KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Stanislovas Jonušas

Pasitikėjimo tinklaraščių įrašais metodo sudarymas ir tyrimas

Magistro darbas

Darbo vadovas
prof. R. Butleris

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Stanislovas Jonušas

Pasitikėjimo tinklaraščių įrašais metodo sudarymas ir tyrimas

Magistro darbas

Darbo Vadovas
prof. R. Butleris
2009-05

Konsultantas
dokt. V. Taujanskas
2009-05

Recenzentas
doc. dr. E. Karčiauskas
2009-05

Atliko
IFM-3/2 gr. studentas
Stanislovas Jonušas
2009-05

Kaunas, 2009

Turinys

Lentelių turinys

Paveikslų turinys

SUMMARY

Trust of weblog records method development and research

The main aim of this work is to create the system of weblog trust. The system will offer to user subjective evaluation about the article. To calculate the subjective evaluation there is used social network.

There are users, who evaluates articles according the system of evaluation. The same time they form network of trust. Estimators doesn‘t know each other directly. There comes main question, how to calculate trust of unfamiliar users.

In this study there will be elaborating algorithms of users intertrust and right selection. To TidalTrust algorithm, there will be offered extention, which solves the main problem of time counting.

Using calculated algorithm of users intertrust there will be offered approach of calculation, which gives us the recomendationt of article trust.

1 Įvadas

Šiomis dienomis, sparčiai augant tinklaraščių skaičiui, pateiktos informacijos kiekis internete taip pat auga. Tampa sudėtingiau atsirinkti, kurie pateikti straipsniai ir informacija juose yra naudinga, o kurią galima praleisti. Pateiktą didelį kiekį straipsnių būtų lengviau atrinkti, jeigu jie būtų kategorizuojami, įvertintinami ir kaupiami vertinimo sistemose.

Kita sparčiai auganti ir populiarėjanti sritis yra socialiniai tinklai. Žmonės buriasi į virtualias bendruomenes, dalijasi savo patirtimi, sukaupta ir įvertinta informacija. Socialiniai tinklai, kol kas egzistuojančiose tinklaraščių sistemose yra nenaudojami, tačiau taikant šią sritį, būtų galima papildomai išnaudoti socialiniaime tinkle sukauptą informaciją, tokią kaip vartotojų tarpusavio pasitkėjimas.

Darbe analizuojama pasitikėjimo sąvoka socialiniame tinkle, aptariamos pasitikėjimo savybės. Kuriama tinklaraščių sistema remsis socialiniu tinklu, dėl to iš čia išplaukia pagrindinė ploblema, kaip vartotojas turėtų pasitikėti nepažįstamu kitu vartotoju, remdamasis savo socialinio tinklo ryšiais. Šios problemos sprendimui yra detaliau išnagrinėjami galimi vartotojų tarpusavio skaičiavimo algoritmai.

Remiantis išnagrinėtais algoritmas yra parenkamas tinkamiausias metodas – vartotojų pasitikėjimo skaičiavimui galima naudoti Jenifer Ann Golbeck [7] pasiūlytą algoritmą TidalTrust. Tačiau tinklaraščių vertinimo relizacijoje šis algoritmas turi ilgo laiko skaičiavimo problemą, detaliau problema aprašoma 3.1 skyriuje. Pasiūlomas šio algoritmo praplėtimas, kuris sprendžia egzistuojančia ilgo laiko problemą. Taip pat pateikiamas galutinio skaičiavimo metodas, kuris pritaikomas kuriamai tinklaraščių įvertinimo sistemai.

2 Tinklaraščių vertinimo sistemų ir pasitikėjimo modelių analizė

Darbo sritis siejama su socialinių tinklų pasitikėjimo realiziuojančiais modeliais. Darbe bus sprendžiamas pagrindinis uždavinys, kaip vartotojas gali įgyti nepažįstamo žmogaus pasitikėjimą, remdamasis savo patikimų žmonių socialinio tinklo pasitikėjimu. Taip pat, analizės metu bus palyginamos egzistuojančios straipsnių vertinimo sistemos, nusakomos pasitikėjimo savybės bei nagrinėjami algoritmai, kurie gali realizuoti pasitikėjimą, straipsnių vertinimo sistemoje. Straipsnių įvertinimo sistemos kūrimui bus išanalizuotos papildomos sritys: OpenID autentifikacijos mechanizmas ir tam tikri kategorizavimo aspektai.

2.1 Egzistuojančios tinklaraščių vertinimo sistemos

2.1.1 Technorati.com

Technorati.com, vienas iš populiariausių tinklaraščių vertinimo sistemų [2]. Šios sistemos vertinimas remiasi dviem parametrais: autoritetu (*angl. authority*) ir eile (*angl. technorati Rank*). Autoritetas apskaičiuojamas pagal nuorodas į konkretų tinklaraštį, t.y. kiek į šį tinklaraštį yra nuorodų iš kitų tinklaraščių. Įdomus dalykas sistemoje, jog nuorodos skaičiuojamos į pačius tinklaraščius, o ne į straipsnius. Jeigu kažkas pateikė dviejų straipsnių nuorodas į vartotojo tinklarašrį, tai vis tiek skaičiuosis kaip vienas autoriteto įvertinimas. Tokios nuorodos įvertinimas autoritetu galioja 180 dienų. Kitas parametras yra eilė (techorati rank). Ji nurodo kaip toli yra įvertinimas nuo daugiausiai autoritetų turinčio tinklaraščio. Kuo mažesnė šio įvertinimo reikšmė, tuo tinklaraštis labiau yra vertinamas (1 reikšmė reiškia, jog straipsnis yra visų geriausiai vertinamas). Kuo vartotojas turi daugiau nuorodų, įvertintų kitų vartotojų, tuo jo reputacija yra aukštesnė. Panaši skaičiavimo sistema, pagal nuorodas iš kitų svetainių į konkrečią svetainę, yra paremta ir google sukurta PageRank [1].

Papildomai ši įvertinimo sistema siūlo patikusius tinklaraščius dėti į mėgstamiausių sąrašą (favorite it). Kai vartotojas prisijungia prie sistemos, pagal jo pasirinktą mėstamiausią sąrašą yra pateikiama susijusi informacija: straipsniai, atsiliepimai ir komentarai. Taip pat galima atsirinkti tinklaraščių sistemas ne tik pagal pagrindinius autoriteto ir eilės parametrus, bet ir pagal tinklaraščius, kuriuos mėgstamiausiais pažymėjo daugiausia vartotojų. Kad lengviau būtų galima rasti tinkamą informaciją, šiuos mėgstamiausius tinklaraščius, galima pažymėti specialiomis žymėmis (angl. *tags*).

Šioje sistemoje informacijos srautas yra skirstomas į kanalus: verslo, pramogų, gyvenimo būdo, politikos, sporto, technologijų ir tinklaraščių. Pagrindiniuose paslaugos puslapiuose straipsnių pateikimui pagal Technorati sistemos aprašymą yra naudojama beveik realaus veikimo sistema – Percolator. Ši sistema stebi nuorodų atsiradimą ir didelį dėmesį

kreipia į naujų straipsnių atsiradimą, kas tuos straipsnius parašė, kas į juos įdėjo nuorodas, tokių nuorodų spartumo augimo tempą ir kitus faktorius, kurie padeda pateikti palčiausiai plintančią naujieną.

2.1.2 Topix.lt

Topix.lt - lietuviška internetinė sistema, skirta naujienų, video ir straipsnių nuorodų skelbimui [3]. Šioje svetainėje paskelbti strapsniai yra vartotojų reitinguojami (balsuojama už straipsnį). Pagal balsavimus yra sudaromi tam tikri populiariausių straipsnių sąrašai. Balsuoti už patikusius straipsnius gali ir neprisiregistravęs vartotojas. Balsavimo principas – paspaudi mygtuką, atiduodamas savo balsą, t.y. jeigu už šį straipsnį buvo 9 žmonės balsavę, tai jo reitingas po mygtuko paspaudimo pakils iki 10. Sistemoje registruotam vartotojui yra pateikiama statistinė informacija apie jo pateiktus straipsnius, įvertinimus ir panašius dalykus. Visi straipsniai yra suskirstyti į pagrindines kategorijas: Lietuva, kompiuterija, pasaulis ir t..t., kas palengvina staipsnio atradimą reikiamoje kategorijoje.

2.1.3 Zynios.lt

Zynios.lt - straipsnių vertinimo sistema [4], kuri beveik analogiška anksčiau išnagirinėtai Topix.lt sistemai (2.1.2 skyrius). Straipsnio įvertinimo mechanizmas – savo balso atidavimas, t.y. yra sumuojamas balsų skaičius ir kuo straipsnis daugiau surinko balsų, tuo jis turi aukštesnį reitingą. Taip pat pateikiamos specialios kategorijos, kuriuose straipsniai yra surašyti.

2.1.4 Cut.lt

Cut.lt - straipsnių vertinimo sistema [5], kuri beveik analogiška anksčiau išnagirinėtai Topix.lt sistemai (2.1.2 skyrius). Vertinimo mechanizmas balso atidavimas, t.y. balsų skaičius sumuojamas. Straipsniai skirstomi į kategorijas, kuriuose yra naudojamos žymės.

2.1.5 Authorati.com

Tinklaraščių vertinimo sistema [6] Authorati.com, joje galima straipsnius vertinti dviem aspketais: straipsnio turinio kokybe ir autoriaus nustatytais žymių (*angl. tags*) įverčiais. Turinys yra vertinamas skalėje nuo 1 iki 10, o pasrinktos žymės nuo 1 iki 5. Vartotojas, vertindamas straipsnį, negali pasirinkti papildomų žymių, nei straipsnio autorius yra nurodęs, t.y. straipsnių autorius, registruodamasis šioje sistemoje nurodo, kokiose žymėse kiti vartotojai turės vertinti jo straipsnius.

Paslaugos svetainėje galima matyti tokią informaciją apie straipsnių autorių: turinio vidutinį įvertinimą, žymių vidutinį įvertinimą, visų autoriaus pateiktų žymių vidutinius įvertinimus, laiko juostą, kurioje sužymėti įkelti straipsniai. Laiko juostoja, galima filtruoti straipsnius pagal tam tikrą turinio įvertinimą, t.y. vartotojas pasirenka įvertinimo balą, ir laiko juostoje yra parodomi visi straipsniai, kurie atitinka šį balą.

Pateiktus straipsnius gali vertinti ir neprisiregistravę vartotojai, dėl to sistema yra neapsaugota nuo sukčiavimų. Pabandžius vertinti iš keleto skirtingų interneto naršyklių, sistema priėmė visus įvertinimus. Taip pat žymės neturi klasifikavimo sistemos, dėl to negalima palyginti straipsnių, kurie turėtų aukštus įvertinimus tose pačiose srityse.

2.2 Pasitikėjimo sąvoka ir savybės

Pasitikėjimas naudojamas daugelyje disciplinų, tokių kaip socioliogija, psichologija, ekonomika, politikos mokslai, istorija, filosofija ir informatika. Kiekvienoje iš minėtų disciplinų šią sąvoką bando apibrėžti tam tikru konceptualiu lygiu. Pasitikėjimą apibrėžti yra problematiška todėl, kad yra daug pasitikėjimo tipų, taip pat ši sąvoka yra kiekvienam žmogui skirtingai suprantama ypač skirtiguose kontekstuose.

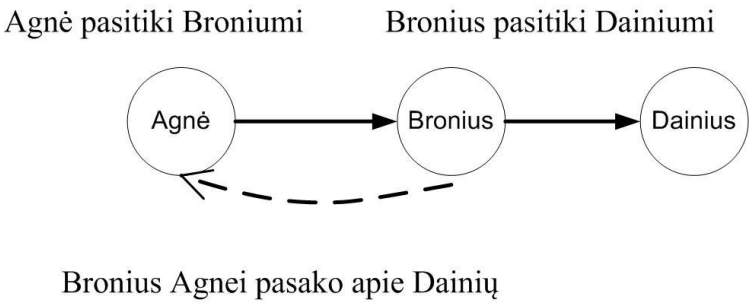
Kadagi nagrinėjama magistrinio darbo probleminė sritis yra susijusi su informatika, tai galime remtis Jennifer Ann Golbeck desertacijoje [7] pateiktu pasitikėjimo apibrėžimu: „Socialiniame tinkle vartotojas priskiria pasitikėjimą kaip vieną reikšmę, kuri nusako ryšį su kitais asmenimis, nenaudojant konteksto ar istorijos“. Golbeck savo disertacijoje pabrėžia, kad pasitikėjimas socialineme tinkle neturi priklausyti nuo konteksto ir istorijos, kaip teigė Stephen Paul Marsh [8] apibrėžtamas pasitikėjimą informatikos srityje.

2.2.1 Pasitikėjimo savybės

2.2.1.1 Tranzityvumas

Socialiniame tinkle pasitikėjimas - atrodo, kad yra ne visiškai tranzitivus matematine prasme, t.y. tarkim Agnė visiškai pasitiki Bronium, o Bronius visiškai pasitiki Dainiumi, tai dar nereiškia, kad Agnė visada visiškai pasitikės Dainiumi (1 pav.). Tačiau yra suprantama, kad auksčiau paminėtame pavyzdyje, vis dėlto mes pasitikėjimą galime perduoduoti tarp asmenų. Dažniausiai, mes ko nors nežinodami, klausiame patikimų draugų ir tų draugų nuomonę priimame kaip savo. Pazydžiui, kai Agnė visiškai pasitiki Broniumi, ji Broniaus paklausia nuomonės apie konkretų staipsnį. Ir jeigu Bronius nieko negali pasakyti, tai jis klausia savo draugo Dainiaus, kuriuo visiškai pasitiki. Taigi Bronius gaus atsaskymą iš Dainiaus ir priims tai kaip savo nuomonę ir šį atsakymą pateiks Agnei, kuri taip pat priims kaip savo nuomonę. Taigi galiausiai gaunasi, kad Agnė priims Dainiaus nuomonę. Tokio

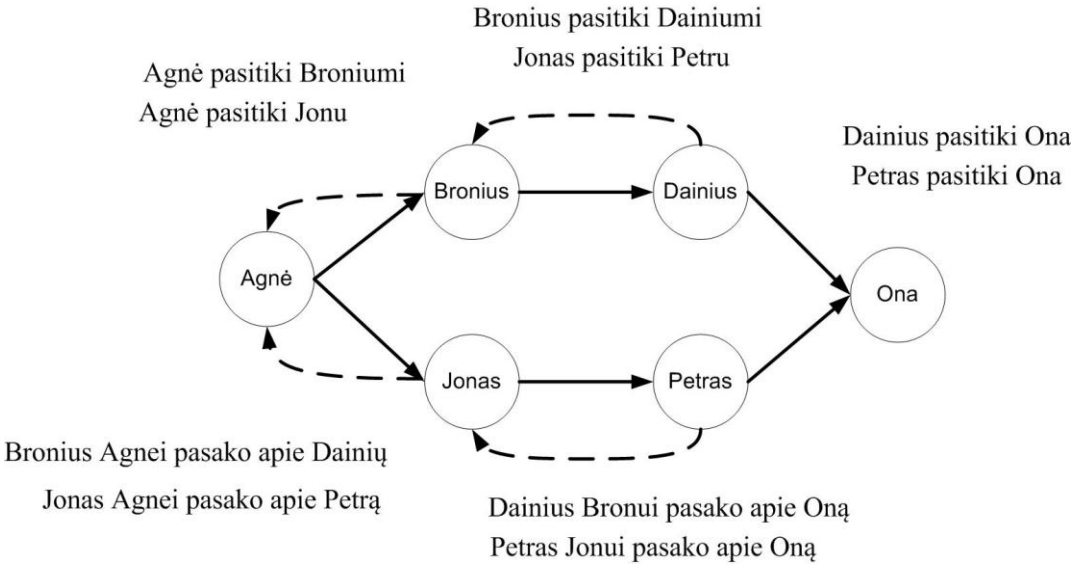
argumento šaltinio pagrindu galima sakyti, kad pasitikėjimo tranzitivumas yra tada, kai yra atgalinis informacijos perdavimas. Galima apibrėžti, jog Bronius yra tranzityviai patikimas Agnės socialiniame tinkle, jeigu informacija gali būti perduota iš Broniaus Agnei atgaliniu pasitikėjimo ryšiu tinkle. Taigi, visa pasitikėjimo informacija gali būti perduota tam tikra žmonių grandine – tai jeigu mums reikia sužinoti apie nežinomą asmenį ką nors, tai mes galime jį susirasti per tranzityvinius pasitikėjimo kelius.



1 pav. Tranzityvumo pavyzdys

2.2.1.2 Komponavimas

Tranzityvumas apibūdina, kaip pasitikėjimo įvertis gali būti perduotas per žmonių grandinę. Tačiau, kaip reiktų elgtis, kai rekomendacijas teikia daugiau nei vienas asmuo (2 pav.). Tokioje situacijoje Agnė turėtų surinkti informaciją ir nuspręsti tikėti ar netikėti Ona. Ši sudarymo savybė yra labai svarbus atributas skaičiuojant pasitikėjimo įvertinimus.



2 pav. Komponavimo pavyzdys

Šiame paveiksle matosi tokia pasitikėjimo grandinė, kuriame Agnei reikia sužinoti, kaip pasitikėti Ona. Agnė pasitiki tiesiogai Broniumi ir Jonu, Jonas pasitiki Petru, o Bronius - Daniumi, Dainius ir Petras pasitiki Ona. Taigi pasitikėjimo struktūra tampa sudėtingesnė, nes Agnė gauna informaciją iš dviejų asmenų Broniaus ir Jono, jei reikia sudėti

ir nuspręsti apie Oną. Kuo daugiau informacijos gauname iš kitų asmenų, tuo yra daugiau argumentų nuspręsti apie pasitikėjimą.

2.2.1.3 Suasmeninimas

Pasitikėjimas iš prigimties gali būti asmens subjektyvi nuomonė. Du asmenys gali dažnai turėti skirtingą nuomonę ir pasitikėjimą trečiu asmeniu. Paprastu pavyzdžiu gali būti politinis klausimas: „ar pasitikima dabartiniu prezidentu ir jo gebėjimu valdyti šalį?“. Gyventojai greičiausiai pasidalins į dvi grupes, kurie pasitiki ir kurie ne. Taigi, pasitikėjimo sąvoka įtraukia tikėjimą patikimu asmeniu, priklausomai nuo to, kokio mes iš jo rezultato laukiame. Koks rezultatas yra geras, čia kiekvienas asmuo turi skirtingą vertinimą. Kadangi kiekvienas iš mūsų turime savo interesus, prioritetus ir savo nuomonę, kuri gali nesutapti su kitų, tai kitų asmenų pasitikėjimas, galil būti taip pat skirtingas. Dėl to, pasitikėjimo skaičiavimai turėtų būti aliekami iš individualios perspektyvos, kad atitiktų konkretaus asmens interesus.

2.2.1.4 Asimetriškumas

Kaip du draugai gali turėti skirtingą nuomonę tam tikrose srityse, taip ir jie gali skirtingai pasitikėti vienas kitu, pavyzdžui Agnė gali visiškai pasitikėti Broniumi, o Bronius Agne - nepasitikėti. Tokiu atveju, jeigu socialinį tinklą atvaizduotume grafu, matytume pasitikėjimo lygių asimetriškumą.

2.2.2 Pasitikėjimo įverčių reikšmės

Prieš tai, 2.2 skyriuje pasitikėjimą apibrėžėme kaip tam tikrą reikšmę, kuri nusako, kaip mes pasitikime tinkle kitu asmeniu. Kiekviena vertinimo sistema turi savo vertinimo intervalus ir reikšmes. Nagrinėtos lietuviškos straipsnių vertinimo [3],[4],[5] sistemos naudojo balų sistemą, kur įvertinimas prasideda 0 (nuliu) ir nėra nustatytos viršutinės ribos. Nagrinėtoje [2] sistemoje vertinimo skalė irgi yra neapibrėžta, t.y. abiejų matavimo vienentų reikšmės teoriškai gali būti nuo 1 iki begalybės.

Mohsen Lesani [9] savo straipsnyje siūlo įvertinimams naudoti ne skaitines reikšmes, o žmonėms labiau suprantamas lingvistines reikšmes, tokias kaip: mažas, mažas-vidutinis, vidutinis, vidutinis-aukštas, aukštas patikimumas. Tokius pasitikėjimo ryšius vartotojui lengviau priskirti draugams socialiniame tinkle.

2.3 Pasitikėjimo algoritmų skirstymas

Pasitikėjimo algoritmus galima skirstyti į dvi kategorijas: lokalaus ir globalaus pasitikėjimo.

Globalaus tipo algoritmams galima priskirti objektyvaus skaičiavimo požiūrį. Šie algoritmai skaičiuoja universalų vartotojo pasitikėjimą tinkle. Tokiu atveju, pasitikėjimo įvertis bus vienodas nepriklausomai nuo asmens, kuris klausia. Iš skačiavimo pusės, tokie algoritmai reikalauja mažiau skaičiavimo resursų, nes nereikia individualiai skaičiuoti pasitikėjimo priklausomai nuo to, kas klausia. Vartotojas, nepriklausomai ar plėtoja ar ne savo socialinį tinklą, gali matyti tam tikras rekomendacijas pagal globalių vartotojų įvertinimus. Tokių algoritmų trūkumas tas, kad tarkim jeigu paklaustume tokio politinio klausimo: „Ar pasitikima, kaip dabartinis prezidentas valdo šalį?“ – tai vartotojai greičiausiai pasidalintų į dvi stovyklas. Toks pasidalinimas greičiausiasi duotų rezultatą per vidurį, t.y. jeigu vertintume dešimtbalėje sistemoje, kur 10 pasitiki, o 1 – nepasitiki rezultatas būtų apie 5.

Lokalaus tipo algoritmai atitinka subjektyvų požiūrį, t.y. kiekvienas pasitikėjimo įvertinimas yra pritaikomas konkrečiam asmeniui. Taigi, pačius algorimus įtakoja vartotojo pasirinkti pasitikėjimo faktoriai (socialiniame tinkle pasitikėjimo ryšiais nusakyti asmenys). Tokie algoritmai turi tendenciją žymiai geriau atspindėti vartotojui skirtas rekomendacijas. Tačiau šių algoritmų pagrindinis trūkumas – starto taškas ir didelių resursų reikalavimas. Tarkim vartotojas prisijungia prie socialinio tinklo ir jis nepažįsta nė vieno vartotojo arba negali išreikšti nuomonės apie kitą vartotoją. Tada lokalus algoritmas negalės pritaikyti jokių skaičiavimų, nes informacija nebus susijusi su nauju vartotoju. Jeigu tai būtų socialinis tinklas, tai mes negalėtume parinkti kitų vartotojų, pagal prisijungusio vartotojo „skonį“. Šie algoritmai pasidaro labiau veiksmengesni, kai surenkame didesnę informaciją, t.y. socailinime tinkle vartotojas turi daugiau pažįstamų ar jį daugiau kiti vartotojai pažįsta.

2.4 Vartotojų tarpusavio pasitikėjimo apskaičiavimo algoritmai

Internetinį socialinį tinklą galima nupiešti kaip orentuotą grafą, dėl to nagrinėjimi algoritmai remiasi grafų principais. Grafe viršūnės atitika vartotojus, o briaunos su svoriais – vartotojo pasitikėjimo lygį. Galimi keturi pagrindiniai principai pasitikėjimo skaičiavimui: trumpiausias ir stipriausias kelias; tumpiausias ir visi įmanomi keliai; visi įmanomi keliai ir visų keliu pasitikėjimas; visų kelių tik stipriausias pasitikėjimas. Metodai, kurie ieško trumpiausių kelių (šiuo atveju: trumpiausias ir stipriausias kelias; tumpiausi ir visi įmanomi keliai) niekada negrįžta į prieš tai nagrinėtą viršūnę paieškos metu, nes trumpesnis kelias būna rastas anksčiau – atitinka grafo paieškos algoritmus. Kiti du algoritmai paieškos metu naudoja prieš tai buvusias viršūnes, dėl to veikimas panašus į medžio paieškos algoritmus. Toliau detaliau bus išnagrinėtos „trumpiausio ir stipriausio kelio“ (TidalTrust) ir „visų kelių tik stipriausias pasitikėjimas“ algoritmai.

2.4.1 Trumpiausias ir stipriausias kelias (TidalTrust)

Jennifer Ann Golbeck pasiūlytas algoritmas TidalTrust [7]. Šis algoritmas remiasi principu – trumpiausias ir stipriausias kelias turi būti labiausias patikimas. Pagal algoritmą, pirmiausiai yra suieškomas trumpiausias kelias grafe. Ši paieška panaši į grafo teorijoje esantį paieškos į plotį algoritmą. Kai pavyksta rasti trumpiausią kelią, kiti tolimesni keliai skaičiavimui yra atmetami – ignoruojami. Taip pat, tarp visų įmanomų trumpiausių kelių yra atrenkami tik stipriausius ryšius turintys keliai. Stiprumo ryšio savybė grafo kelyje yra apibrėžiama, kaip minimalus pasitikėjimo įvertis kelyje, neįtraukiant paskutinių skaičiavimų tame kelyje. Tokio neįtraukimo priežastis yra ta, kad tai nors ir yra pasitikėjimo reikšmės, bet tai yra reikšmės įgytos iš pačio kelio. Visos prieštai apskaičiuotos pasitikėjimo reikšmės einant per grafo viršūnes ir prisideda prie pasitikėjimo informacijos. Kelio stiprumo savybei, kur kelio ilgis lygus 0, yra priskiriama maksimali reikšmė. Stiprumo savybė, nuo šaltinio iki tikslo, yra apibrėžiama kaip maksimalus kelio stiprumas, pereinant visus kelių variantus tarp šių taškų (šaltinio ir tikslo) (3 pav.).

Pasitikėjimas nuo tarpinių viršūnių iki tikslo yra skaičiuojamas atgaliniu būdu, t.y. nuo tikslo einama per tarpines viršūnes iki šaltinio. Skaičiuojant pasitikėjimą, nuo tarpinių viršūnių iki tikslo yra naudojamos tik tos pasitikėjimo reikšmės, kurios yra lygios arba didesnės apibrėžtam pasitikėjimo stiprumui. Tokias kaimynų pasitikėjimo naudojamas reikšmes galime pavadinti dalyviais (*angl.* participating set).

Toliau algoritmas yra pateiktas pseudo kodu.

```
1  for each n in G
2    color(n) = white
3    q = empty
4  TidalTrust (source, sink)
5    push (q, source)
6    depth=1
7    maxdepth = infinity
8    while q not empty and depth ≤ maxdepth
9         n = pop(q)
10        push (d(depth), n)
11        if sink in adj(source)
12             cached_rating(n,sink) = rating(n,sink)
13             maxdepth = depth
14             flow = min(path_flow(n), rating(n,sink))
15             path_flow(sink) = max (path_flow(sink), flow)
16             push (children(n), sink)
17        else
18             for each n2 in adj(n)
19                  if color(n2) = gray
20                       color(n2) = gray
21                       push (temp_q, n2)
22                  if n2 in temp_q
23                       flow = min(path_flow(n), rating(n,n2))
```

```
24                    path_flow(n2) = max (path_flow(n2), flow)
25                    push (children(n), n2)
26        if q empty
27             q = temp_q
28             depth = depth +1
29             temp_q = empty

30 max = path_flow(sink)
31 depth = depth-1

32 while depth>0
33        while d(depth) not empty
34        n = pop(d(depth))
35        for each n2 in children(n)
36             if (rating(n,n2)>=max) and cached_rating(n2,sink)≥0
37                  numerator = numerator + rating(n,n2) *
cached_rating(n2,sink)
38                  denominator = denominator +rating(n,n2)
39        if denominator > 0
40             cached_rating(n,sink) = numerator / denominator
41        else
42             cached_rating(n,sink) = -1
43   depth = depth-1
44 return cached_rating(source, sink)
```

Nagrinėjant detaliau algoritmą matome, kad algoritmas prasideda nuo paieškos eilės, kurioje yra tik šaltinio viršūnė. Paieška vykdoma pakartotinai einant per gretimas viršūnes. Viršūnė, naudojama paieškoje, yra įdedama į steką, tam kad po to galėtų grįžti grafu atgaline tvarka.

Kada paieškos metu viršūnė nėra gretima viršūnei-tikslui, visi viršūnės kaimynai yra pažymimi kaip aplankyti ir įdedami į kitą paieškos lygio eilę (aišku jeigu jie prieš tai jau nebuvo aplankyti). Anksčiau peržiūrėtos viršūnės yra ingnoruojamos, nes jeigu jos jau pažymėtos, vadinasi, jos jau buvo įdėtos į kito lygio eilę. Priešingu atveju, viršūnės priklauso dabartiniam lygui arba prieš tai buvusiam ir todėl jos veda prie ilgesnių kelių.

Stiprumo savybės reikšmė, nuo šaltinio iki kiekvienos kamyninės viršūnės, kurios yra kitoje paieškos lygio eilėje, yra atnaujinama iki maksimumo nuo jos prieš tai buvusios reikšmės ir stiprumo reikšmės naujajame kelyje. Galima pastebėti, kad pagal algoritmą stiprumo reikšmė nuo šaltinio iki kito lygio eilės viršūnės gali būti keletą kartų atnaujinta – tai atsitinka, kai yra randama nauja brauna nuo dabartinės viršūnės.

Jeigu einama viršūnė yra šalia tikslo, vadinasi paieška pasiekė savo tikslą, taigi galima stabdyti paiešką, nustatant, kad viršutinė paieškos gylio reikšmė lygi dabartinei gylio reikšmei. Toks stabdymo būdas padeda apsaugoti nuo paieškos atlikimo einant į gilesnius lygius ir taip pat leidžia dar tame pačiame lygyje ieškoti kelių iki tikslo viršūnės, prireikus atnaujinti stiprumo reikšmę. Reiktų atkreiptį dėmesį, kad pasitikėjimo briaunos iš paskutinio

lygio iki tikslo viršūnes nėra įtraukaimos, skaičiuojant stiprumo reikšmę nuo šaltinio iki tikslo viršūnės, pagal prieš tai buvusį stiprumo savybės apibrėžimą.

Kada visos esamo lygio viršūnės yra pereinamos ir eilė tampa tuščia, gylio reikšmė padidinama ir dabartinio lygio eilė yra pakeičiama kito lygio eilės reikšme. Iteracija vykdoma tol, kol gylio reikšmė yra mažesnė arba lygi maksimaliai gylio reikšmei arba paieškos eilė tampa tuščia. Jeigu iteracija sustabdoma, nes eilė tusčia, vadinasi visos viršūnės buvo peržiūrėtos ir nepavyko rasti kelio nuo šaltinio iki tikslo viršūnės. Maksimalaus gylio reikšmė yra atnaujinama, kada tikslo viršūnė yra pasiekama, norėdami nustatyti ar buvo rastas kelias nuo šaltinio iki tikslo viršūnės, mes palyginame pradinę maksimalaus ilgio reikšmę su dabartine. Jeigu reikšmė nepakitusi, vadinasi kelias nerastas.

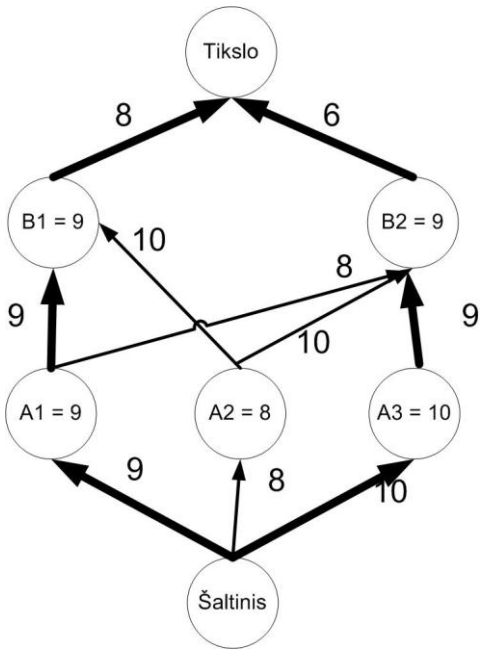
Po to, kai randamas kelias nuo šaltinio iki tikslo viršūnės, viršūnės sudėtos į steką yra išimamos atgaliniam skaičiavimui. Jeigu viršūnė yra arti tikslo viršūnės – tai apskaičiuotas tikslo pasitikėjimas yra paprasčiausias įvertinimas, tikslo viršūnės pasitikėjimo grafe. Priešingu atveju, tikslo viršūnės pasitikėjimas yra apskaičiuojamas naudojant apibrėžtas kaimynų-dalyvių viršūnes. Imama viršūnė iš steko ir nagrinėjami iš jos einantys keliai link tikslo viršūnės (viršūnės vaikai). Pasitikėjimui skaičiuoti yra naudojami tie vaikai, kurie yra stipresni arba lygūs apibrėžtam tinklo stiprumo lygui ir išsaugota pasitikėjimo reikšmė yra didesnė už nulį. Pagal išsaugotą pasitikėjimo reikšmę, mes nustatome, ar kelias veda link tikslo viršūnės, t.y. jeigu daugiau už nulį, vadinasi veda. Kai išnagrinėjami visi viršūnės vaikai, tai šiai viršūnei priskiriame pasitikėjimą – išsaugotą reikšmę. Ši reikšmė apskaičiuojama sudedant kiekvieno vaiko pasitikėjimo reikšmę, padauginant iš išsaugotos reikšmės ir visą šią sumą padalijant iš vaikų sumos pasitikėjimo.

Turint galutinę pasitikėjimo reikšmę, perėjus per visus kelius, kurie atitiko reikalaujamą stiprumą, apskaičiuota reikšme galima pasitikėti tokiu lygiu, kokiu reikalaujamas stiprumas buvo nustatytas.

Pateiktame (3 pav.) pavyzdyje, briaunos atitinka vartotojo pasitikėjimą, o reikšmės viršūnėse – apskaičiuotą kelio stiprumą (bei viršūnės pavadinimą). Kadangi paskutiniame kelio skaičiavime viršūnės iki reišmės lygios 9, vadinasi tai yra maksimalus reikalaujamas kelio stiprumas. Toliau, pagal šią stiprumo reikšmę, yra paryškinti keliai, kurie bus naudojami apskaičiuojant šaltinio-tikslo pasitikėjimo reikšmę.

Apskaičiuotą pasitikėjimą galima užrašyti tokia formule:

$$t_{is} = \frac{\sum_{j \in adj\ (i) \ni t_{ij} \geq max\ t_{ij}\, t_{js}}}{\sum_{j \in adj\ (i) \ni t_{ij} \geq max\ t_{ij}}} \quad (1)$$



3 pav. Pasitikėjimo grafas su stipriausiu keliu.

2.4.2 Visų kelių tik stipriausias pasitikėjimas

Pagrindinius principas šio algoritmo yra pasitikėjimo apskaičiavimas, kuriam yra naudojami visi įmanomi keliai. Algoritmas ieško stipriausių kelių, net jeigu jie yra ilgesni už rastus trumpesnius, t.y. algoritmas nestabdo paieškos, kai atranda trumpiausią kelią iki tikslo viršūnės pirmą kartą, o paieška tęsiama tol kol įmanoma aptikti neaplankytas viršūnes. Toks algoritmo veikimas ledžia tiksliau apskaičiuoti pasitikėjimo įvertinimą, nes yra naudojama daugiau informacijos.

Toliau pateiktas algoritmo pseudo kodas

```
inferAllLengthStrongestTrust ( Node source, Node sink )
{
if ( source = sink )
    return MAX_TRUST_VALUE
queue = new Queue()
priorityQueue = new Queue()

queue.queue( source )

strengthFromSource[ source ] = MAX_TRUST_VALUE
strengthFromSource[ sink ] = -1

while ( (not queue.isEmpty()) or (not priorityQueue.isEmpty()) )
{
    if ( priorityQueue.isEmpty() )
        node = queue.dequeue()
    else
        node = priorityQueue.dequeue()

    isVisited[ node ] = true
    if ( not trustGraph.isAdjacent( node, sink ) )
    {
```

```
        for each adjacentNode in adjacents of node
        {
            pathStrength = min(
                strengthFromSource[ node ],
                trustGraph.getTrust( node, adjacentNode ) )

            previousStrength = strengthFromSource[ adjacentNode ]

            newStrength = max(previousStrength,pathStrength )

            strengthFromSource[ adjacentNode ] = newStrength
            if ( newStrength >= strengthFromSource[ sink ] )
            {
                if ( not isVisited[ adjacentNode ] )
                    queue.queue( adjacentNode )
                else
                    if ( newStrength ≠ previousStrength )
                        priorityQueue.queue( adjacentNode )
            }
        }
    }
    else
    {
        pathStrength = strengthFromSource[ node ]
        strengthFromSource[ sink ] = max(
            strengthFromSource[ sink ], pathStrength )
    }
}

if ( maxDepth == DUMMY_MAX_VALUE )
    return DUMMY

requiredStrength = strengthFromSource[ sink ]

Visi isVisited[ i ] pakeičiami į false

adjacentsToSink = trustGraph.getAdjacentsTo( sink )
for all adjacentsToSink[ i ]
{
        trustsToSink[ adjacentsToSink[ i ] ] =
            trustGraph.getTrust( adjacentsToSink[ i ], sink )
        queue.queue( adjacentsToSink[ i ] )
}

while ( (not priorityQueue.isEmpty()) or (not queue.isEmpty()) )
{
    if ( not priorityQueue.isEmpty() )
        currentNode = priorityQueue.dequeue()
    else
        currentNode = queue.dequeue()
        isVisited[ currentNode ] = true
        adjacentsTo = trustGraph.getAdjacentsTo( currentNode )

        for all adjacentTo in adjacentsTo[ i ]
        {
            if ( adjacentTo = sink )
                continue
            if ( not isVisited[ adjacentTo ] )
```

```
            {
                updateTrustToSink( adjacentTo )
                queue.queue( adjacentTo )
            }
            else
            {
                previousTrust = trustsToSink[ adjacentTo ]
                newTrust = updateTrustToSink( adjacentTo )
                if (previousTrust ≠ newTrust)
                    priorityQueue.queue( adjacentTo )
            }
        }
}
return <trustsToSink[ source ], requiredStrength>
}
```

Pirmiausiai, einama iteraciniu būdu per visų viršūnių lygius, pradedant nuo šaltinio ir ieškant reikalingo stiprumo. Kada atliekama paieškos iteracija – tai nauji keliai atitinka šaltinio kaimynines viršūnes. Stiprumas nuo šaltinio viršūnės iki kaimynų viršūnių yra atnaujinamas tik tada, jeigu stiprumas naujame kelyje yra didesnis už prieš tai buvusį. Jeigu kaimyninė viršūnė dar nebuvo aplankyta – ji yra įdedama į eilę. Priešingu atveju – vadinasi brauna yra atgalinė, t.y. vedanti į prieš tai buvusį lygį. Pakartotinė viršūnės iteracija, jei viršūnė priklauso vienam iš prieš tai buvusių lygių, reikalauja pakartotinai atlikti paiešką keletos prieš tai buvusių lygių viršūnių. Vadinasi yra svarbu atlikti iteraciją prieš pereinant į kitą lygį, kad būtų apribotas ėjimas į gylius. Dėl to peržiūrėtose viršūnėse turėtų būti atliekami skaičiavimai anksčiau, negu esačios viršūnės eilėje. Tam tikslui yra laikoma prioritetinė eilė, kuri saugo peržiūrėtas viršūnes. Viršūnės prioritetinėje eilėje turi didesnį prioritetą negu paprastoje eilėje. Bet kokios kaimynų viršūnės, kurios stiprumas nuo šaltinio link tikslo viršūnės yra mažesnis už naujausią atrastą stiprumą, yra ignoruojamos. Pirmas etapas yra baigiamas po to, kai visos įmanomos viršūnės nuo šaltinio yra apeinamos. Taip pat stiprumo reikšmė nuo šaltinio iki tikslo viršūnės, kuri vadinama reikiamu stiprumu, irgi baigiama skaičiuoti po to, kai apeinamos visos viršūnės.

Antras etapas: viršūnių pasitikėjo reikšmės iki tikslo viršūnės yra skaičiuojamos atgaline tvarka, t.y. pradedama nuo tikslo viršūnės ir einama atgal per surastas nagrinėtas viršūnes link šaltinio. Pasitikėjimo reikšmės turi būti skaičiuojamos tik įtraukiant tas viršūnes, kurios yra lygios arba stiprenės nustatytam stiprumo lygui. Jeigu pasitaiko taip, kad gretima viršūnė nėra peržiūrėta, tai ji yra dedama į eilę vėlesniam peržiūrėjimui, panašiai, kaip buvo vykdomas pirmas skaičiavimo etapas. Pasitikėjimo reikšmė nuo viršūnių link tikslo viršūnės gali buti keletą kartų atnaujinta, kai tik gretimose viršūnėse yra atliekama paieška. Antro etapo pabaiga grąžina pastikėjimo reikšmę.

2.5 Kiti uždaviniai susiję su tinklaraščių pasitikėjimo sistemos kūrimu

2.5.1 Kategorizavimo problema

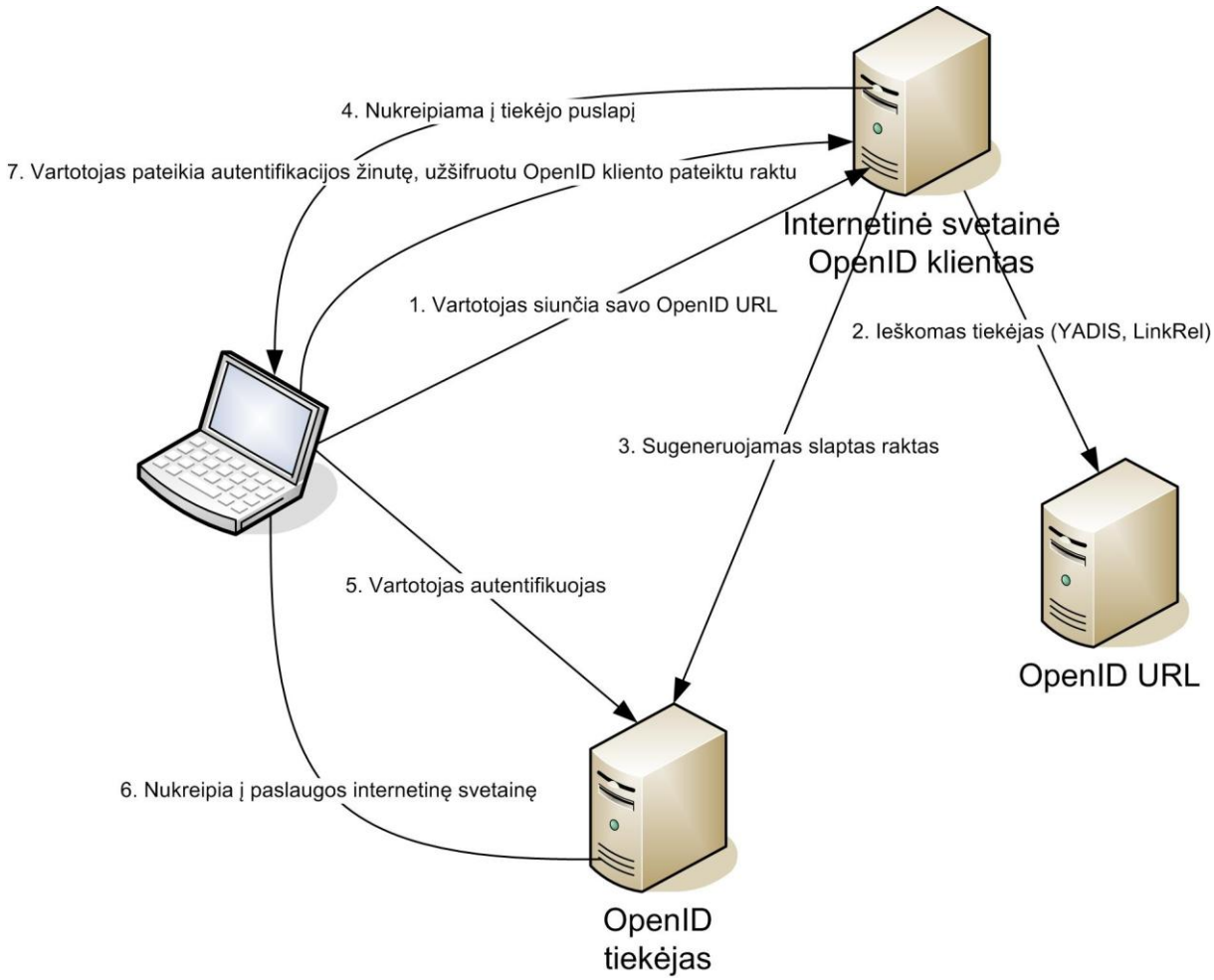
Norint, kad greičiau pasiektų vartotoją reikiama informacija, reikia šią informaciją kategorizuoti. Informacijai kategorizuoti galima naudoti žymes. Kategorizuojant informaciją, galima susidurti su tokiomis problemomis: tarkime jeigu kategorijos žymė vadintųsi Apple (*vertimas iš anglų į lietuvių k.* – obuolys), tai galėtų reikšti vaisių arba kompanijos „Apple“ prasmę ar net kokią kompiuterinę sistemą „Apple records“. Ne tik žymių prasmės gali skirtis pagal vieną pavadinimą, bet ir tos pačios reikšmės gali būti skirtingai užrašytos. Tarkime, kad straipsnis yra susijęs su progamavimo kalbos C# sintakse, šią žymę internete galima rasti užrašyta taip: C#, CSharp, C-sharp ir pan.

MOAT (*angl. Meaning Of A Tag*) yra semantinio tinklo karkasas, kuris leidžia pateikti publikuoti semantinius straipsnius, pagerintus specialiomis žymėmis. Šio projekto tikslas yra išspręsti žymių vienodumo ir suprantamumo problemas, leidžiant vartotojams žymes apibūdinti, naudojant semantinio tinklo URI (galima naudoti URI iš tokių žinių bazių kaip DBpedia[1], geonames[2], DMOZ[3] ar bet kokios tam skirtos žinių bazės). Dėka koncepcijos tokių ryšių tarp žymių ir URI, galima teksto turinio žodžius pažymėti URI žymėmis, o ne paprasto teksto žymėmis išlaikant semantinius ryšius internete, pvz.: *Šitame straipsnyje aš „apple“ reikšme naudoju kaip <http://dbpedia.com/resource/Apple_Records>, o ne kaip vaisų ar kompiuterinės įmonės ženkalą.* Be to, sukurtą šią žymę, mes galime panaudoti kitose socialiniuose tinkluose išlaikant žymės prasmę.

2.5.2 OpenID autentifikacijos mechanizmas

Vienas iš sistemos pateiktų reiklavimų (3.5.2.1 skyriuje) buvo tas, kad registracijos metu arba prisijungiant prie sistemos vartotojas galėtų naudoti OpenID[4]. OpenID pagrindinis privalumas yra tas, kad išsprendžia daug vartotojų ir slaptažodžių problemų, t.y. paprastai registruodamiesi internetinėse svetainėse, mes turime sugalvoti vartotojo vardą ir slaptažodį. Nevisada mes galime panaudoti tą patį vartotojo vardą registruojantis skirtingose sistemose ir aišku sunku atsiminti visus sugalvotus slaptažodžius ir vartotojo vardus. Jeigu dar sistemos laikosi saugumo politikos ir laikas nuo laiko reikia pakeisti slptažodį – tai dar labiau pasunkina slaptažodžių atsiminimą. OpenID pasiūlio vadinamą vieno prisijungimo principą. Žemiau pateiktas 4 pav., kuriame matosi OpenID pagrindinis veikimo principas.

[1] http://dbpedia.org
[2] http://www.geonames.org
[3] http://www.dmoz.org
[4] http://www.openid.net



4 pav. OpenID veikimo schema

Prisijungimo lange vartotojas turi įvesti savo OpenID URL, pvz test.pip.verisignlabs.com . Pagal šį įvestą URL adresą yra atrandamas autentifikacijos tiekėjas, anksčiau pateikto pavyzdžio – tai būtų pip.verisignlabs.com . Atrastam tiekėjui, iš internetinės svetainės yra siunčiamas sugeneruotas slaptas raktas. Jeigu vartotojas dar neautentifikavos pas savo OpenID tiekėjo – tai intenertinė svetainė nukreipia į tiekėjo svetainę atlikti autentifikaciją. Jeigu autentifikacija pas tiekėją yra sėkminga, tiekėjas atgal nukreipiamas į paslaugos puslapį, perduodant autentifikacijos rezultatą, užkoduotą prieš tai siųstu slaptu raktu. Jeigu vartotojas yra jau autentifikavęsis pas savo tiekėją, tai jam nebereikia atlikti tiekėjo autentifikacijos.

2.6 Galimų sprendimų analizė

Ieškant, kokie įmanomi pastikėjimo būdai bei koks metodas gali būti pritaikytas tinklaraščių vertinimo sistemos kūrimui, buvo nagrinėjama papildoma literatūra. Vartotojų pasitikėjimui skaičiuoti rasti algortimai šuose šaltinuose: [10], [11], [14], [15], [19]. Šie algortimai pagal tipą yra lokalūs (skaičiuojama subjektyvi nuomonė) ir pasitikėjimui yra naudojami vartotojų pasitikėjimo ryšiai. Taip pat buvo nagrinėtas [18] straipsnių pasitikėjimo

modelis, kuris skirtas straipsnių infomacijos kitimo pasitikėjimui, pagal savo specifiką šitas modelis labiausiai tinkamas skaičiuoti pasitikėjimą WIKI sistemoms. Kituose nagrinėtose šaltinuose yra apžvelgiamas pasitikėjimo modelis bei ontologijų taikymas semantiniame tinkle [13], [16], [17], nagrinėjamos pasirinktų metodų teigiamos ir neigiamos savybės.

2.7 Analitinės dalies išvados

Pirmiausiai buvo išnagrinėtos egzsituojančios tinklaraščių vertinimo sistemos, aprašomi vertinimo metodai jų funkcionalumas. Galima pastebėti, kad lietuviškos nagrinėtos tinklaraščių sistemos savo veikimo principu yra vienodos, skaičiuojamas balso atidavimas. Tik vienoje iš nagrinėtų tinklaraščių sistemų buvo vertinamas straipsnio turinys bei jos nurodytos sritys (žymės).

Taip pat darbe pateiktos pasitikėjimo savybės ir apibrėžta pasitikėjo sąvoka. Detaliau išnagrinėti galimi pasitikėjimą skaičiuojantys algortimai, bei nusakyti pačių algoritmų tipai.

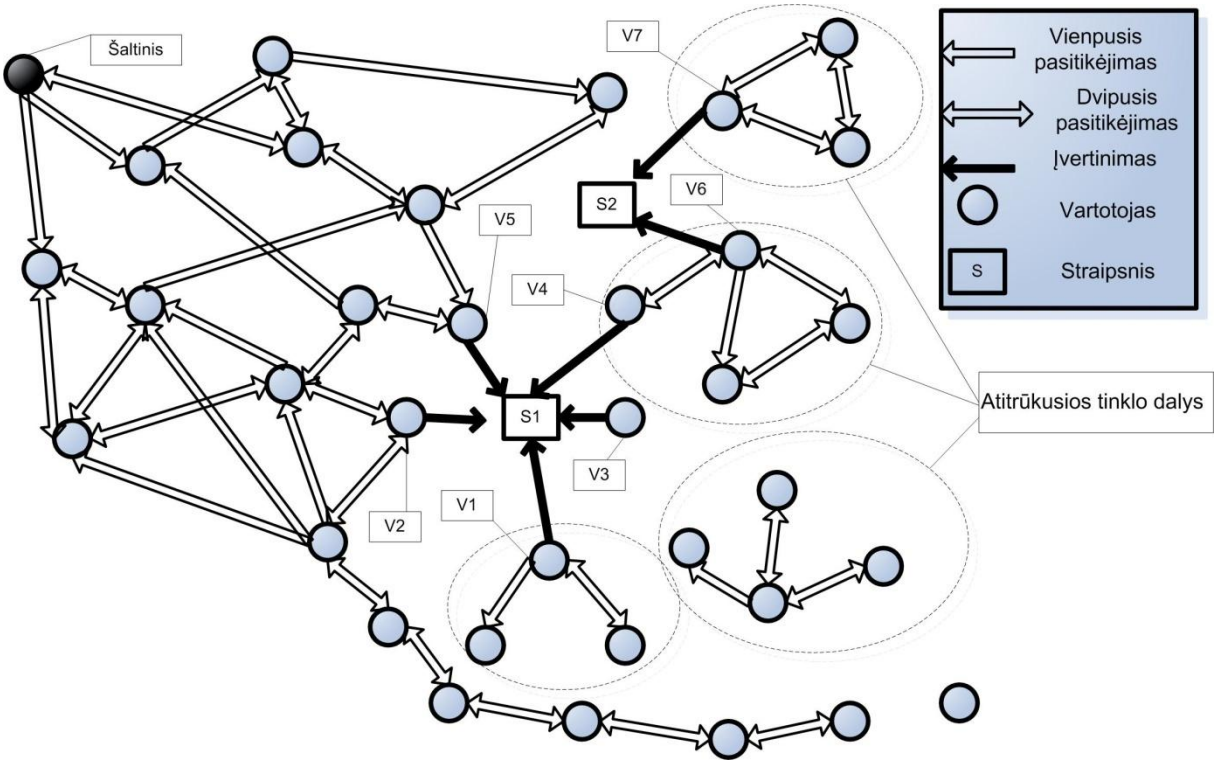
Išanalizuotos susijusios sritys, tokios kaip kategorizavimas ar OpenID naudojimas, kurios bus reikalingos tinklaraščių vertinimo sistemos įgyvendinimui.

3 Tinklaraščių įrašų pasitikėjimo medoto sudarymas ir įgyvendinimas

Keliama užduotis – sukurti tinklaraščių vertinimo sistemą, kurios vertinimai remtųsi socialiniu tinklu. Šiame skyriuje bus parodyta, su kokiomis problemomis susiduriama įgyvendinant TidalTrust algortimą, bei pasiūlytas patobulintas šio algoritmo variantas. Taip pat pateiktas tinklaraščio vertinimo sistemos įgyvendinimo projektas.

3.1 TidalTrust algoritmo taikymo problemos, tinklaraščių vertinimo sistemai

Žemiau pateiktas straipsnių vertinimo sistemos socialinis tinklas, atvaizduotas grafo pavidalu 5 pav. Višūnės atitinka vartotojus, o briaunos - pasitikėjimą kitu vartotoju. Tinkle pasitikėjimas yra asimetrinis (2.2.1.4 skyrius). Šaltinio viršūnė – tai šiai viršūnei bus skaičiuojamos reikiamos straipsnių S1 ir S2 rekomendacijos.



5 pav. Socialinis tinklas atvaizduotas grafu

Panagrinėkime situaciją, kai šiam tinklui skaičiuoti yra panaudojamas TidalTrust algortimas. Norint šaltinio viršūnei (vartotojui) suteikti straipsnių S1 ir S2 rekomendacijas, reikia atsirinkti tas viršūnes, kurios įvertino šiuos straipsnius. Pagal pateiktą pavyzdį, šiuo atveju straipsnio S1 rekomendacijas gali pateikti vartotojai: V1, V2, V3, V4, V5. O straispnio S2 – vartotojai: V6 ir V7.

Pirmuoju atveju šaltinio viršūnei bandome pateikti straipsnio S1 rekomendacijas. Taigi, pagal TidalTrust algoritmą, mums reikės surasti trumpiausius ir stipriausius kelius iki S1 vertinusių vartotojų, t.y. šiuo atveju gauname, kad 5 kartus reikės atlikti TidalTrust skaičiavimus: nuo šaltinio viršūnės iki V1, nuo šaltinio viršūnės iki V2 ir t.t . Pateiktame paveiksle matosi, kad viršūnes V1, V3, V4 yra nepasiekamos šaltinio viršūnei. Taigi šiuo atveju algoritmas tris kartus skaičiuos iki kiek įmanoma giliausio grafo viršūnės taško, neduodamas naudingo rezultato. Galutiniame rezultate gausime tik dvi viršūnes, pagal kurias galėsime apskaičiuoti straipsnio rekomenduojamą įvertinimą.

Nagrinėjant antrą atvejį, kai bus bandoma šaltinio viršūnei suteikti rekomendaciją apie straipsnį S2, mes skaičiuosime du kelius iki tikslo viršūnės (šaltinio -> V7, šaltionio -> V6). Kadangi nėra jokių kelių iki šių viršūnių, algoritmas du kartus pereis per visą įmanomą grafą iki tolimiausios viršūnės, tačiau jokio rezultato neduos.

Abiem atvejais, nagrinėjamos tik kelios nepasiekiamos tikslo viršūnės, tačiau socialiniuose tinkluose tų viršūnių skaičius gali būti gana didelis. Galima daryti išvadą, jog kuo daugiau šių viršūnių ir kuo tinklas daugiau turi trūkių, tuo daugiau resursų ir laiko reikės, kad būtų galima apskaičiuoti pasitikėjimą.

3.2 Prielaidos siūlomam pasitikėjimo medoto skaičiavimui

Svarbiausia siūlomo modelio prielaida – trumpiausias ir stipriausias kelias turi būti labiausiai patikimas. Šį teiginį Jenifer Ann Golbeck [7] savo disertacijoje įrodė, remdamasi filmų pasitikėjimo sistema, gauti jos rezultatai pateikiami žemiau esančioje 1 lentelėje.

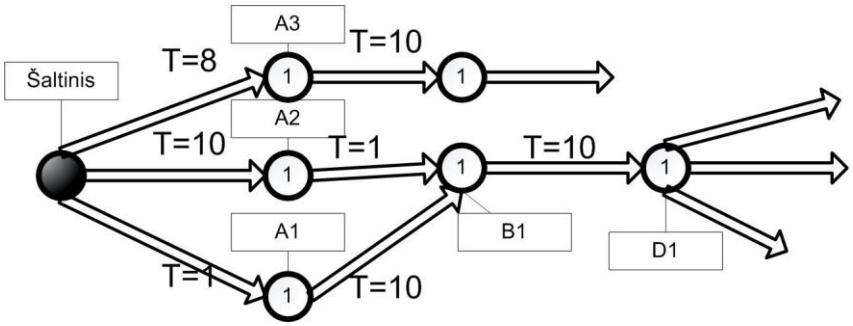
1 lentelė. Pasitikėjimo nuokrypio duomenys

	Kelio ilgis				
Pasitikėjimo reikšmė		**2**	**3**	**4**	**5**
	10	0,953	1,52	1,92	2,44
	9	1,054	1,588	1,969	2,51
	8	1,251	1,698	2,048	2,52
	7	1,5	1,958	2,287	2,79
	6	1,702	2,076	2,369	2,92

Pagal Golbeck (1 lentelė) pateiktus rezultatus galime pastebėti, jog kai kelio ilgis lygus 2 , o pasitikėjimo reikšmė 7, vidutinis nuokrypis yra 1,5 , kas beveik atitinka kelio ilgui 3 ir pasitikėjimo reikšmei 10 – reikšmė 1,52 . Vadinasi, ne tik trumpesnis kelias gali išduoti tikslesnę reikšmę, bet gali būti, kad turint stipresnį pasitikėjimą iš ilgesnio kelio, mes gausime

tikslesnį įvertinimą. Remiantis šiuo faktu, siūlomo modelio algoritmas suras trumpiausią kelią iki artimiausios viršūnės ir paiešką tęs tik tuo atveju, jeigu bus galimybė rasti stipresnį pasitikėjimą turinčią viršūnę.

Kelio stiprumo reikšmė bus lygi arba maženė, jeigu tolimenis keliai ves iš šios viršūnės.



6 pav. Srauto apribojimas

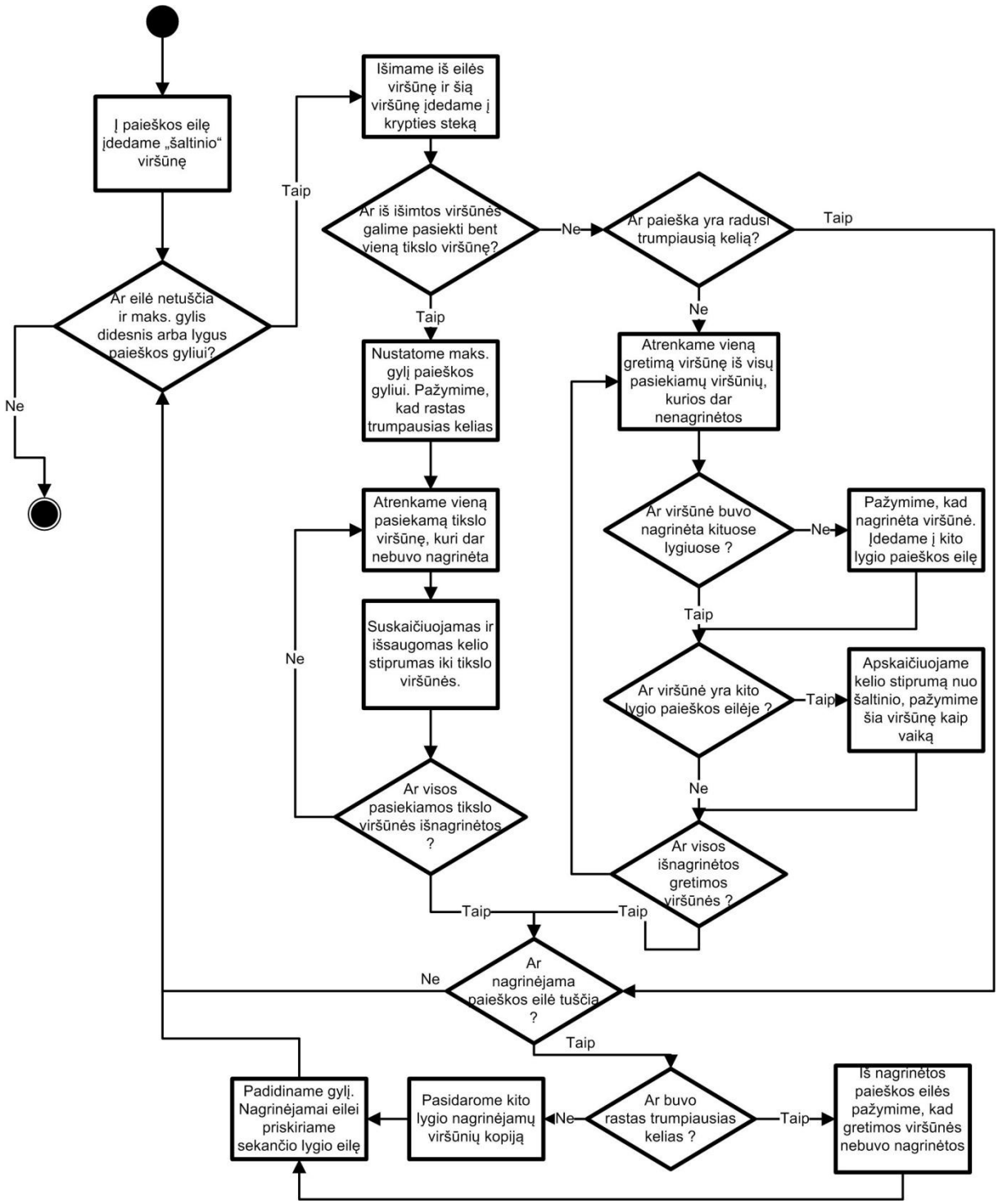
Pateiktame pavyzdyje (6 pav.) matome, kad jeigu tolimesni keliai seks tik iš B1 viršūnės, tai šie keliai, nepriklausomai nuo to, kokio stiprumo, nebus stipresni nei kelio stiprumas B1 viršūnėje, t.y. šiame paveiksle iš A3 viršūnės nei vienas kelias neveda į kelius ateinančius iš B1 viršūnės. Jeigu kelias A3 vestų į bent vieną iš B1 kelių, vadinasi kelio stiprumas einančios iš B1 viršūnės gali įgyti didesnę reikšmę. Dėl to, skaičiuojant pasitikėjimą, mes tikriname, ar įmanomi stipresni keliai, t.y. ar galima rasti stipresnį pasitikėjimą.

3.3 Pasitikėjimo algoritmas

Tinklaraščių vertinimų pasitikėjimo algoritmas – tai 2.4.1 skyriuje nagrinėto TidalTrust algoritmo praplėtimas. Pasinaudoję Golbeck atrastais pastebėjimais dėl kelio ilgio ir pasitkėjimo reikšmės, bei norint užkirsti kelią dideliems skaičiavimams, bus nagrinėjama ne viena tikslo viršūnė o iškarto visos reikalingos tikslo viršūnės. Algoritmą galima sudalinti į tris etapus: pirmajame etape, surandamas trumpiausias kelias iki vienos ar kelių tikslo viršūnių. Antrame etape, nustatę maksimalią grafo stiprumo reikšmę, atliekame pakartotinę paiešką per nagrinėtas paskutines viršūnes, ieškant tenkinančią apibrėžtą grafo stiprumo reikšmę (lygus arba stipresnis kelias). Trečiame etape skaičiuojame atgaline tvarka pereitas viršūnes, skaičiuojame pasitikėjimą nuo pereitų viršūnių, link rastų tikslo viršūnių.

3.3.1 Pirmasis etapas – paieška iki artimiausių tikslo viršūnių

Žemiau pateikta (7 pav.) pirmojo algoritmo etapo blokinė diagrama. Šiame etape pradedama paieška nuo šaltinio ir iteraciniu būdu einama per grafą į jo gylį, ieškodami trumpiausio kelio bent iki vienos tikslo viršūnės.



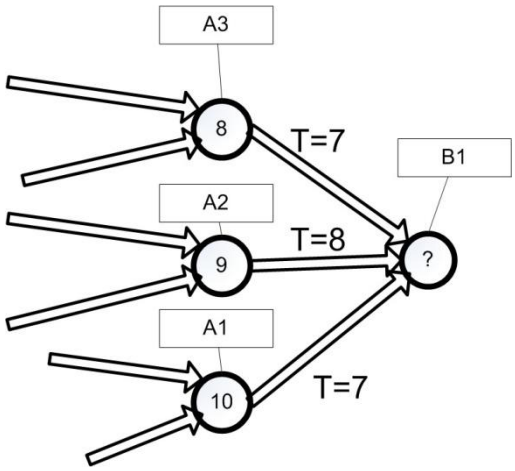
7 pav. Pirmo etapo algoritmo blokinė diagrama

Pirmiausiai, į paieškos eilę įdedama šaltinio viršūnė. Patikriname, ar ši viršūnė turi gretimas viršūnes. Jei gretimų viršūnių nėra, vadinasi paieška baigta. Šitas algoritmo etapas bus vykdomas tol, kol bus eilėje viršūnių arba nustatytas maksimalus gylis taps mažesnis už paieškos gylį.

Išimame iš paieškos eilės vieną viršūnę (pavadinkime *nagrinėjama viršūne*) ir įdedame į krypties steką (krypties stekas bus naudojamas trečiajame algoritmo etape). Tikriname ar ši nagrinėjama viršūnė turi bent vieną gretimą viršūnę, kuri būtų tikslo viršūnė.

Jeigu gretimos viršūnės nėra tikslo, tai patikriname ar algortimas jau rado trumpiausią kelią. Jeigu algorimtas dar neturi trumpiausio kelio – tai nagrinėjame kiekvieną nagrinėjamos viršūnės gretimą viršūnę: pirmiausiai patikriname ar ši gretima viršūnė nebuvo dar anksčiau nagrinėta. Jeigu nenagrinėta – pažymima kaip nagrinėta ir įdedame į sekančio lygio formuojamą eilę; antra, patikriname ar gretima viršūnė priklauso sekančio lygio eilei, jeigu priklauso – skaičiuojame kelio stiprumą nuo šaltinio iki gretimos viršūnės. Ciklas kartojamas tol, kol išnagrinėjamos visos naginėjamos viršūnės gretimos viršūnės.

Kelio stiprumas apskaičiuojamas tokiu principu: yra imama mažesnė reikšmė, lyginant kelio stiprumo reikšmę iš ateinančios viršūnės su pasitikėjimo įvertinimu skaičiuojamai viršūnei. Ir iš visų šių minimumų, atrenkama maksimali reikšmė. Žemiau (8 pav.) pateiktas skaičiavimo pavyzdys. Viršūnėse esančios reikšmės – kelio stiprumas, o ant briaunų – pasitikėjimas.



8 pav. Kelio stiprumo skaičiavimo grafo pavyzdys.

$$min_{A1,B1} = \min(stiprumasNuoŠaltinio[A1], pasitikėjimas(A1,B1)) = \min(10,7) = 7$$
$$min_{A2,B1} = \min(stiprumasNuoŠaltinio[A2], pasitikėjimas(A2,B1)) = \min(9,8) = 8$$
$$min_{A3,B1} = \min(stiprumasNuoŠaltinio[A3], pasitikėjimas(A3,B1)) = \min(8,7) = 7$$
$$stiprumasNuoŠaltinio[B1] = \max(min_{A1,B1}, min_{A2,B1}, min_{A3,B1}) = (7,8,7) = 8$$

Jeigu nagrinėjamosios viršūnės randame bent vieną gretimą viršūnę, vadinasi trumpiausias kelias atrastas. Nustatome maksimalaus paieškos gylio reikšmę į dabartinės paieškos gylio reikšmę. Iš nagrinėjamos viršūnės gretimų viršūnių, atrenkame tikslo viršūnes. Kiekvienai iš šių tikslo viršūnių apskaičiuojame tinklo stiprumą. Tikslo viršūnėms skaičiuojant tinklo stiprumą, nėra įtraukiama pasitikėjimo reikšmė – tiesiog paimama maksimali tinklo stiprumo reikšmė, iš kurios į tikslo viršūnę galime patekti.

Kai nagrinėjamas paieškos lygis bus tuščias, patikrinsime ar buvo rasta bent viena tikslo reikšmė. Jeigu nerasta, tada dabartinei paieškos eilei priskirsimę kitos eilės sąrašą, pasidarysime kitos eilės kopiją ir padidinsime paieškos gylio reikšmę. Priešingu atveju, vadinasi yra rastas trumpiausias kelias. Antrajame algoritmo etape, dar kartą pereisime per paskutinę nagrinėtą viršūnių eilę (iš eilės kopijos pasiėmę), todėl reikia pažymėti kad gretimos viršūnės iš paskutinės nagrinėtos eilės buvo dar neaplankytos (nenagrinėtos).

Po pirmo algoritmo etapo turėtume turėti rastą bent vieną tikslo viršūnę, jeigu ji nebuvo rasta – vadinasi visas algortimas stabdomas, apskaičiuoti neįmanoma. Rastos tikslo viršūnės saugos maksimalų tinklo stiprumą (šis stiprumas bus reikalingas trečiajame algoritmo etape). Taip pat žinosime, kokiame paieškos gylyje buvo rastas trumpiausias kelias.

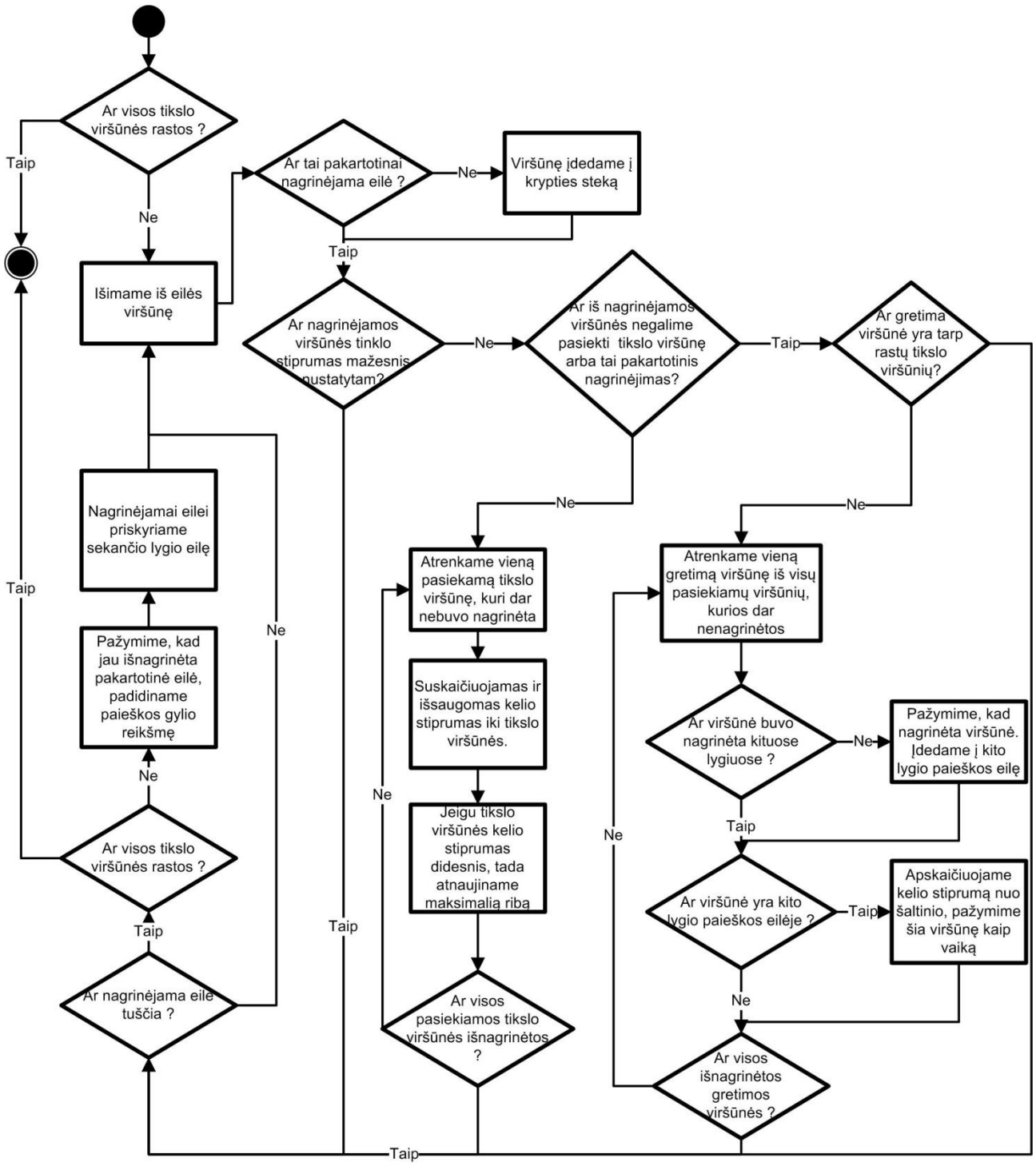
3.3.2 Antras etapas – tolimesnė tinklo paieška

Antras etapas prasideda nuo patikrinimo ar yra nerastų tikslo viršūnių. Jeigu visos viršūnės buvo rastos – algoritmas pereina į trečią etapą. Priešingu atveju, skaičiuojama tol, kol randamos visos tikslo viršūnės arba tolimesni keliai pasidaro mažesni už maksimalią nustatytą tinklo stiprumo reikšmę. Maksimalią tinklo reikšmę nustatome taip - iš visų rastų tikslo viršūnių, paimame stipriausią kelio reikšmę. Ši maksimali tinklo reikšmė bus atnaujina po kiekvienos naujos tikslo viršūnės atradimo, jeigu to tikslo viršūnės kelio reikšmė bus didesnė, negu buvo prieštai buvusi tinklo reikšmė. Algoritmas etapas detalizuotas 9 pav.

Pirmiausiai išimame iš nagrinėjamos eilės vieną viršūnę (ją vadinsime *nagrinėjamaja*). Patikriname ar nagrinėjamoji viršūnė yra iš pirmojo etapo nagrinėtos paskutinės eilės, jeigu taip – vadinasi mes jau krypties steke turime pasižymėję šią viršūnę, dėl to į krypties steką šios viršūnės nededame. Patikriname, ar šios viršūnės kelio stiprumas yra didesnis už nustatyto tinklo stirpumą, jeigu mažesnis, nagrinėjame kitą eilės viršūnę.

Patikriname, ar nagrinėjamosios viršūnės gretimos viršūnės nėra ieškomos tikslo viršūnės. Nagrinėjant pakartotinai pirmo etapo paskutinę eilę iš nagrinėjamosios viršūnės, mes nerasime tikslo viršūnės, nes pirmajame etape jos jau buvo įtrauktos į rastų tikslo viršūnių sąrašą. Tačiau galima situacija, kad einant per ilgesnį grafo tinklo kelią, mes rasime nagrinėjamajai viršūnei gretimą viršūnę, kuri jau yra rasta trumpesniame kelyje. Vadovaudamiesi tuo, kad trumpesnis kelias turi didesnį prioritetą, mes šios viršūnės nenagrinėsim.

Toliau nagrinėjamosios viršūnės gretimos viršūnės pereinamos tokiu pačiu principu kaip ir pirmajame etape, t.y. skaičiuojamas kelio stiprumas.



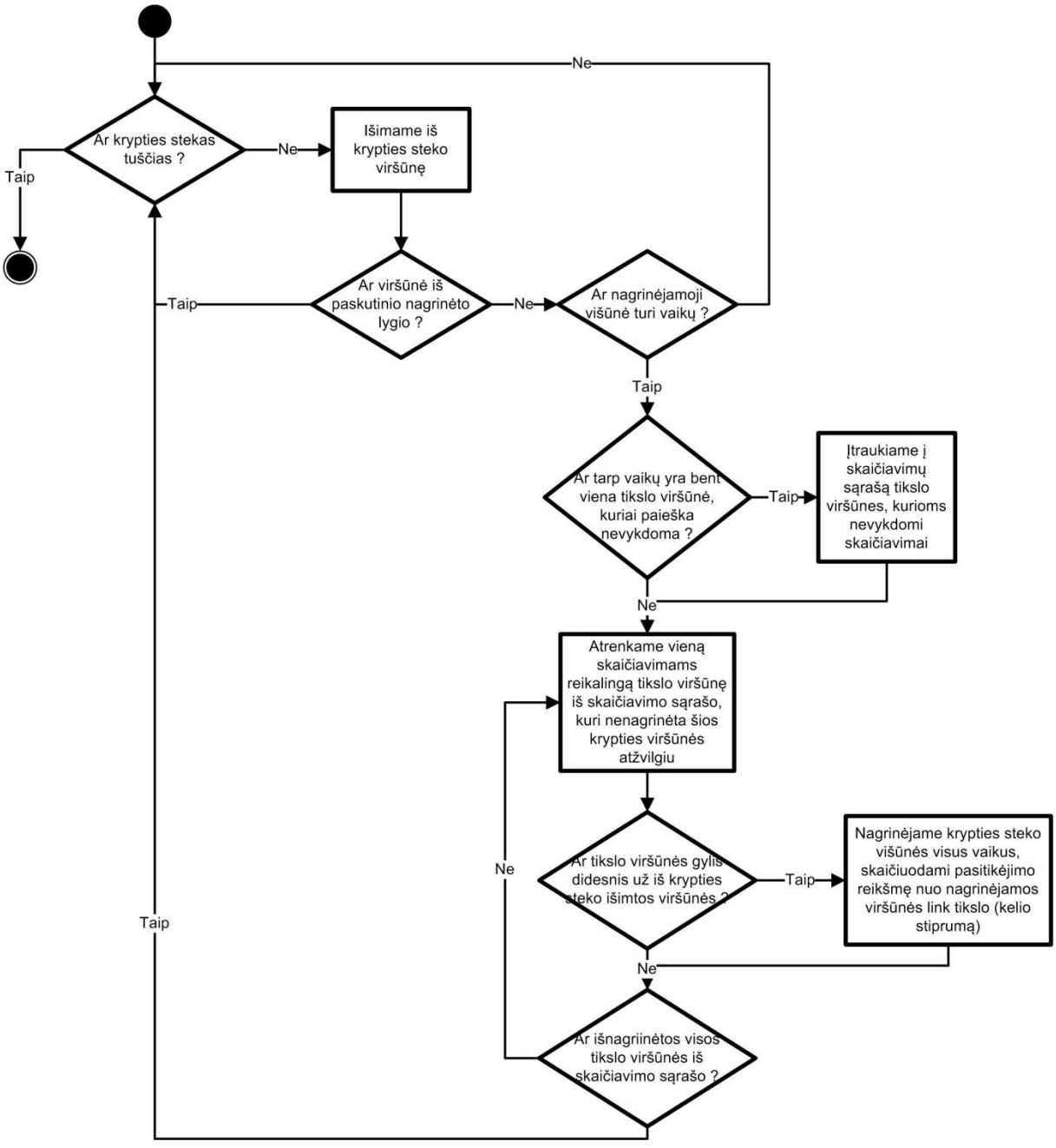
9 pav. Antro etapo algoritmo blokinė diagrama

Jeigu pavyksta rasti nagrinėjamąją viršūnę, kuri turi bent vieną gretimą tikslo viršūnę, tai nagrinėjame šitas rastas tikslo viršūnes – apskaičiuojant kelio stiprumą iki tikslo viršūnės. Taip pat lyginame rastos tikslo viršūnės kelio stiprumą su tinklo stiprumu, jeigu kelio stiprumas didesnis – tai pasižymime naują tinklo stiprumą, kurį pakeisime po to, kai bus išnagrinėtos visos rastos nagrinėjamosios viršūnės gretimos tikslo viršūnės.

Po antrojo etapo algortimo pabaigos, bus rastos visos paieškos, reikalingos tikslo viršūnės arba jeigu nepavyktų rasti visų viršūnių, tai žinosime, kad tolimesnės tikslo viršūnės ves prie mažesnių pasitikėjimo reikšmių.

3.3.3 Trečias etapas – pasitikėjimo skaičiavimas rastoms tikslo viršūnėms

Žemiau (10 pav.) pateikta paskutino etapo blokinė schema.



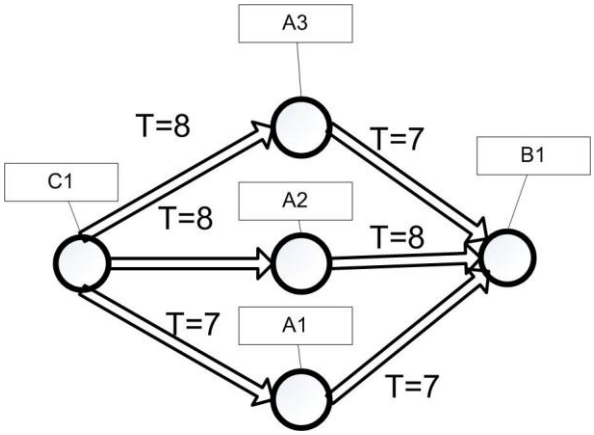
10 pav. Trečiojo etapo algoritmo blokinė diagrama

Trečiajame etape yra apskaičiuojamas pasitikėjimas rastoms tikslo viršūnėms. Imame iš krypties steko viršūnę (pavadinsime *nagrinėjamoji*) ir žiūrime kuriame lygyje ši viršūnė yra. Jeigu nebuvo vykdytas antras algoritmo etapas, vadinasi paskutinis nagrinėtas

lygis atitiks pirmojo etapo maksimalų gylį, priešingu atveju naudosime į steką įdėtos paskutinės viršūnės gylio reikšmę.

Etapas bus vykdomas tol, kol steke bus bent viena viršūnė. Patikriname ar nagrinėjamoji viršūnė turi išsaugotų gretimų viršūnių (kurias galima vadinti *vaiko* viršūnėmis). Einant nuo tikslo viršūnės link šaltinio, mes pradėsime skaičiuoti tikslo viršūnės reikšmę, kai nagrinėjamos viršūnės gylis bus mažesnis už nagrinėjamos viršūnės (išimtos iš steko). Tuo tikslu yra formuojamas skaičiavimo sąrašas, kurio pagalba mes žinome, nuo kokio gylio pradėti skaičiuoti tikslo viršūnės pasitikėjimą.

Kai iš skaičiavimo sąrašo paimtos tikslo viršūnės gylis tampa didesnis už nagrinėjamos viršūnės gylį, mes apskaičiauojame pasitikėjimą nuo nagrinėjamos viršūnės link tikslo viršūnės. Skaičiavimo pavyzdys pateiktas 11 pav. .



11 pav. Pasitikėjimo skaičiavimo grafo pavyzdys

Šiame paveiksle B1 viršūnė yra tikslo viršūnė, kelio stiprumas iki tikslo viršūnės lygus 8. Tarkime, kad pirmajame arba antrajame etape ėjome per viršūnes tokia tvarka (C1, A3, A2, A1) iš steko šios viršūnės bus išimtos atgaline tvarka (A1, A2, A3, C1). Pradedama nuo A1 viršūnės: kadangi ši viršūnė turi gretimą tikslo viršūnę, vadinasi suformuotame apskaičiaivmo sąrašę, ji bus tokio pačio lygio, o mums reikia žemesnio lygio. Apskaičivimo viršūnės lygis – tai tas pats lygis, iš kurio mes galime pasiekti tikslo viršūnę (pačiame grafe tai būtų esančios tikslo viršūnės lygis plius vienam). Kandagi ir A2 ir A3 yra tame pačiame lygyje, tai mes jų nenagrinėjame. Išsaugotos apskaičiuotos reikšmės viršūnėms A1, A2, A3 yra paimamos iš pirmo arba antro etapo, šiuo pavyzdžio metu, tarkime, kad A1 = 7, A2 = 8, A3 = 7. Toliau iš steko seka C1 viršūnė, jos lygis maženis lyginant už tikslo lygį. Vadinasi, bus atliekami tokie skaičiavimai konkrečiam pavyzdžiui (naudojami du kintamieji: vardiklis ir daliklis): pirmuoju atveju nagrinėjame, ar pastikėjimas C1 viršūnės A3 yra didesnis arba lygus nustatym kelio stiprumui, t.y. didesnis arba lygus 8, ir ar išsaugota stiprumo reikšmė didesnė arba lygi nuliui. Jeigu taip:

$$vardiklis_1 = pasitikėjimas(C1, A3) * \ išsaugotaKelioReikšmė(A3, B1) = \ 56$$
$$daliklis_1 = pasitikėjimas(C1, A3) = 8$$

Sekančiu etapu patikriname ar pasitikėjimas C1 -> A2 >= 8 ir išsaugotaKelioReikšmė A2 >= 0, kadangi ši sąlyga pavyzdyje yra tenkinama, vadinasi:

$$vardiklis_2 = pasitikėjimas(C1, A2) * \ išsaugotaKelioReikšmė(A2, B1) = \ 64$$
$$daliklis_2 = pasitikėjimas(C1, A2) = 8$$

Trečiu atveju pastikėjimas C1 ->A1 yra mažesnis už 8, vadinasi šio kelio neįtraukiame. Taigi, kai suskaičiavome visus vaikus, kurie tenkino pasitikėjimo stiprumą, galutinis pasitikėjimas nuo C1 viršūnės iki B1 gaunasi toks:

$$išsaugotaKelioReikšmė(C1, B1) = \frac{\sum vardikliai}{\sum dalikliai} = \frac{56 + 64}{8 + 8} = 7{,}5$$

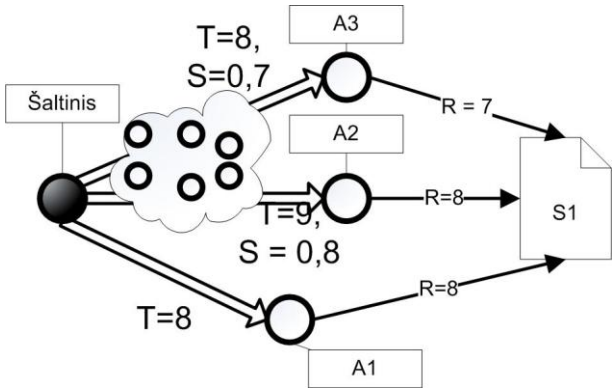
Kai trečiasis algoritmo etapas bus baigtas, mes turėsime išsaugotas reikšmes nuo šaltinio iki visų rastų tikslo viršūnių.

3.4 Straipsnio rekomendacijos skaičiavimas

Galutinei straipsnio rekomendacijai suskaičiuoti yra naudojamas vartotojų įgytas arba turimas pasitikėjimas. Žemiau pateikta formulė

$$R_{straipsnio} = \frac{\sum kelioStiprumas * vartotojoPasitikėjimas * straipsnioĮvertinimas}{\sum kelioStiprumas * vartotojoPasitikėjimas}$$

Žemiau pateiktas skaičiavimo pavyzdys.



12 pav. Straipsnio rekomendacijos skaičiavimo pavyzdys

S - kelio stiprumas iki ieškotos tikslo viršūnės, T – įgytas arba priskirtas vartotojo pasitikėjimas, R- vartotojo straipsnio įvertinimas.

Pateiktame 12 pav. matome tris viršūnes (vartotojus), kurios buvo įvertinusios straipsnį ir pagal kurias mes galime apskaičiuoti straipsnio rekomendaciją. Viršūnė A1 yra

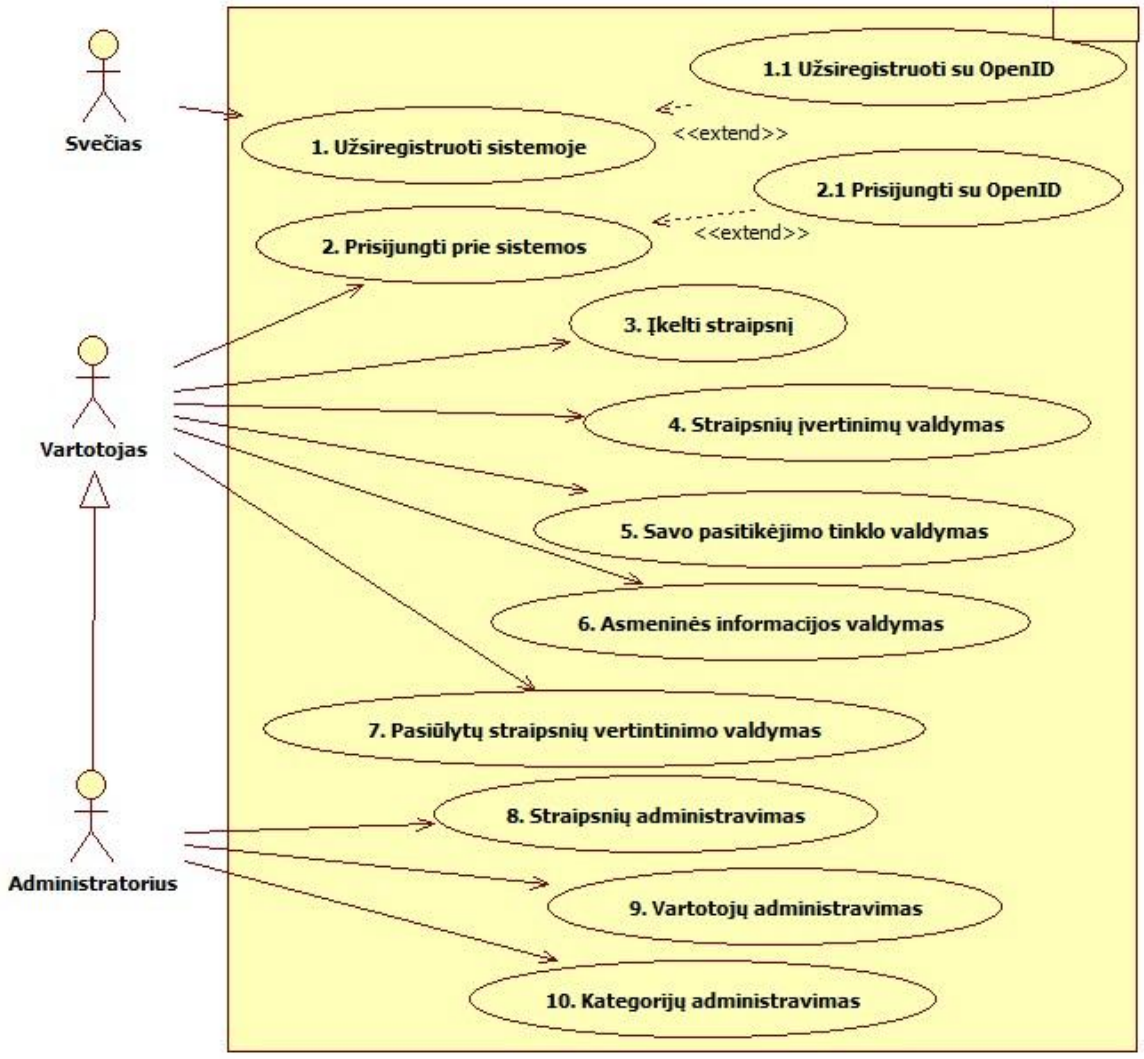
gretima šaltinio viršūnei, dėl to jos kelio stiprumo reikšmė yra maksimali (lygus 1). Viršūnių A2 ir A3 pastikėjimas yra įgytas. Taigi straipsnio rekomedacija bus apskaičiuota taip:

$$R_{S1} = \frac{1 * 8 * 8 + 0{,}8 * 9 * 8 + 0{,}7 * 8 * 7}{8 + 0{,}8 * 9 + 0{,}7 * 8} \approx 7{,}7$$

3.5 Pasitikėjimo metodo įgyvendinimas tinklaraščių vertinimo sistemoje

3.5.1 Aukščiausio lygio panaudojimo atvejis

Žemiau (13 pav.) pateikta tinklaraščių pasitikėjimo sistemos panaudos atvejų diagrama.



13 pav. Panaudos atvejų diagrama

1. Užsiregistruoti sistemoje: Norėdamas matyti sistemoje esančius straipsnius ar juos užsiregistruoti-vertinti, svečias privalo užsiregistruoti sistemoje. Registracijos metu vartotojas turi įvesti minimalius duomenis apie save.

1.1 Užsiregistruoti su OpenID: Vartotojas registracijos metu vietoj įprastinės formos pildymo gali panaudoti OpenID siūloma metodika.

2. Prisijungti prie sistemos: Vertinti straipsnius gali tik sistemoje prisijungę vartotojai, dėl to vartotojams privaloma pirmiausiai prisijungti prie sistemos. Prisijungimui yra naudojamas vartotojo vardas ir slaptažodis.

2.1 Prisijungti su OpenID: Autentifikacija vyksta pagal OpenID prisijungimo scenarijų (detalesnis scenarijus pateiktas 2.5.2 skyriuje).

3. Įkelti straipsnį: Vartotojas gali įtraukti naują straipsnį į vertinimų sistemą. Straipsnio įvedimo metu yra pateikiama informacija apie straipsnį, jo buvo vietą. Galima įkelti tik tuos straipsnius kurie URI sistemoje nėra užregistruoti.

4. Straipsnių įvertinimų valdymas: Vartotojai gali vertinti patikusius straipsnius ir taip pat, jeigu yra įmanoma, matyti jų apskaičiuotus įvertinimus remiantis socialiniu tinklu. Į valdymą įeina: įvertinti straipnį, matyti įvertinimą, redaguoti įvertinimus.

5. Savo pasitikėjimo tinklo valdymas: Vartotojas gali formuoti savo asmenų pasitikėjimo ratą: įtraukti patikimus asmenis, jiems suteikiant pasitikėjimo įvertinimą, redaguoti suteiktą pasitikėjimą arba pašalinti iš savo patikimų asmenų sąrašo.

6. Asmeninės informacijos valdymas: Vartotojas gali keisti savo pateiktą informaciją, t.y. vardą, pavardę, slaptažodį ir pan. Taip pat gali valdyti pasiūlymų uždraudimo/leidimo sąrašą (plačiau apie straipsnių pasiūlymų funkcionalumą - 7 panaudos atvejis)

7. Pasiūlytų straipsnių vertintinimo valdymas: Vartotojas gali siųsti savo patikimam asmeniui straipsnį su prašymu įvertinti arba atvirkščiai – gali gauti iš kito prašymą, kad jis įvertintų pasiūlytą straipsnį.

8. Straipsnių administravimas: Sistemos administratorius gali trinti, redaguoti sistemoje egzistuojančius straipsnius.

9. Vartotojų administravimas: Administratorius gali trinti, redaguoti ir blokuoti sistemoje esančius vartotojus.

10. Kategorijų administravimas: Administratorius gali sukurti, redaguoti ar trinti sistemoje esančias kategorijas.

3.5.2 Sistemos reikalavimai

3.5.2.1 Reikalavimai sistemai

Straipsnių vertinimų sistema turi būti realizuota internetinės svetainės pagrindu. Matyti ir vertinti straipsnius ar įvesti kitą informaciją į paslaugos svetainę, gali tik autentifikuoti vartotojai. Autentifikacijai ir registracijai vartotojai gali naudoti OpenID, tačiau tai yra nebūtina sąlyga. Turi būti galimybė registruoti ir autentifikuoti, naudojantis paslaugos svetainėje pateiktomis formomis. Naują straipsnį galima įvesti tik vieną kartą, t.y. jeigu kitas kartotojas vedamą straipsnį jau yra įvedęs, tai sistema antro tokio straipsnio neturi leisti įvesti. Įvedant straipsnį, turi būti galimybė priskirti jį vienai ar keliom kategorijom. Sistema turi turėti galimybę papildyti vartotojo pasiūlytas kategorijas.

3.5.2.2 Architektūros reikalavimai

Architektūra turi būti realizuota paskirstant jos komponentus į veiklos sluoksnius pagal n-sluoksnio metodiką. Išskyriami 3 sluoksniai, kuriuose numatomi pagrindiniai reikalavimai sistemai pateikiami 2 lentelėje.

2 lentelė. Architektūros reikalavimai

Sluoksnis	Reikalavimas
Atvaizdavimo	Kliento aplikacija turi būti pasiekama per interneto naršyklę.
	Turi būti validuojami kliento įvedami duomenys (kurie nepriklauso nuo logikos), kaip pavyzdžui: ar ne per ilgas įvestas pavadinimas, ar URI adreso formatas tinkamas.
	Visose naršyklėse atvaizdavimas turi būti vienodas.
	Realizacija turi būti įgyvendinta ASP.NET technologija.
Logikos	Pateikti duomenys turi būti validuojami.
	Rekomendacijas apskaičuoti panaudojamas siūlomas metodas.
	Dirbama su duomenų objektais, panaudojant duomenų sluoksnį
	Bendravimas su duomenų sluoksniu naudojant LINQ techonologija.
Duomenų sluoksnio	Naudojama „LINQ to SQL“ technologija realizuojanti bendravimą su MS SQL duomenų baze.

3.5.2.3 Nefunkciniai reikalavimai ir apribojimai

Sistemos nefunkciniai reikalvaimai aptarti žemiau esančioje 3 lentelėje.

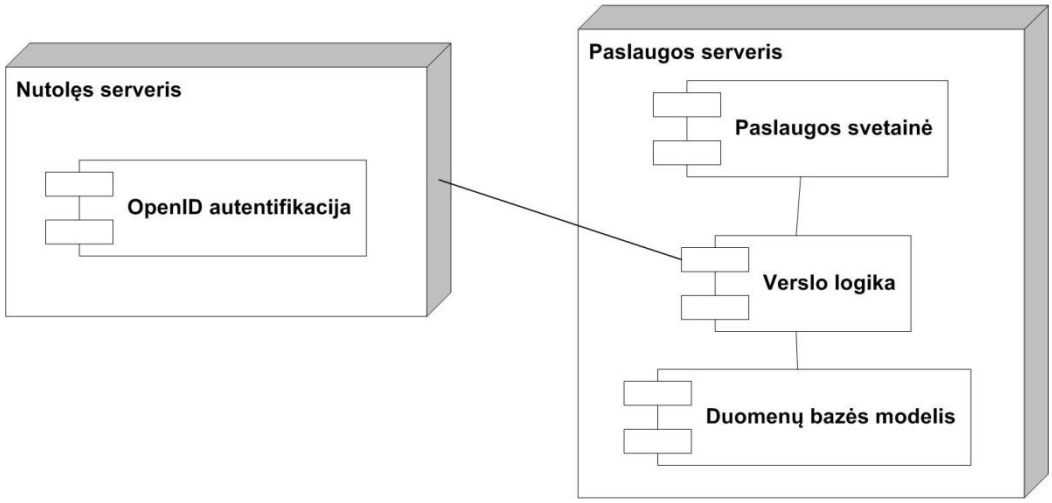
3 lentelė. Nefunkciniai reikalavimai ir apribojimai

Reikalavimas	Išpildymas
Reikalavimai veikimui	Sistemos rekomendacijų pateikimo laikas yra svarbesnis už

	kokybę, tačiau šios kokybės praradimas turi būti proto ribose, kokybės praradimas negali viršyti 10% lyginant su TidalTrust algorimto gautais rezultatais.
Reikalavimai vartotojo sąsajai.	Vartotojas turi intuityviai suprasti sistemos veikimą pateiktoje paslaugos svetainėje: vertinti straipsnius, gauti rekomendacijas ir panašiai. Sistemoje turi būti pateikta sąsaja, kad vartotojas galėtų matyti orginalų straipsnį, kartu su papildomų vertinimo įrankų juosta.
Reikalavimai praplėtimui	Sistemoje gali būti palikti papildomi interfeisai duomenų eksportui į ontologijas.

3.5.3 Sistemos architektūra

Žemiau 14 pav. pateikta bendra architektūros diagrama.

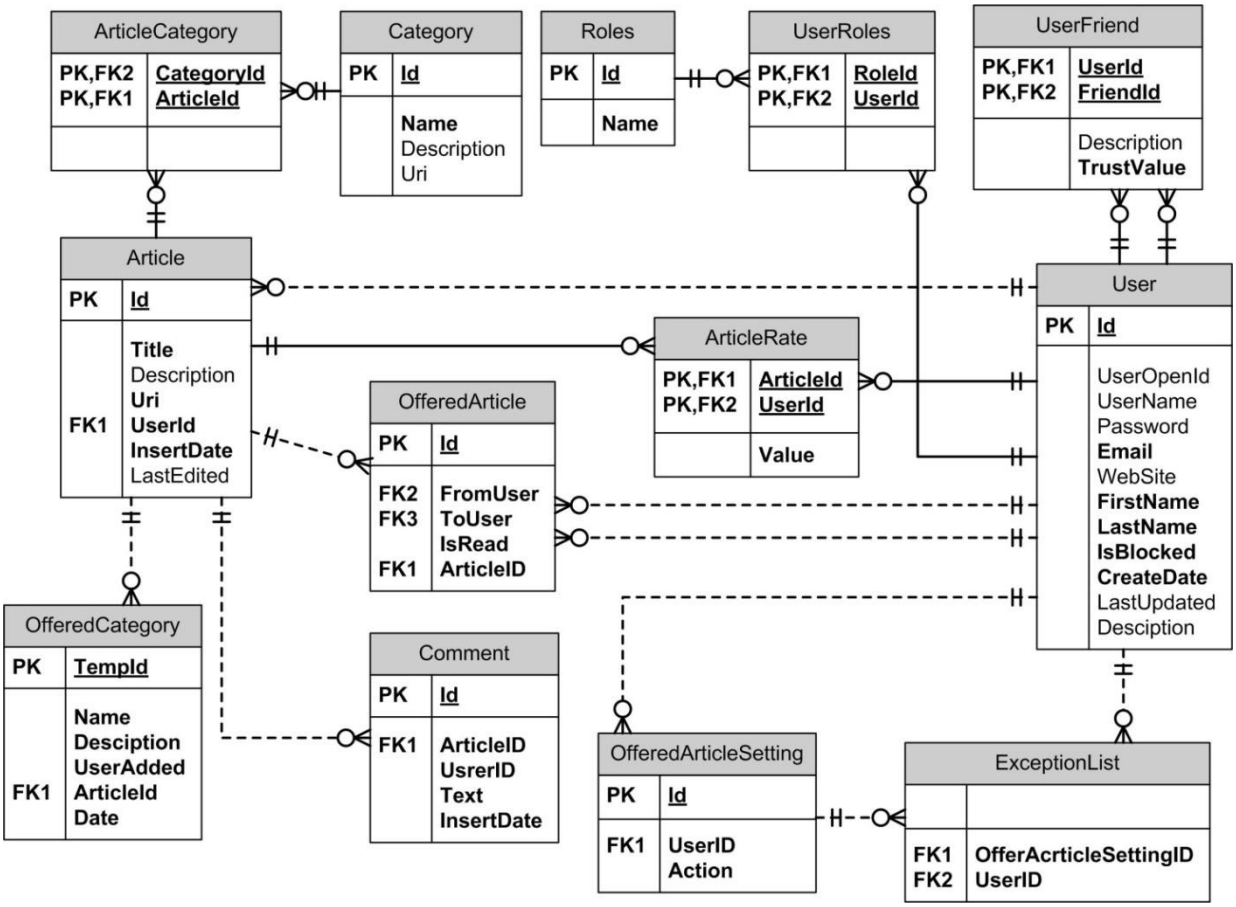


14 pav. Architektūros modelis

Paslaugos svetainė yra standartinė informacinė sistema, kurioje yra kaupiama informacija apie tinklaraščių straipsnius bei jų įvertinimą. Paslaugos serveryje sudiegti pagrindiniai programos komponentai: duomenų bazės sluoksnis, verslo logikos ir atvaizdavimo. Taip pat paslaugos serveris naudoja nutolusį OpenID komponentą, kurio pagalba yra atutentifikuojamasi arba registruojamasi prie paslaugos svetainės.

3.5.4 Duomenų bazės loginė schema

Žemiau (15 pav.) pateikta tinklaraščio pasitikėjimo sistemos duomenų bazės loginė schema.



15 pav. Duomenų bazės loginė schema

Duomenų bazėje, vartotojų tarpusiavio pasitikėjimui apskaičiuoti yra naudojamos dvi lentelės: *User* ir *UserFriend*. Straipsnio rekomendacijai pateikti yra naudojamos vartotojo pasitikėjimo apskaičiuotos reikšmės (prieš tai paminėtos lentelės) ir reikšmės iš lentelių: *ArticleRate* ir *Article*. Sistemoje naudojamas pasiūlymų mechanizmas, kurių duomenys saugomi šiose lentelėse: *OfferedArticle, OfferedArticleSetting ir ExceptionList*. *ExceptionList* ir *OfferedArticleSetting* reikalinga tam, kad apsaugoti vartotoją nuo šiukšlių (angl. *spam*) siuntinėjimo. Vartotojas gali išsisaugoti taisykles: priimti/nepriimti gaunamus pasiūlymus (lentelėje *OfferedArticleSetting.Action* stulpelio reikšmė), išskyrus šį sąrašą (iš lentelės *ExceptionList*). Šių dviejų lentelių pagalba, vartotojas gali nustatyi sau taisyklę pavyzdžiui: nepriimti iš kitų vartotojų jokių siūlymų, išskyrus šį pateiktą mano vartotojų sąrašą.

3.6 Įgyvendinimo išvados

Šiame skyriuje buvo pateiktos problemos, susijusios su tinklaraščių pasitikėjimo sistemos įgyvendinimu, kai pasitikėjimui realizuoti yra naudojamas TidalTrust algoritmas. Taip pat pasiūlytos prielaidos, kuriomis vadovaujantis buvo sukurtas patobulintas TidaTrust algoritmas, detalizuotas kiekvieno algortimo etapas.

Pateikti tinklaraščių sistemos projekto įgyvendinimo reikalavimai, bei panaudos atvejai. Taip pat įgyvendinti pagrindinai architektūrinai sprendimai: duomenų bazės loginė schame ir paslaugos sistemos komponentų architektūra.

4 Metodo eksperimentinis tyrimas

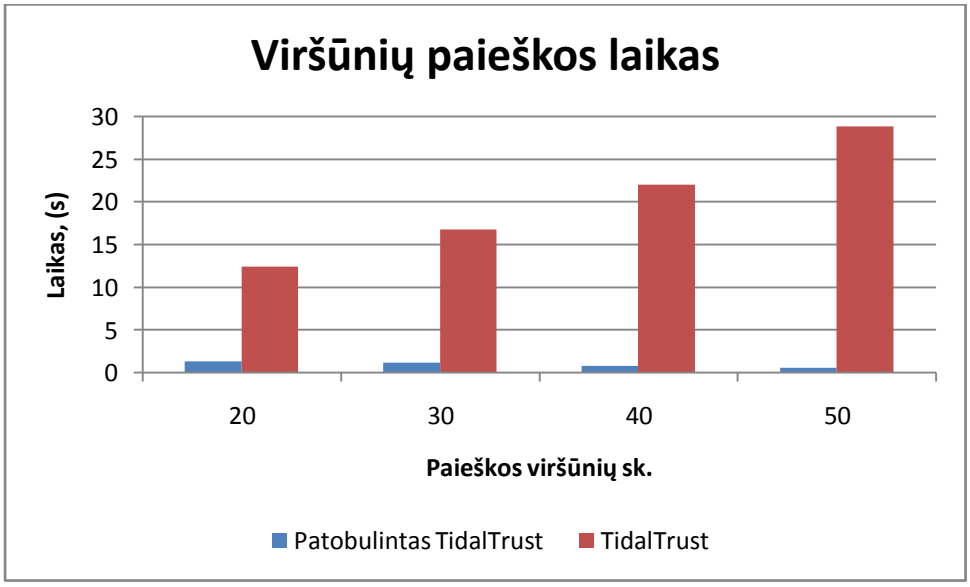
4.1 Tyrimui naudojamas sugeneruota socialinis tinklas

Pasiūlyto metodo tyrimui buvo panaudotas sugeneruotas socialinis tinklas. Šis tinklas buvo naudojamas nustatyti, kaip kinta laiko charakteristikos viršūnių skaičiui augant. Taip pat norint ištirti kokią įtaką padaro neneagrinėtos viršūnės, rezultatai buvo lyginami su TidalTrust algoritmo gautais skaičiavimais. Tyrimui buvo sugeneruotas grafas su 100 viršūnių, kiekvienai viršūnei buvo generuojami pasitikėjimo ryšiai, viršūnės pasitikėjimo ryšių skaičius siekė nuo 1 iki 15, o pasitkėjimo reikšmės nuo 6 iki 10. Šiame tinkle buvo vykdoma viršūnių paieška tokias kiekiais: 20, 30, 40, 50. Paieškai atsitiktine tvarka buvo pasirenkamos viršūnės. Žemiau pateikta lentelė, kurioje pateikiamas vidutinis laikas (sekundėmis) užtruktas reikalingai viršūnių skaičiaus paieškai atlikti.

4 lentelė. Paieškos laikas pagal ieškomų viršūnių skaičių.

	Paieškoje ieškomų viršūnių skaičius			
	20	**30**	**40**	**50**
Patobulintas TidalTrust	1,35	1,17	0,85	0,58
TidalTrust	12,4	16,76	22,03	28,81

Žemiau (16 pav.) pateikti rezultatai grafiniu pavidalu.



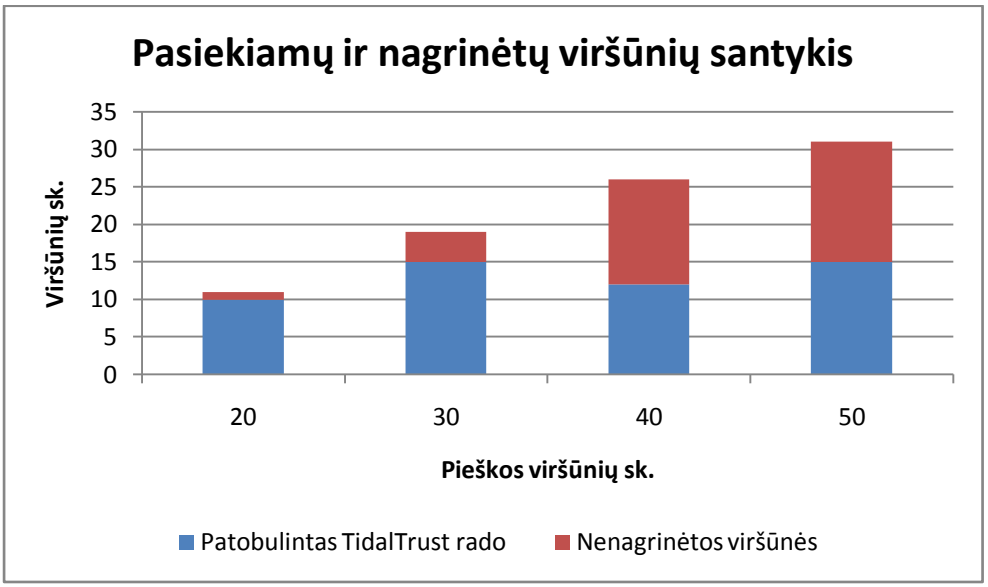
16 pav. Viršūnių paieškos laiko grafikas

Pagal pateiktus duomenis, matome, kad paiešką vykdant pagal TidalTrust algoritmą laikas auga, o patobulintas algoritmas iš laiko panašų laiko intervalą. Šis tinklas, kaip ir realus socailinis tinklas, turėtų turėti paieškos viršūnės atžvilgiu nepasiekaimų viršūnių. Toliau pateikta, kiek viršūnių sugeneruotame tinkle buvo pasiekiamos ir kiek patobulintas algortimas paieškai jų naudojo.

5 lentelė. Pasiekiamų ir nagrinėtų viršūnių skaičius.

	Paieškoje ieškomų viršūnių skaičius			
	20	**30**	**40**	**50**
Patobulintas TidalTrust nagrinėjo	10	15	12	15
Pasiekiamos viršūnės	11	19	26	31

Grafike galima matyti, kokią dalį sudaro patobulinto metodo metu nagrinėtos viršūnės ir kokia dalis buvo atmesta.



17 pav. Pasiekiamų ir nagrinėtų viršūnių grafikas

Kaip matome pagal pateikta 5 lentelę, pasiūlytas metodas atmeta tam tikrą viršūnių skaičių, tad buvo pabandyta panagrinėti, kokį poveikį nenagrinėtos viršūnės padaro galutiniam pasitikėjimo rezultatui. Žemiau pateikta formulė, pagal kurią buvo lyginama:

$$T = \frac{\sum viršūnėsPasitikėjimoĮvertinimas * kelioStiprumasIkiViršūnės}{\sum kelioStiprumasIkiViršūnės}$$

Pagal šią formule gavome tokius rezultatus

6 lentelė. Pasitikėjimo skirtumas lyginant gautus algortimo rezultatus

	Paieškoje ieškomų viršūnių skaičius			
	20	**30**	**40**	**50**
Patobulintas TidalTrust	6,941	7,28	7,52	7,180
TidalTrust	6,936	7,43	7,60	7,179
Skirtumas(absoliutus)	*0,005*	*0,15*	*0,08*	*0,001*

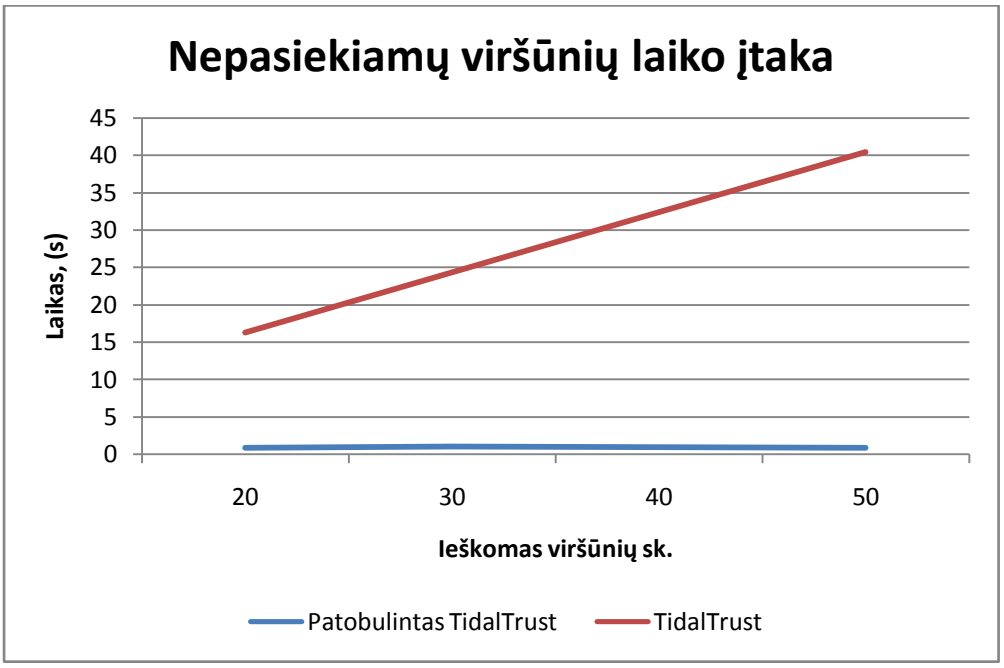
Kaip matome pagal pateiktus rezultatus 6 lentelėje, atmestos viršūnės nepadaro didelės įtakos bendram vidutiniam pasitikėjimui.

Toliau buvo atliktas eksperimentas, siekiant nustatatyti laiko įtaką, kai paieškos viršūnės yra nepasiekiamos, t.y. tinkle buvo pridėtos papildomos viršūnės, nepasiekiamos nagrinėjamos viršūnės atžvilgiu. Žemiau lentelėje pateikta paieška, kuri užtrukto tam tikrą laiką (sekundėmis) ieškant neagzistuojančių viršūnių.

7 lentelė. Nepasiekiamų viršūnių paieškos laiko trukmė.

	Paieškoje ieškomų viršūnių skaičius			
	20	**30**	**40**	**50**
Patobulintas TidalTrust	0,95	1,07	0,97	0,93
TidalTrust	16,32	24,37	32,41	40,47

Rezultatu grafinis pavidalas pateiktas žemiau 18 pav.



18 pav. Nepasiekiamų viršūnių laiko įtaka

Pagal gautus rezultatus galime teikti, kad praplėstam TidalTrust algorimtui įtakos nepasiekiamų viršūnių skaičius neturi, taip kaip lyginant su nepatobulintu TidalTrust algoritmu.

4.2 ATS tinklaraščių sistemos sukaupti duomenys

Projekto metu buvo sukurta tinklaraščių vertinimo sistema (ATS), kurioje vartotojai galėjo vertinti straipsnius, bei kurti savo socialinį pasitikėjimo tinklą. Šios paslaugos svetainės pagrindu buvo norima atlikti tyrimą, ar taikomi pasitikėjimo algoritmai atitinka vartotojo įvertintus straipsnius, t.y. koks yra vidutinis nuokrypis tarp vartotojo

įvertinto straipsnio ir apskaičuoto taikant TidalTrust ir patobulinta TitadlTrust algoritmus. Taip pat buvo bandoma išsiaiškinti, ar naudojama subjektyvi nuomonė labiau pasiteisino nei objektyvi, tinklaraščių vertinimo sistemoje.

Tačiau projekto metu buvo sukaupta per mažai informacijos, kad būtų galima palyginti vartotojo įvertintus straipsnius su rekomendacijom, kuriems apskaičiuoti buvo naudojami anksčiau paminėti algoritmai. Šioje sistemoje sukurto socialinio tinklo tarpusiavio artumas yra labai didelis, t.y praktiškai kiekvienas nagrinėtas vartotojas turi tiesiogiai prieinamą tikslo viršūnę.

4.3 Tinklaraščių vertinimo sistemų kiekybinis palyginimas

Pateiktas (8 lentelėje) analitinėje dalyje nagrinėtų sistemų kiekybinis palyginimas su sukurta tinklaraščių vertinimo sistema (ATS).

8 lentelė. Tinklaraščių vertinimo sistemų kiekybinis palyginimas

	Technorati.com	Topix.lt	Zynios.lt	Authority.com	Cut.lt	ATS
Ar pasitikėjimas subketyvus ?	-	-	-	-	-	+
Vertinimo skalė turi ribas	-	-	-	+	-	+
Vertinami konkretūs straipsniai	-/+	+	+	+	+	+
Informacija kategorizuojama	+	+	+	+	+	+
Vertinama straipsnio kategorija	-	-	-	+	-	-
Vertinimas apibūdinamas kelias parametrais	+	-	-	+	-	-
Vartotojas priskiria įvertinimą	-	+	+	+	+	+

Pagrinidinis išskirtinumas sukurtos ATS sistemos yra tas, kad pasitikėjimo įvertinimo skaičiavimui yra naudojama subjektyvi vartotojo nuomonė.

4.4 Ekspermento tyrimo išvados

Patobulinto TidalTrust algoritmo skaičiavimams buvo panaudotas sugeneruotas socailinis tinklas. Šio tinklo pagalba buvo nustatyta, kad skaičiavimui didelės įtakos nepadaro nepasiekiamos viršūnės, t.y. poveikis žymiai mažesnis negu klasikinio TidalTrust algoritmo.

Taip pat, pagal gautus rezultatus, kai nenagrinėjamos visos viršūnės, galima matyti, kad pasitikėjimo lygio įvertis nedaug kuom skiriasi nuo TidalTrust metu išnagrinėtų visų viršūnių.

5 Išvados

1. Išanalizavus vartotojo tarpusiavio pasitikėjimo algorimus, kurie remiasi grafo paieškos metodais, buvo pastebėta, kad dėl greitaveikos labiau tinka trumpesnių kelių paieškos metodai, negu visų kelių. Dėl to galutinei sistemos realizacijai buvo pasirinktas trumpiausio ir stipriausio (TidalTrust) metodo įgyvendinimas.
2. Detaliau išanalizavus trumpiausio ir stipriausio kelio algoritmą (TidalTrust), buvo pastebėta, kad realizacijos metu šis algoritmas turi tam tikrų trūkumų: kiekviena viršūnė yra nagrinėjima atskirai, dėl to didėjant tinklui arba nepasiekiamų višūnių skaičiui, skaičiaivmo laikas labai išauga. Išlaikant pagrindinus algoritmo principus buvo pasiūlytos galimos prielaidos ir metodas, kaip spręsti, kad algoritmo veikimo laikas būtų trumpesnis.
3. Algortimo veikimo laiko pagerinimui buvo pasirinkta ne tik nagrinėti visas viršūnes iškarto, bet papildomai įterpti vieną etapą, kurio pagalba būtų galima atmesti labiausiai nutolusias ir silpniausias viršūnes. Visų viršūnių nagrinėjimas leido pagerinti laiką tuo atveju, kai nepasiekiamų viršūnių skaičius auga ir jeigu būtų nagrinėjama po vieną viršūnę, tai paieška būtų vykdoma tiek kartų iki giliausio grafo taško, kiek tų viršūnių yra.
4. Atlikus ekspermentą buvo nustatyta, kad patobulinto TidalTrust algoritmo apskaičuoti pasitikėjimo įvertinimai nelebai skiriasi nuo orginaliaus TidalTrust algortimo gautų rezultatų. Tačiau patobulintas algoritmas savo greitaveika žymiai lenkia orginalų TidalTrust.
5. Patobulintas TidalTrust metodas gali būti pritaikytas ne tik tinklaraščių vertinimo sistemose, bet ir kituose srityse, kur reikalingi pasitikėjimo skaičiavimai: aukcijonų, knygų, filmų, kelionių ir kitose rekomendacinio pobudžio sistemose.

6 Literatūra

[1] Page Lawrence; Brin Sergey; Motwani Rajeev; Winograd Terry. „*The PageRank Citation Ranking: Bringing Order to the Web",* 1999;
<http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf>

[2] Technorati.com sistema [Žiūrėta 2009-04-17], prieiga internete < http://www.technorati.com>

[3] Topix.lt sistema [Žiūrėta 2009-04-17], prieiga internete <http://www.topix.lt>

[4] Zynios.lt sistema [Žiūrėta 2009-04-17], prieiga internete <http://www.zynios.lt>

[5] Cut.lt sistema [Žiūrėta 2009-04-17], prieiga internete <http://www.cut.lt>

[6] Authority.com sistema [Žiūrėta 2009-04-17], prieiga internete <http:// www.authority.com >

[7] Jennifer Ann Golbeck; „Computing and applying trust in web-based social networks", <www.lib.umd.edu/drum/bitstream/1903/2384/1/umi-umd-2244.pdf >

[8] Stephen Paul Marsh; „*Formalising Trust as a Computational Concept*", <http://reference.kfupm.edu.sa/content/f/o/formalising_trust_as_a_computational_con_33819.pdf>

[9] Mohsen Lesani; Niloufar Montazeri; „*Fuzzy trust aggregation and personalized trust inference in virtual social networks*", Computer engineering deparment sharif university of technology Tehran, Iran;
< ce.sharif.edu/~montazeri/downloads/Papers/2009TrustPaper.pdf >

[10] Syavash Nobarany; Mona Haraty; Dan Cosley; „*GePuTTIS: General Purpose Transitive Trust Inference System*", School of ECE, University of Tehran, Iran;
< www.cs.cornell.edu/~danco/research/papers/trust-nobarany-aaai-ss-2008.pdf >

[11] Ugur Kuter; „SUNNY: A New Algorithm for Trust Inference in Social Networks Using Probabilistic Confidence Models", < www.cs.umd.edu/~golbeck/downloads/Sunny.pdf >

[12] Jaideep Srivastava; „*Data Mining for Social Network Analysis*", Australasian Data Mining Conference (AusDM) 2007, <http://kdl.cs.umass.edu/papers/jensen-neville-nas2002.pdf>

[13] Nima Dokoohaki; „*Modeling and Representing Trust Relations in Semantic Web-Driven Social Networks: An Ontological Analysis*", The Royal Institute of Technology Stockholm, May 2007, < http://web.it.kth.se/~nimad/Nima_Thesis.pdf >

[14] Yarden Katz; „*Using Social Network-based Trust For Default Reasoning On The Web*", Maryland Information and Network Dynamics Lab, University of Maryland,

< http://www.cs.umd.edu/~golbeck/papers/jwsSocDefaults.pdf >

[15] Ilya Zaihrayeu; Paulo Pinheiro da Silva; Deborah L. McGuinness; „*IWTrust: Improving User Trust in Answers from the Web*“, Proceedings of 3rd International Conference on Trust Management (iTrust2005), Springer, France, 2005;
<http://www.ksl.stanford.edu/people/pp/papers/Zaihrayeu_iTrust_2005.pdf>

[16] Deborah L. McGuinnes; Paulo Pinheiro da Silva; „*Explainig Answer from the Semantic Web: The Inference WebApproach*“, <ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-04-01.pdf >

[17] Qing Zhang; Ting Yu; Keith Irwin; „*A Classification Scheme for Trust Functions in Reputation-Base Trust Management*“, <http://trust.mindswap.org/trustWorkshop/papers/4-f.pdf >

[18] Honglei Zeng; Maher A. Alhossaini1; Li Ding; Richard Fikes1; Deborah L. McGuinness; „*Knowledge Systems: Computing Trust from Revision History*“, Proceedings of the 2006 International Conference on Privacy, Security and Trust,
< http://ebiquity.umbc.edu/_file_directory_/papers/302.pdf>

[19] John O‘Donovan; Vesile Evrim; Barry Smyth; „*Personalizing Trust in Online Auctions*“,
<http://imsc-dmim.usc.edu/publications/STAIRS_06_trustonebay_11.pdf >

7 Terminų ir santrumpų žodynas

Tinklaraštis (angl. Weblog, Blog) – internetinis dienoraštis arba automatiškai formatuojami naujienų puslapiai. Dažnai lietuviškuose portaluose šis terminas vadinamas „blogas“.

URI (angl. Uniform Resource Identifier) – vieningas resursų identifikatorius, kuriame pateikiama simbolių eilutė, kurios pagalba galima nusakyti ar nurodyti resursus internete.

OpenID – atviras, necentralizuotas vartotojo autentifikacijos standartas, leidžaintis vartotojui prisijungti prie daugelio sistemos paslaugų naudojant vieną identifikatorių.

Socialinis tinklas – tinklinė struktūra, sudaryta iš mazgų ir ryšių. Mazgais nusakomi socialinio tinklo nariai: asmenys, organizacijos. Ryšiai tarp mazgų nusako socialinio tinklo narių tarpusavio sąveikas (pasitikėjimą).

LINQ to SQL – Microsoft sukurta ORM (angl. Object-Relational mapping) technologija, skirta duomenų modelio konvertavimui iš realiacinės duomenų bazės į objektine orentuotą programavimo kalbą

Stekas (angl. Stack) – duomenų struktūra, kuriame paskutinis įdėtas kintamasis yra pirmiausiai išimamas.

Eilė (angl. Queue) – duomenų struktūra, kuriame pirmas įdėtas kintamasis į sąraša yra išimamas pirmiausiai.

MOAT (angl. Meaning Of A Tag) – yra semantinio tinklo karkasas, kuris leidžia publikuoti semantinius straipsnius pagerintus specialiomis žymėmis.

Priedai

Patobulinto TidalTrust algoritmo pseudo kodo etapai

Pirmas etapas

```
     // Pirmasis etpas - paieška tikslo viršūnės
     while ( (not eilė.isEmpty()) ir (gylis <= maksGylis) )
     {
          viršūnė = eilė.dequeue()
          kryptiesStekas.push( viršūnė )
          if ( not pasitikėjimoGrafas.yraGretimas(viršūnė, tiksloViršunių) )
          {
               if ( rastaBentVienaViršūnėVedantiLinkTikslo)
               {
                    continue;
               }
               foreach ( gretimaViršūnė in viršūnė.gretimosViršūnės)
               {
                    if ( not aplankyta[ gretimaViršūnė ] )
                    {
                         aplankyta[ gretimaViršūnė ] = true
                         kitoLygioEilė.queue( gretimaViršūnė )
                    }

                    if ( kitoLygioEilė.contains( gretimaViršūnė ) )
                    {
                         kelioStiprumas = min( stiprumasNuoŠaltinio[ viršūnė ],
                         pasitikėjimoGrafas.gautiPasitikėjima(viršūnė,
gretimaViršūnė))
                         stiprumasNuoŠaltinio[ gretimaViršūnė] = max (
                                   stiprumasNuoŠaltinio[ gretimaVirsūnė ],
                                   kelioStiprumas )
                         vaikas[viršūnė].push(gretimaViršūnė)
                    }
               }
          }
          else
          {
               maksGylis = gylis
               rastaBentVienaViršūnėVedantiLinkTikslo = true
```

```
            foreach(tikslo in viršūėePasiekiantiTiklo)
            {
                kelioStiprumas = stiprumasNuoŠaltinio[ viršūnė ]
                stiprumasNuoŠaltinio[tikslo] =
                    max( stiprumasNuoŠaltinio[tikslo],  kelioStiprumas )
                rastuVirsuniuSrasas.Add(tikslo)
            }
        }

        if ( eilė.isEmpty() )
        {
            if (rastaBentVienaViršūnėVedantiLinkTikslo)
            {
                //Reikia korekcijos
                while (not kitoLygioEilė.isEmpty())
                {
                    laikinaViršūnė = kitoLygioEilė.dequeue()
                    aplankyta[laikinaViršūnė].remove()
                    //Išimame vaiko viršūnę
                    Ištrinti(vaikas[viršūnė], laikinaViršūnė)
                }
            }
            if (maksGylis > gylis)
            {
                //Pasidarome kopija nagrinejamos sekancios eiles
                paskutineEile = kitoLygioEilė
            }
            eilė = kitoLygioEilė
            gylis = gylis + 1
            kitoLygioEilė = new Queue()
        }
    }
```

Antras etapas

```
    // Antras etpas
    if ( YraNerastuTiksloViršūnių() )
    {
        maksRibineReiksme = MaksimaliVisųTiksloViršūniųReikšmė()
        kitoLygioEilė = new Queue()

        while ( not paskutineEile.isEmpty() )
        {
            viršūnė = paskutineEile.dequeue()
            if ( not pirmaPaieška )
            {
                kryptiesStekas.push( viršūnė )
            }
```

```
                if ( stiprumasNuoŠaltinio[viršūnė] >= maksRibineReiksme )
                {
                     if      (      not      pasitikėjimoGrafas.yraGretimas(viršūnė,
tiksloViršunių)
                          ir  pirmaPaieška)
                     {
                          vaikas[viršūnė].push(gretimaViršūnė)
                          if ( ArAtrastaTiksloViršūnėJau( viršūnė) )
                          {
                               continue;
                          }
                          if ( not aplankyta[ gretimaViršūnė ] )
                          {
                               aplankyta[ gretimaViršūnė ] = true
                               kitoLygioEilė.queue( gretimaViršūnė )
                          }

                          if ( kitoLygioEilė.contains( gretimaViršūnė ) )
                          {
                               kelioStiprumas = min( stiprumasNuoŠaltinio[ viršūnė
],
                               pasitikėjimoGrafas.gautiPasitikėjima(viršūnė,
                                                                    gretimaViršūnė))
                               stiprumasNuoŠaltinio[ gretimaViršūnė] = max (
                                    stiprumasNuoŠaltinio[ gretimaVirsūnė ],
                                    kelioStiprumas )
                          }
                     }
                     else
                     {
                          foreach(tiklo in  NerastosTiksloViršūnėsGretimosViršūnei
                                                                    (viršūnė) )
                          {
                               kelioStiprumas = min( stiprumasNuoŠaltinio[ viršūnė
],
                               pasitikėjimoGrafas.gautiPasitikėjima(viršūnė,
                                                                 gretimaViršūnė))
                               stiprumasNuoŠaltinio[ gretimaViršūnė] = max (
                                    stiprumasNuoŠaltinio[ gretimaVirsūnė ],
                                    kelioStiprumas )
                               vaikas[viršūnė].push(gretimaViršūnė)
                          }
                     }
                }
                if ( paskutineEile.isEmpty() )
                {
                     pirmaPaieška = false
```

```
                    gylis = gylis + 1

                    if (not YraNerastuTikslоViršūnių() )
                    {
                         break;
                    }
                    paskutineEile = kitoLygioEilė
                    kitoLygioEilė = new Queue()
               }
          }
     }
```

Trečias etapas

```
     // Jeigu pirmuoju etapu radoms visas tikslo viršūnes, tai maksGylis lieka
     // iš pirmojo etapo, priešingu atveju, maksGylio reikmšė nustatoma pagal
paskutinės
     // krypties viršūnės gylį.

     while (not kryptiesStekas.isEmpty() )
     {
          viršūnė = kryptiesStekas.pop()
          if ( viršūnė.gylis == maksGylis)
          {
               continue;
          }
          if ( ArVišūnėTuriVaikų( viršūnė) )
          {
               tiksloViršūnės = IšrintiTarpVaikųEsančiasTiksloViršūnes()
               foreach ( tikslo in tiksloViršūnės )
               {
                    if (not ArPriklausoGalutinėsNagrinėjamosmTiksloVir ( tikslo) )
                    {
                         nagrinejamosTiksloViršūnės.add(tikslo,lygis)
                    }
               }

               foreach ( nagrinėjama in nagrinejamosTiksloViršūnės )
               {
                    if ( nagrinėjama.gylis > viršūnė.gylis)
                    {
                         skaitiklis = 0
                         vardiklis = 0
                         foreach (Viršūnė vaikas in vaikas[viršūnė])
                         {
                              if ( pasitikėjimas(viršūnė, vaikas) >= max
                                   ir issaugotaReikšmė(vaikas,tikslas) >= 0)
                              {
```

```
     skaitiklis = skaitiklis + pasitikejimas(viršūnė,vaikas) *
                issaugotaReikšmė(vaikas,tikslas)
     vardiklis = vardiklis + pasitikejimas(viršūnė,vaikas)
                                   }
                              }
                              if ( vardiklis > 0 )
                              {
                              issaugotaReikšmė(viršūnė,tikslas)    =    skaitiklis    /
vardiklis
                              }
                              else
                              {
                              išsaugotaReikšmė(viršūnė,tikslas) = -1
                              }
                         }
                    }
               }
          }
     }
```