

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

**Algoritmas maksimaliam srautui dinaminuose tinkluose rasti**  
(Algorithm for the maximal flow in dynamic networks)

Atliko: 4 kurso 2 grupės studentas

Aleksas Vaitulevičius (parašas)

Darbo vadovas:

lekt. Irmantas Radavičius (parašas)

Recenzentas:

doc. dr. Vardauskas Pavardauskas (parašas)

Vilnius  
2018

## Santrauka

Paskutiniuose dviejuose dešimtmečiuose yra plačiai domimasi dinaminiais grafais, dėl jų naudotos tokiose srityse kaip: komunikacijos tinkluose, VLSI kūrime, kompiuterinėje grafikoje. Šis darbas apima dinaminių grafų problemą, maksimalaus srauto radimą, viena iš labiausiai fundamentalių optimizavimo problemų. Šio darbo tikslas yra įrodyti, kad pateikiamo maksimalaus srauto radimo algoritmas yra efektyvesnis už statinio grafo Fordo Fulkersono algoritmą skaičiavimui panaudojamų briaunų atžvilgiu. Pateikiamo algoritmo veikimas yra pagrįstas modifikuotu grupavimo metodu ir modifikuotu Fordo Fulkersono algoritmu. Grupavimo metodas yra skirtas grafo išskirtymui į grupes, o modifikuotas Fordo Fulkersono algoritmas yra skirtas apskaičiuoti konkretaus regiono maksimaliems srautams. Atmintyje bus saugomas regionų išsidėstymas ir jų talpos grafo pavidale. Perskaičiavus grupes kuriuose įvyko pokytis, galima rasti pakitusio grafo maksimalų srautą.

Raktažodžiai: dinaminiai grafai, maksimalūs srautai, grupavimo metodas, Fordo Fulkersono algoritmas, euristiniai bandymai.

## Summary

In the last two decades, there was a growing interest in dynamical graphs, because of their use in such fields like: communication network, VLSI design, graphics. This work is only about dynamical graph problem, maximum flow, one of the most fundamental optimization problems. The main task of this work is to prove that provided algorithm for solving maximum flow problem is more efficient than Ford Fulkerson algorithm for static graph in the context of used edges for calculation. This algorithm is based on modified clustering method and modified Ford Fulkerson algorithm. Clustering method is used for clustering graph into clusters and modified Ford Fulkerson algorithm is used for computing maximum flows in particular cluster. Positioning and capacity of clusters will be saved in memory. Once updated clusters are computed, it is possible to find maximum flow for the whole updated graph.

Keywords: dynamic graphs, max flow, clustering, Ford Fulkerson algorithm, heuristic experiments.

## Turinys

Santrauka .....	1
Summary .....	2
Sąvokų apibrėžimai .....	4
Įvadas .....	6
1. Algoritmas maksimaliam srautui dinaminuose tinkluose rasti .....	8
1.1. Grupės apibrėžimas .....	8
1.2. Fordo Fulkersono algoritmas .....	8
1.3. Grupavimo funkcija .....	10
1.4. Sukurtas algoritmas .....	13
1.4.1. Inicializavimo fazė .....	13
1.4.2. Po kiekvieno pokyčio fazės pagalbinės funkcijos .....	14
1.4.3. Po kiekvieno pokyčio fazės funkcijos kiekvinam pokyčio tipui .....	15
2. Sukurto algoritmo įgyvendinimas .....	18
3. Empiriniai bandymai .....	19
3.1. Tinklų generuojantis modulis .....	19
3.2. Bandymus atliekantis modulis .....	20
3.3. Statistiniai skaičiavimai ir jų rezultatai .....	20
Išvados .....	21
Conclusions .....	22
Literatūra .....	23
Priedas Nr.1	
Priedas Nr.2	

## Sąvokų apibrėžimai

1. Dinaminiai grafai (angl. Dynamic graphs) - tai grafai, kuriuose galima atlikti tokias operacijas: pridėjimo ( pridėti briauną, pridėti viršūnę), atėmimo (atimti briauną, atimti viršūnę), papildomos operacijos priklausomai nuo grafo savybių (pavyzdžiui jei grafas yra svorinis, tai jis galėtų turėti papildomą operaciją keisti svorį).
2. Dalinai dinaminis grafas (angl. Partial dynamic graph) - tai dinaminis grafas, kuriame yra uždrausta bent viena, bet ne visos dinaminio grafo operacijos.
3. Pilnai dinaminis grafas (angl. Fully dynamic graph) - tai dinaminis grafas, kuriame yra leidžiamos visos operacijos.
4. Inkrementalus dinaminis grafas (angl. Incremental dynamic graph) - dinaminis grafas, kuriame yra leidžiama pridėjimo operacijos.
5. Dekrementalus dinaminis grafas (angl. Decremental dynamic graph) - dinaminis grafas, kuriame leidžiamos atėmimo operacijos.
6. Grupė (angl. Cluster) - tinklo subgrafas, kuris yra naudojamas grupavimo metode.
7. Grupavimo metodas (angl. Clustering) - algoritmo dinaminiam grafiui konstravimo būdas, kuriame grafas yra suskirstomas į grupes.
8. Tinklas (angl. Network) - orientuotas grafas, kurio briaunos turi talpas.
9. Srautas (angle. Flow) - tai leistinas kelias konkrečiam kiekiui iš šaltinio į tikslą.
10. Šaltinis (angl. Source) - tinklo viršūnė, kuri yra srauto pradžios taškas.
11. Tikslas (angl. Sink) - tinklo viršūnė, kuri yra srauto pabaigos taškas.
12. Talpa (angl. Capacity) - briaunos savybė, kuri nurodo koks kiekis gali ja praeiti.
13. Maksimalus srautas (angl. Max flow) - srautas, kurio dydis yra didžiausias iš visų leistinų srautų.
14. Vieno šaltinio ir kelių tikslų maksimalus srautas (angl. Single source multi sink max flow) - tai maksimalaus srauto tipas, kuriame yra vienas šaltinis ir daugiau nei vienas tikslas.
15. Kelių šaltinių ir vieno tikslo maksimalus srautas (angl. Multi source single sink max flow) - tai maksimalaus srauto tipas, kuriame yra daugiau nei vienas šaltinis ir vienas tikslas.
16. Kelių šaltinių ir kelių tikslų maksimalus srautas (angl. Multi source multi sink max flow) - tai maksimalaus srauto tipas, kuriame yra daugiau nei vienas šaltinis ir daugiau nei vienas tikslas.

17. Vieno šaltinio ir vieno tikslo maksimalus srautas (angl. Single source single sink max flow)  
- tai maksimalaus srauto tipas, kuriame yra vienas šaltinis ir vienas tikslas.
18. BFS (angl. Breadth first search) - paieška į plotį.
19. Euristinis eksperimentas (angl. heuristic experiment) -

## Įvadas

Paskutiniuose dviejuose dešimtmečiuose yra plačiai domimasi algoritmais skirtais spręsti įvairias problemas dinaminuose grafuose. Šių problemų sprendimai optimizuoja tokias sritis kaip: komunikacijos tinklus, VLSI kūrimą, kompiuterinę grafiką [DFI01; EGI98]. Šios problemos yra statinių grafų problemų poaibis. Tačiau dinaminių grafų problemų sprendimai gali būti labiau optimizuoti nei statinių, nes yra daugiau informacijos apie grafą nei statiniame grafe (pavyzdžiui, jei buvo apskaičiuotas grafo maksimalus srautas ir prie grafo buvo pridėta briauna, tai bus žinomas grafo poaibio, kuriam nepriklauso naujai pridėta briauna, maksimalus srautas). Vienai iš šių problemų, maksimalaus srauto paieškos problemai, buvo sukurtas algoritmas kursiniame darbe. Sukurtasis algoritmas šiame darbe buvo įgyvendintas ir ištirtas. Įgyvendinimo metu buvo pastebėta daug netikslumų, tad algoritmas buvo pataisytas.

Dinaminis grafas - tai grafas, kuriam yra galima atlikti bent vieną iš šių operacijų: pridėjimo (pridėti briauną, pridėti viršūnę), atėmimo (atimti briauną, atimti viršūnę), papildomos operacijos priklausomai nuo grafo savybių (pavyzdžiui jei grafas yra svorinis, tai jis galėtų turėti papildomą operaciją keisti svorį). Pagal leidžiamas operacijas dinaminiai grafai yra skirstomi į: dalinai dinaminis grafus, kurie yra skirstomi į inkrementalius (vykdoma tik pridėjimo operacija) ir dekrementalius (vykdoma tik atėmimas), ir pilnai dinaminis grafus, kuriuose vykdomos visos operacijos. Sukurtas algoritmas yra skirtas spręsti pilnai dinaminio grafo uždavinį.

Maksimalus srautas - tai didžiausias galimas srautas tinkle iš viršūnių  $s_i$  (šaltinių) iki viršūnių  $t_i$  (tikslų). Tinklas - tai orientuotas grafas  $G = \{V, E, u\}$ , kur  $V$  yra viršūnių aibė,  $E$  - briaunų aibė, o  $u$  - briaunų talpų aibė ( $u : E \rightarrow R$ ). Pagal šaltinių ir tikslų skaičių ši problema yra skirstoma į:

- Vieno šaltinio ir kelių tikslų - tai srautas, kuriame yra vienas šaltinis ir daugiau nei vienas tikslas.
- Kelių šaltinių ir vieno tikslo - tai srautas, kuriame yra vienas tikslas ir daugiau nei vienas šaltinis.
- Kelių šaltinių ir kelių tikslų - tai srautas, kuriame yra daugiau nei vienas šaltinis ir daugiau nei vienas tikslas.
- Vieno šaltinio ir vieno tikslo - tai srautas, kuriame yra vienas šaltinis ir vienas tikslas.

Sukurtas algoritmas gali rasti tik vieno šaltinio ir kelių tikslų maksimalų srautą. Tačiau tarpinėms reikšmėms gauti algoritmas turi rasti ir visų kitų tipų maksimalius srautus. Tam naudojamas Fordo Fulkersono algoritmas [JF62] pritaikytas rasti kelių šaltinių ir kelių tikslų maksimalius srautus. Taip modifikuotas Fordo Fulkersono algoritmas gali rasti maksimalius srautus grupėse. Grupės - tai tinklo, kuriame ieškomas maksimalus srautas, subgrafai. Metodas, kuriame yra naudojamos grupės yra vadinamas grupavimo metodu [Fre85], kuriame tinklas yra padalinamas į grupes. Šis metodas buvo pritaikytas konstruojant algoritmą.

Sukurto algoritmo korektiškumui įrodyti reikia pateikti formalų įrodymą. Tačiau formalus įrodymas yra sudėtingas procesas. Todėl verta ištirti algoritmo korektišką veikimą naudojantis empiriniais tyrimais prieš formalų įrodymą. Šie tyrimai suteikia galimybę lengvai nustatyti ar sukurtas algoritmas veikia nekorektiškai. Jei empiriniais tyrimais yra nustatomas nekorektiškas algoritmo veikimas, tai reiškia, kad formalus įrodymas yra neprasmingas. Tad šiame darbe yra atliekami empiriniai tyrimai, o ne formalus įrodymas. Taip pat nėra nustatyta ar sukurtas algoritmas yra efektyvesnis už algoritmą randantį maksimalų srautą statiniame tinkle. Jei sukurtas algoritmas nėra efektyvesnis, tai jo formalus įrodymas yra neprasmingas, nes algoritmai skirti statiniams grafams, gali būti pritaikyti ir dinaminiais. Tad algoritmai, skirti dinaminiais grafams, yra naudojami tik tuo atveju jei jie yra efektyvesni už algoritmus, skirtus statiniams grafams.

Tad šio darbo **TIKSLAS** yra nustatyti ar sukurtą algoritmą verta formaliai įrodyti. Šiam tikslui pasiekti reikia atlikti šiuos uždavinius:

1. Pateikti sukurtą algoritmą:
  - (a) Pateikti Fordo Fulkersono algoritmą ir kaip jis buvo pritaikytas kelių šaltinių ir kelių tikslų problemai.
  - (b) Pateikti grupavimo funkciją.
  - (c) Pateikti funkciją, kuri randa viso tinklo maksimalų srautą su pakitusiu bent vienu maksimaliu srautu vienoje ar keliose grupėse.
  - (d) Pateikti sukurto algoritmo pagrindines funkcijas.
2. Pateikti sukurto algoritmo panaudojimo pavyzdžius.
3. Įgyvendinti sukurtą algoritmą.
4. Įgyvendinti ir atlikti empirinius tyrimus.
5. Atlikti statistinius skaičiavimus su atliktų tyrimų rezultatais.

Darbas susideda iš trijų dėstymo skyrių. Pirmame skyriuje yra pateikiamas sukurtas algoritmas (1. užduotis). Jame išdėstoma kaip Fordo Fulkersono algoritmas buvo pritaikytas sukurtam algoritmui (1.a užduotis), grupavimo funkcija (1.b užduotis), funkcija, kuri randa viso tinklo maksimalų srautą su pakitusiu bent vienu maksimaliu srautu vienoje ar keliose grupėse (1.c užduotis), ir sukurto algoritmo pagrindinė funkcija (1.d užduotis). Antrame skyriuje pateikiami sukurto algoritmo pavyzdžiai (2 užduotis). Trečiame skyriuje yra aprašomas sukurto algoritmo įgyvendinimas (3 užduotis). Ketvirtame skyriuje yra aprašomi empiriniai bandymai, pateikiami jų rezultatai (4 užduotis), aprašomi atlikti statistiniai skaičiavimai ir jų rezultatai (5 užduotis).



# 1. Algoritmas maksimaliam srautui dinaminuose tinkluose rasti

## 1.1. Grupės apibrėžimas

Grupė - tai grupuojamo tinklo  $T$  subgrafas  $G = \{V, E, u\}$ . Subgrafo  $G$  šaltiniai  $s_i$  yra:

1. tinklo šaltinis, jei jis yra subgrafo  $G$  viršūnių aibėje,
2. menamos viršūnės. Jei  $\exists x : x \in V$  ir grupė  $G_i : G_i \neq G$  turi viršūnę  $y$ , kuri nepriklauso grafui  $G$ , bei egzistuoja briauna  $y \rightarrow x$ , tai egzistuoja menama viršūnė  $x'$  ir briauna  $x' \rightarrow x$ , kurios talpa yra lygi grupės  $G_i$  maksimaliam srautui su tikslu  $x$ .

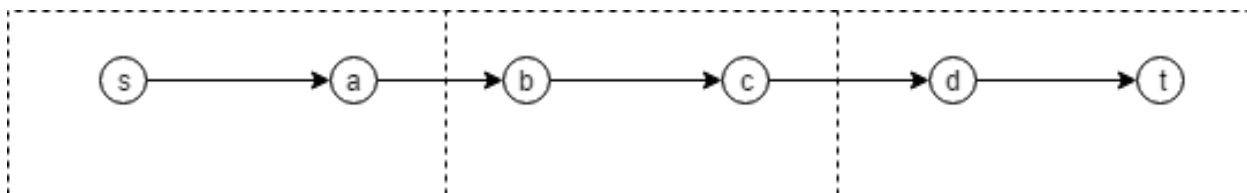
subgrafo  $G$  tikslai  $t_i$  yra:

1. tinklo tikslas, jei jis yra subgrafo  $G$  viršūnių aibėje,
2. menama viršūnė. Jei  $\exists x : x \in V$  ir grupė  $G_i : G_i \neq G$  turi viršūnę  $y$ , kuri nepriklauso grafui  $G$ , bei egzistuoja briauna  $x \rightarrow y$ , tai egzistuoja menama viršūnė  $x'$  ir briauna  $x \rightarrow x'$ , kurios talpa yra lygi briaunos  $x \rightarrow y$  talpai.

Kiekviena tinklo  $T$  viršūnė  $v$  priklauso tik vienai grupei. Kiekviena tinklo  $T$  briauna  $e$  priklauso tik vienai grupei, nebent  $e = x \rightarrow y : x \in G_i, y \in G_j, i \neq j$ . Taip pat jei tinkle egzistuoja subgrafas, kuriame yra Eulerio ciklas, tai visos viršūnės priklausančios tam subgrafui turi būti vienoje grupėje.

Pavyzdys: tarkime turime tinklą  $G = V = s, a, b, c, d, t, E = s \rightarrow a, a \rightarrow b, b \rightarrow c, c \rightarrow d, d \rightarrow t$ , kuris yra sugrupuotas į grupes, kurių  $V$  yra lygūs  $s, a, b, c, d, t$ . Šis grupavimas pavaizduotas paveikslėlyje - 1.

1 pav. Grupavimo pavyzdys



Tada subgrafo  $b, c$  šaltinis yra menama viršūnė  $s_a$ , kuri yra sujungta briauna, kurios talpa yra subgrafo  $s, a$  maksimalus srautas iki tikslo  $b$ , o tikslas yra  $d$ .

## 1.2. Fordo Fulkersono algoritmas

Maksimaliems srautams grupėse rasti algoritmas naudoja modifikuotą Fordo Fulkersono algoritmą, kuris naudoja BFS [BAP13] galimam srautui rasti. Originalus Fordo Fulkersono algoritmas

[JF62] yra skirtas rasti vieno šaltinio ir vieno tikslo maksimalų srautą, tačiau sukurto algoritmo atveju gali susidaryti grupės, kurios turi kelis tikslus ir arba kelis šaltinius. Tad modifikuotas Fordo Fulkersono algoritmas yra skirtas rasti kelių šaltinių ir kelių tikslų maksimalų srautą.

BFS algoritmas, aprašytas publikacijoje :

1. Inicializuojami masyvai  $V$ ,  $Q$  ir  $FLOW$ , į  $V$  ir  $Q$  patalpinamas tinklo šaltinis.
2. Jei  $Q$  yra tuščias, tai einama į žingsnį 9.
3. Išimamas paskutinis masyvo  $Q$  elementas  $y$ .
4. Jei  $\nexists$  viršūnė  $x : y \rightarrow x, x \notin V$ , tai einama į žingsnį 2.
5. Briauna  $y \rightarrow x$  patalpinama į masyvą  $FLOW$ .
6. Viršūnė  $x$  patalpinama į masyvą  $V$ .
7. Viršūnė  $x$  patalpinama į  $Q$  masyvo pradžią.
8. Einama į žingsnį 4.
9. Baigiamas algoritmas.

Klasikinis Fordo Fulkersono algoritmas, kuris yra aprašytas publikacijoje bei naudojantis BFS:

1. Maksimaliam srautui priskiriama reikšmė nulis. Sukuriama tinklo kopija  $G$  ir inicializuojama tinklo  $MAX$  reikšmė  $V=\{\}$ ,  $E=\{\}$ ,  $u=\{\}$ .
2. Naudojant BFS randamas srautas nuo šaltinio iki tikslo.
3. Jei nėra vieno srauto nėra randama einama į žingsnį 8.
4. Sumažinama visų briaunų, kurie priklauso rastam srautui, talpas per rasto srauto dydį tinkle  $G$ .
5. Rastas srautas pridedamas prie tinklo  $MAX$ .
6. Maksimalaus srauto reikšmė yra padidinama per rasto srauto dydį.
7. Einama į žingsnį 2.
8. Baigiamas algoritmas.

Modifikuotas Fordo Fulkersono algoritmas, naudojantis BFS:

1. Masyvo maksimalaus srauto dydžiai, kurio dydis yra lygus tikslų skaičiui, reikšmės nustatomos į nulį. Sukuriama tinklo kopija  $G$  ir inicializuojama tinklo  $MAX$  reikšmė  $V=\{\}$ ,  $E=\{\}$ ,  $u=\{\}$ .

2. Naudojant BFS randami srautai nuo visų šaltinių iki visų tikslų.
3. Jei nėra vieno srauto nėra randama einama į žingsnį 8.
4. Sumažinama visų briaunų, kurie priklauso rastiems srautams, talpas per rasto srauto dydį tinkle G.
5. Rasti srautai pridedamas prie tinklo MAX.
6. Masyvo maksimalaus srauto dydžiai elementų, kurie atitinka pasiektus tikslus, reikšmės padidinamos per rastų srautų dydžius.
7. Einama į žingsnį 2.
8. Baigiamas algoritmas.

### 1.3. Grupavimo funkcija

Šiame darbe tiriamas algoritmas yra paremtas Frederiksono suformuluotu grupavimo metodu [Fre85]. Grupavimo metodas - tai metodas, kuris yra pagrįstas grafo dalinimu į subgrafus vadinamus grupėmis. Grafas yra padalinamas taip, kad kiekviena atlikta operacija turėtų įtakos tik daliai grupių, bet ne visoms. Todėl tiriamas algoritmas veikia pagal grupavimo metodą tik tada kai yra patenkinta sąlyga: jei tinkle egzistuoja subgrafas, kuriame yra Eulerio ciklas, tai visos viršūnės priklausančios tam subgrafui yra vienoje grupėje. Šitai sąlygai patenkinti yra naudojama grupavimo funkcija, kuri naudoja šias pagalbines funkcijas:

Subgrafų su Eulerio ciklais radimo funkcija - tinklo EG, kurio viršūnės yra grupių, tenkinančių pateiktą sąlygą (subgrafai su Eulerio ciklais), viršūnių masyvai, kūrimo funkcija:

1. Iš apskaičiuojamo tinklo  $C = \{V_C, E_C, u_C\}$  sukuriamas tinklas EG, kurio viršūnės būtų apskaičiuojamo tinklo viršūnės patalpintos masyvuose, o briaunos atitiktų apskaičiuojamo tinklo briaunas.
2. Sukuriamas masyvas B, kuriame talpinamos briaunos, su kuriomis reikia daryti skaičiavimus, stekas PATH, kuriame talpinamos aplankytos viršūnės, ir viršūnių iteratorius x, jam suteikiama C šaltinio reikšmė ir patalpinamas į steką PATH.
3. Jei PATH yra tuščias, einama į žingsnį 19.
4. Jei masyve B  $\exists x \rightarrow y : y \in V_C$ , tai sukuriamas masyvas B', į kurį yra sudedamos visos viršūnės y iš B masyvo.
5. Jei masyve B  $\nexists x \rightarrow y : y \in V_C$ , tai sukuriamas masyvas B', į kurį yra sudedamos visos viršūnės y iš tinklo V.
6. Jei B' yra tuščias tai einama į žingsnį 17.

7. Iš masyvo  $B'$  yra išimamas pirmas elementas  $y$ .
8. Jei  $\nexists y \in V_C$ , tai surandama viršūnė  $z$ , kuri turi visus  $y$  elementus (toliau  $y := z$ ).
9. Jei  $PATH$  neturi elemento  $y$ , tai elementas  $y$  įdedamas į  $PATH$  ir einama į žingsnį 15.
10. Inicializuojama nauja viršūnė  $n$  su visais  $y$  elementais.
11. Iš  $PATH$  išimamas elementas  $z$ .
12. Jei elementas  $z = y$ , tai einama į žingsnį 15 ir  $y = n$ .
13. Tinklo  $EG$  viršūnės  $z$  ir  $n$  yra pakeičiamos  $x$  ir  $n$  konkatenacija (toliau  $n$  yra  $z$  ir  $n$  konkatenacija).
14. Einama į žingsnį 11.
15. Visi  $x \rightarrow y : y \in B'$  įdedami į masyvą  $B$  ir viršūnių iteratoriui  $x$  suteikiama  $y$  reikšmė.
16. Einama į žingsnį 3.
17. Iš steko  $PATH$  išimamas elementas, iteratoriui  $x$  suteikiama  $PATH$  viršutinio elemento reikšmė.
18. Einama į žingsnį 3.
19. Pabaiga.

Srautui priklausančių grupių sukūrimo funkcija - iš pateikto tinklo  $EG = \{V_E G, E_E G, u_E G\}$ , kurio viršūnės yra apskaičiuojamo tinklo  $C = \{V_C, E_C, u_C\}$  viršūnių, masyvai, sukuriamas grupių tinklas  $R$ .

1. Sukuriamas stekas  $B$ , kuriame talpinamos viršūnės, kurios priklauso konkrečiai grupei, masyvas  $VR$ , kuriame talpinamos viršūnės, kurias reikia ištrinti iš tinklo  $EG$ , stekas  $NC$ , kuriame talpinamos viršūnės kitų grupių kūrimui, ir viršūnių iteratorius  $x$ , jam suteikiama  $EG$  tinklo viršūnės, kurioje yra tinklo  $C$  šaltinis, reikšmė ir patalpinamas į steką  $NC$ .
2. Inicializuojama nauja grupė  $G$ .
3. Jei  $NC$  yra tuščias, einama į žingsnį 21.
4. Viršutinis  $NC$  elementas yra perkeliamas į  $B$  ir iteratoriui  $x$  yra suteikiama to elemento reikšmė.
5. Jei  $x$  dydis = 1, tai į tinklą  $G$  patalpinama viršūnė elemento  $x$  reikšmė ir einama į žingsnį 10.
6. Tinklui  $G$  priskiriama reikšmė  $G = \{x, Y, Y_u\}$ , kur  $Y$  yra tinklo  $C$  briaunos tarp  $x$  viršūnių, o  $Y_u$  - tai tų briaunų svoriai.

7. Naudojantis apibrėžimu sukuriamos tinklo R briaunos  $G \rightarrow Q$ , kur Q yra menamos viršūnės, nustatomi grupės G šaltiniai ir tikslai.
8. Tinklas G patalpinamas į tinklą R.
9. Tinklui G inicializuojama naujos grupės reikšmė.
10. Jei B yra tuščias einama į žingsnį 16.
11. Iš steko B išimamas viršutinis elementas, kurio reikšmė priskiriama iteratoriui x.
12. Jei  $\exists y : x \rightarrow y$ , y dydis  $> 1$ ,  $y \notin NC$ ,  $y \notin B$ , tai y talpinamas į steką NC ir einama į žingsnį 12.
13. Jei  $\exists y : x \rightarrow y$ , y dydis  $= 1$ ,  $y \notin NC$ ,  $y \notin B$ , tai y elemento reikšmė patalpinama į steką B, to elemento reikšmė pridedama į grupę G kaip viršūnė ir einama į žingsnį 12.
14. Į steką VR įdedama iteratoriaus x reikšmė.
15. Einama į žingsnį 10.
16. Tinklui G priskiriamos tinklo C briaunos, kurių viršūnės atitinka G viršūnes.
17. Naudojantis apibrėžimu sukuriamos tinklo R briaunos  $G \rightarrow Q$ , kur Q yra menamos viršūnės, nustatomi grupės G šaltiniai ir tikslai.
18. Tinklas G patalpinamas į tinklą R.
19. Tinklui G inicializuojama naujos grupės reikšmė.
20. Einama į žingsnį 3.
21. Iš tinklo EG ištrinamos visos viršūnės, kurios yra VR masyve.
22. Pabaiga.

Pati grupavimo funkcija:

1. Kviečiama subgrafų su Eulerio ciklais radimo funkcija (rezultatas tinklas EG).
2. Kviečiama srautui priklausančių grupių sukūrimo funkcija su tinklu EG.
3. Likusios tinklo EG viršūnės konkatenuojamos į masyvą X.
4. Sukuriamas tinklas  $G = \{X, Y, Y_u\}$ , kur Y yra tinklo C briaunos tarp X viršūnių, o  $Y_u$  - tai tų briaunų savoriai.
5. Naudojantis apibrėžimu sukuriamos tinklo R briaunos  $G \rightarrow Q$ , kur Q yra menamos viršūnės, nustatomi grupės G šaltiniai ir tikslai.

6. Tinklas  $G$  patalpinamas į tinklą  $R$ .

7. Pabaiga.

## 1.4. Sukurtas algoritmas

Sukurtas algoritmas turi dvi fazes: inicializavimo ir skaičiavimo po kiekvieno pokyčio. Inicializavimo fazėje yra išskaidomas pateiktas tinklas į grupes, apskaičiuojami maksimalūs srautai kiekvienoje grupėje, kuri priklauso maksimaliam pateikto tinklo srautui, ir iš rastų srautų yra gaunamas maksimalus pateikto tinklo srautas. Tada skaičiavimo po kiekvieno pokyčio fazėje yra laukiama pokyčio. Kai jis įvyksta, yra kviečiama funkcija priklausomai nuo to koks pokytis įvyko. Visos funkcijos gali pasiekti ir keisti grupių tinklą  $CLUSTERS = \{V_{CLUSTERS}, E_{CLUSTERS}, u_{CLUSTERS}\}$ , apskaičiuojamą tinklą  $NETWORK = \{V_{NETWORK}, E_{NETWORK}, u_{NETWORK}\}$ .

### 1.4.1. Inicializavimo fazė

Inicializavimo fazė (maksimalus  $NETWORK$  tinklo srautas yra grupėje su apskaičiuojamo tinklo tikslu):

1. Tinklui  $NETWORK$  yra kviečiama grupavimo funkcija, kuri grąžina grupių tinklą, kuris patalpinamas į tinklą  $CLUSTERS$ .
2. Inicializuojamas sąrašas  $MF$ , kuriame laikoma kokiam tikslui koks maksimalus srautas buvo apskaičiuotas, ir masyvas  $CAL$ , kuriame laikomos visos grupės, kurios jau buvo apskaičiuotos.
3. Apskaičiuojama grupės, kurioje yra tinklo  $NETWORK$  šaltinis, maksimalus srautas, kviečiant modifikuotą Fordo Fulkersono algoritmą. Rezultatas įsimenamas sąrašė  $MF$  ir pačioje grupėje.
4. Grupė  $G$  patalpinama į masyvą  $CAL$ .
5. Jei masyve  $CAL$  yra grupė, kurioje yra tinklo  $NETWORK$  tikslas tai einama į žingsnį 11.
6. Jei  $\nexists$  grupė  $G : Y \rightarrow G, \forall Y \in CAL, G \in V_{CLUSTERS}, G \notin CAL$ , tai einama į žingsnį 11.
7. Kiekvienam grupės  $G : Y \rightarrow G, \forall Y \in CAL, G \in V_{CLUSTERS}, G \notin CAL$  briaunai  $m \rightarrow s$ , kur  $m$  yra menama viršūnė, kuri yra grupės  $G$  šaltinis, yra suteikiama talpa iš sąrašo  $MF$  elemento, kuris atitinka tikslą  $s$ .
8. Grupei  $G$  apskaičiuojamas maksimalus srautas naudojant modifikuotą Fordo Fulkersono algoritmą. Rezultatas įsimenamas sąrašė  $MF$  ir pačioje grupėje.
9. Grupė  $G$  patalpinama į masyvą  $CAL$ .

10. Einama į žingsnį 5.

11. Algoritmo pabaiga.

#### 1.4.2. Po kiekvieno pokyčio fazės pagalbinės funkcijos

Dalis funkcijų, kurios yra kviečiamos po konkretaus pakeitimo, naudoja šias pagalbines funkcijas:

Perskaičiuoti pokyčio paveiktas grupes funkcija. Šios funkcijos parametrai yra grupių, kuriuose įvyko pokytis masyvas *AFFECTED* ir trinama viršūnė *DELETE* (jei viršūnė nėra nurodoma, tai jokia viršūnė neištrinama):

1. Inicializuojamas sąrašas *CHANGES*(*t*, *change*) elementų, kuriuose yra tikslų ir maksimalių srautų iki jų pokyčių poros.
2. Jei *AFFECTED* yra tuščias, tai einama į žingsnį 7.
3. Kiekvienai grupei  $G = \{V_G, E_G, u_G\}$  iš masyvo *AFFECTED*:
  - (a) Apskaičiuojamas naujas maksimalus srautas naudojantis Fordo Fulkersono algoritmu. Rezultatas patalpinamas sąrašė *RESULTS*.
  - (b) Kiekvienam grupės *G* tikslui *t*, iki kurio maksimalus srautas pakito:
    - i. Iš grupės *G* gaunama sena maksimalaus srauto iki *t* reikšmė *OLD*.
    - ii. Iš sąrašo *RESULTS* gaunama nauja maksimalaus srauto iki *t* reikšmė *NEW*.
    - iii. Į sąrašą *CHANGES* patalpinama pora *t* ir *NEW* - *OLD*.
    - iv. Jei  $NEW \neq 0$ , tai Grupėje *G* išsaugomas nauja srauto reikšmė iki *t*, *NEW*.
    - v. Jei  $NEW = 0$ , tai iš grupės *G* ištrinamas tikslas *t*.
4. Išimami visi elementai iš sąrašo *AFFECTED*.
5. Kiekvienai grupei  $NG = \{V_{NG}, E_{NG}, u_{NG}\} : NG \in V_{CLUSTERS}$ , kurios šaltinis *s* yra yra sąrašė *CHANGES*(*s*, *change*):
  - (a) Jei briaunos  $s \rightarrow x$ , kur  $x \in V_{NG}$ , talpa  $z : z + change \neq 0$ , tai briaunai  $s \rightarrow x$  suteikiama reikšmė  $z + change$ , einama į 5.a su kita grupe *NG*.
  - (b) Iš grupės *NG* ištrinamas šaltinis *s*.
  - (c) Jei  $DELETE = x$ , kur  $x : s \rightarrow x$ , tai viršūnė *x* yra ištrinama iš grupės *NG*.
  - (d) Jei grupės *NG* nėra masyve *AFFECTED*, tai grupė *NG* įdedama į masyvą *AFFECTED*.
6. Einama į žingsnį 2.
7. Algoritmo pabaiga.

Eulerio ciklo aptikimo funkcija. Ši funkcija nustato ar pridėta briauna  $x \rightarrow y$  sukuria naują subgrafą, kuriame yra Eulerio ciklas. Funkcijos parametrai yra briauna  $x \rightarrow y$ . Funkcija grąžina masyvą EULER, kuriame yra visos viršūnės priklausančios Eulerio ciklams:

1. Inicializuojamas stekas POSSIBLE, kuriame talpinami viršūnių masyvai, kurie gali sudaryti Eulerio ciklą.
2. Į steką POSSIBLE patalpinamas masyvas, kurio elementai yra viršūnės  $x$  ir  $y$ .
3. Inicializuojamas masyvas EULER.
4. Jei POSSIBLE yra tuščias, tai einama į žingsnį 10.
5. Sukuriamas masyvas CYCLE, jam suteikiama iš POSSIBLE steko išimto elemento reikšmė.
6. Jei  $\nexists a \rightarrow b$ , kur  $a$  yra paskutinė CYCLE masyvo elemento reikšmė, o  $b \in V_{NETWORK}$ , tai einama į žingsnį 4. Kitu atveju pasirenkamas bet kuris  $b$ .
7. Jei  $x = b$ , tai visos viršūnės  $z : z \in CYCLE, z \notin EULER$  yra pridedamos į masyvą EULER.
8. Jei  $x \neq b$ , tai sukuriamas CYCLE kopija CYCLE', į CYCLE' patalpinama viršūnė  $b$  ir CYCLE' patalpinamas į steką POSSIBLE.
9. Einama į žingsnį 6.
10. Algoritmo pabaiga.

#### 1.4.3. Po kiekvieno pokyčio fazės funkcijos kiekvinam pokyčio tipui

Jei įvyksta viršūnės  $x$  pridėjimas, tai tada yra įvykdoma funkcija:

1. Jei  $\exists$  grupė  $G : G \in CLUSTERS$ , kuri neturi šaltinių ir tikslų (iki grupės nėra kelio nuo NETWORK tinklo šaltinio), tai grupei  $G$  yra pridedama viršūnė  $x$  ir einama į žingsnį 5.
2. Sukuriama grupė  $N$ .
3. Grupei  $N$  yra pridedama viršūnė  $x$ .
4. Grupė  $N$  yra pridedama į tinklą CLUSTERS.
5. Algoritmo pabaiga.

Jei įvyksta viršūnės  $x$  atėmimas, tai tada yra įvykdoma funkcija:

1. Jei  $\nexists s \rightarrow x$ , kur  $s$  yra grupės  $SG \in CLUSTERS$ , šaltinis, tai randama grupė  $G = \{V_G, E_G, u_G\} : x \in V_G, G \in CLUSTERS$ , grupė  $G$  patalpinama į masyvą AFFECTED ir ištrinamas  $x$  iš grupės  $G$ .



2. Jei  $\exists s \rightarrow x$ , kur  $s$  yra grupės  $G : G \in CLUSTERS$  šaltinis ir grupės  $PG : PG \in CLUSTERS$  tikslas, tai iš  $PG$  ištrinama viršūnė  $x$  ir į  $AFFECTED$  įdedama grupė  $PG$ .
3. Kviečiama perskaičiuoti pokyčio paveiktas grupes funkcija su argumentais  $AFFECTED$  ir viršūnę  $x$ .
4. Algoritmo pabaiga.

Jei įvyksta briaunos  $x \rightarrow y$  pridėjimas, tai tada yra įvykdoma funkcija:

1. Kviečiama Eulerio ciklo aptikimo funkcija. Rezultatas patalpinamas į viršūnių masyvą  $EULER$ .
2. Jei  $EULER$  nėra tuščias:
  - (a) Sukuriamas masyvas  $AFFECTED$ , į masyvą  $AFFECTED$  sudedamos visos grupės, kurios turi bent vieną viršūnę, kuri yra  $EULER$  masyve.
  - (b) Grupės masyve  $AFFECTED$  yra sujungiamos į vieną grupę  $G$ .
  - (c) Ištrinamos grupės iš tinklo  $CLUSTERS$ , kurios yra masyve  $AFFECTED$ .
  - (d) Grupė  $G$  patalpinama į tinklą  $CLUSTERS$ .
  - (e) Kviečiama perskaičiuoti pokyčio paveiktas grupes funkcija su argumentu masyvu, į kurį patalpinta grupė  $G$ .
3. Jei  $EULER$  yra tuščias:
  - (a) Sukuriamas masyvas  $AFFECTED$ .
  - (b) Į  $AFFECTED$  masyvą įdedamos visos grupės, kurios turi viršūnę  $x$  ir priklauso tinklui  $CLUSTERS$ .
  - (c) Kviečiama perskaičiuoti pokyčio paveiktas grupes funkcija su argumentu  $AFFECTED$ .

4. Algoritmo pabaiga.

Jei įvyksta briaunos  $x \rightarrow y$  atėmimas, tai tada yra įvykdoma funkcija:

1. Randama grupė  $G : G \in V_{CLUSTERS}$  su briauna  $x \rightarrow y$
2. Grupės  $G$  briauna  $x \rightarrow y$ .
3. Grupė  $G$  patalpinama į masyvą  $AFFECTED$ .
4. Kviečiama perskaičiuoti pokyčio paveiktas grupes funkcija su argumentu  $AFFECTED$ .
5. Algoritmo pabaiga.

Jei įvyksta briaunos  $x \rightarrow y$  talpos pakeitimas į  $u$ , tai tada yra įvykdoma funkcija:

1. Randama grupė  $G : G \in V_{CLUSTERS}$  su briauna  $x \rightarrow y$ .
2. Grupės  $G$  briaunos  $x \rightarrow y$  talpa pakeičiama į  $u$ .
3. Grupė  $G$  patalpinama į masyvą  $AFFECTED$ .
4. Kviečiama perskaičiuoti pokyčio paveiktas grupes funkcija su argumentu  $AFFECTED$ .
5. Algoritmo pabaiga.

## 2. Sukurto algoritmo įgyvendinimas

Tam, kad pasiekti šio darbo tikslą buvo įgyvendintas sukurtas algoritmas. Įgyvendinimas buvo parašytas JAVA kalba versija 9, visos naudojamos bibliotekos yra įdiegiamos į algoritmo įgyvendinimą naudojantis įrankiu MAVEN versija 3. Šis įgyvendinimas naudoja šių bibliotekų funkcionalumą:

1. Jgrapht - ši biblioteka yra naudojama grafų saugojimui, generavimui ir skaičiavimams.
2. Jgraphx - ši biblioteka yra naudojama grafų atspausdinimui vartotojo grafinėje sąsajoje.
3. lombok - naudojama kodo generavimui, kuris yra skirtas pasiekti ir modifikuoti privačius laukus.
4. junit - naudojamas kodo modulių testavimui.

Įgyvendinto algoritmo architektūra pavaizduota 2. Šioje architektūroje kiekviena klasė atlieka šias funkcijas:

1. DynamicNetworkWithMaxFlowAlgorithm - šioje klasėje yra laikomas apskaičiuojamas dinaminis algoritmas NETWORK, yra atliekami pokyčiai tinklui NETWORK ir yra atliekama pagrindinės sukurto algoritmo funkcijos.
2. FordFulkerson - ši klasė atlieka modifikuoto Fordo Fulkersono algoritmo funkciją.
3. BFS - ši klasė atlieka paieška platyn algoritmo funkciją.
4. DividerToClusters - ši klasė atlieka grupavimo funkciją.
5. Network - ši klasė yra tinklas.
6. DynamicNetwork - ši klasė yra dinaminis tinklas.
7. EulerCycleWarps - ši klasė yra tinklas, kurio viršūnės yra grupių, tenkinančių sąlygą subgrafai su Eulerio ciklais, viršūnių masyvai.
8. ClustersNetwork - ši klasė yra grupių tinklas.

Klasės SimpleDirectedGraph<List<int>, DefaultEdge> ir SimpleDirectedWeightedGraph<int, WeightedEdge> yra klasės iš jgrapht bibliotekos.

### 3. Empiriniai bandymai

Tam, kad nustatyti ar verta formaliai įrodyti sukurta algoritma buvo atlikti empiriniai bandymai, kurie skirti ištirti algoritmo korektišką veikimą ir efektyvumą. Šiems bandymams atlikti buvo įgyvendintas sukurta algoritmas, modulis, generuojanti tinklus, ir modulis, kuris atlieka bandymus su tinklus generuojančio modulario rezultatais. Su bandymus atliekančio modulario rezultatais yra atliekami statistiniai skaičiavimai.

#### 3.1. Tinklus generuojantis modulis

Šiame darbe reikia ištirti kuo įvairesnius tinklus, iš kurių parametrų būtų paprasta sukonstruoti regresinius modelius. Tad tinklus generuojantis modulis turi generuoti tinklus  $N_i = \{V_{N_i}, E_{N_i}, uN_i\}$ , kurių parametrai tenkintų šias sąlygas:

- Viršūnių aibių  $V_{N_i}$  dydžių aibės  $SV$  augimo greitis būtų linijinis ir  $SV$  turi būti baigtinė.
- Kiekvieną kartą generuojant tinklą  $N_i$  yra tikimybė sugeneruoti jungų tinklą.
- Sugeneruotų tinklų aibėje egzistuoja tinklai su skirtingais dydžiais ir jungiu tinklu su yra vidutiniu galimų briaunų skaičiumi. Vidutinis galimų briaunų skaičius yra apskaičiuojamas  $SE_a = \frac{SE_{max} + SE_{min}}{2}$ , kur  $SE_{max}$  yra maksimalus galimų briaunų skaičius, o  $SE_{min}$  - minimalus galimų briaunų skaičius.
- Sugeneruotų tinklų aibėje egzistuoja tinklai su skirtingais dydžiais ir vidutiniškai silpnai jungiu tinklu. Tinklas yra laikomas vidutiniškai silpnai jungiu, jei tenkinama sąlyga:  $SE_{a_{min}} = \frac{SE_a + SE_{min}}{2}$ , kur  $SE_a$  yra vidutinis galimų briaunų skaičius, o  $SE_{min}$  - minimalus galimų briaunų skaičius.
- Sugeneruotų tinklų aibėje egzistuoja tinklai su skirtingais dydžiais ir vidutiniškai stipriai jungiu tinklu. Tinklas yra laikomas vidutiniškai stipriai jungiu, jei tenkinama sąlyga:  $SE_{a_{max}} = \frac{SE_a + SE_{max}}{2}$ , kur  $SE_a$  yra vidutinis galimų briaunų skaičius, o  $SE_{max}$  - maksimalus galimų briaunų skaičius.

Žinant, kad minimalus galimų briaunų skaičius tinkle yra  $SE_{min}(SV_i) = SV_i - 1$ , o maksimalus galimų briaunų skaičius tinkle yra  $SE_{max}(SV_i) = SV_i \times (SV_i - 1)$ , kur  $SV_i$  yra tinko viršūnių skaičius, tai gauname, kad  $SE_a(SV_i) = \frac{SV_i^2 - 1}{2}$ ,  $SE_{a_{min}}(SV_i) = \frac{SV_i^2 + 2 \times SV_i - 3}{4}$  ir  $SE_{a_{max}}(SV_i) = \frac{3 \times SV_i^2 - 2 \times SV_i - 1}{4}$ .

Tad tinklus generuojantis modulis sugeneruoja tinklą  $N_i = \{V_{N_i}, E_{N_i}, uN_i\}$  aibę A. Aibėje A yra 10 poaibių  $A'_j$ , kurių tinklų viršūnių aibių  $V_{N_i}$  dydžiai  $SV_i$  yra lygūs. Tad kiekvienas poaibis  $A'_j$  turi skirtingą dydį  $SV_j$ . Modulis sugeneruoja poaibius  $A'_j$  su šiais  $SV_j = 10 \times j : j = 1..10$ . Kiekviename poaibyje  $A'_j$  yra po 3 poaibius  $A''_j$ , kuriuose yra po 10 tinklų  $N_i$ , kurių briaunų aibių  $E_{N_i}$  dydžiai yra lygūs. Tad kiekvienas poaibis  $A''_j$  turi skirtingą dydį  $SE_j$ . Modulis sugeneruoja poaibius  $A''_j$  su šiais  $SE_j : SE_a(SV_i), SE_{a_{min}}(SV_i), SE_{a_{max}}(SV_i)$ .

### 3.2. Bandymus atliekantis modulis

Šiame darbe buvo sukurtas modulis, kuris atlieka bandymus su kiekvienu tinklu, kurį sugeneravo tinklus generuojantis modulis. Šių bandymų metu yra skaičiuojamas briaunų panaudotų skaičiavimuose skaičius. Šio bandymo rezultatai yra masyvai *INCORRECT*, *ACTION*,  $Algorithm_{\{T\}}$  ir  $Test_{\{T\}}$ , kur *T* yra kiekvieno dinaminio tinklo operacijos tipas. Šio bandymo eiga su tinklu *NETWORK*:

1. Apskaičiuojamas tinklo *NETWORK* maksimalus srautas *ACTUAL*, naudojant sukurto algoritmo įgyvendinimą.
2. Apskaičiuojamas tinklo *NETWORK* maksimalus srautas *EXPECTED*, naudojant modifikuotą Fordo Fulkersono algoritimą.
3. Jei  $ACTUAL \neq EXPECTED$ , tai į masyvą *INCORRECT* įdedama grupė *G*, o į masyvą *ACTION* reikšmė *INIT*.
4. Su kiekvienu dinaminio tinklo operacijos tipu *TYPE* yra atliekami šie veiksmai:
  - (a) Tinklui *NETWORK* atliekama operacija *TYPE*.
  - (b) Apskaičiuojamas tinklo *NETWORK* maksimalus srautas *ACTUAL*, naudojant sukurto algoritmo įgyvendinimą. Į  $Algorithm_{TYPE}$  įdedamas skaičiavime panaudotų briaunų skaičius.
  - (c) Apskaičiuojamas tinklo *NETWORK* maksimalus srautas *EXPECTED*, naudojant modifikuotą Fordo Fulkersono algoritimą. Į  $Test_{TYPE}$  įdedamas skaičiavime panaudotų briaunų skaičius.
  - (d) Jei  $ACTUAL \neq EXPECTED$ , tai į masyvą *INCORRECT* įdedamas tinklas *NETWORK*, o į masyvą *ACTION* įdedamas *TYPE*.

### 3.3. Statistiniai skaičiavimai ir jų rezultatai

## **Išvados**

Išvadose ir pasiūlymuose, nekartojant atskirų dalių apibendrinimų, suformuluojamos svarbiausios darbo išvados, rekomendacijos bei pasiūlymai.

## **Conclusions**

Šiame skyriuje pateikiamos išvados (reziumė) anglų kalba.

## Literatūra

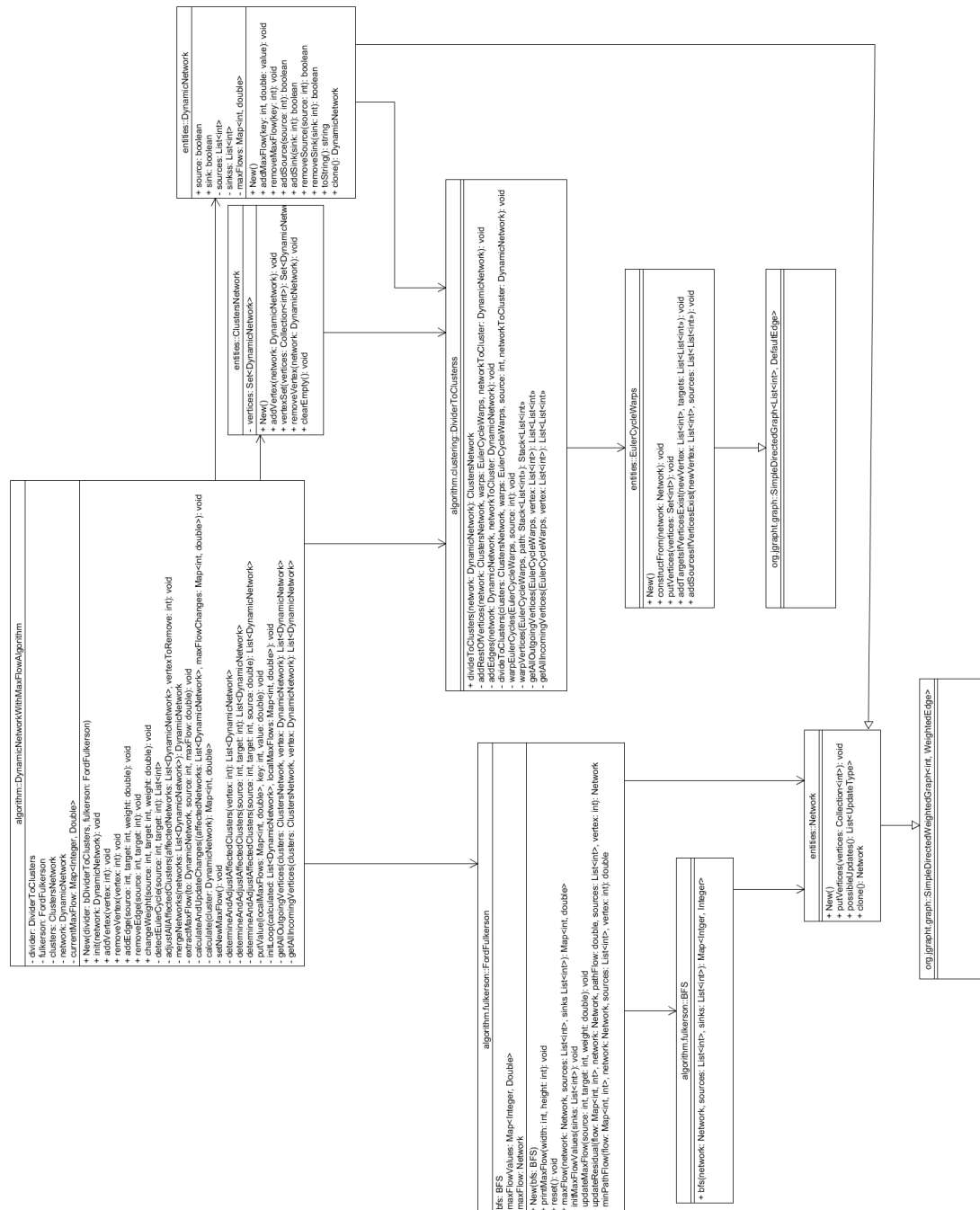
- [BAP13] S. Beamer, K. Asanovic, and D Patterson. Direction-optimizing breadth-first search. , 2013. pages 137–148, 1685 KB accessed 2018-05-15.
- [DFI01] C. Demetrescu, I Finocchi, and G. F. Italiano. Dynamic graphs. <http://www.diku.dk/PATH05/CRC-book1.pdf>, 2001. pages 1-2, 215 KB, accessed 2018-05-13.
- [EGI98] D. Eppstein, Z. Galil, and G. F. Italiano. Dynamic graph algorithms. <https://pdfs.semanticscholar.org/d381/f9a7234fcfb57c2f615e5c99cc7362ab60c9.pdf>, 1998. pages 13-18, 324 KB, accessed 2018-05-15.
- [Fre85] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees. <https://epubs.siam.org/doi/abs/10.1137/0214055>, 1985. accessed 2018-05-14.
- [JF62] L. R. Ford Jr. and D. R. Fulkerson. Flows in networks. <https://books.google.lt/books?hl=lt&lr=&id=fw7WCgAAQBAJ&oi=fnd&pg=PP1&dq=ford+fulkerson>, 1962. pages 1 - 16, accessed 2018-05-15.



# Priedas Nr. 1

## Igyvendinto algoritmo architektūra

2 pav. Sukurto algoritmo įgyvendinimo UML klasių diagrama



**Priedas Nr. 2****Eksperimentinio palyginimo rezultatai**

1 lentelė. Lentelės pavyzdys

Algoritmas	$\bar{x}$	$\sigma^2$
Algoritmas A	1.6335	0.5584
Algoritmas B	1.7395	0.5647