VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Baigiamasis bakalauro darbas

Tvarkaraščio realizacija žiniatinklio terpėje taikant grafo viršūnių spalvojimo algoritmus

Atliko:
Andrius Kankevičius parašas

Vadovas:
dr. Igor Katin

Vilnius
2019

TURINYS

Santrauka **3**

Summary **4**

Įvadas **5**

1. Tvarkaraščio sudarymo problema **7**
1.1. Paskaitų tvarkaraščio sudarymas 7
1.2. Ribojimai....8
1.2.1. Pagrindiniai būtini tvarkaraščio ribojimai....8
1.2.2. Nepatogumų ribojimai....8
1.3. Panašių sistemų analizė....8

2. Grafų teorijos elementai **12**
2.1. Pagrindinės sąvokos ir apibrėžimai. 13
2.2. Spalvų grafas....15
2.2.1 Godusis algoritmas grafo viršūnių spalvinimui 16
2.2.2 Grįžimo algoritmas grafo viršūnių spalvinimui 17
2.3. Spalvų grafo pritaikymas realiems uždaviniams spręsti 18
2.4. Grafo viršūnių spalvinimo algoritmų pritaikymas tvarkaraščio sudarymui 19
2.4.1. Grafo sudarymas tvarkaraščiui formuoti....19
2.4.2. Sudaryto grafo spalvojimo realizacija....20

3. Tvarkaraščio sudarymo realizavimas žiniatinklio terpėje **23**
3.1. Techninės sistemos charakteristikos....23
3.2. Tvarkaraščio matematinis modelio formavimas....24
3.3. Sistemos duomenų modelis....25
3.4. Realizuojamos programos struktūra 29
3.5. Tvarkaraščio sistemos realizacijos pavyzdys....31

Išvados **35**

Ateities tyrimų planas **36**

Literatūros šaltiniai **37**

Priedai **39**

Santrauka

Tvarkaraščio realizacija žiniatinklio terpėje taikant grafo viršūnių spalvojimo algoritmus

Akademinio tvarkaraščio sudarymas – tai tam tikros baigtinės aibės paskirstymas laike procesas, kuris ribojamas tam tikrų resursų ištekliais. Tvarkaraščių sudarymo problema yra aktuali daugeliui mokymo įstaigų ir dažnai sprendžiama grafo spalvinimo būdu.

Šio darbo tikslas – sudaryti studentų savaitinį tvarkaraštį tipinio universiteto fakultetui, naudojant grafo viršūnių spalvojimo algoritmus. Sudarant tvarkaraštį atsižvelgta į įvairius savaitinio tvarkaraščio sudarymo reikalavimus, tokius kaip dėstytojas ar studentas negali būti dvejose vietose vienu metu, ribotas paskaitų skaičius per dieną, specializuotos auditorijos, tinkamos teoriniams ar praktiniams dalykams, dėstytojų reikalavimai darbo laikui.

Tvarkaraštis realizuotas žiniatinklio terpėje, taikant grafo viršūnių godžiojo ir grįžimo spalvojimo algoritmus. Sukurta tvarkaraščio programa orientuota į vartotoją, lengvai valdoma ir tinkamai sudaro tvarkaraštį, kuris gali funkcionuoti ir realioje aplinkoje.

Summary

Web Based Implementation of Shedule Generation by Graph Coloring Algorithms

Academic scheduling it is a process of distributing a finite set over time which is limited by the certain recources. The problem of scheduling is relevant to various educational institutions and is often solved by graph coloring.

The main task of the article is to compile a student weekly schedule for a typical University Faculty, basically by using the graph vertices coloring algorithms. Developing the scheduler various requirements of weekly scheduling was taken into account, such as the fact that the teacher or student is unable to attend in two same places at the same time, limited number of lectures per day, specialized audiences, suitable for theoretical or practical activities and teachers working hours preferences.

The timetable was depicted in the web by using „greedy“ and „backtracking“ graph vertices coloring algorithms. Created scheduler program is user-oriented, easy to operate and properly schedules a timetable that can fuction in a real-life environment.

Įvadas

Kiekviena švietimo įstaiga susiduria su viena dažniausių akademinio planavimo problemų – paskaitų tvarkaraščiu. Tvarkaraštis formuojamas kiekvienai studentų grupei pagal studijuojamą studijų programą. Tvarkaraštis turi apimti dėstytojus, studentus, dalykus, auditorijas taip, kad būtų išvengta konfliktų, t. y. turi būti tenkinami griežti ir minkšti reikalavimai. Tvarkaraščių formavimo uždaviniai priklauso NP (*node–point*) sudėtingumo klasei, kadangi šiems uždaviniams nėra sukurtų polinominio sudėtingumo metodų. Taigi, norint rasti artimiausią optimalų sprendimą per protingą laiką, pagrindinis dėmesys skiriamas euristiniams metodams. Grafo spalvinimas yra vienas iš tokių euristinių algoritmų, pagal kurį galima sudaryti tvarkaraštį. Šiame darbe realizuoti keletas grafo spalvinimo algoritmų gali būti pritaikomi daugeliui aukštojo mokslo švietimo įstaigų, kuriose yra skirtingi duomenys ir reikalavimai. Darbe dėmesys sutelkiamas į studijų tvarkaraščio sudarymą, atsižvelgiant į griežtus ir minkštus ribojimus.

Studijuojami dalykai laikomi grafo viršūnėmis – mazgais, o briaunos brėžiamos tarp konfliktuojančių dalykų, pavyzdžiui, jeigu juos studijuoja tie patys studentai arba juos veda tie patys dėstytojai.

Darbo tikslas – sudaryti studentų savaitinį tvarkaraštį tipinio universiteto fakultetui, naudojant grafo viršūnių spalvojimo algoritmus.

Darbo uždaviniai:

- Išanalizuoti tvarkaraščio sudarymo problemą ir jų galimus sprendimo būdus.
- Išnagrinėti grafų teorijos pagrindines sąvokas ir grafo viršūnių spalvojimo algoritmus, kurių pagalba bus sudaromas tvarkaraštis.
- Automatiškai sugeneruoti tvarkaraščius taikant grafo viršūnių spalvojimo algoritmus ir juos atvaizduoti žiniatinklio terpėje.

Sudarant tvarkaraštį, susiduriama su įvairiais savaitinio tvarkaraščio sudarymo reikalavimais, tokiais kaip ribotas paskaitų skaičius per dieną, specializuotos auditorijos, tinkamos teoriniams ar praktiniams dalykams, dėstytojų reikalavimai darbo laikui.

Atsižvelgiant į šiuos reikalavimus, kiekvienai studentų grupei sudaromas semestro tvarkaraštis. Tvarkaraščio sudarymo problemos automatizavimas yra svarbi užduotis, nes pateikia optimalius sprendimus, dėl kurių institucijos gali sutaupyti daug darbo valandų, padidinti produktyvumą, švietimo ir paslaugų kokybę.

Taigi pagrindinės tvarkaraščių formavimo nuostatos buvo šios:

- Darbo dienų skaičiaus per savaitę ribojimas: paprastai visas tvarkaraštis sudaromas tik vienai savaitei ir kartojamas tam tikrą savaičių skaičių.

Tvarkaraščiai buvo sudaromi:

- Kiekvienos studentų grupės paskaitų tvarkaraštis.
- Auditorijų užimtumo tvarkaraštis.
- Kiekvieno dėstytojo darbo tvarkaraštis, nurodant patalpą, kur jis dirba ir dalyką, kurį dėsto.

Pagrindinės problemos formuojant tvarkaraštį buvo šios:

- Dėstytojas ar studentas negali būti dvejose vietose vienu metu.
- Auditorijoje tuo pačiu metu negali dirbti keletas grupių ar keletas dėstytojų.
- Dėstytojas dirba jam tinkama laiku, pavyzdžiui, ketvirtadieniais ir trečiadieniais.

Sukurta programa turi būti realizuota žiniatinklio terpėje, orientuota į vartotoją ir lengvai valdoma.

1. Tvarkaraščio sudarymo problema

Pagrindinė problema sudarant optimalų, neturintį konfliktų tvarkaraštį yra susijusių išteklių priskyrimas tam tikram laikui, tenkinant griežtus ir minkštus reikalavimus.

Pateikiama keletas tvarkaraščio sudarymo uždavinių:

- Studentų auditorinio darbo ir egzaminų tvarkaraščiai. Sudaromas paskaitų ir egzaminų grafikas atitinkamoms grupėms, nurodant kiekvienai grupei studijuojamo dalyko dėstytoją, patalpą ir laiką.
- Lėktuvų / orlaivių tvarkaraščiai. Atitinkamam reisui turi būti priskirtas orlaivis, lakūnas.
- Darbo grafikų sudarymas – efektyviai sudėlioti darbo laiką turimam darbuotojų ir įrenginių kiekiui, optimizuojant resursų sąnaudas [3].
- Susitikimų planavimas. Reikia nustatyti tokį susitikimo laiką, kad visi suinteresuoti asmenys galėtų dalyvauti.

1.1. Paskaitų tvarkaraščio sudarymas

Paskaitų tvarkaraštis yra susijusių dalykų, organizuojamų tam tikru laiko tarpsniu, grafikas, laikantis reikalavimų, kad daugiau nei vienas įvykis vienu metu nereikalautų tų pačių išteklių. Paprastai švietimo įstaigoje ištekliai, kurių paskaitos metu gali prireikti, yra studentai, auditorijos ir dėstytojai.

Norint suformuoti aukštosios mokyklos tvarkaraštį, iš pradžių reikia turėti pradinių duomenų aibę. Šio konkretaus tvarkaraščių formavimo ir optimizavimo uždavinio sprendimui didelę įtaką turi pradinių duomenų aibėje aprašyti tokie euristiniai parametrai:

- Grupių dalykų skirstymas: sudaromas kiekvienos grupės dėstomų dalykų sąrašas. Aprašant dalykus buvo naudojami tokie duomenys: pavadinimas, valandų skaičius per savaitę, tipas (teorinis ar praktinis). Aprašant grupes nurodomas kursas ir grupės numeris, taip pat konkrečiai grupei priskiriami reikiami dalykai.
- Dėstytojų skirstymas: dėstytojų duomenys: vardas ir pavardė, dėstomų dalykų pavadinimai. Taip pat nurodoma pasirinktas pageidaujamas darbo laikas (nurodoma, kokios pageidaujamos darbo dienos ir laikas).

Norint suformuoti optimalų tvarkaraštį, reikia laikytis griežtų tvarkaraščio sudarymo reikalavimų. Tvarkaraščiai skirstomi taip:

- Studentų grupių užsiėmimų tvarkaraščiai: paskirstyti užsiėmimus, kai mažiausias vienetas tvarkaraštyje yra studentų grupė, turinti studijuojamų dalykų sąrašą, juos dėstančius dėstytojus, priskirti tinkančią auditoriją ir laiką.
- Auditorijų apkrovimo tvarkaraščiai: pagal sugeneruotus grupių užsiėmimo tvarkaraščius suformuoti auditorijų apkrovimo grafikus.
- Dėstytojų užimtumo tvarkaraščiai: pagal sugeneruotus grupių užsiėmimo tvarkaraščius suformuoti dėstytojų užimtumo grafikus.

1.2. Ribojimai

Jei institucija yra didelė, sukurti optimalų, visus būtinus ir pageidaujamus ribojimus atitinkantį tvarkaraštį, fiziškai yra neįmanoma. Atsiranda reikalavimai, kurie prieštarauja vieni kitiems. Paskaitos ir egzaminai, kuriems reikalingi tie patys ištekliai, turi būti sudaryti išvengiant konfliktų: vienai grupei tam tikru laiku priskiriamas tik vienas dalykas, to dalyko dėstytojas ir atitinkama auditorija. Dėstytojas vienu metu negali dirbti su keliomis studentų grupėmis ar auditorijoje vykti keletas užsiėmimų. Atsižvelgiant į apribojimų griežtumo laipsnį, apribojimai paprastai skirstomi į kietus ir minkštus. Sudarytas tvarkaraštis turi būti priimtinas tiek studentams, tiek dėstytojams.

1.2.1. Pagrindiniai būtini tvarkaraščio ribojimai

Griežti ribojimai yra pagrindinės sąlygos, kurias būtina įvykdyti, kad būtų sudarytas veikiantis grafikas. Jei kurio nors iš griežtų apribojimų negalima sėkmingai įgyvendinti, toks tvarkaraštis atmetamas. Pavyzdžiui, negalima sudėti vienai grupei, vienam dėstytojui keleto dalykų tuo pačiu metu, taip pat negalima numatyti keleto paskaitų toje pačioje auditorijoje. Žemiau pateikiamas griežtų ribojimų sąrašas:

1. Darbo dienų d skaičius per savaitę turi būti $d \leq 5$.
2. Dėstytojas negali dirbti vienu metu keliose skirtingose vietose.
3. Studentas vienu metu negali dalyvauti keliose paskaitose.
4. Ribotas paskaitų p skaičius per dieną.
5. Vienoje auditorijoje negali vykti kelios skirtingos paskaitos vienu metu.
6. Dalykai, reikalaujantys specialios įrangos, turi vykti specialiai tam skirtose patalpose (pavyzdžiui, *Kūno kultūra*, *Programavimas*).
7. Keletas grupių gali studijuoti tą patį dalyką (bendros paskaitos).

1.2.2. Nepatogumų ribojimai

Nepatogumų (minkštieji) ribojimai yra lengvatinės sąlygos, kurios yra neprivalomos. Dažniausiai sudėtinga į tvarkaraštį įtraukti visus švelnius apribojimus. Tvarkaraštis gali būti tinkamas, net jei jis neatitinka švelnių ribojimų, tačiau yra laikomasi visų griežtų ribojimų. Pavyzdžiui, dėstytojas gali pageidauti vesti paskaitas tik antroje dienos pusėje arba tik penktadieniais ir pan., dėstytojo ar studentų tvarkaraštyje esantys langai turi būti eliminuoti.

1.3. Panašių sistemų analizė

Tvarkaraščių sudarymo uždaviniai sprendžiami jau seniai, tačiau jų sprendimas yra sudėtingas, todėl geresnių sprendimų paieška aktuali ir dabar. Tvarkaraščio uždavinys priskiriamas NP (angl. *Nondeterministic Polynomial*) uždavinių klasei. Tai reiškia, jog naudojant tiesinį visų galimų kombinacijų perrinkimą, uždavinys yra praktiškai neišsprendžiamas. Kadangi galimų variantų yra labai daug, šioje srityje sėkmingai taikomi euristiniai algoritmai, kurie sprendžia tokius uždavinius nepalyginamai greičiau, bet su tam tikra paklaida.

Tvarkaraščių programos pagal savo funkcionalumą yra skirstomos į:

- automatinės – vartotojas įveda duomenis, o visą tvarkaraštį automatiškai sugeneruoja programa;
- pusiau automatinės – pagal įvestus duomenis programa sugeneruoja tvarkaraštį, tačiau vartotojui leidžiama jį koreguoti;
- pagalbinės – tvarkaraščio negeneruoja, tačiau sukuria vartotojui patogią aplinką tvarkaraščiui sudaryti.

Buvo pasirinkta peržiūrėti pagal uždavinį labiausiai atitinkančias tvarkaraščių sudarymo programas.

„aSc TimeTables“. Tai yra tvarkaraščių sudarymo programa, kuria galima greitai ir paprastai sukurti mokyklos, gimnazijos, profesinės mokyklos, licėjaus, aukštosios mokyklos ar kitos mokslo įstaigos tvarkaraščius, kurie atitiks visus organizacinius, psichologinius bei pedagoginius reikalavimus [1]. Programą sukūrė Slovakijos IT kompanija *Applied Software Consultants s.r.o*. ir teikia ją lietuviškoms mokykloms. Šią programą sėkmingai naudoja daugiau nei 7500 mokymo įstaigų visame pasaulyje. Vartotojo sąsaja yra paprasta ir lengvai suprantama.

Šioje tvarkaraščių programoje galima:

- Įvesti ir išsaugoti pagrindinius duomenis: klases, mokytojus, kabinetus, pamokas, savaitės dienas kada vyksta pamokos ar užsiėmimai, skambučių laiką ir t.t.
- Aprašyti įvairias sąlygas: laiką, kada gali vykti pamokos, mokytojų darbo laiką, sąryšius tarp skirtingų disciplinų (kaip jos turi būti paskirstytos per savaitę, kokios disciplinos negali eiti viena po kitos, kada turi vykti ir pan.), langų mokytojams laiką, pietų pertraukas, dėstomų pamokų skaičių ir daugelį kitų dalykų.
- Sukurti tvarkaraštį: programa turi galingą algoritmą, leidžiantį jai sudaryti tvarkaraštį, kurį galima redaguoti, keičiant sąlygas ar rankiniu būdu išdėstant pamokas. Jeigu pasirodo, kad tvarkaraščio sukurti neįmanoma, programa turi testavimo įrankius, leidžiančius surasti problemą, ar tai būtų įvestų duomenų klaida, ar pernelyg griežtos jai užduotos sąlygos.
- Atnaujinti tvarkaraštį per visus mokslo metus, atspausdinti kiekvienos klasės, kabineto ar mokytojo tvarkaraštį, ar bendrą klasių, mokytojų ar kabinetų tvarkaraštį.
- Pavadavimų programa leidžia parinkti pavaduojančius mokytojus, kai paskirtieji mokytojai dėl tam tikrų priežasčių negali vesti užsiėmimų. Programa sudaro suvestines apie nedalyvavusius mokytojus pasirinktam laikotarpiui, dėl kokių priežasčių jie nevedė pamokų.

Vartotojai programą verina labai gerai, galima būti tik paminėti, kad vartotojo sąsaja nėra intuityvi, tačiau šį trūkumą kompensuoja labai geras šios programos gidas. Programa taip pat neturi tvarkaraščio optimizavimo galimybės ir visiškai nevertina mokinių langų. Todėl tvarkaraštį vertinant iš individualaus mokinio pozicijų, jis nėra visiškai geras. Tvarkaraščio dizainas nėra pritaikytas veikti planšetiniuose kompiuteriuose ir išmaniuosiuose telefonuose.

***Mimosa*.** Programos pagrindinė idėja yra labai paprasta. Praktiškai neįmanoma automatiškai sukurti gero tvarkaraščio. Todėl programa to padaryti ir nesistengia – ji tik sukuria aplinką maksimaliai patogiai sudėlioti ir derinti tvarkaraštį, pateikdama efektingą ir reikalingą informaciją.

Programa leidžia:

- klases skirstyti į kiek norimą grupių skaičių (ribiniu atveju grupių bus tiek, kiek klasėje yra mokinių);
- sujungti kelių klasių ar jų grupių mokinius per atskirų dalykų pamokas;
- tos pačios klasės mokiniams vienu metu sudėti kelias skirtingas pamokas ar priskirti kelis mokytojus dėstyti tą patį;
- sudaryti atskirus pamokų tvarkaraščius kiekvienam mokiniui;
- nustatyti tvarkaraščio apribojimus (laisvas pamokas ar dienas) mokytojams ir mokiniams.
- nurodyti užsiėmimų svarbą (privalomas ar pasirenkamas dalykas, maža ar didelė mokinių grupė jį pasirinko);
- atspausdinti individualius pamokų tvarkaraščius mokytojams, mokiniams ar klasėms;
- atspausdinti kabinetų užimtumo tvarkaraščius ir padaryti internetinę tvarkaraščių versiją.

Programa *Mimosa* yra licencijuota, tačiau visos Lietuvos mokyklos šios programos licenciją gali gauti nemokamai. Daug laiko ir papildomo darbo reikalauja ne tik duomenų suvedimas į programą, bet ir programos sudaryto tvarkaraščio koregavimas. Automatiškai sudarytas tvarkaraštis ne tik, kad nėra geras, bet ir netenkina visų jam keliamų reikalavimų, dažnai palieka daug langų ir mokiniams, ir mokytojams. Lyginant su kitomis programomis, *Mimosa* sistema sudėtinga naudotis. Ji priskiriama prie pagalbinių produktų pogrupio.

FET. Nemokama tvarkaraščių sudarymo programa, suderinama su *XML* duomenų standartu, palaiko *CSV* bylų importavimą bei eksportavimą, tvarkaraščius galima pateikti *HTML*, *XML* ir *CSV* rinkmenose, galimas visiškai automatinis, pusiau automatinis ar rankinio paskirstymo generavimo algoritmas. Programa lokalizuota daugelyje kalbų ir yra realizuotas nepriklausomas platformos įgyvendinimas.

Programa turi didelę lanksčią paletę laiko apribojimams:

- minimalus ir maksimalus užsiėmimų skaičius per savaitę,
- mokytojams tinkančių dienų parinkimas,
- maksimalus akademinių valandų per dieną nustatymas,
- negalimi pamokoms laiko intervalai,
- maksimalus pastatų pasikeitimas per dieną,
- moksleivių dienos pabaiga ir dar daug kitų galimybių [4].

Programa leidžia įgyvendinti vietos apribojimus:

- Riboti pagrindines erdves;
- Apriboti užimtas klases/auditorijas;
- Apriboti užsiėmimo pageidaujamą klasę/auditoriją;
- Riboti dalyko pageidaujamą klasę/auditoriją;
- Apriboti keičiamus pastatus studentams ir mokytojams (minimalus tarpas tarp pakeistų pastatų, maksimalus skaičius pastatų pakeitimo per dieną, maksimalus skaičius pastatų pakeitimo per savaitę).

Literatūroje sunku rasti šios programos trūkumų, kadangi visi naudotojai ją vertina labai gerai. Galima paminėti tik vieną trūkumą: įvedus daug pasirinkimo kriterijų, galutinis tvarkaraščio sugeneravimas užtrunka, todėl programa imli laiko resursams.

Prime Timetable. Internetinė priemonė Prime Timetable [12] yra skirta mokymo įstaigų (bendrojo lavinimo mokyklų, aukštųjų mokyklų, neformalaus švietimo įstaigų) tvarkaraščių sudarymui. Internetinė priemonė *Prime Timetable* turi daug tvarkaraščiui sudaryti reikalingų funkcijų [15]:

- Duomenų suvedimo ir išsaugojimo;
- Automatinio tvarkaraščio sudarymo pagal įvestus duomenis;
- Tvarkaraščio koregavimo rankiniu būdu;
- Duomenų, kurie rodomi tvarkaraštyje, pasirinkimo;
- Tvarkaraščio kūrimo ir koregavimo galimybės turėjimas bet kuriame įrenginyje;
- Tvarkaraščio išsaugojimo įvairiais formatais;
- Tvarkaraščio spausdinimo;
- Tvarkaraščio peržiūros internete.

Tvarkaraščio kūrimo priemonės *Prime Timetable* naudojimas paprastas ir intuityvus. Tinkamai suvedus pradinius duomenis tvarkaraštis automatiškai suformuojamas. Sugeneruotame tvarkaraštyje galima pasirinkti, kokia informacija bus rodoma, perkelti pamokas iš vienos vietos į kitą. Vienas naudojimo trūkumų yra tas, kad dizainas nėra pritaikytas veikti planšetiniuose kompiuteriuose ir išmaniuosiuose telefonuose.

Apibendrinta šių automatinių tvarkaraščių generavimo priemonių palyginimo analizė pateikiama lentelėje:

1 lentelė. Sistemų palyginimas

Pavadinimas	Duomenų įvedimas	Sąlygų aprašymas	Tvarkaraščio spausdinimas	Koregavimas	Išvada
aSc TimeTables	taip	nustatomas laikas, kada gali vykti paskaitos, mokytojų laikas, sąryšiai tarp disciplinų ir kt.	individualūs pamokų tvarkaraščiai mokytojams, mokiniams, klasėms	taip	Sudaromas geras tvarkaraštis, tačiau neturi tvarkaraščio optimizavimo galimybės. Programa visiškai nevertina mokinių langų.
Mimosa	taip	laisvos pamokos ar dienos) mokytojams ir mokiniams, užsiėmimų svarba	individualūs pamokų tvarkaraščiai mokytojams, mokiniams ar klasėms	taip	Programa sugeneruoja tvarkaraštį, kuriame trūksta pamokų. Toks tvarkaraštis neatitinka būtinųjų reikalavimų, todėl mokykloje pagal jį dirbti būtų negalima. Siekiant padaryti, kad tvarkaraštis tenkintų visus būtinuosius ribojimus tvarkaraštis gali būti koreguojamas rankiniu būdu. Šiam darbui atlikti,

					tvarkaraštį sudarantis asmuo turi skirti daug laiko.
FET	taip	Platus spektras laiko, vietos apribojimų	taip	taip	Programa lanksti, tinka mokykloms ir universitetams [14], tačiau Lietuvoje nėra plačiai naudojama
Prime Timetable	taip	trūksta pamokų tvarkos apribojimų	taip	taip	Ši programa sugeneruoja pakankamai gerą tvarkaraštį – nepalieka laisvų pamokų mokiniams, tačiau neleidžia įvesti pamokų tvarkos apribojimų, kurie reikalingi siekiant padaryti tvarkaraštį priimtinesnį.

Apžvelgus šias pačias populiariausias tvarkaraščio sudarymo programines priemones, galime įsitikinti, kad jas naudoti yra problematiška, nes:

- Sudėtinga įvedimo ir tvarkarščio pateikimo sąsajos;
- Nėra optimalu naudoti universalią sistemą, nes ji reikalauja žymiai didesnių resursų; didžioji dalis jų teikiamų galimybių gali būti nepanaudota dėl jų sudėtingumo.

Tokiu būdu tikslinga sukurti sistemą, kurioje būtų galima neperkraunant papildomomis funkcijomis realizuoti norimas kompiuterizuoti funkcijas, o taip pat įgyvendinti ir specifinius mokymo įstaigos poreikius, kurių gali neturėti universalios sistemos. Šiuo darbu sukurta sistema yra nesudėtinga, lengvai valdoma, greitai ir efektyviai pateikianti tvarkaraštį, atitinkantį pagrindinius mokymo įstaigos reikalavimus.

2. Grafų teorijos elementai

Grafų teorija – matematikos sritis, nagrinėjanti grafus. Vienas pirmųjų matematikų, galvojęs apie grafus ir tinklus, buvo vienas didžiausių matematikų istorijoje Leonhardas Euleris (1707–1783). Euleris išrado didžiąją dalį šiuolaikinės matematinės terminijos ir žymėjimo bei padarė svarbių atradimų skaičiavimo, analizės, grafų teorijos, fizikos, astronomijos ir daugelyje kitų temų. Pirmasis grafų teorijos uždavinys, nagrinėtas 1736 m., buvo uždavinys apie Karaliaučiaus tiltus. Maždaug 100 metų laikotarpyje šio uždavinio sprendinys buvo vienintelis grafų teorijos rezultatas. 1852 m. A. De Morganas iškėlė keturių spalvų hipotezę, kuri grafų teorijos vystymesi suvaidino analogišką vaidmenį, kaip didžioji *Ferma* teorema skaičių teorijoje.

Laikoma, kad grafų teorija, kaip savarankiška matematikos šaka, atsirado 1936 m. , kai matematikas D. Kionigas išleido monografiją "Baigtinių ir begalinių grafų teorija". Jis pirmasis vietoje įvairiuose moksluose naudojamų skirtingų schemų pavadinimų: sociogramos (psichologija), simpleksai (topologija), grandinės (fizika), diagramos (ekonomika), ryšių tinklai, vandentiekio tinklai, geneologiniai medžiai ir t. t., pasiūlė naudoti terminą – "grafas". Taigi grafai yra įvairiausios fizinės prigimties reiškinių matematiniai modeliai. Tai ir lemia grafų teorijos, kaip savarankiškos matematikos šakos, audringą vystymąsi ir plačias taikymo galimybes.

2.1. Pagrindinės sąvokos ir apibrėžimai.

1 apibrėžimas. Orientuotu grafu G vadiname porą *(V;E),* kur V yra baigtinė aibė $V=\{v_1;v_2;\dots;v_n\}$, o $E \subset V \times V$ yra aibė sudaryta iš V elementų porų. Aibė *V* vadinama grafo ***viršūnių aibe***. Aibės *V* elementų skaičius yra lygus grafo viršūnių skaičiui ir dažnai vadinamas grafo ***eile***: $|V|=n$.

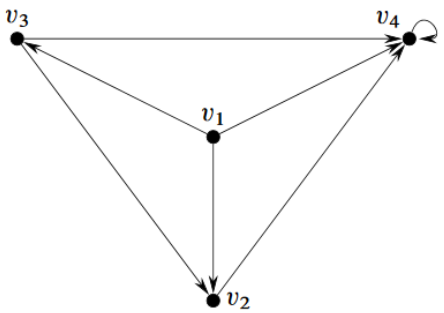
Orientuotą grafą $G = (V,E)$ mes galime grafiškai pavaizduoti plokštumoje atidėję $n = |V|$ taškų ir kiekvienai porai $(v_i,v_j) \in E$ priskyrę rodyklę einančią iš taško v_i į tašką v_j.

Kompiuterio atmintyje grafas $G = (V, E)$ gali būti saugomas kaip matrica $M =(a_{ij})$, kur

$$a_{ij} = \begin{cases} 1, & jeigu\ (v_i, v_j) \in E \\ 0, & jeigu\ (v_i, v_j) \notin E \end{cases}$$

Taip sukonstruota matrica *M* yra vadinama grafo *G* gretimumo matrica [13].

Orientuoto grafo pavyzdys. Tarkime G = (V, E), kur $V\{v_1, v_2, v_3, v_4\}$ ir $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_3, v_2), (v_3, v_4), (v_2, v_4), (v_4, v_4)\}$. Tuomet tokį grafą galima pavaizduoti taip:



$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

1 pav. Orientuoto grafo pavyzdys

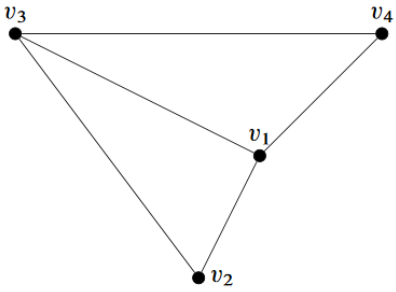
Briaunos (v_j,v_j) kurių pradžia ir galas sutampa yra vadinamos kilpomis.

2 apibrėžimas. Neorientuotu grafu vadinama pora (V;E), kur E yra nesutvarkytų porų (v_i,v_j) aibė. Kitaip tariant neorientuoto grafo atveju mes nelaikome, kad poromis (v_i,v_j) ir (v_j,v_i) nusakomos briaunos yra skirtingos.

Aibė *U* neorientuotojo grafo atveju vadinama grafo ***briaunų aibe***, o orientuotojo grafo (orgrafo) atveju – ***lankų aibe***. Tuo būdu viršūnių v_1 ir v_2 pora (v_1,v_2) sudaro arba briauną arba lanką $|U|=m$.

Neorientuotus grafus be kilpų kompiuterio atmintyje galima išsaugoti kaip simetriškas matricas, kurių įstrižainėje yra nuliai.

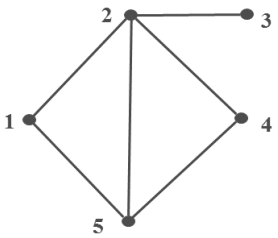
Neorientuoto grafo pavyzdys. Tarkime G= (V;E), kur V={v_1, v_2, v_3, v_4} ir E = $\{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_3, v_2), (v_3, v_4)\}$. Tuomet tokį grafą galima pavaizduoti taip:



$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

2 pav. Neorientuoto grafo pavyzdys

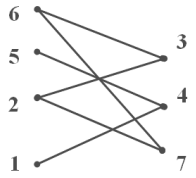
Viršūnės v laipsnis, tai skaičius viršūnių gretimų viršūnei ***v***. Viršūnės laipsnį žymėsime ***ρ(v)*** arba ***d(v).*** Aibė viršūnių, gretimų viršūnei *v*, žymima ***N(v)*** ir vadinama ***viršūnės aplinka***. Aišku, kad $\rho(v)=|N(v)|$.



3 pav. Grafo pavyzdys

Pagal paveikslėlyje pateiktą grafą galime suskaičiuoti kiekvienos viršūnės laipsnį: ρ(1)=2 ; ρ(2)=4; ρ(3)=1; ρ(4)=2; ρ(5)=3.

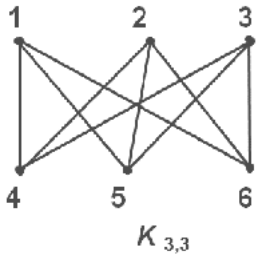
Dvidalis grafas. Grafas, kurio viršūnių aibę galima išskaidyti į du poaibius *A* ir *B* taip, kad kiekvienos briaunos galai priklausytų skirtingiems poaibiams, vadinamas *dvidaliu* grafu.



4 pav. Dualusis grafas

A={1,2,5,6}, B={3,4,7}, $V = A \cup B$, $A \cap B = Ø$

Simboliu $K_{s,t}$ žymime ***pilnąjį dvidalį*** grafą, t. y. grafą, kai kiekviena aibės A viršūnė jungiama briauna su visomis aibės B viršūnėmis.



5 pav. Pilnasis dvidalis grafas $K_{s,t}$

$V = A \cup B$, A = {1, 2, 3}, B = {4, 5, 6}.

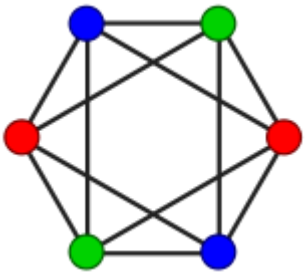
2.2. Spalvų grafas

Grafo spalvinimas arba grafo dažymas – grupė grafų teorijos uždavinių, kuriuose siekiama grafo elementams priskirti spalvas taip, kad būtų tenkinamos tam tikros sąlygos (paprastai – kad gretimi grafo elementai turėtų skirtingas spalvas). Iš tokių uždavinių dažniausiai naudojamas viršūnių spalvinimas, kai kiekvienai viršūnei priskiriama spalva taip, kad gretimos viršūnės turėtų skirtingas spalvas, kiek rečiau – briaunų spalvinimas, kai spalvos priskiriamos briaunoms.

Viršūnių spalvinimas – grafo spalvinimo variantas, kai grafo viršūnėms priskiriamos spalvos taip, kad gretimos viršūnės turėtų skirtingas spalvas. Dvi viršūnės yra gretimos, jeigu jos yra sujungtos briaunomis, o dvi briaunos yra gretimos, jei abi jos turi bendrą viršūnę. Taigi, viršūnių spalvinimas yra ne kas kita, kaip tai, jog dvi gretimos viršūnės nėra tokios pat spalvos.

3 apibrėžimas. Tarkime, kad turime grafą $G = (V, E)$ ir spalvų rinkinį C. Tuomet grafo nuspalvinimu yra vadinamas toks atvaizdis $f : V(G) \rightarrow C$, kad kiekvienai grafo briaunai $(v_i, v_j) \in E(G)$ teisinga nelygybė $f(v_i) \neq f(v_j)$. Kartais toks spalvinimas vadinamas korektišku.

Mažiausias skaičius spalvų, kuriomis galima nudažyti grafo viršūnes, vadinamas grafo **chromatiniu skaičiumi**. Bendru atveju patikrinti, ar n viršūnių turintis grafas gali būti nudažytas k spalvų galima peržiūrint visus derinius iš n elementų po k (kiekvienai viršūnei priskiriama viena iš k spalvų). Ieškant chromatinio skaičiaus tokius patikrinimus reikia atlikti k reikšmėms nuo 1 iki n, tad algoritmo laikinis sudėtingumas yra $O((n+1)!)$. Kadangi tikslus algoritmas yra lėtas, paprastai naudojami euristiniai algoritmai.



6 pav. Grafo viršūnių spalvinimo pavyzdys

2.2.1 Godusis algoritmas grafo viršūnių spalvinimui

Godusis algoritmas – tai euristinių algoritmų klasei priskiriamas metodas. Šis algoritmas paprastai yra efektyvus, tačiau nebūtinai gaunamas globalusis optimumas. Atsižvelgiant į godžiojo algoritmo savybes, pastarasis tinkamas grafo viršūnių spalvinimo problemai spręsti. Godžiojo dažymo metodo atveju spalvos yra sunumeruojamos vis didėjančia pirmenybės tvarka ir kiekvieną kartą, kai pasiekiama nauja viršūnė, jos nuspalvinamos mažiausio skaičiaus spalva, kad būtų patenkinti apribojimai [10]. Algoritmą sudaro du etapai:

1. Pirmajai viršūnei priskiriama pirma spalva.
2. Likusioms viršūnėms V–1:
    - Iš eilės priskiriama galima spalva mažiausiu indeksu, jeigu viršūnės v kaimyninės viršūnės tokios spalvos neturi, priešingu atveju parenkama nauja spalva.

Literatūroje minimos kelios godžiojo metodo modifikacijos, kurios skirtos grafo dažymo problemai nagrinėti: atsitiktinė eilė, išrikiavimas pagal didžiausią laipsnį, paskutinė – mažiausia.

Atsitiktinė eilė

1. Grafo viršūnės yra išrikiuojamos atsitiktine tvarka.

2. Taikomas godusis algoritmas.

Išrikiavimas pagal didžiausią laipsnį (angl. *Largest Degree Ordering*)

Viršūnės su mažesniu laipsniu dažniausiai turi didesnį galimų spalvų pasirinkimą.

1. Grafo viršūnės yra išrikiuojamos laipsnių mažėjimo tvarka.

2. Taikomas godusis algoritmas.

Mažiausia – paskutinė

h – laipsnis: šiai viršūnei negalimų spalvų skaičius;

k – laipsnis: viršūnei gretimų nenudažytų viršūnių skaičius.

Rūšiavimas: pirmenybė teikiama viršūnei su didžiausiu *h* laipsniu. Jei kelių viršūnių *h*

laipsniai sutampa tuomet imama viršūnė, kurios *k* laipsnis didžiausias (jei ir šie laipsniai sutampa imama viršūnė, kurios numeris mažiausias).

1. Grafo viršūnės yra išrikiuojamos;

2. Taikomas godusis algoritmas;

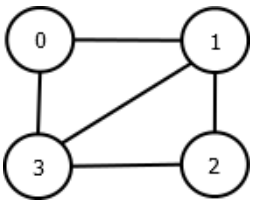
3. Po kiekvieno žingsnio perskaičiuojami *h* ir *k* laipsniai.

2.2.2 Grįžimo algoritmas grafo viršūnių spalvinimui

Šis algoritmas pagrįstas grįžimo metodu (angl. *backtracking*). **Grįžimo metodas** – tai sistemingas būdas spręsti uždaviniams, kurių sprendinys yra kintamųjų $p_1, p_2, \ldots, p_n$ reikšmių rinkinys, tenkinantis kokius nors reikalavimus. Pagrindinė grįžimo metodo idėja tokia: paeiliui renkamos visų galimų kintamųjų reikšmės ir tikrinama, ar tenkinami reikalavimai, o radus sprendinį arba situaciją, kai reikalavimai netenkinami, grįžtama per vieną žingsnį atgal ir parenkama nauja atitinkamo kintamojo reikšmė. Programuojant grįžimo metodą dažnai naudojama rekursija.

Norint nuspalvinti grafą *G(V,E)* naudojant grįžimo algoritmą, reikia nusakyti norimą spalvų skaičių *m* ir iš grafo viršūnių sudaryti gretimumo matricą *M[L][L]*, kur *L* yra viršūnių skaičius grafe. *M[i][j]* reikšmė yra 1, jei $(v_i,v_j) \in E$ t .y. tos viršūnės yra sujungtos briauna, kitu atveju *M[i][j]* vertė yra 0.

Grafo ir gretimumo matricos pavyzdys:



7 pav. Grafo pavyzdys, n=4, m=3

n	0	1	2	3
0	1	1	0	1
1	1	1	1	1
2	0	1	1	1
3	1	1	1	1

8 pav. Gretimumo matrica M

Grįžimo algoritmą grafo viršūnių nuspalvinimui sudaro šie etapai:

1. Pasirenkama grafo viršūnė *v* (pirmoje algoritmoje iteracijoje *v=0*)
2. Viršūnei bandoma priskirti po vieną spalvą $c_{mi} \in [0, m]$, kur *m* – norimų spalvų skaičius.
   - Prieš priskiriant spalvą, grafo gretimumo matricoje *M[L][L]* patikrinama, ar galima norimą spalvą priskirti viršūnei t.y. patikrina, ar briauna egzistuoja *M[v][i] = 1* ir ar sprendiniuose *spalvos[i]* jau priskirta spalva c_{mi}, čia $i \in [0, L]$, *L* – matricos ilgis.
     - Jeigu spalvą priskirti galima, ji priskiriama *spalvos[i]* = c_{mi}. Imama viršūnė *v+1* ir algoritmas kartojamas nuo pirmo žingsnio.
     - Jei viršūnei *v+1* nepavyksta priskirti spalvos rekursyviai grįžtama atgal (angl.

backtracking), nuimama spalva nuo *v* viršūnės ir toliau bandomos kitos galimos spalvų kombinacijos. Jeigu nei viena kombinacija netenkina grafo, algoritmas sustabdomas ir grąžinama reikšmė *false*, kuri reiškia, kad grafo *G(V,E)* negalima nuspalvinti su *m* spalvų.

2.3. Spalvų grafo pritaikymas realiems uždaviniams spręsti

Daugelis optimizavimo uždavinių yra susijusių su situacijomis, kai tam tikri įvykiai negali įvykti tuo pačiu metu arba kai kai kurie objektų aibės nariai negali būti greta. Pavyzdžiui, bandant sudaryti susitikimų tvarkaraščius planuotojai susiduria su sunkumais patenkinant visų asmenų suvaržymus ir pan. Šiems ir panašiems uždaviniams spręsti dažnai pasitelkiami viršūnių spalvojimo grafai, kurie padeda spręsti daugelį uždavinių. Šiame skyrelyje pateikiama keletas realių uždavinių.

Septynių Königsbergo tiltų problema. Dar 1735 m. Eulerį suintrigavo sena Karaliaučiaus miesto prie Baltijos jūros problema. Pregelio upė dalija Karaliaučių į keturias atskiras dalis, kurias jungia septyni tiltai. Problema formuluojama taip: ar galima vaikščioti po miestą kertant visus tiltus vieną kartą, bet ne daugiau kaip vieną kartą?

Žemėlapio spalvinimo problema. Turime nuspalvinti žemėlapį taip, kad dvi turinčios bendrą sieną valstybės būtų nudažytos skirtingomis spalvomis. Šį uždavinį galima konvertuoti į dvigubą grafą, kuriame kiekvienas regionas yra viršūnė, o du regionai sujungiami briauna, kai šalys turi bendrą sieną. Šiuo atveju turime plokštųjį grafą, kas palengvina spalvinimą.

Radijo dažnio paskirstymas. Turime n radijo, televizijos ar kt. stočių transliuojančių radijo signalus. Jeigu dvi stotys yra pakankamai arti viena kitos jos transliuodamos tuo pačiu dažniu gali viena kitai trukdyti. Uždavinio tikslas – parinkti kiekvienai stočiai transliavimo dažnį taip, kad stotys esančios arti viena kitos transliuotų skirtingais dažniais. Be to, norima panaudoti kuo mažiau skirtingų dažnių. Ši problema taip pat yra grafo spalvinimo problemos pavyzdys, kai kiekviena stotis žymi viršūnę, o riba tarp dviejų stočių rodo, ar jie yra vienas kito diapazone.

Užduočių planavimas. Užduotys laikomos grafo viršūnėmis ir jei tarp dviejų užduočių yra briauna, jos negali vykdomas vienu metu. Skirtingi darbuotojai gali atlikti skirtingas užduotis, galima taikyti spalvų grafą.

Sudoku. Sudoku (9x9) gali būti laikoma grafu, turinčiu 81 viršūnę, po vieną kiekvienai ląstelei, o dvi viršūnės sujungtos briauna, jei joms negalima priskirti tos pačios vertės. Pavyzdžiui, visi tos pačios eilutės, stulpelio ar bloko langeliai turi briaunas tarp atitinkamų viršūnių. Galvosūkis yra išsprendžiamas, jeigu galime nuspalvinti šį grafą 1 – 9 spalvomis.

Registrų paskirstymas. Daugelyje programavimo kalbų programuotojas gali naudoti daugybę kintamųjų. Kompiuteris gali greitai nuskaityti ir įrašyti registrus procesoriuje, todėl kompiuterio programa veikia greičiau, kai CPU registruose gali būti daugiau kintamųjų. Tačiau registrų skaičius yra ribotas. Kompiliatorius turi nuspręsti, kaip paskirstyti kintamuosius ribotam CPU registrų skaičiui. Ši problema taip pat yra grafo spalvinimo problema. [2]

2.4. Grafo viršūnių spalvinimo algoritmų pritaikymas tvarkaraščio sudarymui

Daugelis mokslininkų grafų viršūnių spalvinimo algoritmus taiko efektyviam paskaitų, egzaminų tvarkaraščio sudarymui ([3], [5], [6],[7]). Norint išspręsti paskaitų tvarkaraščio uždavinį naudojant grafo viršūnių spalvinimo algoritmą, pirmiausia reikia suformuluoti problemą grafo pavidalu, kur dalykai laikomi viršūnėmis – mazgais. Pasirinkus konfliktišką grafą, brėžiamos briaunos tarp konfliktuojančių dalykų, pavyzdžiui, jeigu juose studijuoja tie patys studentai arba dėsto tie patys dėstytojai. Paskaitos negali konfliktuoti dėl naudojamų bendrų išteklių: studentų, dėstytojų, auditorijų. Kursai taip pat gali konfliktuoti dėl bendrų dėstytojų, auditorijų ar studentų. Tokiais atvejais konfliktų diagramoje turi būti atsižvelgiama į studentų, dėstytojų ir auditorijų konfliktus vienu metu. Pasirinkus viršūnę – dalyką, kiti veiksniai, tokie kaip atitinkama grupė, dėstytojas veikia kaip papildomi duomenys, reikalingi sudarant tvarkaraštį.

Toliau šiame darbe pateikiama keletą tvarkaraščių formavimo pavyzdžių su konkrečiais duomenimis.

2.4.1. Grafo sudarymas tvarkaraščiui formuoti

Formuojant grafą reikalingas dėstomų disciplinų sąrašas, iš kurių sudaromas konfliktinis grafas *G(V, E)*, V – įvesti dalykai grafe vaizduojami kaip viršūnės. E – ryšių tarp viršūnių porų aibė, kuri sudaroma atsižvelgiant į nurodytus parametrus:

- Vienu metu viena studentų grupė gali mokytis tik vieną dalyką.
- Vienas dėstytojas tuo pačiu dėsto tik vieną paskaitą.

Įvesties duomenų rinkinys: dėstomų dalykų sąrašas: *{D1, D2, D3, D4, D5, D6, D7, D8};* dėstytojų sąrašas: *{T1, T2, T3, T4, T5};* grupių sąrašas: *{G1, G2}*. 2 lentelėje pavaizduota dėstytojų ir dalykų pasiskirstymo matrica. Pavyzdžiui, dėstytojas T1 dėsto D1 ir D6 dalykus.

2 lentelė Dėstytojų ir dalykų paskirstymų matrica

Periodas	D1	D2	D3	D4	D5	D6	D7	D8
T1	x					x		
T2		x					x	
T3			x		x			
T4				x				
T5								x

Dėstytojo – dalyko problema. Problemos apibūdinimas: tam tikram *D* dalykų skaičiui priskiriamas *T* dėstytojų skaičius, ir *P* laiko vienetų skaičiui reikia sudaryti tvarkaraštį grupėms *G*.

Šiai problemai išspręsti naudojamas dvidalis grafas, kuris veikia kaip konfliktinis grafas. Dvi atskiros savarankiškos aibės yra dėstytojai ir dalykai. 3 lentelėje pavaizduoti įvesties duomenys, kur matome, jog 1 grupė (1G) studijuos D1, D2, D3, D4 ir D5 dalykus, o antra (2G) – D5, D6, D7 ir D8 studijų dalykus. Prie kiekvieno dalyko priskiriamas ir jį dėstantis dėstytojas.

3 lentelė. Įvesties duomenys: grupės, dalykai ir dėstytojai.

1G		2G	
D1	T1	D6	T1
D2	T2	D7	T2
D3	T3	D8	T5
D4	T4	D5	T3
D5	T3		

Atsižvelgiant į 3 lentelės duomenis, galima surasti konfliktiškas kraštines. Pavyzdžiui, *1G* studijuoja dalykus *D1, D2, D3, D4, D5,* o dėstytojas *T1* dėsto *D1, D6* dalykus, todėl visi šie dalykai negali vykti vienu metu, ir grafe bus sujungti briaunomis.

Tokiu principu grafui *G(V,E)* galime surasti visas konfliktuojančias viršūnes jungiančias briaunas ir sudaryti gretimumo matricą:

E = [{D1, D2},{D1, D3}, {D1, D4}, {D1, D5}, {D1, D6},{D2, D3}, {D2, D4},{D2, D5},{D2,D7}, {D3, D4}, {D3, D5},{D4, D5}, {D5, D6}{D5, D7},{D5, D8},{D6, D7}, {D6, D8}, {D7, D8}].

4 lentelė. Grafo *G(V, E)* gretimumo matrica

	D1	D2	D3	D4	D5	D6	D7	D8	Viršūnės laipsnis
D1	0	1	1	1	1	1	0	0	5
D2	1	0	1	1	1	0	1	0	5
D3	1	1	0	1	1	0	0	0	4
D4	1	1	1	0	1	0	0	0	4
D5	1	1	1	1	0	1	1	1	7
D6	1	0	0	0	1	0	1	1	4
D7	0	1	0	0	1	1	0	1	4
D8	0	0	0	0	1	1	1	0	3

2.4.2. Sudaryto grafo spalvojimo realizacija

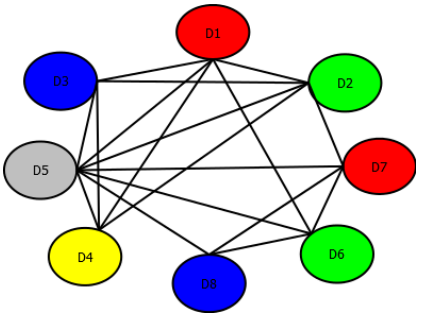
Godžiojo algoritmo realizacija. Suradus visas grafo *G(V,E)* briaunas, galima taikyti spalvojimo algoritmus. Pritaikius godžiojo algoritmo modifikaciją – išrikiavimą pagal didžiausią laipsnį mažėjimo tvarka (angl. *Largest Degree Ordering*), gaunami rezultatai pateikti 5 lentelėje. Matome, kad daugiausiai konfliktų turinčiai viršūnei D5 buvo priskirta pirmoji spalva.

5 lentelė. Grafo viršūnių – dalykų nuspalvinimo rezultatai (duomenys surikiuoti pagal didžiausią laipsnį mažėjimo tvarka)

Spalva 0 (pilka)	Spalva 1 (raudona)	Spalva 2 (žalia)	Spalva 3 (mėlyna)	Spalva 4 (geltona)
D5	D1 D7	D2 D6	D3 D8	D4

9 paveikslėlyje pavaizduotas spalvų grafas, kuris gaunamas įvedus studijų dalykus ir visus

apribojimus, kad būtų išvengta konfliktų. Iš paveikslėlio matyti, kad D5 negali vykti nei su vienu dalyku (šis dalykas bendras abiems grupėms), D1, D2 dalykai negali vykti kartu su kitais 5 dalykais, D3, D4, D6, D7 dalykai negali vykti kartu su kitais keturiais dalykais (t. y. tos pačios grupės dalykai arba tas pats dėstytojas), D8 – su trimis kitais dalykais.



9 pav. Spalvotas viršūnių grafas, kur viršūnės – studijų dalykai

Sėkmingai nuspalvinus grafo viršūnes, tampa nesunku grafo viršūnėms priskirti konkrečius tvarkaraščio laiko vienetus. Vienai spalvai priskiriamas tas pats laiko vienetas (žr. 6 lentelę).

6 lentelė. Dėstytojų ir dalykų paskirstymas tvarkaraštyje – godžiojo algoritmo realizacija

1 tvarkaraščio laiko vienetas	2 tvarkaraščio laiko vienetas	3 tvarkaraščio laiko vienetas	4 tvarkaraščio laiko vienetas	5 tvarkaraščio laiko vienetas
D5–T3	D1–T1 D7–T2	D2–T2 D6–T1	D3–T3 D8–T5	D4–T4

Godžiojo algoritmo įvykdymo charakteristika:

1) Nė viename stulpelyje nėra tų pačių dalykų ir atitinkamu laiku dėsto tik vienas dėstytojas.

2) Tvarkaraštis sudaromas dviem grupėms, todėl kiekviename stulpelyje yra tik 2 dalykai – dviem skirtingoms grupėms t. y. vienu laikotarpiu reikia daugiausiai 2 auditorijų. 5 dalykas yra dėstomas abiems grupėms, todėl jam reikia atskiro laiko, kad abi grupės būtų laisvos, 4 dalykas lieka vienas, kadangi jam nebėra poros. Abi grupės turi ne po vienodą paskaitų skaičių.

Grįžimo algoritmo realizacija. Sprendžiant neorientuoto grafo uždavinį grįžimo spalvų grafo algoritmu, galima taip pat spręsti tvarkaraščio formavimo uždavinį. Naudojami tie patys duomenys, kaip ir sprendžiant uždavinį godžiojo algoritmu. Šiame uždavinyje viršūnių skaičius – 8. Uždavinio tikslas yra, kad nė viena gretima grafo viršūnė nebūtų nudažyta ta pačia spalva. Įvedamas skaičius m – viršūnių skaičius, kuris yra maksimalus spalvų skaičius ir grafo *G(V,E)* gretimumo matrica, surikiuota pagal didžiausią laipsnį mažėjimo tvarka (angl. *Largest Degree Ordering*).

Pritaikius kitą spalvoto grafo algoritmą, gautas toks pat rezultatas (spalvas sudaro tokie patys dalykų rinkiniai):

7 lentelė. Grįžimo algoritmo realizacija (duomenys surikiuoti pagal didžiausią laipsnį mažėjimo tvarka)

Spalva 0 (pilka)	Spalva 1 (raudona)	Spalva 2 (žalia)	Spalva 3 (mėlyna)	Spalva 4 (geltona)
D5	D1 D7	D2 D6	D3 D8	D4

Išnagrinėtas pavyzdys, kurio duomenys buvo dviejų grupių studijuojami 8 dalykai. Pateikus abiems algoritmams surikiuotą tą patį grafą *G(V,E)*, viršūnės nuspalvinamos ta pačia spalva. Taigi, abiejų algoritmų rezultatas priklauso nuo tvarkos, pagal kurią viršūnės perduodamos spalvinimui.

3. Tvarkaraščio sudarymo realizavimas žiniatinklio terpėje

Įgyvendinant pasirinktą sistemą buvo pasirinkta *Java* programavimo kalba ir *Spring Boot karkasas* (angl. *Framework*), operacijoms duomenų bazės lygyje *Hibernate/JPA*, grafinei vartotojo sąsajai sukurti buvo pasirinkta *ReactJS* biblioteka. Klientinei programai reikalingas būdas gauti duomenis iš serverio ir juos dinamiškai atvaizduoti. Interneto paslaugų konstrukcija leidžia įvairių aplinkų sistemoms sąveikauti per tinklą, t. y. interneto paslaugos leidžia skirtingoms programoms bendrauti tarpusavyje, net jeigu jos parašytos skirtingomis programavimo kalbomis ar veikia ant skirtingų platformų. Bendravimui tarp vartotojo sąsajos ir vidinio kodo (angl. *Back–end*) naudotos *Java API REST* internetinės paslaugos, kurių pagalba duomenys pasiekiami per URL (angl. *Uniform Resource Locator*).

3.1. Techninės sistemos charakteristikos

Grafinė vartotojo sąsaja – kliento dalis (angl. *front–end*) buvo kurta *JavaScript*, pasitelkiant UI komponentų kūrimo biblioteką *ReactJS*, *Bootstrap* 4 karkasą. Serverio dalis (angl. *back–end*) buvo programuojama *Java* programavimo kalbos pagalba redaktoriumi *IntelliJ IDEA*, naudojant *Spring Boot* karkasą.

Spring Boot yra *Java* programavimo kalbai sukurtas karkasas, kuris gimė iš panašaus pavadinimo *Spring* karkaso. *Spring Boot* buvo sukurtas tam, kad palengvintų programuotojams naujo projekto kūrimo ir konfigūravimo iššūkius. Sugeneravus *Spring Boot* naują projektą, jį parsisiuntus iš oficialios svetainės, jis jau turi *Tomcat* serverį ir testavimui parengtą duomenų bazę, taip pat nurodžius, kokias priklausomybes (angl. *Dependency*) nori naudoti, galima iš karto paleisti programą, ir beliks tik programuoti. Be greitos ir lengvos pradžios *Spring Boot* turi ir daugiau privalumų:

- Priklausomybių įskiepiai (angl. *Dependency injection*), leidžia rašyti lengvai skaitomą ir testuojamą kodą.
- Paprastas duomenų bazės transakcijų valdymas.
- Lengva integracija su kitais karkasais pvz. *JPA/Hibernate.*

REST (angl. *Representational State Transfer*) yra alternatyvus architektūrinis stilius, skirtas interneto paslaugoms kurti gaunant ir manipuliuojant ištekliais su nustatytomis standartinėmis operacijomis, tokiomis kaip GET, POST, PUT ir DELETE. REST interneto paslaugos klientas tiesiog prisijungia prie interneto paslaugos per URI (angl. Uniform Resource Identifier) ir interneto paslauga grąžina reikiamus išteklius. Grąžinti ištekliai gali būti paprastas tekstas, HTML, XML ir JSON dokumentas. *REST* paslaugos yra nesudėtingos ir nesilaiko jokių standartų, kas gali būti laikoma trūkumais. Kaip teigia K. Stankevičius, REST trūkumai yra šie: nėra nustatytų specifikacijų, nėra priimto atvaizdo šablono kaip WSDL (angl. *Web Service Description Language*) [11]. REST nereikalauja jokių specializuotų priemonių, išskyrus įprastas, kurios leidžia naudotis tinklo ištekliais per HTTP ir dirbti su XML dokumentais.

ReactJS biblioteka šiuo metu viena dažniausiai naudojamų *JavaScript* bibliotekų aplikacijų kūrimui, kuri yra palaikoma didelius resursus ir puikius inžinierius turinčios kompanijos *Facebook*. React nėra priskiriamas prie web karkasų, pagrindinė ReactJS funkcija yra kurti UI komponentus. React yra įrankis, kurį naudojame vartotojo sąsajos konstravimui iš blokų (komponentų). Toks principas yra patogus, nes komponentus yra lengva perpanaudoti skirtingose

aplikacijos vietose, bei leidžia lengvai skaityti kodą.

3.2. Tvarkaraščio matematinis modelio formavimas

Formuojamas tvarkaraštis turi atitikti šiuos reikalavimus.

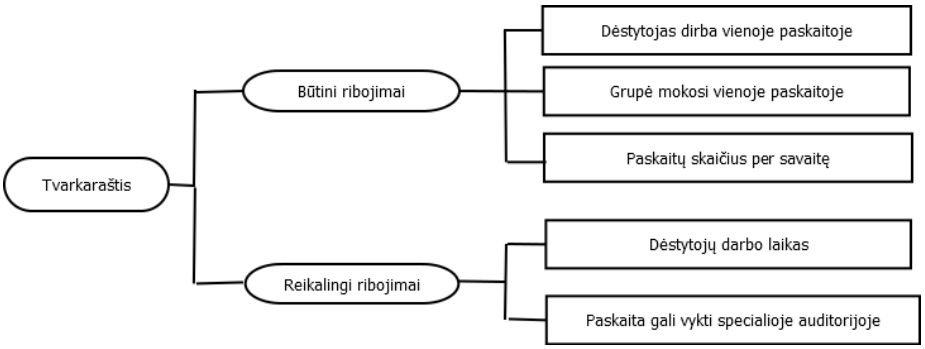
Būtini reikalavimai:

1. Dėstytojas dirba vienoje paskaitoje
2. Grupė studijuoja vieną paskaitą
3. Paskaitų skaičius turi neviršyti 25 paskaitų per savaitę

Reikalingi apribojimai:

1. Dėstytojai gali dirbti ne visas darbo dienas
2. Paskaita vyksta atitinkamoje pagal jos profilį auditorijoje.

Grafiškai tvarkaraščio sudarymo parametrai atvaizduoti 10 paveikslėlyje.



10 pav. Tvarkaraščio sudarymą įtakojantys kriterijai

Formuojant aukštosios mokyklos tvarkaraštį reikia įvertinti tai, kad kai kurie dėstytojai dirba kitose įstaigose ir darbui universitete gali skirti ribotą laiką.

Studentai gali dirbti 5 dienas iš eilės po 5 paskaitas. Tvarkaraščio suformavimui pakanka suformuoti keturių matmenų matricą:

tvarkaraštis [[S[T], L[T]], [V], [G], [K]],

čia S – visų disciplinų matrica; T – dėstytojų skaičius; L – kiekvieno dėstytojo galimo laiko matrica, V – galimų dirbti paskaitų skaičius per savaitę; G – visų studentų grupių matrica; K – patalpų skaičius, kuriose gali vykti paskaitos. Realiai maksimalus paskaitų skaičius V yra 25, tai galima pavaizduoti masyvu:

V=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25].

Čia skaitinės reikšmės reiškia vietą tvarkaraštyje: 1 – Pirmadienis, 8:00–9:30, 2 – Pirmadienis, 9:40–11:10,..., 25 – Penktadienis, 14:40–16:10.

Kiekvieno dėstytojo galimą dirbti laiką galima suraštyti į matricas taip pat. Pavyzdžiui:

L_i *=[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]*.

Tai reiškia, kad *i–tasis* dėstytojas gali dirbti nuo antradienio iki penktadienio visų paskaitų laiku.

Kad būtų lengviau interpretuoti aukštojoje mokykloje esantį užimtumą, aprašysime šią matricą kaip dvejetainę. Taigi, jeigu šios matricos elementai būtų vienetai arba nuliai, matytume, ar pamoka yra vykdoma, ar ne. Pavyzdžiui,

tvarkaraštis [s[t]][l][g][k]=1,

reiškia, kad s[t] – dalykas, kurį gali dėstyti dėstytojas t, vyksta laiku l, kurio metu yra dėstomas nurodytas dalykas, studentų grupės g studentams auditorijoje k. Jei

tvarkaraštis [s[t]][l][g][k]=0,

reiškia, kad pamoka nevyksta. Pagal sudarytą tvarkaraštį, kiekvieno dėstytojo tvarkaraščio matricą galima aprašyti taip: *tvarkaraštis* T_i[s]][l][g][k].

Kiekvienos auditorijos užimtumo tvarkaraštis galėtų būti aprašomas taip: K_i*[s[t]][l][g]*. Tai reiškia, kad *i–tojoje* auditorijoje vyksta dalyko s paskaita, kurią veda dėstytojas t laiku l ir grupei g.

Auditorijos turi savo tipą, pavyzdžiui, skirta auditoriniams arba praktiniams dalykams vesti, dalykai savo ruožtu taip pat turi atitinkamą tipą – praktinis arba teorinis dalykas. Tai svarbu priskiriant auditoriją dalykui. Dar būtų galima įvesti papildomą parametrą – auditorijos vietų skaičių ir studentų, priskirtų paskaitai, skaičių. Parenkant auditoriją, programa turėtų į tai atsižvelgti ir parinkti tinkamą auditoriją.

3.3. Sistemos duomenų modelis

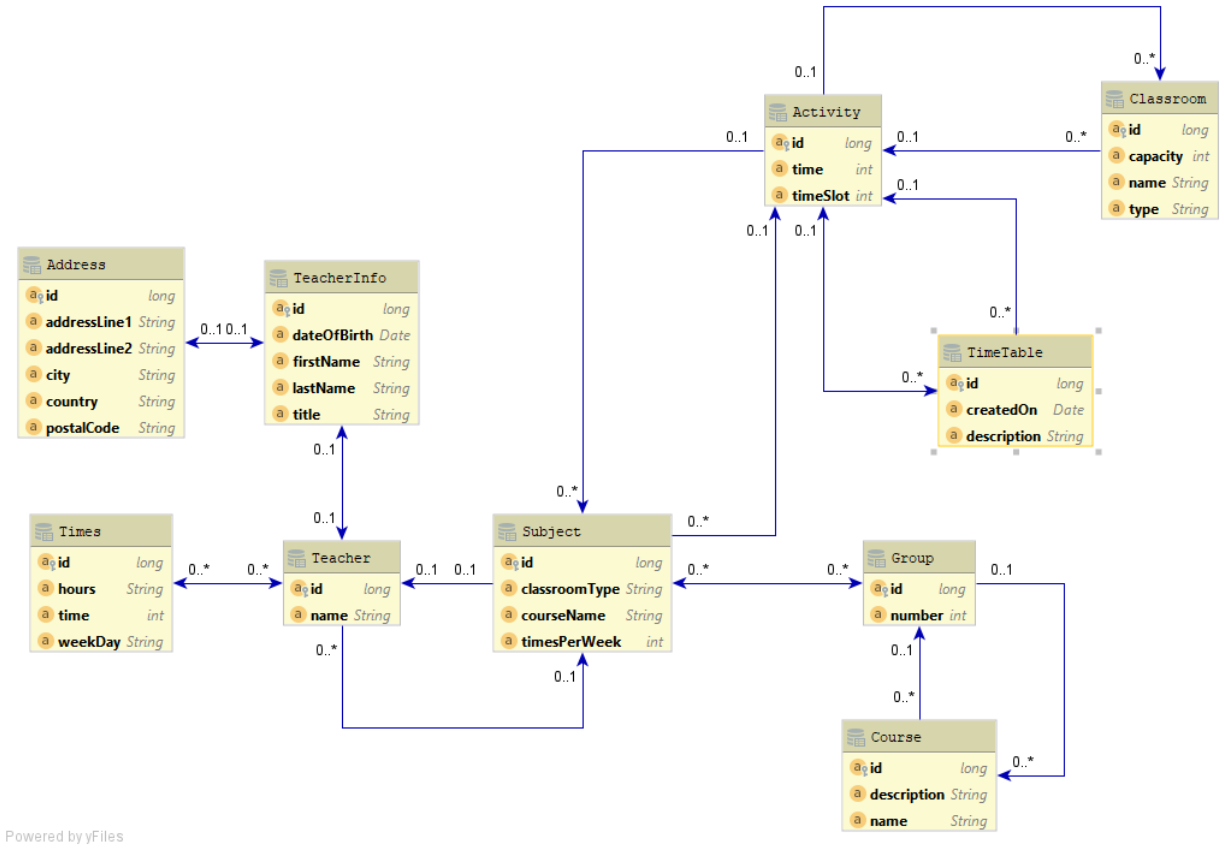
Darbe pateiktame duomenų bazės modelyje viskas prasideda nuo kurso (pvz. *IT*), o kursą sudaro įvairios grupės, todėl ryšys tarp kurso ir grupės yra 1–N. Grupei yra dėstoma daug įvairių dalykų, tačiau kartais paskaitos sujungia visas grupes į vieną, pavyzdžiui teorinės paskaitos vyksta visam kursui t. y. visoms to kurso grupėms, todėl čia ryšys yra daug–su–daug (angl. *Many–to–Many*).

Dalykas (angl. *Subject*) yra esminė programos dalis, nes dalykai spalvotame grafe sudaro viršūnes ir jų visumą reikia surikiuoti korektiškai teisingai. Dalyką dėsto vienas dėstytojas, tačiau dėstytojas gali dėstyti daug dalykų, todėl čia ryšys 1–N.

Dėstytojo lentelėje taip pat yra kelios pagalbinės lentelės, kurios leidžia saugoti platesnę dėstytojo informaciją, tai yra *TeacherInfo* bei *Address* lentelės, tarp visų trijų šių lentelių ryšys yra 1–1. Dalykas nėra konkreti paskaita, tai tik kurso dalis, koks dalykas yra dėstomas studentams, o konkreti paskaita yra saugoma lentelėje *Activity*.

Vienas dalykas gali vykti, pavyzdžiui, 3 kartus per savaitę, tokiu atveju susikurs trys įrašai lentelėje *Activity,* todėl čia ryšys taip pat yra 1–N. *Activity* lentelės ryšiai leidžia konkrečiai

identifikuoti, kokia paskaita vyksta ir kur, todėl taip pat *Activity* lentelė yra sujungta su *Classroom* lentele, kadangi vienoje klasėje gali vykti daug skirtingų paskaitų ir čia ryšys taip pat 1–N. Programai suskaičiavus visas konkrečias paskaitas su duomenimis, jų kolekcija yra saugoma *TimeTable* lentelėje, todėl ir čia ryšys yra 1–N.

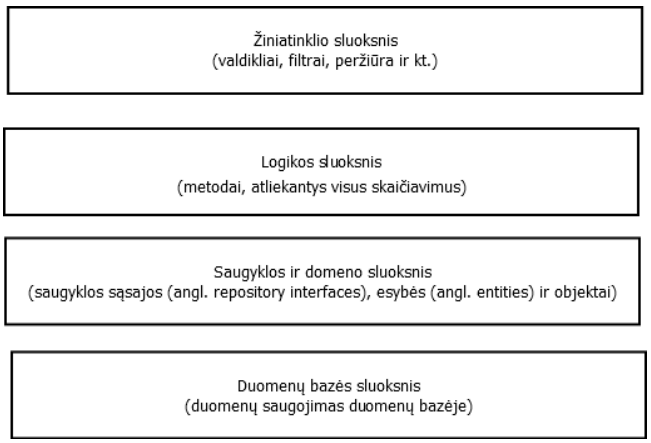


11 pav. Tvarkaraščio sukūrimo duomenų bazės modelis

Modelyje matomos sistemą sudarančios duomenų bazės lentelės: dėstytojų duomenis saugančios lentelės (*Teacher, Teacher info, Address*), dalykų lentelė (*Subject*), kursų lentelė (*Course*), grupių (*Group*), auditorijų (*classrooms*), laiko lentelė (*Times*), ir išvestinės lentelės: sugeneruotų tvarkaraščių lentelė, paskaitų (*Activity*) lentelė, grupių dalykų lentelė (*group_subject*), dėstytojų laikų lentelė (*Teacher–time*).

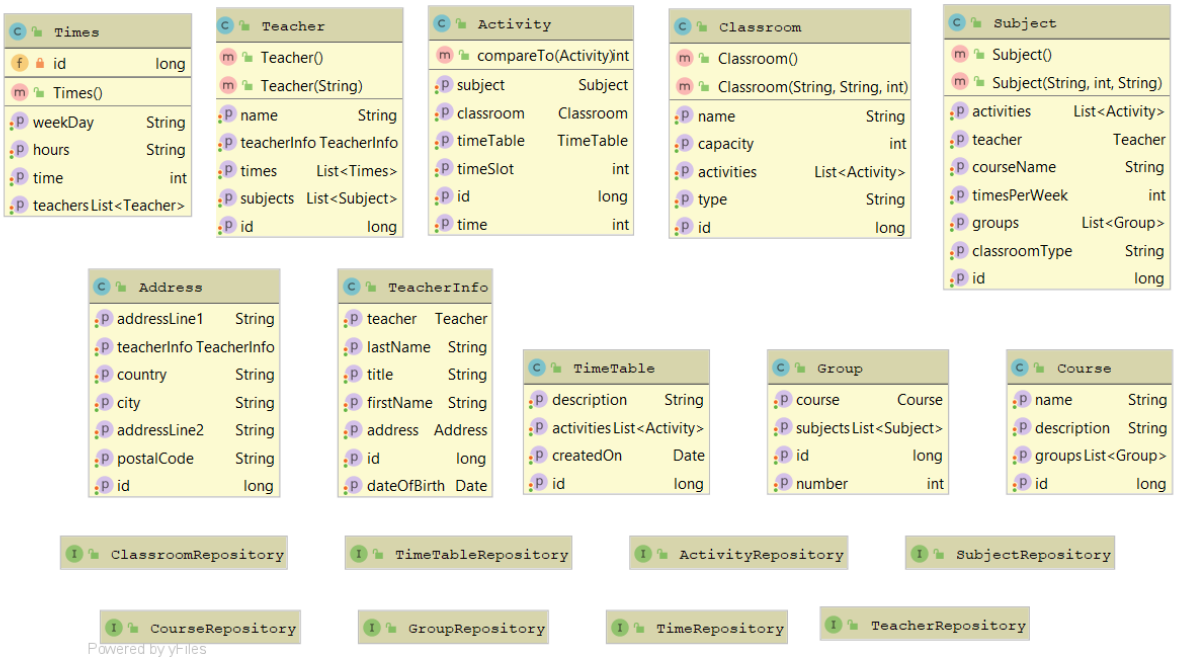
Sukurta sistema naudoja *PostgreSQL* duomenų bazę, kiekvienas įrašas – objektas, kuris turi identifikacinį numerį (toliau – ID).

Serverio pusė sukurta pasinaudojant Spring tradicine architektūra, kuri yra paremta kodo išskaidymu į 4 sluoksnius (žr. 12 pav.).



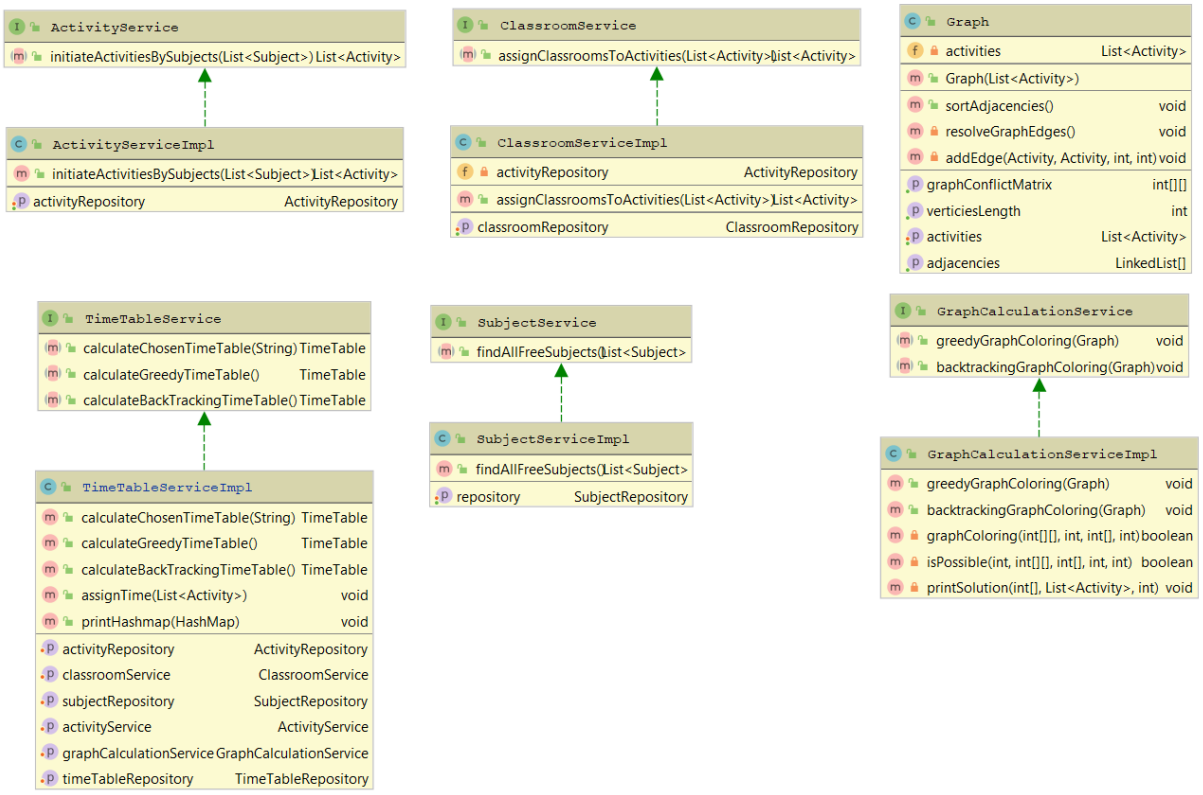
12 pav. Sukurtos sistemos architektūra

Visus sluoksnius representuojančios klasės pateikiamos atskiruose paveikslėliuose, su visais klasėse esančiais metodais ir atributais. Saugyklos ir domeno sluoksnyje pateikiami duomenis representuojančios klasės ir saugyklos, kurios atlieka veiksmus su jomis (sukurti, nuskaityti, koreguoti ir ištrinti (angl. *CRUD*)).



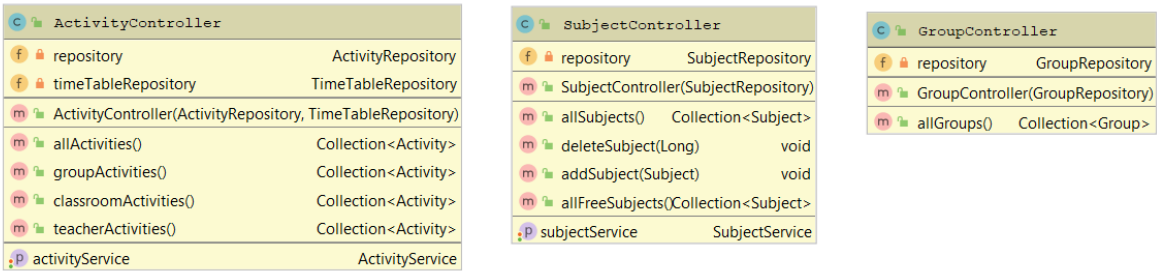
13 pav. Saugyklos ir domeno sluoksnis

Logikos sluoksnyje pavaizduoti visi metodai, atliekantys skaičiavimus, reikalingus naujo tvarkaraščio generavimui. Svarbu paminėti vieną modelio klasę – „Graph“, kurioje aprašoma, koks bus paieškos grafas. Jis turi viršūnes (dalykai) ir briaunas, sudarytas iš tų viršūnių.



14 pav. Logikos sluoksnis

Žiniatinklio sluoksnyje pavaizduotos valdiklių klasės, kurios yra tiesiogiai kviečiamos iš grafinės vartotojo sąsajos.



15 pav. Žiniatinklio sluoksnis

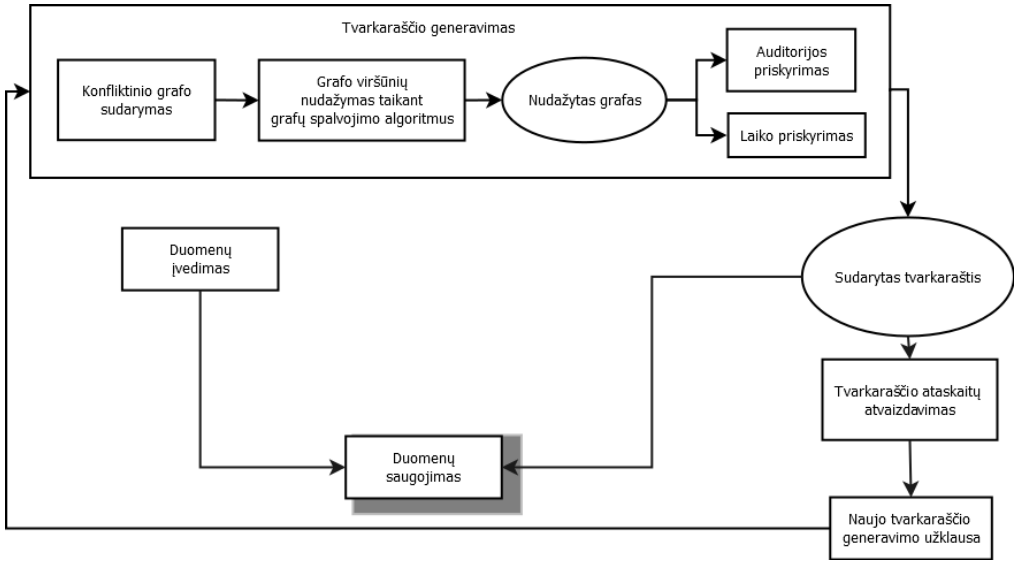
Svarbu paminėti, kad duomenų bazės sluoksnis atsako už duomenų saugojimą pasirinktoje duomenų bazėje *PostgreSQL*.

3.4. Realizuojamos programos struktūra

Šiame skyrelyje aprašyta sukurtos internetinės aplikacijos, skirtos švietimo įstaigoms sudaryti tvarkaraščius, struktūra. Sprendimo algoritmus sudaro:

- Atsitiktinės eilės godžiojo modeliavimo algoritmas.
- Išrikiavimo pagal didžiausią laipsnį mažėjimo tvarka godžiojo modeliavimo algoritmas.
- Atsitiktinės eilės grįžimo modeliavimo algoritmas.
- Išrikiavimo pagal didžiausią laipsnį mažėjimo tvarka grįžimo modeliavimo algoritmas.

Tvarkaraščio sistemą sudaro dvi dalys – kliento ir serverio. 1 ir 2 prieduose pateikiama abiejų programos dalių failų struktūra. Šiame darbe sukurta prototipinė tvarkaraščių sudarymo sistema, kuri leidžia sudaryti tvarkaraščius švietimo įstaigoms. Įvedus pradinius duomenis, sistema pritaiko grafo spalvojimo algoritmus – t. y. išsprendžia grafo uždavinį, priskiriant spalvas studijų dalykams. Apie algoritmą ir jo veikimą plačiau aprašoma 3 skyriuje, o matematinį modelį – 4.2 skyrelyje. 16 paveikslėlyje pateikiama tvarkaraščio generavimo sistemos veikimo schema:



16 pav. Tvarkaraščio generavimo sistemos veikimo schema

Plačiau aprašomi uždavinio sprendimo moduliai:

1. Duomenų įvedimo modulis. Prototipinėje versijoje buvo naudotas duomenų priskyrimas vidinėje programavimo (serverio) dalyje. Inicializuojami šie duomenys: kursai, grupės, dalykai, dėstytojai, auditorijos. Pagal duomenų bazėje suformuotus ryšius dalykai priskiriami grupėms, dėstytojams, nurodomas kiekvieno dėstytojo tinkamas darbo laikas. Auditorijos ir dalykai taip pat turi savo tipus, kad galėtume priskirti auditoriją pagal dėstomo dalyko specifiką.

2. Naujo tvarkaraščio generavimo užklausos formavimas. Klientinėje programos dalyje, klientas pasirenka norimą tvarkaraščio generavimo algoritmą, ir tas pasirinkimas tampa parametru iškviestame RESTful API servise, kuris kreipiasi į tvarkaraščio generavimo modulį.

3. Tvarkaraščio generavimo modulis . Iškvietus šį metodą, atliekami veiksmai:

a) Sukuriamas naujas tvarkaraščio objektas.

b) Suformuojamos iš dalykų paskaitos (*Activity* objektai), paskaitų skaičius priklauso nuo to, kiek kartų dalykas bus dėstomas per savaitę. Tarkime, grupė studijuoja 5 dalykus, kurie visi vyksta po 2 kartus per savaitę. Taigi, ši grupė per savaitę turės 10 paskaitų. Kiekviena paskaita yra laikoma grafo viršūne, todėl joms priskiriamos pradinė spalva –1. Šis paskaitų formavimo metodas yra aprašytas klasėje *ActivityService* ir įgyvendintas *ActivityServiceImpl*.

c) Sukurti paskaitų objektai, priskiriami prie a) punkte naujai sukurto tvarkaraščio objekto. Tokiu būdu kiekviena paskaita priskiriama tvarkaraščiui, o tai leidžia generuoti ne vieną skirtingą tvarkaraštį.

d) Iš sukurtų paskaitų objektų sukuriamas grafas (sistemoje apibūdinamas kaip *Graph* objektas), kiekviena paskaita tampa grafo viršūne, o briaunos tarp viršūnių nustatomos atsižvelgiant į būtinus tvarkaraščio sudarymo reikalavimus, t.y. dėstytojai ir studentai vienu metu negali būti keliose vietose.

e) Pagal pasirinktą vieną iš 4 (atsitiktinės eilės godžiojo, išrikiavimo pagal didžiausią laipsnį mažėjimo tvarka godžiojo, atsitiktinės eilės grįžimo, išrikiavimo pagal didžiausią laipsnį mažėjimo tvarka grįžimo) viršūnių spalvinimo metodų kiekvienai paskaitai suskaičiuojama ir priskiriama spalva.

f) Atsižvelgiant į dalyko ir auditorijos tipą kiekvienai paskaitai priskiriama auditorija. Praktinės paskaitos vedamos praktinio pobūdžio auditorijose, teorinės – teorinio. Taip pat būtų galima naudoti specializuotas auditorijas specifinio tipo paskaitoms, tarkim, kūno kultūros užsiėmimams vesti ir pan. Priskiriant auditorijas, atsižvelgiama į du dalykus – tipą ir užimtumą. Pasirenkama tik ta auditorija, kurios tipas tinka ir ji tuo metu yra laisva t.y. ji negali tuo pačiu metu priklausyti kelioms paskaitoms, nudažytoms ta pačia spalva, nes jos vyksta tuo pačiu metu. Auditorijų priskyrimo metodas yra aprašytas *ClassroomService ir įgyvendintas ClassroomServiceImpl*.

g) Atsižvelgiant į paskaitai priskirtą spalvą, dėstančio dėstytojo bei grupės užimtumą, paskaitai priskiriamas laikas. Pažymėtina tai, kad tos pačios spalvos paskaitoms priskiriamas analogiškas laikas. Pasirinktas grafo spalvojimo algoritmas, priskyręs spalvas paskaitoms, užtikrina, kad neįvyktų konfliktas ir neatsitiktų taip, kad tas pats dėstytojas dėstytų kelioms grupėms, ar grupėms tuo pačiu metu nebūtų priskirti keli užsiėmimai. Kuomet grafas jau nuspalvotas, ir spalvos tvarkaraštį apsaugo nuo konfliktų, paskaitoms priskiriant laiką yra atsižvelgiama ir į minkštus reikalavimus, jeigu tai yra įmanoma. Taigi uždavinys yra kiekvienai spalvai priskirti laiką. Paimame po vieną spalvą ir sudarome tos spalvos visų dėstytojų laiką, kada jie visi gali dirbti vienu metu. Tarkime, spalvą sudaro 3 paskaitos, jas dėsto 3 dėstytojai: pirmas dėstytojas dirba pirmadieniais – trečiadieniais, antras – trečiadieniais – ketvirtadieniais, o trečias – gali dirbti bet kuriuo laiku. Vadinasi bendras visų dėstytojų galimas laikas yra trečiadienis. Jei

grupė trečiadienį gali mokytis, paskaitai priskiriamas trečiadienio laikas.

h) Su visais paskaitų duomenimis naujasis tvarkaraščio objektas perduodamas grafinei vartotojo sąsajai.

4. Tvarkaraščio ataskaitų atvaizdavimo modulis. Šio proceso rezultatas – žiniatinklyje pateikiami tvarkaraščiai – ataskaitos. Sugeneruoti tvarkaraščiai atvaizduojami vartotojo sąsajoje pagal grupes, dėstytojus ir auditorijų užimtumą. Vartotojai – studentai ar dėstytojai gali pasirinkti ir peržiūrėti norimą tvarkaršt̨.

3.5. Tvarkaraščio sistemos realizacijos pavyzdys

Toliau yra pateikiamas tvarkaraščio formavimo pavyzdys, kuris generuojamas 3 grupėms. Šiame pavyzdyje kai kurių dalykų paskaitos vyksta 1 kartą per savaitę, o kai kurių – 2. Taip pat 2 dalykus: „Operacinės sistemos“ ir „Programavimo pagrindai“, klausys jungtinė studentų grupė (*G1, G2*). Įvedimo duomenys pateikiami 8 lentelėje:

8 lentelė. Įvesties duomenų rinkinys: 3 grupių studijuojami dalykai.

Nr	Dalykas	Paskaitų skaičius per savaitę	Grupė
1.	Anglų k.	1	I IS
2.	Duomenų bazės	2	I IS
3.	Kompiuterių architektūra ir tinklai	1	I IS
4.	Matematika	1	I IS
5.	Operacinės sistemos	1	I IS, II IS
6.	Programavimo pagrindai	2	I IS, II IS
7.	Duomenų sauga	2	II IS
8.	Aplinkos ir žmonių sauga	1	II IS
9.	Duomenų struktūros ir algoritmai	1	II IS
10.	Informacinės sistemos	2	II IS
11.	Verslo pagrindai	1	II IS
12.	Kompiuterinių sistemų projektavimas	2	III IS
13.	Internetinės sistemos	2	III IS
14.	Internetinių sistemų praktika	2	III IS
15.	Kompiuterinė grafika	1	III IS
16.	Verslo informacinės sistemos	2	III IS
17.	Virtualizacija	2	III IS

Tai įvertinus, tvarkaraštyje turi būti sugeneruojamos 26 paskaitoms 3 studentų grupėms. Dalykų studijavimui numatytas 8 auditorijų sąrašas: 4 teorinio tipo ir 4 praktinio tipo auditorijos.

Visas savaitės paskaitų laikas suskirstytas į 25 dalis, paskirstant po 5 dalis kiekvienai savaitės dienai nuo pirmadienio iki penktadienio. Pavyzdžiui, pirmadienį sudaro laiko intervalai nuo 1 iki 5, antradienį – nuo 6 iki 10 ir t.t.

Dėstytojų sąrašas su jų galimais dirbti laikais pateikiamas 9 lentelėje:

9 lentelė. Dėstytojų galimas dirbti laikas

Dėstytojas	**Laiko intervalas**	**Savaitės dienos**
T1	1 – 25	pirmadienis – penktadienis
T2	16 – 25	ketvirtadienis – penktadienis
T3	1 – 25	pirmadienis – penktadienis
T4	11 – 20	trečiadienis – ketvirtadienis
T5	6 – 20	antradienis – ketvirtadienis
T6	6 – 20	antradienis – ketvirtadienis
T7	1 – 25	pirmadienis – penktadienis
T8	11 – 20	ketvirtadienis – ketvirtadienis
T9	1 – 25	pirmadienis – penktadienis
T10	1 – 25	pirmadienis – penktadienis
T11	11 – 20	trečiadienis – ketvirtadienis
T12	1 – 25	pirmadienis – penktadienis

Perdavus duomenis algoritmui, programa sugeneruoja 11 spalvų, tai reiškia, kad grąžinamas 11 laiko periodų tvarkaraštis. Taip pat galima pastebėti, kad nėra jokių konfliktų, tvarkaraštis grįžo optimalus, vienu laiko periodu negali vykti daugiau nei 3 paskaitos, nes tvarkaraštis sudaromas 3 grupėms. 10 lentelėje pateikti sugeneruoto tvarkaraščio duomenys.

10 lentelė. Paskaitų priskyrimas spalvoms, taikant surikiuoto godžiojo algoritmą

Spalva	**Dalykai**	**Dėstytojas**	**Grupė**	**Auditorija**	**Laiko vienetas**
spalva 0	Programavimo pagrindai Virtualizacija	doc. T5 lekt. T10	I IS, II IS III IS	1 audit. 2 audit.	7
spalva 1	Virtualizacija Programavimo pagrindai	lekt. T10 doc. T5	III IS I IS, II IS	2 audit. 1 audit.	6
spalva 2	Kompiuterinių sistemų projektavimas Operacinės sistemos	lekt. T7 lekt. T6	II IS I IS, II IS	2 audit. 1 audit.	8
spalva 3	Duomenų struktūros ir algoritmai Kompiuterinių sistemų projektavimas Matematika	lekt. T2 lekt. T7 doc. T3	II IS II IS I IS	2 audit. 1 audit. 5 audit.	16
spalva 4	Duomenų bazės Informacinės sistemos Internetinės sistemos	lekt. T2 lekt. T7 doc. T5	I IS II IS III IS	3 audit. 2 audit. 1 audit.	17
spalva 5	Duomenų bazės Informacinės sistemos Internetinės sistemos	lekt. T2 lekt. T7 doc. T5	I IS II IS III IS	3 audit. 2 audit. 1 audit.	18
spalva 6	Duomenų sauga Kompiuterių architektūra ir tinklai Internetinių sistemų praktika	lekt. T10 asist. T4 doc. T5	II IS I IS II IS	2 audit. 3 audit. 1 audit.	9
spalva 7	Anglų k. Duomenų sauga Internetinių sistemų praktika	lekt. T1 lekt. T10 doc. T5	I IS II IS II IS	5 audit. 2 audit. 1 audit.	10
spalva 8	Verslo informacinės sistemos Verslo pagrindai	lekt. T11 lekt. T9	III IS II IS	1 audit. 2 audit.	13
spalva 9	Verslo informacinės sistemos Aplinkos ir žmonių sauga	lekt. T11 lekt. T8	III IS II IS	1 audit. 5 audit.	12
spalva 10	Kompiuterinė grafika	lekt. T12	III IS	1 audit.	1

Grafinėje vartotojo sąsajoje galima pakeisti norimą spalvojimą algoritmą. Algoritmų

modeliavimo pasirinkimo langas pavaizduotas 17 paveikslėlyje.

17 pav. Algoritmų modeliavimo pasirinkimo langas

Rastas uždavinio sprendimas apdorojamas ir išvedamas tvarkaraščių pavidalu. 18–19 paveiksluose, taip pat B ir C prieduose pavaizduoti sprendimai: tvarkaraščiai pagal grupės, pagal dėstytojus ir auditorijų užimtumo tvarkaraščiai. Languose pateikiama tik po vieną tvarkaraščio pavyzdį: 1 grupės, 1 auditorijos ir 5 dėstytojo:

Pradinis Dėstytojai Studentų grupės Dalykai Tvarkaraštis Dėstytojų užimtumas Auditorijų apkrovimas

Surikiuotas_Godumo

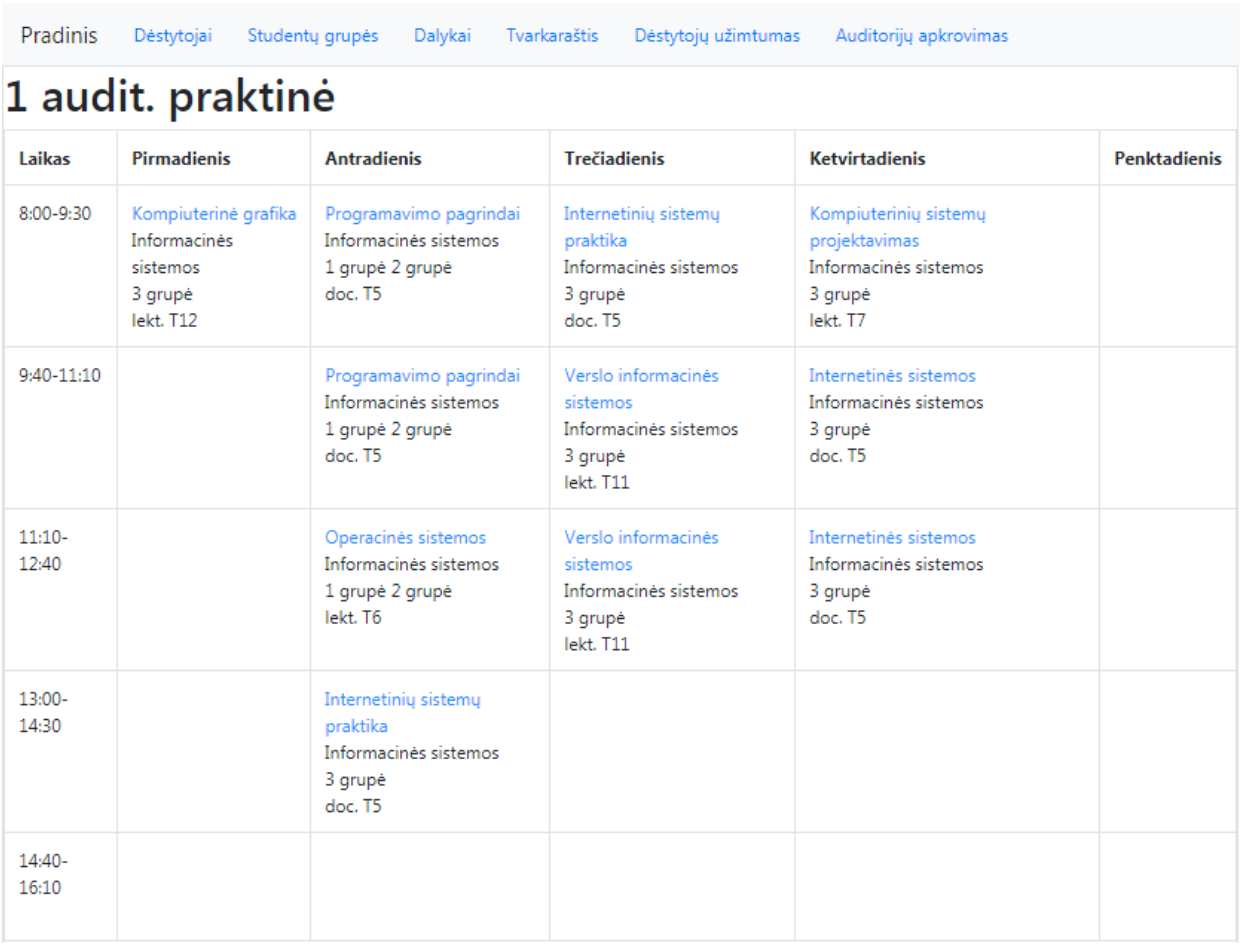
1 grupė Informacinės sistemos

Laikas	Pirmadienis	Antradienis	Trečiadienis	Ketvirtadienis	Penktadienis
8:00-9:30		Programavimo pagrindai doc. T5 1 audit.	Anglų k. lekt. T1 5 audit.	Matematika doc. T3 5 audit.	
9:40-11:10		Programavimo pagrindai doc. T5 1 audit.		Duomenų bazės lekt. T2 3 audit.	
11:10-12:40		Operacinės sistemos lekt. T6 1 audit.		Duomenų bazės lekt. T2 3 audit.	
13:00-14:30		Kompiuterių architektūra ir tinklai asist. T4 3 audit.			
14:40-16:10					

18 pav. Grupės tvarkaraščio pavyzdys

Programos funkcionalumas užtikrina duomenų įvedimą, yra leidžiama tą patį dalyką priskirti kelioms grupėms, jas pasirinkus iš sąrašo, kiekvienam dėstytojui priskirti laikus, kada jis gali dirbti. Keletas įvedimo langų pateikta A priede.

19 paveikslėlyje vaizduojamas auditorijų apkrovimo tvarkaraštis, kuriame yra nurodoma auditorija, jos tipas ir visos savaitės užimtumas, nurodant laiką, dalyko pavadinimą, grupę ir dėstytoją.



19 pav. Auditorijos apkrovimo tvarkaraščio pavyzdys

Sistemoje pateikiami ir dėstytojų užimtumo tvarkaraščiai, kuriuose yra nurodomas dėstytojas ir jo visos savaitės užimtumas, nurodant laiką, dalyko pavadinimą, grupę ir auditoriją (žr. C priedą).

Sugeneruotas tvarkaraštis visiškai atitinka užduotus parametrus: nurodo paskaitos dėstytoją, priskiria paskaitai auditoriją, laiką pagal dėstytojų pageidavimus. Jeigu pagal pasirinktą algoritmą sugeneruotame tvarkaraštyje paskaitai negalima priskirti tinkamo laiko, tuomet sistema daro išimtį ir priskiria kažkuriam dėstytojui netinkamą laiką arba reikia pakoreguoti įvedimo duomenis. Pavyzdžiui, spalvai 0 priskirti 3 dalykai, tačiau, tarkim, pirmo dalyko dėstytojas dirba pirmadieniais, antro – antradieniais, trečio – ketvirtadieniais. Šiuo atveju neįmanoma rasti laiko, kuris būtų patogus visiems dėstytojams ir sistema parenka laiką, neatsižvelgdama į kažkurio dėstytojo pageidaujamą dirbti laiką. Jei netinka sistemos parinktas laikas, galima pakeisti dėstytojų laikus, kad tavarkaraštis atitiktų jų lūkesčius. Laiko priskyrimo algoritmą galima koreguoti pagal aukštosios mokyklos poreikius ir pageidavimus. Tarkim, per dieną negali būti daugiau nei 4 paskaitos ir t.t.

Išvados

- Tvarkaraščių sudarymo uždavinių sprendimai yra sudėtingi ir naudojant tiesinį visų galimų kombinacijų perrinkimą, praktiškai neišsprendžiami. Kadangi galimų variantų yra labai daug, šioje srityje sėkmingai taikomi euristiniai algoritmai, kurie sprendžia tokius uždavinius nepalyginamai greičiau, bet su tam tikra paklaida.
- Taikant grafo viršūnių spalvojimo algoritmus per trumpą laiką galima sudaryti pakankamai optimalius paskaitų tvarkaraščius dėl kurių institucijos gali sutaupyti daug darbo valandų, padidinti produktyvumą, švietimo ir paslaugų kokybę.
- Realizuotas tvarkaraštis žiniatinklio terpėje, taikant grafo viršūnių spalvojimo godžiojo ir grįžimo algoritmus. Tai leidžia efektyviai suplanuoti paskaitas, kuriose tvarkaraščiai turi būti priskirti įvairių išteklių deriniams. Sukurta tvarkaraščio programa tinkamai sudaro tvarkaraštį ir gali funkcionuoti realioje aplinkoje.

Ateities tyrimų planas

Siekiant patobulinti šio darbo rezultatus, darbą galima būtų plėtoti sekančiai:

- Patobulinti tvarkarščių sudarymo sistemą, pridedant tvarkaraščio koregavimo funkciją. Tai leistų tvarkaraščių sudarytojams rankiniu būdu patobulinti tvarkaraštį.
- Sudarant tvarkaraštį atsižvelgti į daugiau minkštų reikalavimų: kiek įmanoma eliminuoti langus, keletą pasikartojančių paskaitų vykdyti iš eilės.
- Pagerinti sukurtos tvarkaraščio sudarymo aplikacijos funkcionalumą, kad ji būtų patogi naudoti visų lygių programos naudotojams: administratoriui, studentams, dėstytojams ir mokyklos administracijai.

Literatūros šaltiniai

[1] aSc Timetables. [Tikrinta 2019–12–20]
http://www.ibn.lt/asc–tvarkarasciai–kompiuterine–tvarkarasciusudarymo–programa

[2] Bincy A.K., Presitha B.Jeba. Graph Coloring and its Real Time Applications an Overview. International Journal of Mathematics And its Applications Volume 5, Issue 4–F, 2017, 845–849. ISSN: 2347–1557. [Tikrinta 2019–12–25]
http://ijmaa.in/v5n4–f/845–849.pdf

[3] Cauvery N. K. Timetable scheduling using graph coloring. *International Journal of P2P Network Trends and Technology (IJPTT),* Volume 1 Issue 2, 2011. [Tikrinta 2019–10–10]
https://pdfs.semanticscholar.org/ab34/ee867408ff5388b034a93410920b2dcadcad.pdf.

[4] FET. *Tvarkaraščių sudarymo programa.* [Tikrinta 2019–11–11]
http://lalescu.ro/liviu/fet/doc/

[5] Ganguli R., Siddhartha Roy. A Study on Course Timetable Scheduling using Graph Coloring Approac. *International Journal of Computational and Applied Mathematics. ISSN 1819–4966 Volume 12, Number 2, 2017.* [Tikrinta 2019–12–11]
https://www.ripublication.com/ijcam17/ijcamv12n2_26.pdf

[6] Poddar N., Mondal2 B. An instruction on course timetable scheduling applying graph coloring approach. *International Journal of Recent Scientific Research* Vol. 9, Issue, 2(C), pp. 23939–23945, 2018.

[7] Nandhini V., Selvanayaki K., Rubine L., Saranya S. A Study onCourse Timetable Scheduling and ExamTimetable Scheduling using Graph Coloring Approach. *International Journal forResearch in Applied Science & Engineering Technology (IJRASET).* ISSN: 2321–9653; IC Value: 45.98; SJ Impact Factor: 6.887Volume 7 Issue III, Mar 2019. [Tikrinta 2019–12–11]
http://www.ijraset.com/fileserve.php?FID=20496

[8] PrimeTimeTable. Tvarkaraščių sudarymo programa. [Tikrinta 2019–10–11].
https://primetimetable.com

[9] Pupeikienė L. Optimizavimo metodų tyrimas ir taikymas profiliuotų mokyklų tvarkaraščių sudarymo uždaviniuose. *Daktaro disertacija.* 2009.

[10] Saha, Muskan & Behera, Bikash & Panigrahi, Prasanta. (2019). Quantum Algorithms for Colouring of Graphs. 10.13140/RG.2.2.19222.19523. [Tikrinta 2019–12–20]
https://www.researchgate.net/publication/330069797_Quantum_Algorithms_for_Colouring_of_Graphs

[11] Stankevičius K. REST architektūrinio stiliaus palyginimas su SOAP, įgyvendinant šiuolaikines interneto paslaugas Vilniaus Gedimino technikos universitetas. 2013 5(2): 92–95. Elektronika ir elektrotechnika.

[12] TimeTabler. Tvarkaraščių sudarymo programa. [Tikrinta 2019–11–10]. http://www.timetabler.com/

[13] Zacharovas V. Grafų teorija. Paskaitų konspektas. 2018. [Tikrinta 2019–12–15] https://klevas.mif.vu.lt/~vytzach/cgi–bin/grafu_teorija.pdf

[14] Muhamad W., Zuki A. W. Solving university course timetabling problems using FET software (2018). AIP Conference Proceedings. 2013. 020052. 10.1063/1.5054251. [Tikrinta 2019–12–15]. https://www.researchgate.net/publication/328044145_Solving_university_course_timetabling_problems_using_FET_software

[15] Navakauskaitė L. Tvarkaraščio sudarymo uždaviniai. Magistro baigiamasis darbas. VU. Matematikos ir informatikos fakultetas. 2016.

[16] JavaScript - programavimo kalba, skirta manipuliuoti HTML žymėmis. [Tikrinta 2019-12-28] https://www.javascript.com/

- [17] Postman įrankis, skirtas API kūrimui [Tikrinta 2019-12-27] https://www.getpostman.com/

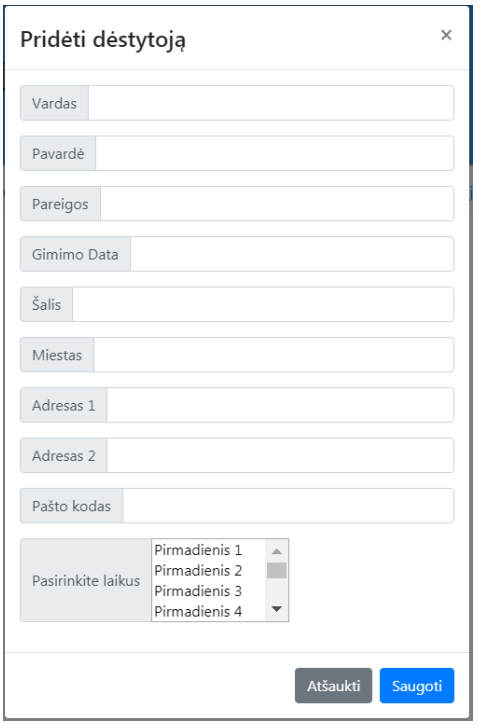
[18] PostgreSQL duomenų bazė [Tikrinta 2020-01-04] https://www.postgresql.org/

[19] React – Javascript kalbos programavimo karkasas [Tikrinta 2019-12-28] https://reactjs.org/

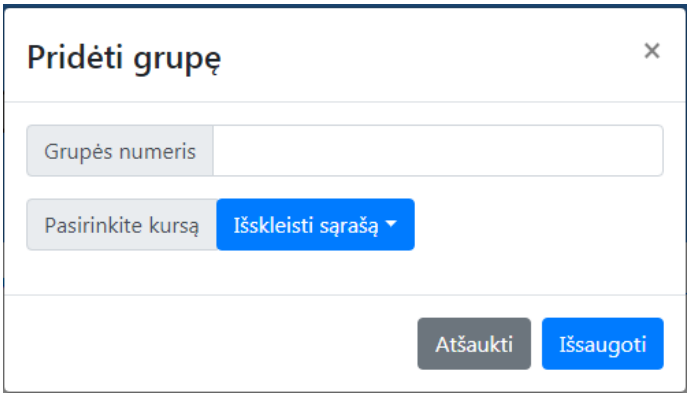
[20] Spring boot karkaso dokumentacija [Tikrinta 2019-12-28] https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

Priedai

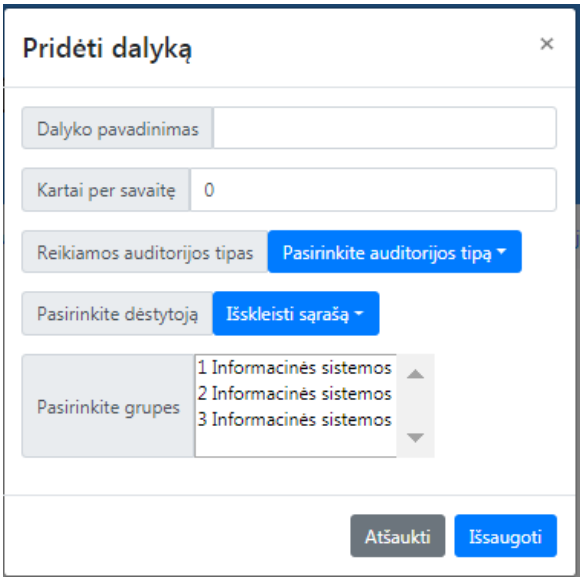
A. Duomenų įvedimo langai



Dėstytojo įvedimo langas



Grupės įvedimo langas



Dalyko įvedimo langas

B. Sugeneruoti tvarkaraščiai skirtingoms grupėms

2 grupė Informacinės sistemos

Laikas	Pirmadienis	Antradienis	Trečiadienis	Ketvirtadienis	Penktadienis
8:00-9:30		Programavimo pagrindai doc. T5 1 audit.	Duomenų sauga lekt. T10 2 audit.	Duomenų struktūros ir algoritmai lekt. T2 2 audit.	
9:40-11:10		Programavimo pagrindai doc. T5 1 audit.	Verslo pagrindai lekt. T9 2 audit.	Informacinės sistemos lekt. T7 2 audit.	
11:10-12:40		Operacinės sistemos lekt. T6 1 audit.	Aplinkos ir žmonių sauga lekt. T8 5 audit.	Informacinės sistemos lekt. T7 2 audit.	
13:00-14:30		Duomenų sauga lekt. T10 2 audit.			
14:40-16:10					

3 grupė Informacinės sistemos

Laikas	Pirmadienis	Antradienis	Trečiadienis	Ketvirtadienis	Penktadienis
8:00-9:30	Kompiuterinė grafika lekt. T12 1 audit.	Virtualizacija lekt. T10 2 audit.	Internetinių sistemų praktika doc. T5 1 audit.	Kompiuterinių sistemų projektavimas lekt. T7 1 audit.	
9:40-11:10		Virtualizacija lekt. T10 2 audit.	Verslo informacinės sistemos lekt. T11 1 audit.	Internetinės sistemos doc. T5 1 audit.	
11:10-12:40		Kompiuterinių sistemų projektavimas lekt. T7 2 audit.	Verslo informacinės sistemos lekt. T11 1 audit.	Internetinės sistemos doc. T5 1 audit.	
13:00-14:30		Internetinių sistemų praktika doc. T5 1 audit.			
14:40-16:10					

C. Sugeneruoti dėstytojų užimtumo tvarkaraščiai

doc. T5

Laikas	Pirmadienis	Antradienis	Trečiadienis	Ketvirtadienis	Penktadienis
8:00-9:30		Programavimo pagrindai Informacinės sistemos 1 grupė 2 grupė 1 audit.	Internetinių sistemų praktika Informacinės sistemos 3 grupė 1 audit.		
9:40-11:10		Programavimo pagrindai Informacinės sistemos 1 grupė 2 grupė 1 audit.		Internetinės sistemos Informacinės sistemos 3 grupė 1 audit.	
11:10-12:40				Internetinės sistemos Informacinės sistemos 3 grupė 1 audit.	
13:00-14:30		Internetinių sistemų praktika Informacinės sistemos 3 grupė 1 audit.			
14:40-16:10					

lekt. T10

Laikas	Pirmadienis	Antradienis	Trečiadienis	Ketvirtadienis	Penktadienis
8:00-9:30		Virtualizacija Informacinės sistemos 3 grupė 2 audit.	Duomenų sauga Informacinės sistemos 2 grupė 2 audit.		
9:40-11:10		Virtualizacija Informacinės sistemos 3 grupė 2 audit.			
11:10-12:40					
13:00-14:30		Duomenų sauga Informacinės sistemos 2 grupė 2 audit.			
14:40-16:10					