропускная способность, angl. capacity). Jei $(u,v)\notin E$, tai laikoma, kad $c(u,v)=0$. Grafe yra dvi ypatingos viršūnės: srauto pradžios viršūnė - **šaltinis** (rus. исток, angl. source) s ir srauto pabaigos viršūnė - **nuotakis** (rus. сток, angl. sink) t. Paprastumo dėlei laikoma, kad grafe nėra "nenaudingų" viršūnių (kiekviena viršūnė $v\in V$ priklauso kažkuriam keliui $s \to v \to t$.



1 pav. Tinklas

Srautu (rus. поток, angl. flow) grafe $G=(V,E)$ vadinama funkcija $f: V\times V\to R$, pasižyminti šiomis savybėmis:

- $f(u,v) \leq c(u,v) \quad \forall\, u\in V, \forall\, v\in V;$
- $f(u,v) = -f(v,u) \quad \forall\, u\in V, \forall\, v\in V;$
- $\sum_{v\in V} f(u,v) = 0 \quad \forall\, u\in V-\{s,t\}.$

Reikšmė $f(u,v)$ gali būti kaip teigiama, taip ir neigiama. Ji nusako, kiek srauto juda nuo viršūnės u link viršūnės v. Neigiama reikšmė nusako judėjimą priešinga kryptimi.

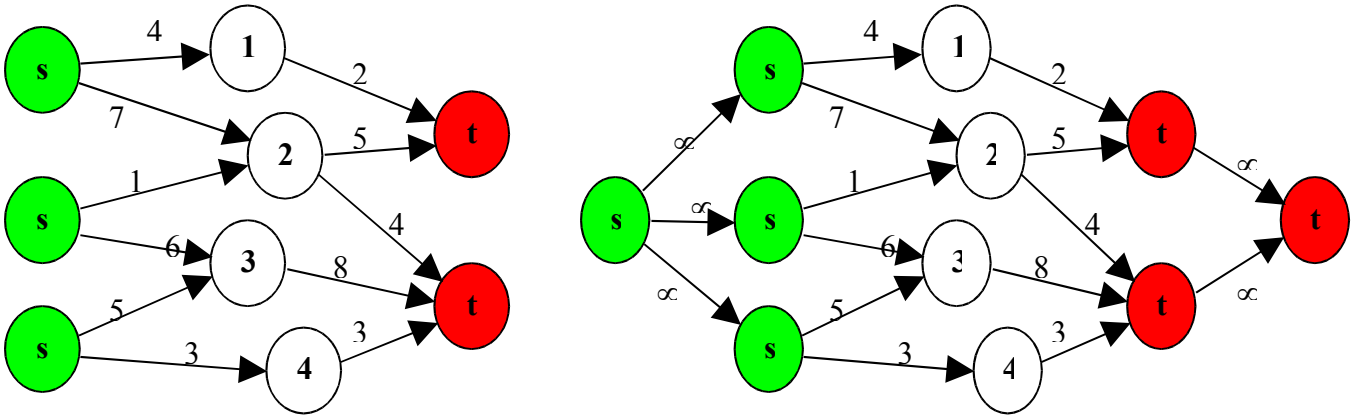
Srauto dydis (rus. величина, angl. value) f apibrėžiamas kaip suma

$$\sum_{v\in V} f(s,v)$$

(sudedami visų lankų, išeinančių iš pradinės viršūnės, srautai). Jis žymimas $|f|$.

Maksimalaus srauto uždavinys (rus. задача о максимальном потоке, angl. maximum flow problem) yra rasti didžiausią srautą duotajam tinklui *G* su šaltiniu *s* ir nuotakiu *t*. Maksimalaus srauto apskaičiavimo uždavinys yra tiesinio programavimo uždavinys, todėl jam gali būti taikomi tiesinio programavimo uždavinių sprendimo metodai. Tačiau, įvertinant uždavinio specifiką, yra sukurti žymiai efektyvesni šio uždavinio sprendimo metodai.

Tinklai su keliais šaltiniais ir nuotakiais. Maksimalaus srauto uždavinys bendru atveju gali turėti ne vieną, o kelis šaltinius. Tas pat pasakytina ir apie nuotakius (2 pav., a) :



2 pav. Tinklas su keliais šaltiniais ir nuotakiais

Šis iš pirmo žvilgsnio komplikuotas uždavinys iš tiesų nėra sunkesnis už uždavinį su vienu šaltiniu ir vienu nuotakiu.

Maksimalaus srauto uždavinys su keliais šaltiniais (tarkime, jų yra *m*) suvedamas į uždavinį su vienu šaltiniu pridedant vadinamą **superšaltinį *s*** ir lankus (s, s_i), kiekvienam $i = 1, 2, \ldots, m$. Lankams (s, s_i) suteikiamas pralaidumas c (s, s_i) = ∞. Analogiškai elgiamasi ir su nuotakiais (2 pav, b).

Akivaizdu, kad srautas 2 pav., (a) pavaizduotu tinklu atitiks srautą 2 pav. (b) pavaizduotu tinklu ir atvirkščiai.

1.2. Srauto algoritmų apžvalga

Maksimalaus srauto uždavinys bei dualus jam minimalaus pjūvio uždavinys intensyviai nagrinėjami jau daugiau kaip tris dešimtmečius.

Danzigo [2] simplekso metodas transportavimo uždaviniui, paskelbtas 1951 metais, sprendė maksimalaus srauto problemą kaip atskirą atvejį. Neužilgo Fordas ir Falkersonas [3] maksimalaus srauto

uždaviniui spręsti pasiūlė didinančių grandinių (angl. augmenting path) metodą. Dinicas [4], [5] bei Edmondsas ir Karpas [6] įrodė, kad šio metodo sudėtingumas yra polinominis.

Efektyvesni algoritmai maksimaliam srautui rasti yra pagrįsti blokuojančio srauto (angl. blocking flow) ir priešsraučio koregavimo (angl. push-relabel) metodais. Pirmąjį blokuojančio srauto metodą pasiūlė Dinicas [4]. Karzanovas [7] buvo pirmasis, kuris blokuojančių srautų radimą suformulavo kaip atskirą uždavinį ir jo sprendimui pasiūlė naudoti priešsraučius. Priešsraučio koregavimo metodą, įeinantį į Goldbergo algoritmą [8], galutinai išvystė Goldbergas ir Tarjanas [9].

Čia pateikiama maksimalaus srauto algoritmų vystymosi istorija (n reiškia tinklo viršūnių skaičių, m – tinklo lankų skaičių, U – didžiausias lanko pralaidumas) [10] :

metai	autorius (iai)	sudėtingumas
1951	Dantzig	$O (n^2 m U)$
1956	Ford & Fulkerson	$O (n m U)$
1970	Dinitz, Edmonds & Karp	$O (n m^2)$
1970	Dinitz	$O (n^2 m)$
1972	Edmonds & Karp, Dinitz	$O (m^2 \log U)$
1973	Dinicas, Gabow	$O (n m \log U)$
1974	Karzanov	$O (n^3)$
1977	Čerkasskij	$O (n^2 m^{1/2})$
1978	Malxotr ir kiti	$O (n^3)$
1980	Galil & Naamad	$O (n m \log^2 n)$
1983	Sleator & Tarjan	$O (n m \log n)$
1986	Goldberg & Tarjan	$O (n m \log(n^2/m))$
1987	Ahuja & Orlin	$O (n m + n^2 \log U)$
1987	Ahuja ir kiti	$O (n m \log (n \sqrt{\log U} / m))$
1989	Cheriyan & Hagerup	$E (n m + n^2 \log^2 n)$
1990	Cheriyan ir kiti	$O (n^3 / \log n)$
1990	Alon	$O (n m + n^{8/3} \log n)$
1992	King ir kiti	$O (n m + n^{2+e})$
1993	Phillips & Westbrook	$O (n m (\log_{m/n} n + \log^{2+e} n))$
1994	King ir kiti	$O (n m \log_{m/(n \log n)} n)$
1997	Goldberg & Rao	$O (\min (n^{2/3}, m^{1/2}) m \log (n^2 / m) \log U)$

Išsamesnė maksimalaus srauto algoritmų apžvalga pateikiama [10], [11].

1.3. Fordo – Falkersono metodas

Derėtų skirti Fordo – Falkersono metodą nuo algoritmo. Metodas yra vienas, o jį realizuojantys algoritmai gali būti įvairūs. Esminės šio metodo sąvokos yra trys: liekamieji tinklai, papildomi keliai ir pjūviai.

Liekamieji tinklai. Tegul turime tinklą ir srautą jame. Neformaliai tariant, liekamasis tinklas yra sudarytas iš tų lankų, kuriais galima padidinti srautą. Formalus apibrėžimas toks: tegul $G = (V,E)$ – tinklas, turintis pradžios viršūnę s ir pabaigos viršūnę t. Tegul f – srautas tame tinkle. Kiekvienai viršūnių porai u ir v apibrėžkime liekamąjį pralaidumą iš u į v (rus. остаточная пропускная способность, angl. residual capacity of (u,v)) taip:

$$c_f(u,v) = c(u,v) - f(u,v).$$

Jis nusako, kiek dar srauto galima nukreipti iš u į v. Pavyzdžiui, jei $c(u,v) = 16$, o $f(u,v) = 11$, tai lanku (u,v) dar galima papildomai perduoti $c_f(u,v)=5$ vienetus srauto. Liekamasis pralaidumas $c_f(u,v)$ gali būti didesnis už $c(u,v)$, jei duotu momentu $f(u,v)$ yra neigiamas. Pavyzdžiui, jei $c(u,v) = 16$, o $f(u,v) = -4$, tai $c_f(u,v) = 20$. Tada galima srautą padidinti 4, pašalinus priešinį srautą, ir dar nukreipus 16 vienetų, neviršijant lanko (u,v) pralaidumo.

Tinklas $G_f = (v,E_f)$, kur

$$E_f = \{ (u,v) \in V \times V : c_f(u,v) > 0 \},$$

vadinamas tinklo G liekamuoju tinklu (rus. остаточная сеть, angl. residual network).

Didinančiosios grandinės. Tegul f – srautas tinkle $G = (V,E)$. Didinančiaja grandine (rus. дополняющий путь, angl. augmenting path) vadinamas kelias iš pradinės viršūnės s į paskutinę viršūnę t liekamuoju tinklu G_f. Iš liekamojo tinklo apibrėžimo išplaukia, kad visais papildomo kelio lankais (u,v) galima perduoti kažkiek srauto, neviršijant nei vieno lanko pralaidumo. Didžiausias srautas, kurį galima perduoti didinančiąja grandine p, vadinamas liekamuoju kelio p pralaidumu (angl. residual capacity of p):

$$c_f(p) = \min \{ c_f(u,v) : (u,v) \in p \}.$$

Pjūviai tinkluose. Tinklo $G = (V,E)$ pjūviu (cut) vadinamas toks aibės V padalinimas į dvi dalis S ir $T = V \setminus S$, kad $s \in S$, o $t \in T$. Pjūvio (S,T) pralaidumu (angl. capacity of the cut) vadinama pjūvį kertančių lankų pralaidumų suma $c(S,T)$. Srautui f apibrėžiamas srauto dydis per pjūvį (S,T) (angl. net flow across (S,T)) – tai suma $f(S,T)$ pagal pjūvį kertančius lankus. Minimaliu pjūviu (angl. minimal cut) vadinamas pjūvis su mažiausiu pralaidumu.

Maksimalaus srauto uždavinys Fordo – Falkersono metodu [3] realizuojamas pažingsniui. Pradžioje laikoma, kad srautas lygus nuliui. Kiekvienu sekančiu žingsniu srautas didinamas. Tai atliekama randant didinančiąją grandinę, kuria galima praleisti dar truputį srauto, ir ją panaudojant. Šios iteracijos atliekamos tol, kol įmanoma surasti nors vieną didinančiąją grandinę. Metodą galima užrašyti taip:

1 laikyti, kad srautas $f = 0$

2 **while** egzistuoja papildomas kelias p

3 **do** papildyti *f* keliu *p*

4 **return** *f*

Bendroji Fordo – Falkersono algoritmo schema. Naudojantis Fordo – Falkersono metodu, kiekviename žingsnyje laisvai pasirenkama didinančioji grandinė *p* ir srautas *f* didinamas dydžiu $c_f(p)$. Toliau pateikiamas algoritmas einamosioms srauto reikšmėms saugoti naudoja masyvą *f[u,v]*. Laikoma, kad funkcija *c(u,v)* paskaičiuojama per laiką O(1).

1 **for** kiekvienam lankui $(u,v) \in E[G]$

2 **do** $f[u,v] \leftarrow 0$

3 $f[v,u] \leftarrow 0$

4 **while** likusiame tinkle G_f egzistuoja kelias *p* iš *s* į *t*

5 **do** $c_f(p) \leftarrow \min \{ c_f(u,v) : (u,v) \in p \}$

6 **for** kiekvienam kelio *p* lankui (u,v)

7 **do** $f[u,v] \leftarrow f[u,v] + c_f(p)$

8 $f[v,u] \leftarrow -f[u,v]$

1-3 eilutėse nustatomos pradinės srauto reikšmės, ciklas 4-8 eilutėse randa didinančiąją grandinę *p* grafe G_f ir padidina srautą *f*. Jei didinančiųjų grandinių nėra, reiškia rastas srautas yra maksimalus.

Fordo – Falkersono algoritmo analizė. Fordo – Falkersono procedūros darbo trukmė priklauso nuo to, kaip ieškomas kelias *p* (4-ji eilutė). Teoriškai algoritmas gali iš viso nesustoti, jei srauto prieaugis vyks vis mažėjančiais žingsneliais, taip ir nepasiekiant maksimumo. Vienok jeigu didinančiąją grandinę parinkinėti naudojant paieškos platyn algoritmą [1], tai algoritmas dirbs polinominį laiką.

1.4. Edmondso – Karpo metodas

Jeigu Fordo – Falkersono metodą galima užrašyti taip:

1 laikyti, kad srautas $f = 0$

2 **while** egzistuoja papildomas kelias *p* iš *s* į *t*

3 **do** papildyti *f* keliu *p*

4 **return** *f*

tai Edmondso – Karpo metodas [6] apibūdinamas taip:

1 laikyti, kad srautas $f = 0$
2 **while** egzistuoja papildomas kelias p iš s į t
3 tegul p yra trumpiausias kelias iš s į t
4 **do** papildyti f keliu p
5 **return** f

Taigi, iš esmės tai yra tas pats Fordo – Falkersono metodas, tik vietoj bet kokios didinančiosios grandinės ieškoma trumpiausios didinančiosios grandinės. Tokia didinančioji grandinė gali būti surasta per $O (m)$ laiko. Akivaizdu, kad jos ilgis yra ne didesnis už $n – 1$. Iš to išplaukia, kad ir skirtingų ilgių skaičius yra ne didesnis už $n – 1$. Lieka sunkiausias klausimas: kiek yra tokių fiksuoto ilgio didinančiųjų grandinių? Įrodyta, kad jų skaičius neviršija m. Dėl tos priežasties Edmondso – Karpo algoritmas gali būti įvykdomas per $O (n m^2)$.

1.5. Dinico metodas

Dinico metodas [4] taip pat remiasi trumpiausios didinančiosios grandinės paieška, bet efektyvesniu būdu, nei Edmondso – Karpo metode: kiekvieną tokią grandinę jis randa per $O (n)$ laiko, o viso metodo greitis tada yra $O (n^2 m)$. Tokio efektyvumo Dinico metodas pasiekia, pakartotinai naudodamas paieškos platyn metodą (rus. поиск в ширину, angl. Breadth First Search, BFS) sukauptą informaciją. Ieškoma ne vieno, o keleto trumpiausių kelių. Tuo tikslu, ieškant trumpiausios didinančiosios l ilgio grandinės, konstruojamas pagalbinis, sluoksniuotas, tinklas (angl. layered network), kuriame yra visi trumpiausi keliai iš s į t. Kelio iš s į t paieška sluoksniuotame tinkle užtrunka tik $O (n)$ laiko. Dinico metodas yra efektyvesnis už Edmondso – Karpo metodą, nes sluoksniuoto tinklo nereikia perskaičiuoti kiekvieną kartą, ieškant didinančiosios grandinės. Vietoj to algoritmas atlieka sluoksniuoto tinklo atnaujinimą šalindamas viršūnes ir briaunas (15 – 16 eilutės toliau pateikiamoje metodo schemoje) :

Dinitz (G, c, s, t)
< *G – tinklas,* c – *lankų pralaidumai,* s – *šaltinis,* t – *nuotakis* >

1 < *inicializavimas* >

2 $f \leftarrow f_0$; konstruojamas liktinis tinklas $\boldsymbol{N}_f = (G_f, c_f, s, t)$

3 konstruojamas sluoksniuotas tinklas $\boldsymbol{L}_f = (L_f, c_f, s, t)$

4 **while** L_f yra kelias iš s į t **do**

5 **repeat**

6 < *iteracijos pradžia:* L_f *ilgis yra* $d_f(s, t) = \boldsymbol{l}$ >

7 L_f randamas kelias p iš s į t; tegul jo mažiausias pralaidumas bus $c_f(p)$

8 < *atnaujinami* f *ir* $\boldsymbol{L}_f$ >

9 **for all** briaunai $< u, v > \in p$ **do**

10 $f(u, v) \leftarrow f(u, v) + c_f(p)$; $f(v, u) \leftarrow f(v, u) - c_f(p)$

11 $c_f(u, v) \leftarrow c_f(u, v) - c_f(p)$

12 **if** $c_f(u, v) = 0$ **then**

13 *iš* $\boldsymbol{L}_f$ *šalinama* $< u, v >$

14 < *išvalomas* $\boldsymbol{L}_f$ >

15 iš $\boldsymbol{L}_f$ šalinamos visos iš s nepasiekiamos viršūnės ir briaunos

16 iš $\boldsymbol{L}_f$ šalinamos visos viršūnės ir briaunos, iš kurių nepasiekiama t

17 < *pastoviai:* $\boldsymbol{L}_f$ *yra visuma* l *ilgio kelių iš* s *į* t *grafe* G_f ;
ir nėra G_f *kelių, trumpesnių nei* l >

18 **until** L_f nėra kelio iš s į t

19 < *repeat ciklui pasibaigus* $d_f(s, t) > l$ >

20 konstruojami tinklai $\boldsymbol{N}_f$ ir $\boldsymbol{L}_f = (L_f, c_f, s, t)$

pasibaigus: f *yra maksimalus srautas tinkle* N

Sluoksniuotasis tinklas konstruojamas naudojant modifikuotą paiešką platyn liekamojo tinklo pagrindu, kuriame yra visi lankai iš vieno sluoksnio į kitą (išskyrus tik pirmą lanką). Kai randamas t, sluoksnių kūrimas baigiamas ir sluoksniuotame tinkle iš t inicijuojama atvirkštinė paieška platyn. Atvirkštinės paieškos platyn tikslas yra pašalinti visas viršūnes ir lankus, iš kurių t yra nepasiekiama.

SluoksniuotojoTinkloKonstravimas (N_f)

1 $V_0 \leftarrow \{ s \}; \; i \leftarrow 0$ > *inicializavimas*

2 **while** $(V_i \neq \emptyset)$ *and* $(t \notin V_i)$ **do**

3 $V_{i+1} \leftarrow \emptyset; \; E_{i+1} \leftarrow \emptyset ;$

4 **for all** $u \in V_i$ **do**

5 **if** $(< u, v > \in E_f)$ **and** $(v \notin V_j , \forall \; j \leq i)$ **then**

6 v įtraukti į V_{i+1}, jei jo ten nėra, o $<u,v>$ įtraukti į E_{i+1};

7 $i \leftarrow i + 1$

8 **if** $(V_i = \emptyset)$ **then**

9 **return** $L_f = \emptyset;$ > *N_f nėra kelio iš s į t*

10 $l \leftarrow i;$

11 atliekama atvirkštinė BFS pradedant nuo t, žyminti visas pasiektas viršūnes ir lankus;

12 šalinamos visos nepažymėtos viršūnės ir lankai;

13 $L_f \leftarrow (V_0 \cup \ldots \cup V_l, E_1 \cup \ldots \cup E_{l-1})$;

14 **return** $\boldsymbol{L}_f = (L_f , c_f , s , t);$

pasibaigus: jei tinkle $\boldsymbol{N}_f$ *t yra pasiekiamas iš s, tai* $\boldsymbol{L}_f$ *yra sluoksniuotas tinklo* $\boldsymbol{N}_f$ *tinklas.*

Dinico metode figūruojanti sąvoka **blokuojantis srautas** apibrėžiamas taip: f_b yra blokuojantis srautas, jei kiekvienam keliui $p = (s, \ldots, t)$ egzistuoja $e \in p : f_b (e) = cap (e)$.

1.6. Goldbergo metodas

Goldbergas 1985 metais [8] pasiūlė naują priešsraučio algoritmą (pirmąjį algoritmą, besiremiantį ne didinančiosiomis grandinėmis, o priešsraučiu, t.y. lankams priskirtų srautų visuma, sudaryta prisilaikant lankų pralaidumo, bet leidžiant kai kurioms viršūnėms priimti daugiau srauto, negu išleisti – t.y. turėti teigiamą ekscesą , 1974 metais pasiūlė Karzanovas [7]), bet sluoksniuotą tinklą jis pakeitė viršūnių

žymėjimu atstumo žymėmis. Žymės apibrėžiamos taip: $d(s) = n$, $d(t) = 0$ ir $d(v) \le d(w) + 1$ kiekvienam liekamajam lankui (v, w). Jei $d(v) < n$, tada žymė reiškia trumpiausią atstumą iš v į t liekamajame tinkle. Jei $d(v) \ge n$, tada $d(v) - n$ yra trumpiausias atstumas tinkle G_f nuo v iki šaltinio.

Algoritmas prasideda visų lankų prisotinimu. Kiekviena viršūnė, srauto dydis į kurią yra teigiamas, yra vadinama aktyvia viršūne; taigi visos šaltiniui gretimos viršūnės iš pradžių tampa aktyviomis. Algoritmas aktyvioms viršūnėms taiko dvi operacijas: *postūmis* (angl. *push*) ir *pernumeravimas* (angl. *relabel*). *Postūmis* nukreipia srautą iš viršūnės kaimyninėms viršūnėms, o *pernumeravimas* atnaujina žymės reikšmę.

```
1     procedure push-relabel ();
2       prisotinti visus iš šaltinio einančius lankus
3       d ( s ) ← n
4       d ( i ) ← 0  ∀ i∈ V \ { s }
5       while yra aktyvi viršūnė i :
6         if yra toks lankas ( i, j ), kad r ( i, j ) > 0 ir d ( i ) = d ( j ) + 1
7           < atlikti operaciją push >
8           δ ← min ( e ( i ),  r ( i, j ) )
9           f ( i, j ) ← f ( i, j ) + δ
10          e ( i ) ←  e ( i ) - δ
11          e ( j ) ←  e ( j ) + δ
12        else :
13          < atlikti operaciją relabel >
14          d ( i ) ← min { d ( j ) + 1 :   r ( i, j ) > 0 })
```

Šis šablonas tarnauja pagrindu šeimai panašių algoritmų, žinomų eile pavadinimų: *push-relabel*, *preflow-push*, *Goldbergo*, *Golgbergo - Tarjano*. Bendrasis algoritmas nenusako jokios *push* ir *relabel* operacijų tvarkos ir yra įvykdomas per $O(n^2 m)$ laiko.

1986 metais Goldbergas ir Tarjanas [9] pasiūlė kelis specialius variantus:

Bendrasis maksimalaus srauto algoritmas. Įrodyta, kad jis įvykdomas per $O(n^2 m)$ operacijų:

```
1       Procedure  Max-Flow ( V, E, s, t, c );
        <  V – viršūnės,  E – lankai,  s – šaltinis,  t – nuotakis,  c – pralaidumai  >
2          < inicializavimas >
3             < inicializuojamas priešsrautis >
4             ∀ ( v, w ) ∈ ( V – {s} ) x ( V – {s} ) do begin
5                f ( v, w ) ← 0;  f ( w, v ) ← 0;
6             end;
7             ∀  v ∈  do begin
8                f ( s, v ) ←  c ( s, v );
9                f ( v, s ) ←  - c ( s, v );
10            end;
11         < inicializuojamos žymės ir ekscesai >
12            d ( s ) ← n;
13            ∀  v ∈  V – {s}  do begin
14               d ( v ) ← 0;
15               e ( v ) ← f ( s, v );
16            end;
17         < ciklas >
18            while ∃ pritaikoma bazinė operacija do
19               parinkti ir įvykdyti bazinę operaciją;
20            return ( f );
21      end.
```

Bazinėmis vadinamos toliau pateikiamos *push* ir *relabel* operacijos:

Push (v, w).

Taikymas: *v* yra aktyvi, $r_f (v, w) > 0$ **and** $d (v) = d (w) + 1$.

Veikimas: nukreipti $\delta = \min (e (v), r_f (v, w))$ *kiekį srauto iš* v *į* w*:*

$f (v, w) \leftarrow f (v, w) + \delta$; $f (w, v) \leftarrow f (w, v) - \delta$;

$e (v) \leftarrow e (v) - \delta$; $e (w) \leftarrow e (w) + \delta$.

Relabel (v).

Taikymas: v yra aktyvi **and** $\forall\, w \in V,\ r_f(v, w) > 0 \Rightarrow d(v) \le d(w)$.

Veikimas: $d(v) \leftarrow \min \{ d(w) + 1 \mid (v, w) \in E_f \}$.

(jei minimumo tenka ieškoti tuščioje aibėje, $d(v) \leftarrow \infty$).

FIFO eilė. Šis metodas įvykdomas per $O(n^3)$ laiko. Čia naudojamos tokios bazinės operacijos: *Push / Relabel (v)* bei *Discharge.*

Dinaminių medžių panaudojimas. Kad pagreitinti neprisotinančio srauto formavimą, pasinaudota Sleator'o ir Tarjano dinaminio medžio duomenų struktūra [12]. Naudojamos operacijos *šaknies-radimas* (angl. *find-root(v)*), *dydžio-radimas* (angl. *find-size(v)*), *reikšmės-radimas* (angl. *find-value(v)*), *mažiausios-radimas* (angl. *find-min(v)*), *reikšmės-keitimas* (angl. *change-value(v,x)*), *susiejimas* (angl. *link(v,w)*), *nukirtimas* (angl. *cut(v)*), *nusiintimas* (angl. *send(v)*) bei *medžio-postūmis* (angl. *tree-push (v)*) ir *medžio-pernumeravimas* (angl. *tree-relabel(v)*). Dinaminio medžio metodas trunka $O(n\, m \log (n^2 / m))$ laiko.

2. PROGRAMINĖ ĮRANGA SRAUTO ALGORITMŲ ANALIZEI

2.1. Reikalavimai projektuojamai sistemai

Programinę įrangą numatoma naudoti Sisteminės analizės katedros veikloje eksperimentiškai tikrinant pasirinktų algoritmų veikimą bei naudoti studentų mokymo procese.

Reikalavimai programinei įrangai aptarti su Kauno technologijos universiteto Sisteminės analizės katedros docentu dr. Kostu Pluku.

Bendras veiklos tikslas yra sukurta ir suderinta programinė įranga, kurios pagalba bus galima praktikoje patikrinti teoriškai įrodytas maksimalaus srauto orientuotame grafe algoritmų savybes. Šiuo metu nėra žinoma tokios programinės įrangos analogų.

Kuriama programinė įranga neturi komercinės paskirties. Ji kuriama išimtinai akademiniams, mokymo tikslams.

2.1.1. Programinės įrangos paskirtis

Programinės įrangos funkcija – pasirinktam tinklui realizuojant Fordo-Falkersono, Edmondso-Karpo, Dinico ir Goldbergo maksimalaus srauto orientuotame grafe radimo algoritmus:

- suskaičiuoti kiekvieno šių algoritmų veikimo trukmę. Gautus rezultatus reikia išsaugoti, kad laikui bėgant susikauptų pakankamas eksperimentinių rezultatų skaičius statistinei analizei;
- vizualizuoti srauto paieškos procesą. Tinklą nupiešti kompiuterio ekrane ir sudaryti galimybę vartotojui stebėti, kaip vyksta srauto paieškos procesas.

2.1.2. Reikalavimai funkcionavimui

Programinė įranga privalo tiksliai realizuoti pasirinktus algoritmus. Jie turi veikti tiksliai taip, kaip yra paskelbta literatūroje.

Lygiagrečius veiksmų organizuoti nėra būtina. Skaičiavimų laikas nėra kritinis. Jį netiesiogiai nusako pats programos vartotojas užduodamas tinklo viršūnių skaičių – kuo daugiau viršūnių ir lankų turi tinklas, tuo ilgesnis bus skaičiavimų laikas.

Programa skirta naudoti Microsoft Windows operacinės sistemos terpėje.

2.1.3. Reikalavimai vartotojo sąsajai

Vartotojo sąsaja – grafinė. Ji turi būti suprojektuota taip, kad – turint galvoje nedidelį įvedamų duomenų bei gaunamų rezultatų kiekį bei trivialumą – būtų suprantama be jokių papildomų aprašymų.

Sąsaja turėtų būti nekintama. Joje nėra reikalo naudoti elementus, kurie kaip nors kistų priklausomai nuo skaičiavimų eigos.

Reikia išvengti sąsajos elementų, kuriuos vartotojui reikėtų kaip nors derinti.

2.1.4. Eksploatavimo aplinka

Programinė įranga kuriama ir eksploatuojama Microsoft Windows aplinkoje. Specialių aparatūrinių resursų šiai programinei įrangai realizuoti nereikia. Projektavimo technologijai apribojimai netaikomi.

Programinė įranga naudojasi bendrais kompiuterio bei vietinio tinklo, kuriuose eksploatuojama, resursais. Duomenų bazių valdymo sistema naudotis nenumatoma. Įvykus avariniam programos sustojimui, neturi būti pažeisti per ankstesnius skaičiavimus sukaupti duomenys.

Siekiant kuo labiau supaprastinti šios programinės įrangos naudojimą skirtingose darbo vietose, reikia pasiekti , kad programinė įrangos nereikėtų instaliuoti į kompiuterį, kuriame ji bus eksploatuojama. Tam, kad perkelti ją į kitą kompiuterį, turi užtekti tik pernešti vykdomąjį modulį ir failą, kuriame kaupiami skaičiavimų rezultatai.

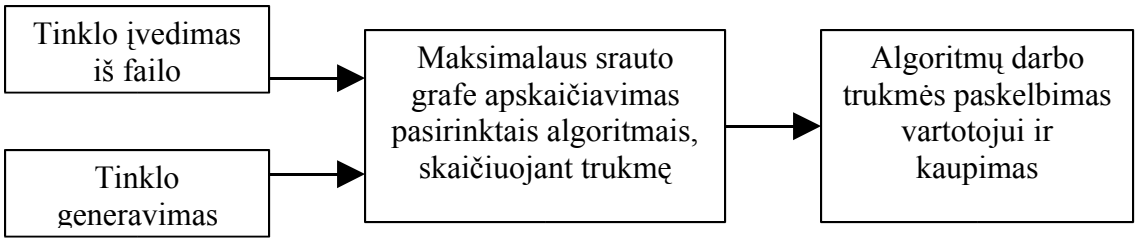
2.1.5. Duomenų srautai

Pradiniai duomenys programinei įrangai nuskaitomi iš failų. Kaip papildomas duomenų šaltinis yra pageidautinas tinklo generatorius, galintis automatiškai sugeneruoti atsitiktinį tinklą.

Programos darbo rezultatas:

- laiko intervalas, kuris reikalingas rasti maksimalų srautą kiekvienu iš pasirinktų metodų. Šie rezultatai turi būti kaupiami, kad vėliau būtų galima atlikti jų statistinę analizę;
- kiekvienu metodu paskaičiuoto srauto aprašas, skirtas srauto paieškos algoritmo vizualizavimui.

Programinės įrangos duomenų transformavimo į rezultatus schema:



2.2. Duomenų struktūra

2.2.1. Duomenų failo struktūra

Duomenų faile yra visa informacija apie tinklą, kuriame maksimalų srautą reikia rasti. Failas susideda iš ASCII simbolių. Eilutės atskiriamos end-of-line simboliu. Reikšmės eilutėse yra skiriamos bent vienu tarpu. Kiekvienos eilutės pirmasis simbolis nusako eilutėje esančių duomenų pobūdį. Eilutės duomenų faile gali būti tokios:

Komentarai. Komentarų eilutės yra skirtos vartotojui. Jos prasideda mažąja raide **c**. Komentarų eilučių skaičius neribojamos ir jos gali būti bet kurioje failo vietoje. Pirmas penkias komentarų eilutes programa traktuoja taip:

Pirmoji – prasideda skaičiumi ir reiškia tinklo tipą. Jei tas skaičius yra 1 .. 10, tai laikoma, kad tinklo tipas atitinka tinklo generatoriuje naudojamą tinklo tipą ir ši informacija naudojama braižant tinklą. Jei minėtas skaičius yra kitoks, jis braižant tinklą nenaudojamas.

Kitos keturios komentarų eilutės nuskaitomos. Jei tinklo tipas yra 1 .. 10, tai jos prasideda skaičiais ir naudojamos tinklo braižymui bei išvedamos į pagrindinio modulio langą informacijai. Jei tipas kitoks, tai tos keturios eilutės gali prasidėti nebūtinai skaičiumi ir yra tik išvedamos į pagrindinio modulio langą.

Likusios komentarų eilutės ignoruojamos.

Uždavinys. Viename duomenų faile gali būti užduotas tik vienas uždavinys. Uždavinio eilutės formatas:

p uždavinys viršūnių_skaičius lankų_skaičius

Kad tai uždavinio eilutė, pasako mažoji **p** raidė pirmoje pozicijoje. “Uždavinys” mūsų atveju yra trijų simbolių ilgio eilutė “max”. “Viršūnių_skaičius” ir “lankų_skaičius” nurodo, kiek viršūnių ir kiek lankų turi tinklas. Jei “lankų_skaičius” nesutampa su lankus nusakančių eilučių skaičiumi (žr. toliau *Lankai*), tai reikšmė “lankų_skaičius” ignoruojama.

Viršūnė. Maksimalaus srauto uždaviniui viršūnės aprašas reikalingas tik tam, kad nurodyti, kuri viršūnė yra srauto šaltinis, o kuri – nuotakis. Viršūnės eilutės formatas:

n viršūnės_numeris žymė

Skiriamasis eilutės požymis – mažoji **n** pirmoje pozicijoje. Viršūnės numeruojamos nuo 1. “Žymė” yra simbolis **s** - šaltiniui, **t** – nuotakiui. Jei duomenų faile nėra nurodytas šaltinis, tai šaltinio laikoma 1-ji viršūnė. Jei nenurodytas nuotakis, nuotakiu laikom viršūnė su didžiausiu numeriu.

Viršūnės koordinatės. Šios eilutės naudojamos, kai vartotojas pats nori nusakyti tinklo viršūnių koordinates. Jei viršūnės koordinatės apibrėžiamos nors vienai tinklo viršūnei, tai jos turi būti apibrėžiamos visoms viršūnėms. Priešingu atveju koordinatės ignoruojamos. Viršūnės koordinačių eilutės formatas:

k viršūnės_numeris x y

Skiriamasis eilutės požymis – mažoji **k** pirmoje pozicijoje. **x** ir **y** reikšmės – natūralieji skaičiai.

Lankas. Šiomis eilutėmis nusakomas kiekvienas tinklo lankas. Eilutės formatas:

e u v c

Skiriamasis eilutės požymis – mažoji **e** pirmoje pozicijoje. Po jos eina lanko pradžios viršūnės numeris **u**, lanko pabaigos viršūnės numeris **v** ir lanko pralaidumas **c**.

Jei eilutė prasideda kitokiu nei paminėta simboliu, ji ignoruojama.

Šios duomenų failo struktūros pagrindu yra paimta DIMACS centro [13] duomenų failo struktūra, pateikiama adresu [14].

Duomenų failo pavyzdys pateiktas 1 priede.

2.2.2. Vidinė duomenų struktūra

Programos viduje tinklas nusakomas lankų matrica. Lankų matricos kaip tinklo saugojimo kompiuterio atmintyje būdo įvertinimas pateiktas [15] knygos 61 puslapyje. Lankų matrica realizuota trimis vienmačiais masyvais: pirmasis masyvas saugo lankų pradžios viršūnių numerius, antrasis – lankų pabaigos viršūnių numerius, trečiasis – lankų pralaidumus.

2.2.3. Rezultatų failo struktūra

Rezultatų failui pasirinkta kableliais atskirtų reikšmių (comma separated values, CSV) failo struktūra. CSV formatas yra bene labiausiai pripažintas formatas, naudojamas perduoti duomenis tarp programų. Beveik visos programos, dirbančios su duomenimis, gali eksportuoti duomenis CSV formatu, ir beveik visos jos gali taip duomenis priimti.

CSV failo struktūros aprašymą galima rasti [16].

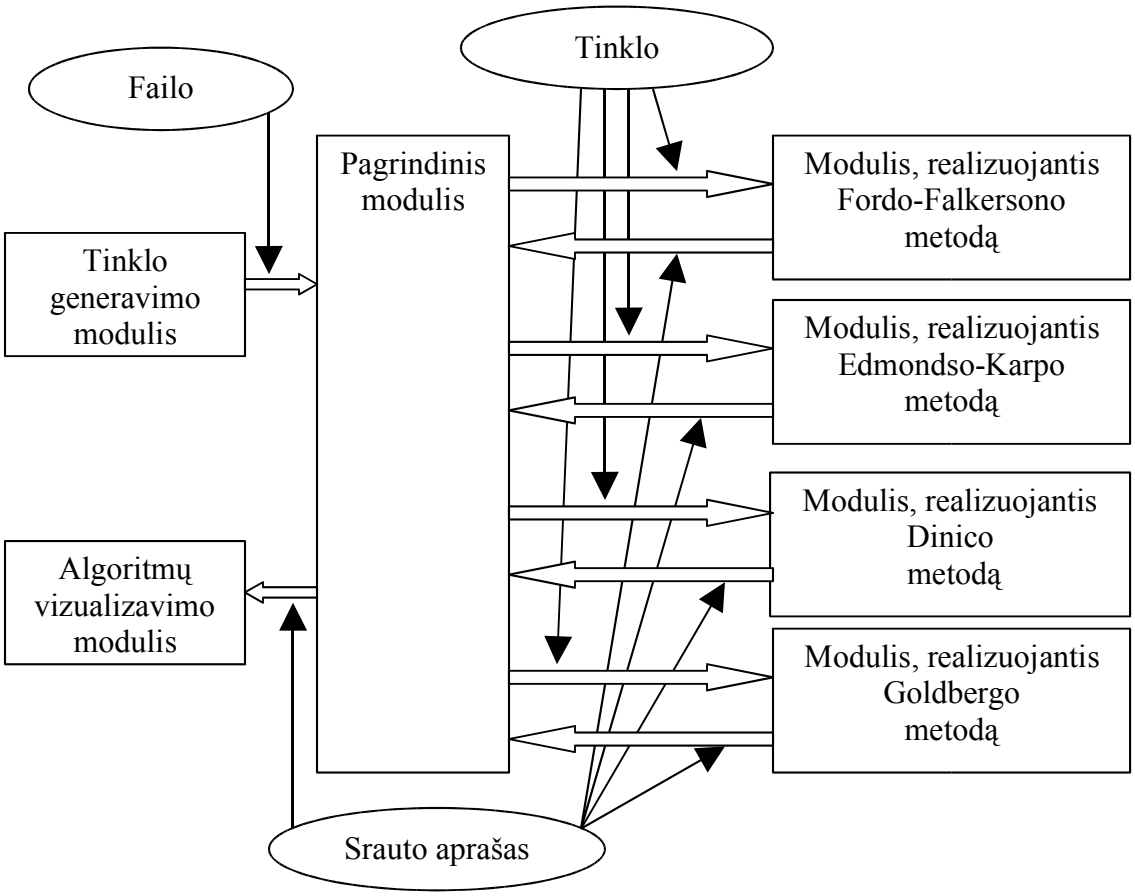
Rezultatų failo laukai:

- taktinis procesoriaus dažnis megahercais. Jis gali būti reikalingas, kai rezultatų failas kaupiamas, skaičiuojant srautą skirtingo pajėgumo kompiuteriais;
- tinklo tipas. Pildomas tik tinklams, kuriuos sugeneravo vidinis programos atsitiktinio tinklo generatorius;
- viršūnių skaičius. Visų viršūnių, įskaitant ir superšaltinį bei supernuotakį, skaičius;
- lankų skaičius;

- eilučių skaičius. Kvadratinio arba stačiakampio tinklo eilučių skaičius. Be superšaltinio ir supernuotakio;
- stulpelių skaičius. Kvadratinio arba stačiakampio tinklo stulpelių skaičius. Be superšaltinio ir supernuotakio.
- mazgų skaičius. Viršūnių skaičius be superšaltinio ir suprnuotakio.
- didžiausias lankų pralaidumas;
- viršūnių laipsnis;
- atsitiktinių skaičių generatoriaus parametras. Turi prasmę, kai norima dar kartą sugeneruoti tokį patį tinklą;
- Fordo-Falkersono metodo trukmė milisekundėmis;
- Edmondso-Karpo metodo trukmė milisekundėmis;
- Dinico metodo trukmė milisekundėmis;
- Goldbergo metodo trukmė milisekundėmis.

2.3. Projektuojamos sistemos architektūra

Duomenų srautas ir programos veikimo algoritmas apsprendžia jos struktūrą, išskaidymą į daugiau ar mažiau savarankiškus modulius:



Šie moduliai tarp savęs per duomenis yra susiję taip:

- tinklo generavimo modulis teikia duomenis (tinklo aprašą) pagrindiniam moduliui;
- pagrindinis modulis gauna duomenis iš tinklo generavimo modulio (failo vardą) ir algoritmų realizavimo modulių (srauto aprašą); teikia duomenis algoritmų realizavimo moduliams (tinklo aprašą) ir vizualizavimo moduliui (srauto aprašą);
- algoritmų realizavimo moduliai gauna duomenis iš pagrindinio programos modulio (tinklo aprašą), ir grąžina jam rezultatus (srauto aprašą);
- vizualizavimo modulis gauna duomenis (srauto aprašą) iš pagrindinio programos modulio.

2.4. Programinių modulių ar objektų specifikacijos

2.4.1. Pagrindinis programos modulis

Modulio paskirtis – valdyti visus kitus programos modulius: nurodyti duomenų failą, paleisti atsitiktinio tinklo generatorių, nurodyti algoritmų realizavimo modulių darbo režimą ir juos aktyvuoti.

Šis modulis dirba cikliniu režimu – atlikus minėtus veiksmus, galima juos kartoti, vėl pasirenkant duomenų failą ir t.t.

Modulio sąsaja – grafinė. Lange yra dvi sritys.

Viena sritis yra skirta pasirinkti tinklą, kuriame bus ieškoma srauto. Šioje srityje yra laukas, skirtas išvesti duomenų failo vardą ir du mygtukai “Ieškoti” ir “Generuoti”. Mygtuko “Ieškoti” pagalba vartotojas gali pasirinkti iš anksto paruoštą duomenų failą su tinklo aprašu. Pasirinkus duomenų failą, šioje srityje išvedamos faile esančios komentaro eilutės, kad vartotojas gautų informaciją apie pasirinktą tinklą. Mygtukas “Generuoti” skirtas iškviesti programoje esantį atsitiktinio tinklo generatorių.

Kita lango sritis skirta nurodyti, ką programa turi daryti su pasirinktu tinklu. Pažymimi algoritmai, kurie ieškos srauto tinkle. Nurodoma, kokiu režimu turės dirbti algoritmų realizavimo moduliai – trukmės registravimo ar duomenų kaupimo vizualizavimui. Mygtukas “Skaičiuoti” suaktyvina pažymėtų algoritmų realizavimo modulius.

2.4.2. Tinklo generavimo modulis

Modulio paskirtis – sugeneruoti atsitiktinį tinklą, tinkantį maksimalias srauto problemai.

Modulio algoritmas nėra originalus. Ruošiant darbą buvo išbandyta eilė viešai skelbiamų grafų generavimo algoritmų: NETGEN, GOTO (Grid On TOrus), GEN, MPV, MMD, RandomGraph ir kt. Prieita išvados, kad daugiausiai galimybių teikia Vašingtono universiteto Kompiuterių mokslo ir inžinerijos profesoriaus Richardo Andersono [17] vadovaujamų studentų sukurtas kelių tinklo tipų generatorius WASHINGTON.

Jis generuoja tokius tinklus:

- Mesh Graph - verčiamas kaip “stačiakampis tinklas”;

- Random Level Graph - verčiamas kaip “lygių tinklas”;
- Random 2-Level Graph – verčiamas kaip “dviejų lygių tinklas”;
- Matching Graph – verčiamas kaip “dvidalis tinklas”;
- Square Mesh – verčiamas kaip “kvadratinis tinklas”;
- Basic Line – verčiamas kaip “tiesinis tinklas”;
- Exponential Line – verčiamas kaip “rodiklinis tinklas”;
- Double Exponential – verčiamas kaip “dvigubas rodiklinis tinklas”;
- DinicBadCase – verčiamas kaip “neparankus Dinico metodui”;
- GoldBadCase – verčiamas kaip “neparankus Goldbergo metodui.

Tinklo generatorius sukuria DIMACS formato failą. Jo struktūra aprašyta skyrelyje 2.2.1.

Modulio sąsaja – grafinė. Ji skirta nurodyti generuojamo tinklo parametrus.

2.4.3. Moduliai, realizuojantys srauto radimo algoritmus

Modulių paskirtis – realizuoti atitinkamą maksimalaus srauto duotajame tinkle paieškos algoritmą. Algoritmai aprašyti ankstesniuose šio darbo skyriuose. Duomenys gaunami iš pagrindinio programos modulio sveiko tipo masyvų pavidalu. Moduliai dirba dviem režimais:

1. darbo režimas, skirtas paskaičiuoti algoritmo darbo trukmę. Šiame režime nekaupiami duomenys vizualizavimui, kad tai neturėtų įtakos trukmei. Pagrindiniam moduliui grąžinamas sveikas skaičius, reiškiantis algoritmo darbo trukmę milisekundėmis.
2. darbo režimas, skirtas parodyti algoritmo darbo eigą. Šiuo atveju yra kaupiami tarpiniai rezultatai, kurie pabaigoje yra grąžinami pagrindiniam moduliui.

2.4.4. Vizualizavimo modulis

Modulio paskirtis – suteikti vartotojui galimybę stebėti, kaip vyksta maksimalaus srauto paieška kiekvienu metodu.

Modulio sąsaja – grafinė. Modulio langas užpildo visą monitoriaus ekraną. Jame išdėstoma:

- po vieną panelę kiekvienam atvaizduojamam metodui. Panelėje yra užrašas su metodo pavadinimu, užrašas su srauto dydžiu ir sritis, kurioje piešiamas tinklas;
- užrašas, kuriame rodomas algoritmo žingsnio numeris;

- du mygtukai “Pirmyn” ir “Atgal”, kuriais pasirenkamas sekantis arba prieš tai buvęs algoritmo žingsnis ir taip yra valdomas vizualizavimo procesas.

2.5. Testavimas

2.5.1. Modulių testavimas

Programos ir atskirų jos modulių testavimas atliekamas neautomatizuotu būdu. Jį atlieka programos autorius. Moduliai testuojami iš karto tik juos suprogramavus. Programa testuojama iš karto, kai tik atskiri moduliai sujungiami į vieną visumą.

Pagrindinis programos modulis. Testuojant šį modulį tikrinama, kaip modulis reaguoja į vartotojo veiksmus su sąsaja, kaip modulis elgiasi, jei atšaukiamas pradėtas veiksmas. Įsitikinama, ar modulis kitiems moduliams atiduoda būtent tą reikšmę, kurią įvedė vartotojas. Patikrinama, ar modulis sukuria teisingo formato rezultatų failą.

Tinklo generavimo modulis. Patikrinama, ar modulis teisingai sukuria tinklą (t.y. ar tinklas tikrai yra tokio tipo, kurį nurodė vartotojas, ar jis turi tiek viršūnių, kiek pageidavo vartotojas), ar suformuoja teisingo formato failą ir ar suteikė jam tokį vardą, kokį turėjo suteikti.

Moduliai, realizuojantys srauto radimo algoritmus. Įsitikinama, ar šie moduliai:

- teisingai realizuoja atitinkamus maksimalaus kelio paieškos algoritmus;
- teisingai suskaičiuoja algoritmo darbo trukmę;
- grąžina teisingus duomenis algoritmų vizualizavimui.

Algoritmų vizualizavimo modulis. Patikrinama, ar modulis sukuria numatytą grafinę sąsają, ar neiškreipia algoritmų darbo eigos, ar teisingai reaguoja į vartotojo pageidavimus parodyti sekantį ar prieš tai buvusį algoritmų žingsnį, ar visiems algoritmams rodo tą patį žingsnį.

2.5.2. Sąsajos testavimas

Vartotojo sąsają turi trys moduliai: pagrindinis, tinklo generavimo ir algoritmų vizualizavimo. Kiekvieno modulio sąsajoje tikrinama, ar:

- teisingai rodomi ir slepiami reikiami grafiniai elementai;
- fokuso suteikimo grafiniams elementams tvarka patogi vartotojui;
- įvedimo laukai teisingai priima įvedamas reikšmes;
- valdymo mygtukai realizuoja numatytas komandas;
- formos reaguoja į <Esc> klavišą.

2.5.3. Visos programos testavimas

Įsitikinama, ar sujungus programos modulius į vientisą programą, ji veikia teisingai. Sukuriamas galutinis vykdomasis failas. Jis perkeliamas į kitą kompiuterį ir įsitikinama, kad taip realizuota programa veikia – priima duomenis iš vartotojo, atlieka numatytus skaičiavimus, išveda rezultatus. Išbandoma, ar programa teisingai veikia šiais atvejais:

- kai vartotojo kompiuteryje greta jos veikia ir dar kelios programos;
- kai skaičiavimų metu kompiuteris išjungiamas.

3. VARTOTOJO DOKUMENTACIJA

3.1. Sistemos funkcinis aprašymas

Šios programinės įrangos paskirtis – paskaičiuoti maksimalų srautą orientuotame grafe pasirinktinai keliais - Fordo-Falkersono, Edmondso-Karpo, Dinico, Goldbergo – metodais ir suteikti galimybę vartotojui palyginti minėtų metodų greitaeigiškumą ir eigą.

Šią analizę galima atlikti bet kokio dydžio ir konfigūracijos tinklui. Tinklą, aprašytą lankų matrica, galima įvesti iš pasirenkamo failo. Failas gali būti bet kurioje vietoje – tiek kietajame kompiuterio diske, tiek pasiekiamas per tinklą. Duomenų faile gali būti nurodytos ir tinklo viršūnių padėtys. Papildomai numatyta galimybė generuoti dešimt atsitiktinio tinklo tipų pačioje programoje esančiu generatoriumi.

Kiekvienu skaičiavimu gauti rezultatai pateikiami programos lange. Juos peržiūrėjus galima išsaugoti, kad laikui bėgant susikauptų pakankamas eksperimentinių rezultatų skaičius statistinei analizei. Šie rezultatai yra kaupiami faile, esančiame tame pat kataloge, kaip ir vykdomasis programos modulis.

Programoje vartojama grafinė sąsaja. Grafinė sąsaja pasirinkta todėl, kad ji dėl savo vaizdumo ir patogumo vartoti puikiai tinka šiam uždaviniui; atskiru atveju (parodant srauto paieškos eigą) netgi būtina. Sąsaja nėra sudėtinga, joje nėra kokių nors specifinių elementų ir suprantama intuityviai. Meniu nenaudojami. Sąsajoje nėra elementų, kurie kaip nors kistų priklausomai nuo skaičiavimų eigos. Taip pat nėra ir elementų, kuriuos vartotojui reikėtų kaip nors derinti.

Laikoma, kad programos vartotojas moka manipuliuoti pele, gali įvesti skaitinius duomenis klaviatūra. Taip pat laikoma, kad vartotojui suprantamos sąvokos “tinklas”, “viršūnė” ir pan.

3.2. Sistemos vadovas

Suaktyvinus programą vartotojui atsiveria jos langas:

Pirmiausiai reikia nurodyti duomenų failą. Jeigu jis jau yra sukurtas, spaudžiamas mygtukas “Ieškoti”. Atidaromas standartinis Windows operacinės sistemos **Open** dialogas, kuris leidžia pasirinkti duomenų failą :

Duomenų failai, kuriuose yra tinklai, skirti maksimalaus srauto uždaviniui, turi plėtinį “max”, todėl šis dialogas pirmiausiai ir siūlo pasirinkti šio tipo failus. Tačiau esant reikalui ši tipą galima pakeisti į kitus:

Pasirinkus duomenų failą jo vardas ir informacija iš komentarų eilučių įrašomi į panelės laukus:

Jei norima sukurti tinklą vidinio tinklo generatoriaus pagalba, spaudžiamas mygtukas “Generuoti”. Tada aktyvuojamas tinklo generavimo modulis, kuris pateikia savo langą tinklo parametrams nurodyti:

Pirmiausiai pasirenkamas tinklo tipas. Priklausomai nuo tinklo tipo dešinioji lango dalis siūlo nurodyti vis kitokius tinklo parametrus:

Nepriklausomai nuo tinklo tipo visada reikia nurodyti atsitiktinių skaičių generatoriaus parametrą. Jis įvedamas į lauką klaviatūros pagalba. Jeigu šį parametrą vartotojas taip pat pageidauja turėti atsitiktinį, reikia paspausti greta lauko esantį mygtuką “parinkti”. Parametras bus paimtas iš sisteminio laikrodžio:

Norint generuoti atsitiktinį tinklą, spaudžiamas mygtukas “Generuoti”. Programa tikrina nurodytus tinklo parametrus: ar yra nurodyti visi reikalingi parametrai ir ar jų reikšmės priklauso leistinam diapazonui. Jei taip, pereinama prie tinklo generavimo. Jei ne, vartotojas informuojamas tokio tipo pranešimu:

ir turi galimybę pataisyti klaidingą parametrą.

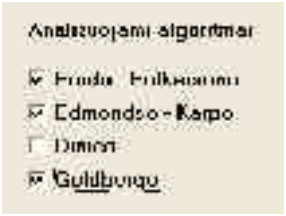
Sugeneruotas tinklas išvedamas į failą, kurio struktūra aptarta 2.2.1. Failas pavadinamas pagal tinklo tipą: jei tipas – Tinklinis grafas, tai failas pavadinamas Tinklinis.max, jei tipas – Dvidalis grafas, tai failas pavadinamas Dvidalis.max ir t.t. Sugeneravus tinklą generatoriaus langas užsidaro ir valdymas grąžinamas pagrindiniam programos moduliui.

Generatoriaus lange paspaudus mygtuką “Atšaukti” arba klavišą <Esc> generatoriaus modulis darbą baigia ir pagrindiniam moduliui nieko negrąžina.

Sugeneruoto failo vardas ir jo apibūdinimas įrašomi į pagrindinio lango panelės laukus.

Parinkus duomenų falą antroje pagrindinio lango panelėje nurodomas analizės tikslas:

ir vartotoją dominantys algoritmai :

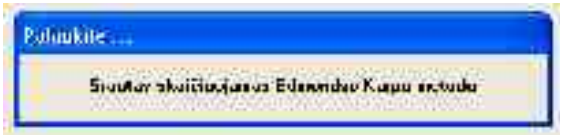


Patarimas vartotojui. Nerekomenduojama rinktis analizės tikslo “parodyti eigą”, jei tinklo viršūnių skaičius didesnis nei 20 – 30. Tada tinklą vaizduojantis piešinys vizualizavimo modulio panelėse bus perdaug tankus ir nevaizdus.

Nurodęs analizės tikslą ir algoritmus, vartotojas mygtuko “Skaičiuoti” paspaudimu gali vykdyti skaičiavimus. Šis procesas susideda iš tokių fazių:

- iš duomenų failo nuskaitomas tinklas;

- jei duomenų faile nebuvo viršūnių koordinačių visai arba jos buvo ne visos, programa pati parenka viršūnėms koordinates;
- paeiliui realizuoja visus pažymėtus algoritmus. Kol vyksta skaičiavimai, vartotojui rodomas toks langas:

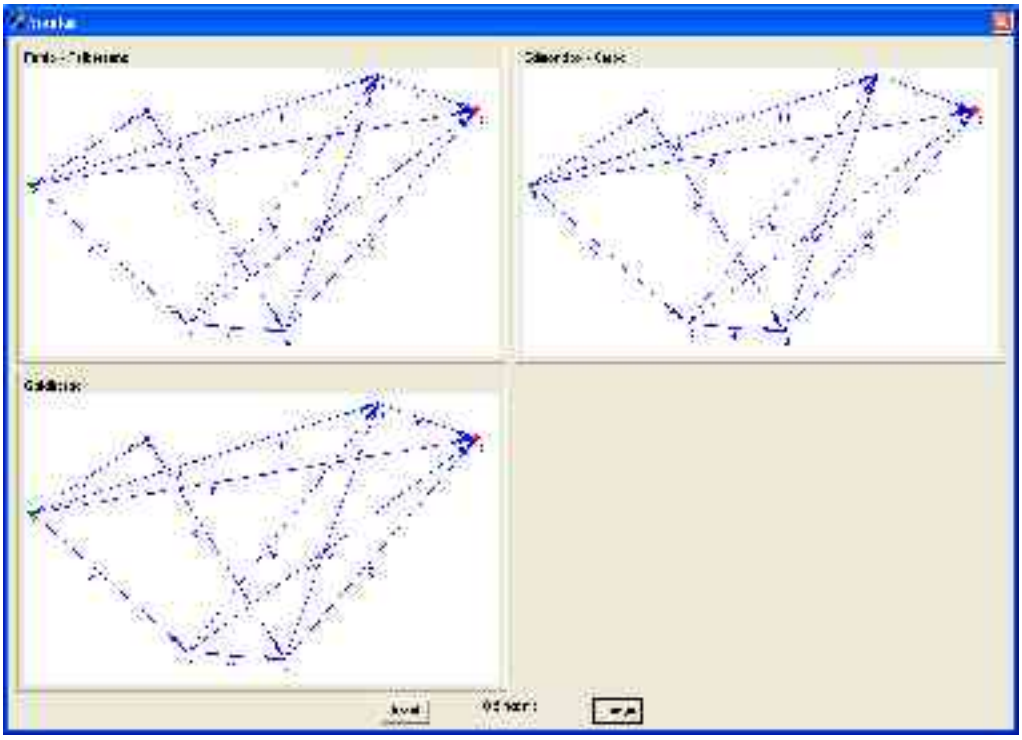


Toliau veiksmas vyksta dvejopai:

1. **Jei vartotojas pageidavo skaičiuoti trukmę**, tai trukmė įrašoma į atitinkamus panelės laukus:

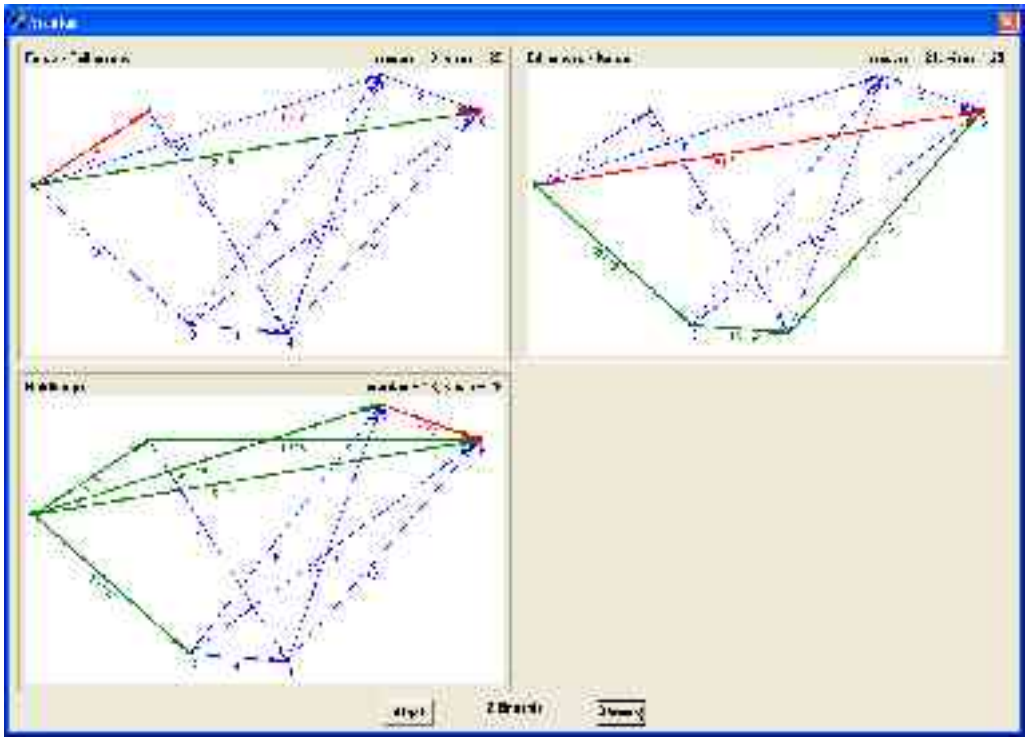
ir vartotojui leidžiama tuos laikus prisiminti. Tai atliekama spaudžiant mygtuką “Prisiminti”. Programa algoritmų darbo trukmę įrašo į 2.2.3 skyrelyje aptartą rezultatų failą;

2. **Jei vartotojas pageidavo matyti algoritmų eigą**, baigus skaičiavimus atsiveria algoritmų vizualizavimo langas:



Lange rodoma tiek panelių, kiek algoritmų pagrindiniame programos lange pasirinko vartotojas.

Kiekvienoje panelėje nupieštas tinklas. Viršūnės sunumeruotos. Šaltinis pažymėtas žaliai. Nuotakis – raudonai. Prie lankų surašyti jų pralaidumai. Tarp mygtukų “Pirmyn” ir “Atgal” parašyta, kuris algoritmo žingsnis atvaizduojamas. 0 žingsnyje paprasčiausiai matomas pradinis tinklas. Perėję į **i**-jį žingsnį ekrane matome tokį vaizdą:



Žalia spalva nupieštas iki **i**-jo žingsnio tinklu praleistas srautas. Raudona spalva nupieštas **i**-jame žingsnyje praleidžiamas srautas.

Vartotojas iš **i**-jo žingsnio gali pereiti į **i-1**-jį arba **i+1**-jį. Taip pat bet kada gali baigti peržiūros procesą, spausdamas ir taip uždarydamas šį algoritmų vizualizavimo langą.

Baigus peržiūrą arba įrašius skaičiavimų trukmę į rezultatų failą programa pasiruošusi priimti iš vartotojo naujus nurodymus ir kartoti aprašytą eigą.

3.3. Sistemos instaliavimo dokumentas

Šis programinis produktas yra nekomercinės paskirties. Jis skirtas naudoti mokymo ir mokymosi procese, todėl ir jis pats, ir jo darbo rezultatai gali būti naudojami be jokių apribojimų. Programinės įrangos į kompiuterį instaliuoti nereikia - užtenka tik į pasirinktą kietojo disko katalogą įrašyti vykdomąjį modulį SRAUTAI.EXE. Jei jau yra sukurtas skaičiavimo rezultatų failas SRAUTAI.CSV ir ateityje planuojama būsimus rezultatus kaupti tame pačiame faile, jį taip pat reikia įrašyti šalia SRAUTAI.EXE. Katalogo vieta diske yra nesvarbi.

Programinės įrangos eksploatavimui reikalingas personalinis kompiuteris, tenkinantis tokius reikalavimus:

- 500 MHz spartos procesorius,
- 64 MB operatyviosios atminties,
- 0,5 MB laisvos vietos kietajame diske vykdomajam moduliui plius atitinkami vietos rezultatų failui,
- Microsoft Windows operacinė sistema.

Programos adaptavimas. Adaptuojant programą dirbti konkrečiame kompiuteryje, reikia vartotojui suteikti operacinės sistemos administratoriaus teises. To nepadarius, programa dėl ribotų vartotojo teisių negali iš sisteminio registro nuskaityti procesoriaus taktinio dažnio ir nedirba.

4. STATISTINIO TYRIMO REZULTATAI

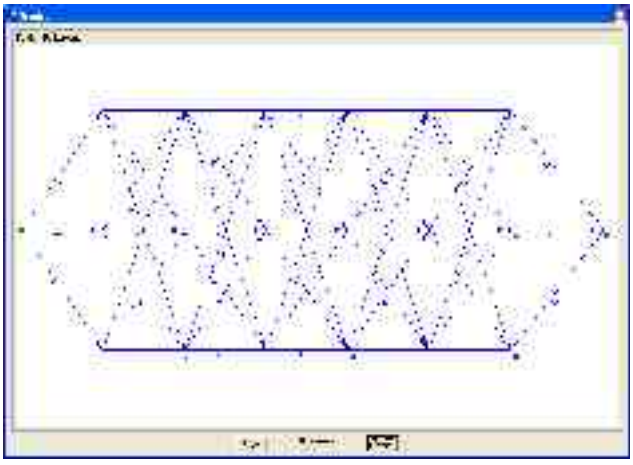
Sukurtos programinės įrangos pagalba atlikta eilė skaičiavimų Visi keturi maksimalaus srauto radimo algoritmai – Fordo-Falkersono, Edmondso-Karpo, Dinico ir Goldbergo – buvo bandomi įvairiais tinklais. Buvo pasinaudota vidinio programos tinklo generatoriaus teikiamomis galimybėmis sugeneruoti įvairių tipų, įvairių dydžių ir įvairių tankių tinklus.

Iš visų įmanomų tam tikram tinklo tipo ir viršūnių skaičiaus variantų buvo pasirenkamas sudėtingesnis, reikalaujantis daugiau skaičiavimų.

Dėl geresnio vaizdumo algoritmų darbo trukmės skalė grafikuose pasirinkta logaritminė.

Nustatant teorinį algoritmų sudėtingumą atsižvelgta į procesoriaus taktinį dažnį.

Eksperimentams stačiakampis, lygių, tiesinis ir rodiklinis tinklai buvo generuojami tokie, kad jų stulpelių skaičius būtų dvigubai didesnis už eilučių skaičių. Pavyzdžiui, šis tinklas (paveikslėlis paimtas iš programos):

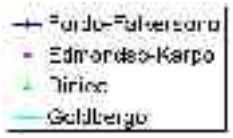


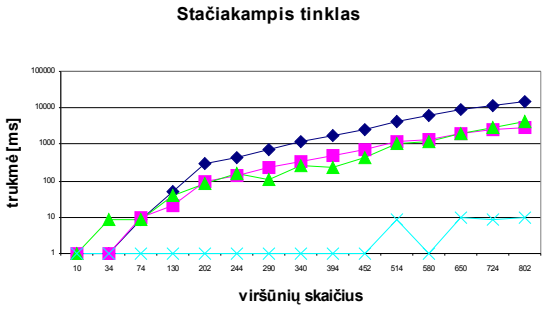
yra stačiakampis; jo eilučių skaičius yra 3 (pirmajai eilutei priklauso 2, 5, 8, 11, 14, 17 viršūnės, antrajai – 3, 6, 9, 12, 15, 18 viršūnės, trečiajai – 4, 7, 10, 13, 16, 19 viršūnės), stulpelių skaičius – 6 (pirmajam stulpeliui priklauso 2, 3, 4 viršūnės, antrajam – 5, 6, 7 viršūnės, trečiajam – 8, 9, 10 viršūnės ir t.t.). Tinklas turo superšaltinį – 1 viršūnę ir supernuotakį – 20 viršūnę).

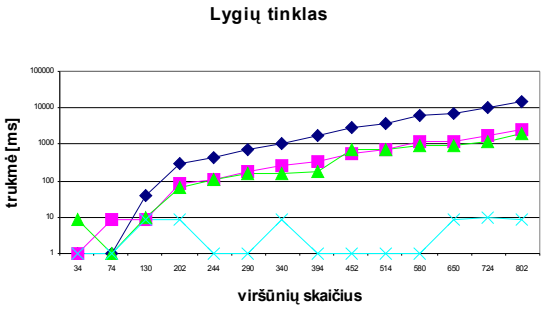
Atsitiktinių skaičių generatoriaus reikšmė visada buvo imama iš sisteminio kompiuterio laikrodžio.

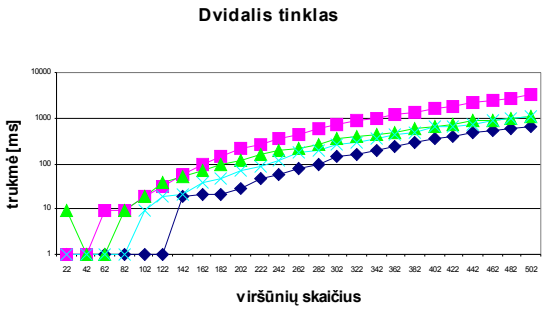
Eksperimentiškai surinktų duomenų kiekį ribojo asmeninio kompiuterio galimybės – esant didesniam tinklo viršūnių ir lankų skaičiui, skaičiavimai užima pakankamai daug laiko.

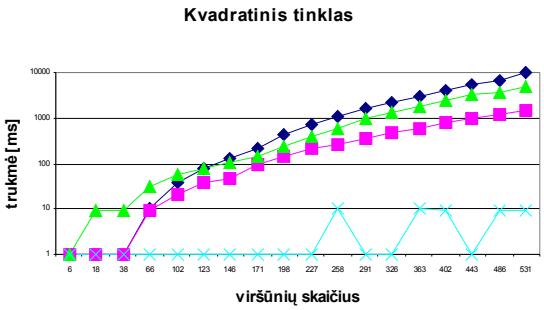
Pirmojoje eksperimentinių rezultatų grupėje pateikiamas palyginimas, kaip skirtingi algoritmai skaičiuoja maksimalų srautą tame pačiame tinkle. Kuri grafiko kreivė kurį algoritmą pavaizduoja, parodyta čia:

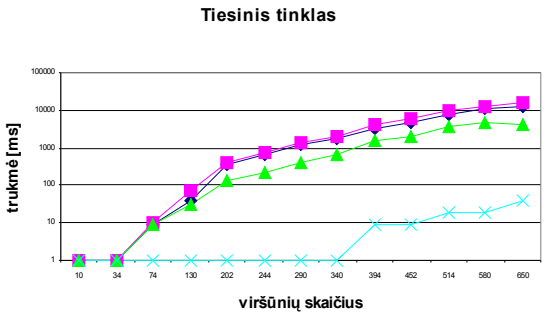


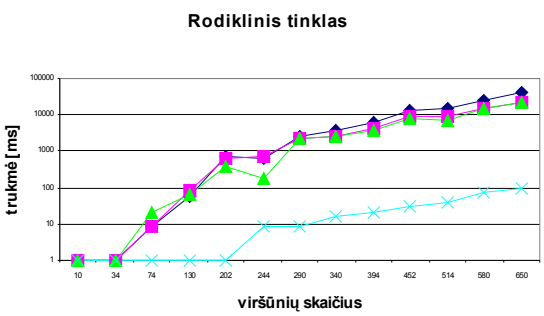












Iš pateiktų eksperimentinių rezultatų matyti, kad algoritmai nėra vienodai efektyvūs bet kokiam tinklui: Fordo-Falkersono algoritmas, pavyzdžiui, geriau už kitus ieško maksimalaus srauto dvidaliame tinkle; kvadratiniame tinkle Edmondso-Karpo algoritmas lenkia Dinico algoritmą. Goldbergo algoritmas, būdamas efektyviausias visuose kituose tinkluose, dvidaliame tinkle tuo nepasižymi. Išvada – norint

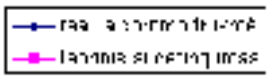
efektyviai ieškoti maksimalų srautą tinkle, pravartu žinoti tinklo struktūrą ir pagal ją pasirinkti geriausiai jam tinkantį algoritmą.

Antroji eksperimentinių rezultatų grupė – tai realios algoritmo darbo trukmės palyginimas su teoriniu sudėtingumu – t.y. ta reikšme, kuria charakterizuojamas algoritmo efektyvumas:

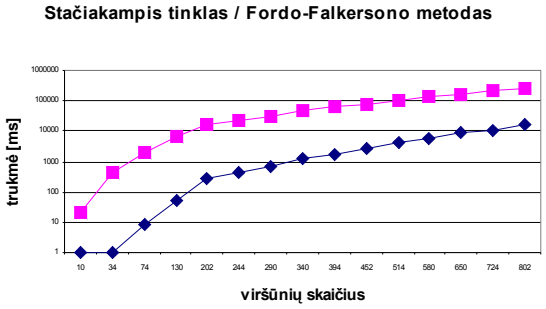
- Fordo - Falkersono algoritmui – O (n m U),
- Edmondso - Karpo algoritmui – O (n m^2),
- Dinico algoritmui – O (n^2 m),
- Goldbergo algoritmui – O (n m log(n^2/m));

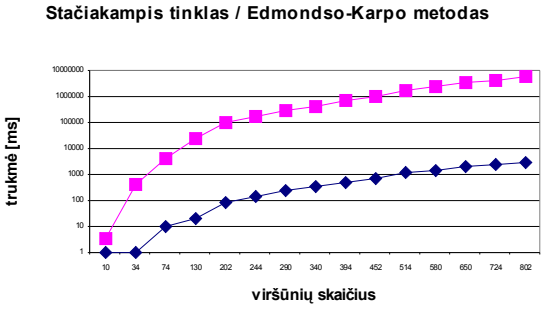
kur n – tinklo viršūnių skaičius, m – lankų skaičius, U – didžiausias lankų pralaidumas.

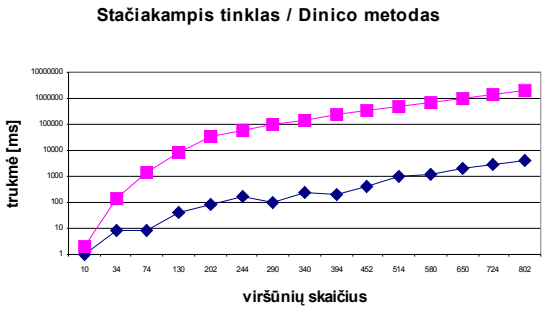
Toliau pareikiamuose grafikuose esančios kreivės reiškia :

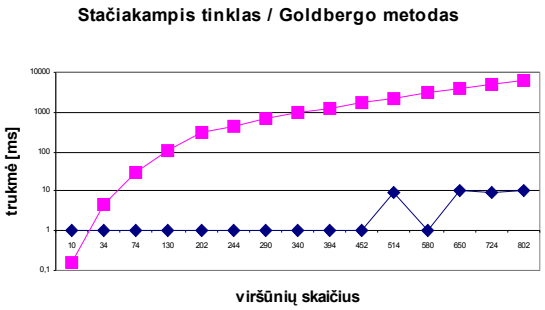


Stačiakampis tinklas :

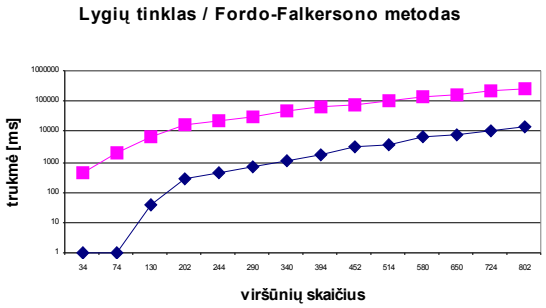


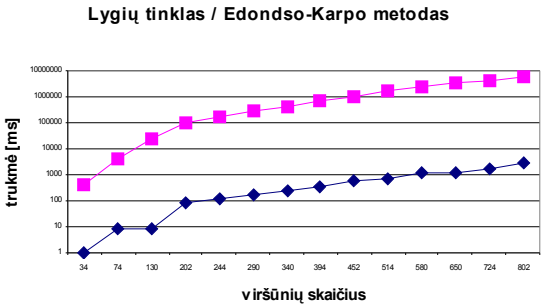


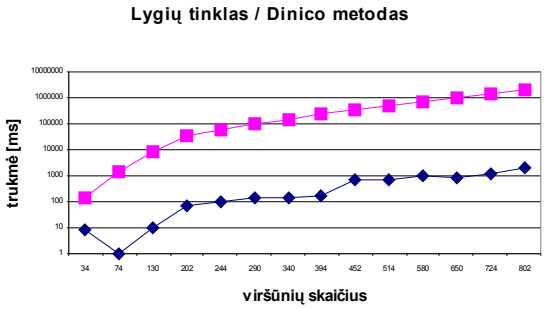


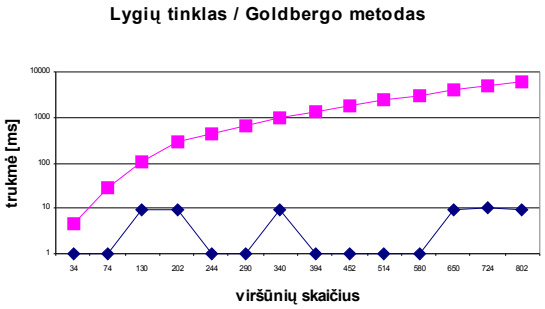


Lygių tinklas :

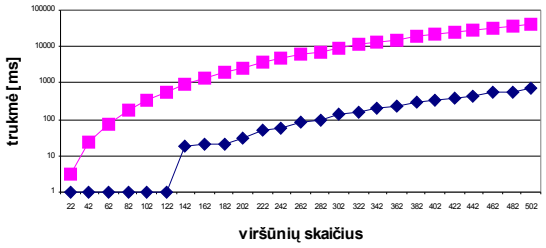


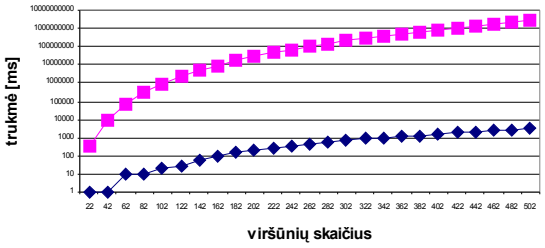




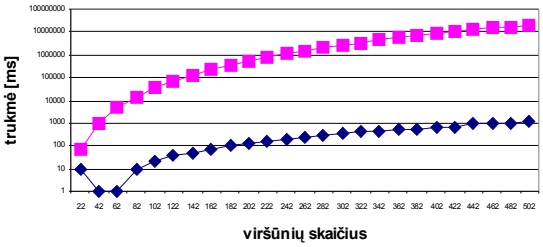


Dvidalis tinklas :

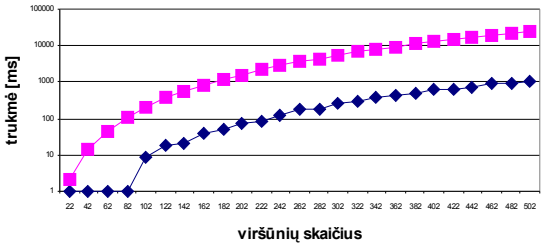




Dvidalis tinklas / Dinico metodas

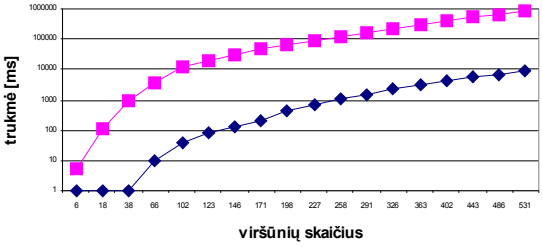


Dvidalis tinklas / Goldbergo metodas

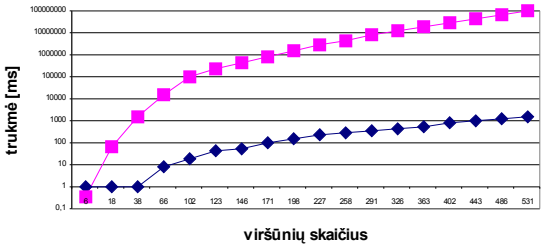


Kvadratinis tinklas :

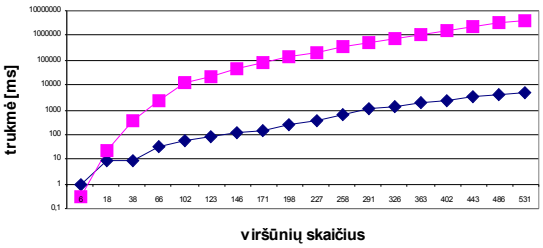
Kvadratinis tinklas / Fordo-Falkersono metodas



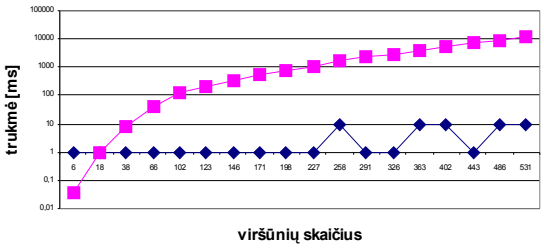
Kvadratinis tinklas / Edmondso-Karpo metodas



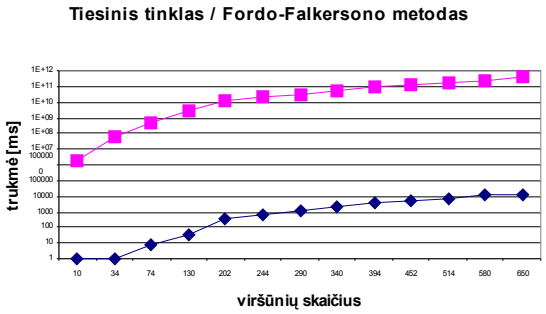
Kvadratinis tinklas / Dinico metodas

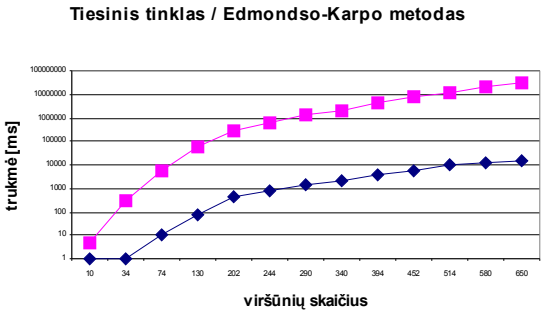


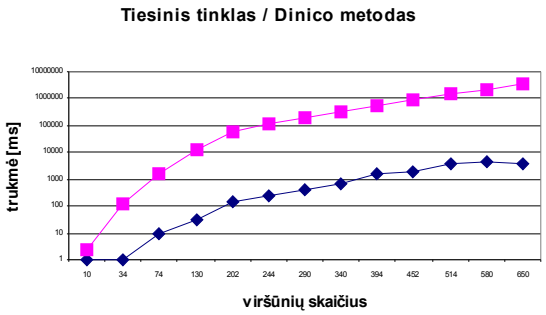
Kvadratinis tinklas / Goldbergo metodas

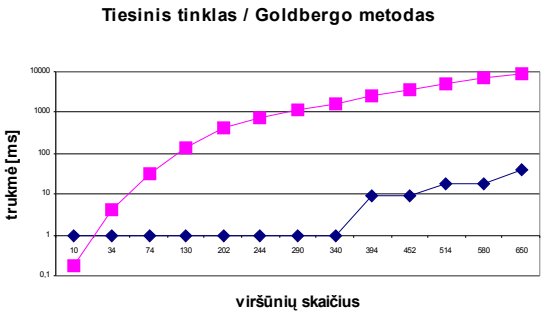


Tiesinis tinklas :

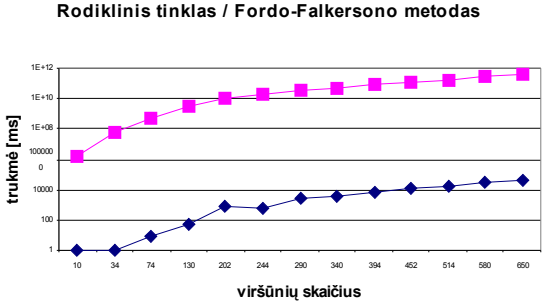


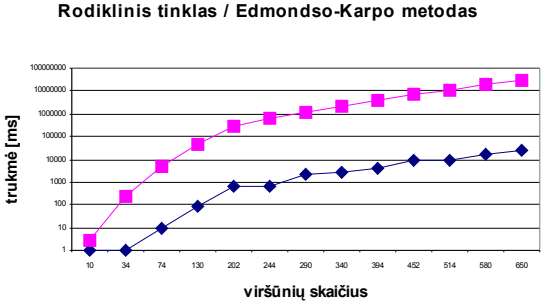


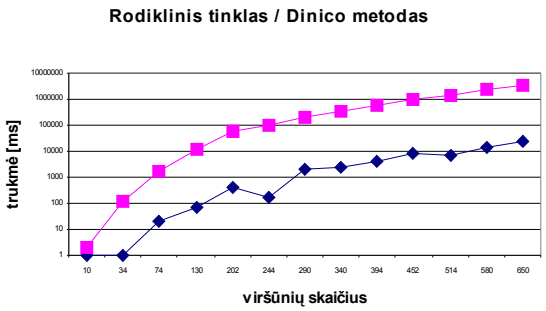


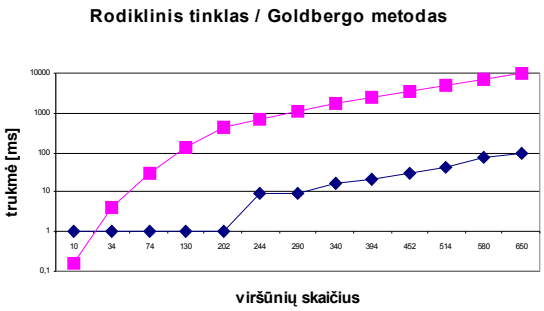


Rodiklinis tinklas :









Kaip matyti iš šių rezultatų, praktiškai sugaištas laikas apskaičiuoti srautą tinkle iš tiesų visada yra mažesnis, negu teoriškai įrodytas.

IŠVADOS

Atlikus šį darbą, gauti tokie rezultatai:

- sukurta programinė įranga, realizuojanti keturis klasikinius maksimalaus srauto skaičiavimo tinkle metodus – Fordo-Falkersono, Edmondso-Karpo, Dinico ir Goldbergo;
- šios programinės įrangos pagalba surinkti statistiniai duomenys apie minėtų metodų efektyvumą. Jie parodė, kad metodų efektyvumas priklauso nuo tinklo, kuriame ieškomas srautas, tipo. Taip pat leido įsitikinti, kad realizuoti kiekvieną metodą galima per trumpesnį laiką, negu teoriškai įrodyta;
- sukurtoji sistema patogi naudoti; jos pagalba galima matuoti kiekvieno aptarto metodo darbo trukmę, kaupti duomenis statistiniam apdorojimui ir vaizdžiai iliustruoti maksimalaus srauto paieškos procesą pasirinktame tinkle kiekvienu aptartu metodu.

LITERATŪRA

[1]. Thomas H. Cormen , Charles E. Leiserson , Ronald L. Rivest Introduction to Algorithms, Second Edition, Cambridge , Massachusetts London, England, 2001 by The Massachusetts Institute of Technology

[2]. G. B. Dantzig. Application of the Simplex Method to a Transportation Problem. In T. C. Koopmans, editor, Activity Analysis and Production and Allocation, pages 359-373. Wiley, New York,1951.

[3]. L. R. Ford, Jr. and D. R. Fulkerson. Maximal Flow Through a Network. Canadian Journal of Math., 8:399-404, 1956.

[4]. E. A. Dinic. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. Soviet Math. Dokl., 11:1277-1280, 1970.

[5]. Е. А. Диниц. Метод поразрядного сокращения невязок и транспортные задачи. Исследования по дискретной математике. Наука, Москва, 1973.

[6]. J. Edmonds and R. M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. J. Assoc. Comput. Mach., 19:248-264, 1972.

[7]. A. V. Karzanov. Determining the Maximal Flow in a Network by the Method of Preflows. Soviet Math. Dok., 15:434-437, 1974.

[8]. A. V. Goldberg. A New Max-Flow Algorithm. Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, M.I.T., 1985.

[9]. A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum Flow Problem. J. Assoc. Comput. Mach., 35:921-940, 1988.

[10]. Andrew V. Goldberg, Satish Rao. Length Functions for Flow Computations. Technical Report # 97-055, NEC Research Institute, Inc., March 1997; revised August 1997.

[11]. Andrew V. Goldberg. Recent Developments in Maximum Flow Algorithms, Technical Report #97-045, NEC Research Institute, Inc., April 1998.

[12]. Daniel D. Sleator and Robert E. Tarjan. A data structure for dynamic trees. Journal Computer Systems Science. 26: 362-391, (1983).

[13]. Center for Discrete Mathematics & Theoretical Computer Science, http://dimacs.rutgers.edu/

[14]. ftp://dimacs.rutgers.edu/pub/challenge/graph/doc/ccformat.tex

[15]. Kostas Plukas, Eugenijus Mačikėnas, Birutė Jarašiūnaitė, Irena Mikuckienė. Taikomoji diskrečioji matematika, Kaunas, Technologija, 2002.

[16]. The Programmer's File Format Collection, http://www.wotsit.org/

[17]. University of Washington, Computer Science & Engineering, http://www.cs.washington.edu/homes/anderson/

1 PRIEDAS. Duomenų failo pavyzdys

```
c 0 – Kažkoks tinklas
p max 6 12
n 1 s
n 6 t
k 1  20 200
k 2  80 300
k 3 100  10
k 4 150   0
k 5 200 350
k 6 250 300
a 1 3 13
a 1 5 16
a 1 2  1
a 1 6  8
a 2 4  6
a 2 6 10
a 3 4 14
a 3 5  2
a 3 6 11
a 4 5  5
a 4 6 17
a 5 6  7
```