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

Magistro baigiamasis darbas

Tvarkaraščio sudarymo modeliavimas taikant kvantinius algoritmus

Atliko:
Modestas Motiejauskas parašas

Vadovas:
Linas Būtėnas

Vilnius
2021

Turinys

Santrauka

Kvantiniai kompiuteriai esant jiems skirtiems kvantiniams algoritmams gali išspręsti tam tikras problemas eksponentiškai greičiau, lyginant su klasikine architektūra paremtais kompiuteriais. Šiame darbe išsikeltas tikslas - sukurti tvarkaraščio sudarymo modelį, panaudojant kvantinį-klasikinį D-Wave hibridinį išsprendėją (angl. quantum-classical hybrid DQM solver). Bendrasis tvarkaraščio formulavimas sudarytas iš reikalavimų įgyvendimo - priklauso apribojimų tenkinimo problemai. Šiame darbe buvo išsikelti šie uždaviniai: pateikti pagrindines sąvokas ir veiksmus, būtinus kvantinių algoritmų bei tvarkaraščio sudarymo modeliavimo analizavimui. Išanalizuoti kelias laisvai prieinamas kvantinių kompiuterių skaičiavimo platformas, leidžiančias atlikti skaičiavimus simuliuojamoje ir tikrame kvantiniame kompiuteryje. Realizuoti tvarkaraščio sudarymo modelį taip, kad jį gebėtų išspręsti D-Wave kvantinių skaičiavimų platforma bei įvertinti gautus rezultatus, pateikti galimus patobulinimus.

Summary

Modeling of Timetable Scheduling Using Quantum Algorithms

Quantum computers can possibly solve certain problems with given quantum algorithms in exponential speedup compared to classical architecture oriented computers. The main goal of this thesis is to create school timetable scheduling model, using quantum-hybrid D-Wave solver. General timetable formulation belongs to constraint satisfaction problem class CSP. In this thesis primary tasks were: display main terms and actions, which are necessary to understand and follow quantum algorithm formulations and general formulation of school timetable scheduling. Also there were analysed few freely available quantum computing infrastructures and tools, which allowed to perform computations simulated environment and also on real quantum hardware. Also implemented timetable scheduling in such way that part of problem could be solved on D-Wave quantum computing platform and given from these results evaluated general applicability.

Įvadas

Kvantinių algoritmų formavimas ir modeliavimas skiriasi nuo klasikinio tuo, jog įprasto kompiuterio algoritmai neįtraukia specialių žingsnių ir būsenų tokių kaip susiejimas (angl. entanglement) bei superpozicija. IBM architektūra paremti kvantiniai algoritmai įprastai analizuojant yra sudaryti iš šių dalių ir procesų: kubitų, bitų, loginių operacijų (angl. quantum gates) transformacijų bei grandinės (angl. circuit). Bitai yra reikalingi išsaugoti ir kaupti išmatuotą kvantinės būsenos reikšmę arba įverčius imant didžiausią tikimybę turinčią reikšmę. O savo ruožtu D-Wave kompanija vysto kitokios architektūros kvantinius kompiuterius, kurie pagrįsti kvantiniu pakaitinimo procesu. Šis procesas dar įprastai yra vadinamas euristiniu bei siekiama rasti globalios funkcijos minimumo. Yra siūlomi ir konstruojami įvairūs algoritmai ir strategijos siekiant paversti kombinatorines optimizavimo problemas, kurios būtų tinkamos išspręsti kvantiniams kompiuteriams. Tačiau tai yra ganėtinai sudėtinga užduotis, nes yra reikalinga modelio kūrimo metu įtraukti apribojimų, reikalavimų ar kitų sąlygų, kurios atsispindėtų ir suformuluotame kvantiniame išsprendimo uždavinyje. D-Wave kvantinis kompiuteris arba jo hibridinis išsprendėjas geba spręsti uždavinius, kurie yra suformuojami kaip dvejetainiais kvadratiniais modeliais arba diskrečiais kvadratiniais modeliais (paremtas kvantiniu-hibridiniu sprendimo būdu). Egzistuoja įvairūs iššūkiai konstruojant modelius, pagrįstais šiais pavidalais - tiesiogiai siunčiant problemą į D-Wave QPU (angl. quantum processing unit) kubitų kiekis ir jų susijungimas yra ribotas dėl techninių ir fizinių struktūrų iššūkių. Taip pat visi kvantiniai kompiuteriai bendrąja prasme yra veikiami išorinės aplinkos, todėl reikalinga taikyti ir ieškoti tinkamo rezultato vykdant bandymus (angl. sample).

Visais atvejais, modeliuojant ir kuriant algoritmus, kuriuos gebėtų išspręsti kvantiniai kompiuteriai, norima įvertinti, ištirti algoritmų efektyvumo spartą bei jų patikimumą. D-Wave kvantinis kompiuteris sprendžia problemas, priklausančias NP-sunkiai (angl. NP-hard) klasei bei jiems nėra žinomi tikslūs, efektyvūs ar, randantys sprendimą per polinominį apibrėžtą laiką). Keletas tokių problemų būtų šios: grafo spalvinimas, kvadratinio krepšelio, kvadratinio priskyrimo, Hamiltono kelio radimas, keliaujančio pirklio, elementų išskaidymo problemos

Šio darbo išsikeltas tikslas - sukurti tvarkaraščio sudarymo modelį, panaudojant kvantinį-klasikinį hibridinį išsprendėją (angl. quantum-classical hybrid DQM solver). Pirmosios dalies tvarkaraščio formulavimas sudarytas iš reikalavimų įgyvendimo - priklauso apribojimų tenkinimo problemai (angl. constraint satisfaction problem). Antrosios dalies įgyvendinime siekta įvertinti kvantinio-hibridinio išsprendėjo galimybes, jo tinkamumą ir efektyvumą sprendžiant įvairios apimties tvarkaraščio sudarymo problemą.

Šio darbo uždaviniai:

- Pateikti pagrindinius terminus ir veiksmus, būtinus kvantinių algoritmų bei tvarkaraščio sudarymo modeliavimo analizavimui.
- Išanalizuoti kelias laisvai prieinamas kvantinių kompiuterių skaičiavimo platformas, leidžiančias atlikti skaičiavimus simuliacinėje aplinkoje ir tikrame kvantiniame kompiuteryje.
- Realizuoti tvarkaraščio sudarymo modelį taip, kad jį gebėtų išspręsti D-Wave kvantinių skaičiavimų platforma.
- Įvertinti gautus rezultatus, pateikti galimus patobulinimus.

Darbe sėkmingai įgyvendintas tvarkaraščio sudarymo modelis ir jo išsprendimas, naudojantis hibridiniu D-Wave išsprendėju, gali būti nesudėtingai performuluotas ir taikomas kitiems sudėtingiems ir didelės probleminės apimties planavimų modeliams arba uždaviniams.

1 skyriuje, išdėstomi terminai ir technologijos kelių algoritmų formulavimas bei pateikiama literatūros apžvalga. 1 skyrių sudaro taip pat dalis skyrelių - būtent, kuriuose kalbama apie IBM kvantinius kompiuterius ir jiems aktualius terminus, algoritmus yra panaudoti iš 2020 pavasario semestro metu turėto mokslo tiriamojo darbo projekto MTDP. 2 skyriuje išdėstomas dvejetainis kvadratinis modelis QUBO, į kurio pavidalą suvedus uždavinį gali išspręsti D-Wave kvantinis kompiuteris. 3 skyriuje aprašomas tvarkaraščio sudarymo modelis, pradinių duomenų gavimo modelis ir visas realizavimas. 4 skyriuje pateikiami gautų tvarkaraščių sudarymo rezultatai, išsprendus juos taikant D-Wave kvantinį hibridinį išsprendėją.

1. Analizė

1.1. Analizės tikslai

Per pastaruosius 10 metų pasiekta pažanga vystant kvantinės kompiuterijos platformas ir techninius sprendimus. Kelios kompanijos vysto ir teikia galimybę tyrėjams vykdyti kvantinius algoritmus NISQ (angl. Noisy Intermediate-Scale Quantum technology) įrenginiuose - IBM Q [4], Rigetti QPU [6], IonQ suspaustų jonų technologiniu pagrindu (angl. trapped ion) [5] ir D-Wave kvantinio pakaitinimo technologijos pagrindu [3]. Tačiau įvykdyti kvantinius algoritmus šiuose įrenginiuose ar mašinose yra nemenkas iššūkis, nes yra reikalinga suformuluoti modelius, metodus, kuriuos kvantiniai įrenginiai galėtų atlikti. Taip pat iškyla problemų gaunant patikimus rezultatus, dėl didelio klaidų kiekio, riboto grandinių gylio (angl. circuit depth), mažo patikimumo. Minėtos kompanijos taip pat vysto ir siūlo mokslininkams naudoti kvantinius simuliatorius, kurių dėka galima performuluoti algoritmus arba modelius taip, kad būtų galima juos vėliau įvykdyti ir techninėje kvantinių kompiuterių įrangoje. Nors kvantiniai simuliatoriai gali atkartoti techninį kvantinių kompiuterių elgesį, simuliavimas reikalauja, jog užduotys būtų nedidelės apimties, nes jie yra vykdomi tyrinėtojų fiziniuose kompiuteriuose. Taip pat kai kurių kompanijų tam tikri skaičiavimų metodai (išsprendėjai) yra mokami, egzistuoja ilgos eilės, norint atlikti bandymus kvantiniuose kompiuteriuose, bet kai kurie modeliai neturi (kol kas) simuliavimo atitikmenų, tad jų tyrinėjimo galimybės yra šiuo metu ribotos. Sekančiame skyrelyje siekiama apsibrėžti terminus norint įsivertinti, kokias problemas ir formulavimus sprendžia D-Wave kvantiniai kompiuteriai.

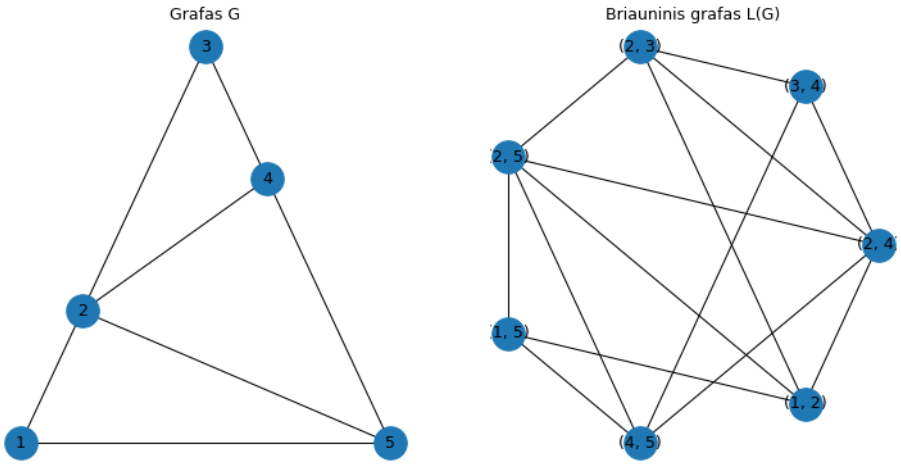
1.2. Terminologija ir technologijos

Šiame skyrelyje apsibrėžiami esminiai terminai, reikalingi dėstymui ir algoritmų analizei.

Grafų teorijos baziniai apibrėžimai: Grafas - $G = (V, E)$, kur V yra ne tuščia viršūnių (angl. vertices) aibė, o E yra briaunų (angl. edges) aibė, jungiančių porą viršūnių. Viršūnės laipsnis $v \in V$ žymimas $\deg(v)$ ir atspindi jungiančių briaunų su konkrečia viršūne v skaičių. Grafo maksimalus ir minimalus laipsniai žymimi $\Delta(G)$ ir $\delta(G)$ bei atspindi maksimalų grafo G viršūnių laipsnius:

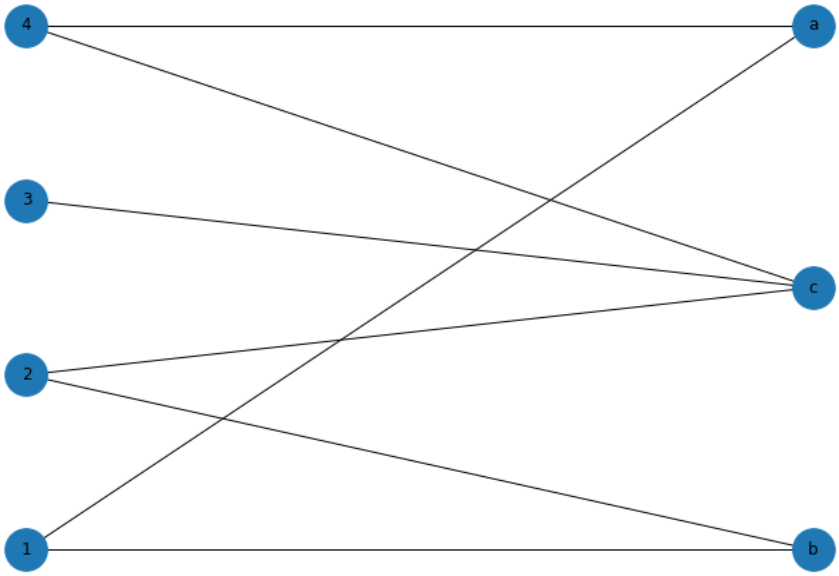
$$\Delta(G) = \max\{\deg(v) \mid v \in V\} \text{ ir } \delta(G) = \min\{\deg(v) \mid v \in V\}$$

Dvidalis (angl. bipartite) grafas - tai grafo tipas, kuris dalina viršūnes į 2 nesusikertančias aibes (angl. disjoint sets) (grupes), neturi nelyginio ilgio grandinių. Viršūnių grafo G spalvinimas yra nuspalvinti viršūnes taip, kad susikertančios viršūnės (turinčios bendrą briauną) įgytų skirtingas spalvas. Mažiausias galimas spalvų skaičius, reikalingas nuspalvinti grafo viršūnes, yra vadinamas chromatiniu grafo skaičiumi. Grafo briaunų spalvinimas yra spalvų priskirimas briaunoms, taip, kad kaimyninės briaunos (einančios iš tos pačios viršūnės) įgytų skirtingas spalvas. Mažiausias skaičius, reikalingas nuspalvinti grafo briaunas skirtingomis spalvomis, yra vadinamas chromatiniu grafo indeksu. Šis chromatinis indeksas dvidalio grafo atveju visada yra lygus - $\Delta(G)$ Briauninis grafas (angl. line graph) - vadinamas grafas L(G), kurio viršūnių aibė turi tiek elementų, kiek briaunų turi grafas G ir viršūnės grafe L(G) jungiamos briauna, tik tada ir tada, kai grafo G atitinkamos viršūnės yra gretimos.



1 pav. Briauninio grafo pavyzdys

1 pav. pateikiamas grafo G ir jo atitinkamas briauninis grafas L(G), kaip matoma grafas G turi 7 briaunas ir jos atitinkamai yra sukonstruojamos į L(G) viršūnes. Grafo briaunų spalvinimo problemai galima paversti į briauninį grafą, nes L(G) viršūnės atitinka G viršūnes.



2 pav. Dvidalio grafo pavyzdys

2 pav. pateikiamas dvidalio (angl. bipartite) grafo pavyzdys, kurio viršūnes nuspalvinti užtenka 2 spalvų, o briaunoms nuspalvinti reikia mažiausiai 3 spalvų ir tai būtų chromatinis šio grafo indeksas. Grafo spalvinimo problema yra laikoma NP-sunki (angl. NP-hard) bei k spalvų spalvinimo užduotis yra NP-pilni (angl. NP-complete) su visais sveikaisiais skaičiais $k \geq 3$ [7, 29]

Performuluoti problemą Skrydžio maršrutų oro uostų terminalų priskyrimas yra aktuali problema, nes oro transporto poreikiai nuolat auga (išskyrus 2020 metus) ir reikalinga suplanuoti efektyvus patikros vartų praėjimus keleiviams. Įprastai šį iššūkį galima išspręsti pridedant daugiau vartų, bet tai yra daug laiko ir kitų resursų atimanti procedūra. Šią problemą galima performuluoti ir išspręsti taikant grafo viršūnių spalvinimo metodą [32, 28].

Komunikacijos stotelių dažnių priskyrimas - technologijų ištobulėjimas mobilaus ir bevielio ryšio tinklų srityje yra pagrindinė priežastis, dėl kurios išaugo bevielio dažnių naudojimas. Esant didesniam vartotojų kiekiui dažnių ruožai tampa perpildyti ir jie persidengia, atsiranda interferencija, prastesnis signalas. Todėl iškyla poreikis optimizuoti ir perskirstyti dažnių kanalus bazinėms stotelėms efektyviau. Yra taikomi įvairūs klasikiniai sprendimo būdai: euristinės technikos, Tabu paieška, simuliacinis pakaitinimas (angl. simulated annealing), bet staigus vartotojų augimas ir bazinių stotelių išplėtimas paverčia šią problemą į NP-sunkia [32]. Tad yra siūlomi sprendimai paverčiant problemą į grafo viršūnių spalvinimo uždavinį, kurį gali išspręsti hibridiniai kvantiniai algoritmai [32].

CPU registrų priskyrimas (kompiliavimo teorijos problema) - kompiliatoriaus didelio kintamųjų kiekio paskirstymas į riboto dydžio CPU registrus, siekiant minimizuoti programų vykdymo laiką. Dabartiniai kompiliatoriai sprendžia šią problemą taikant euristinius metodus. Instrukcijų paruošimo algoritmas siekia suformuoti tarpinį programos kodą, kuris leidžia sumažinti programos bendrą delsą. Pagrindinis CPU registrų priskyrimo tikslas yra sugeneruoti kodą prieš programos vykdymo laiką tokiu būdu, jog būtų naudojama mažiau tarpinių kintamųjų, negu egzistuoja pasiekiamų registrų. Šis kompiliavimo procesas nepakeis programos elgseną, bet gali sumažinti programos vykdymo laiką, nes reikia mažiau kartų pasiekti procesoriaus registrus. Šią problemą galima performuluoti į grafo viršūnių spalvinimo uždavinį ir įrodyta, jog registrų paskirstymas yra NP-pilnas uždavinys [13].

Kubitas - (kvantinis bitas) bazinis [16, 31] informacijos vienetas kvantiniuose skaičiavimuose. Nuo klasikinio bito skiriasi galimybe turėti 2 būsenų reikšmes. Tiksliau apibrėžiant, kubitas yra dvimatė kvantinė sistema. Kubito būsena apibrėžiama šia išraiška [31] :

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

Čia α ir β yra tokie kompleksiniai skaičiai, kurie $|\alpha|^2 + |\beta|^2 = 1$. Dirako (angl. Dirac) žymėjime $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ir $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ yra išreiškimai, kurie užkoduoja dvimatę vektorinę erdvę. Lygtis 1.1 nusako, jog kubito būsena yra dvimatis kompleksinis vektorius $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. Kubito išmatavimas, kurio būsena pateikta 1.1 lygtyje [16], grąžins klasikinę reikšmę nulį ($|0\rangle$) su tikimybe lygia $|\alpha|^2$ arba vienetą ($|1\rangle$) su tikimybe lygia $|\beta|^2$.

Kvantinis kompiuteris turi daugiau nei 1 kubitą, tad naudinga žinoti, kaip apibrėžiama sąveika tarp skirtingų individualių kubitų. Dviejų ar daugiau kubitų sudėtinė sistema yra išreiškiama naudojantis tenzorinę sandaugą (angl. tensor product), kurio operatorius žymimas $\otimes$. Laikant, jog turima 2 kubitų būsenas, kurie išreikšti: $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ ir $|\phi'\rangle = \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}$. Tai pilnoji sistema sudaryta iš dviejų nepriklausomų kubitų atrodys taip:

$$|\phi\rangle \otimes |\phi'\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \begin{pmatrix} \alpha\alpha' \\ \alpha\beta' \\ \beta\alpha' \\ \beta\beta' \end{pmatrix} \tag{1.2}$$

Pavyzdžiui 3 kubitų sistemoje išreikštais 3 kubitais $|\gamma_j\rangle = \alpha_j|0\rangle + \beta_j|1\rangle$, kai $j = 1, 2, 3$, tai

bendroji sistema išreiškiama taip:

$$|\gamma_1\gamma_2\gamma_3\rangle = |\gamma_1\rangle \otimes |\gamma_2\rangle \otimes |\gamma_3\rangle \tag{1.3}$$

$$\begin{aligned}=&\alpha_1\alpha_2\alpha_3|000\rangle + \alpha_1\alpha_2\beta_3|001\rangle + \alpha_1\beta_2\alpha_3|010\rangle + \alpha_1\beta_2\beta_3|011\rangle \\ &+ \beta_1\alpha_2\alpha_3|100\rangle + \beta_1\alpha_2\beta_3|101\rangle + \beta_1\beta_2\alpha_3|110\rangle + \beta_1\beta_2\beta_3|111\rangle\end{aligned} \tag{1.4}$$

Visų trijų kubitų išmatavimas galėtų pateikti vieną iš 8 galimų bitų tekstinės reprezentacijos (angl. bit-string) reikšmių. Iš lygčių pateiktų [1.2 1.3 1.4] [16] būsenos erdvių dimensija auga eksponentiškai nuo n kubitų ir bazinių vektorių skaičius yra 2^n.

Kvantinis susiejimas - tai fizikinis fenomenas [31], kuris teigia, kad pora ar grupė dalelių (šiuo atveju kubitų) yra susieti, dalinasi bendra informacija. Tai reiškia, jog susiję kubitai turės tobulą koreliaciją ir bus vieni nuo kito priklausomi.

Superpozicija - fundamentalusis kvantinės mechanikos principas, kuris teigia, jog iki išmatavimo kvantinė būsena, (kubito reikšmė) nėra apibrėžta - gali turėti būseną, atspindinčią reikšmę 1 arba 0.

1.3. IBM kvantinių kompiuterių algoritmai ir iššūkiai

Vienas iš pagrindinių algoritmų, kurio pagrindu yra konstruojami sudėtingesni algoritmai yra vadinama kvantinė Furje transformacija (angl. Quantum Fourier Transform QFT) [17]. Yra įvairių algoritmo realizavimo siūlymų ir sprendimų [30], kuriais siekiama sumažinti klaidų kiekį kvantiniuose skaičiavimuose. Vienas iš labiausiai žinomų yra vadinamas Šoro (Shor's algorithm) algoritmu [36], skirtas išskaidyti skaičių į 2 pirminius, naudoja kvantinę Furje transformaciją. Taip pat ir kiti algoritmai remiasi QFT, pavyzdžiui Harrow-Hassidim-Lloyd algoritmas [23], skirtas išspręsti tiesines lygčių sistemas.

Pagrindinė problema, dėl kurios iki šiol kvantinių kompiuterių plėtra lėta - didelis klaidų kiekis (angl. error rate). Klaidos kvantiniuose kompiuteriuose atsiranda iš dviejų šaltinių [38]: kubitai sunkiai (trumpai) išlaiko kvantines būsenas (angl. doceherence) bei kvantinės loginės operacijos įneša paklaidas [31, 38]. Palyginimui IBM kvantiniame kompiuteryje CNOT operacija (angl. gate) klaidų kiekis yra 10^{-2} [38] bei praktiškai tokių operacijų skaičius gali siekti tūkstančius ir daugiau vienetų. Yra siūlomi įvairūs klaidas taisantys metodai ir algoritmai [14, 22], tačiau jie yra kritikuojami, nes reikia nemažą dalį kubitų paskirti klaidų taisymams bei iki šiol tai yra aktyvi tyrimų sritis [38]. Taip pat yra iššūkių, susijusių su kvantinių kompiuterinių architektūra - naudojamos labai brangios technologijos, kad būtų galima pasiekti norimas kvantinės sistemos savybes (superpoziciją ir susiejimą).

1.3.1. Bernstein-Vazirani algoritmas

Bernstein-Vazirani algoritmas pirmą kartą pristatytas [10] ir skirtas įvertinti, ar kvantinis kompiuteris galėtų išspręsti tam tikrą problemą greičiau negu įprastas kompiuteris [8].

Algoritmo tikslas - gauti įvedus tekstinės išraiškos bitų seką x, grąžinti 0 arba 1 taip:

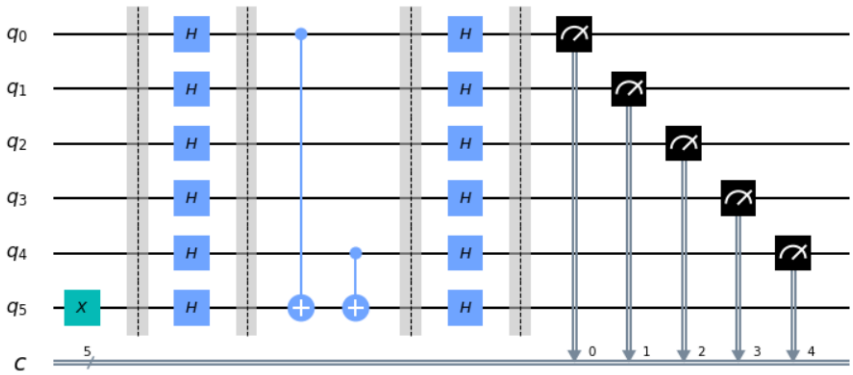
$$f\left(\{x_0, x_1, x_2, \ldots\}\right) \rightarrow 0 \text{ arba } 1 \text{ kur } x_n \text{ yra } 0 \text{ arba } 1 \tag{1.5}$$

Gavus įvestį x (bitų seka) yra randamas paslėptas žodis bitų sekos išraiška. Klasikiniam kompiuteriui šis uždavinys atrodytų taip - $f_s(x) = s \cdot x \mod 2$. Klasikinis kompiuteris išsprendžia šią užduotį per bitų įvestos sekos ilgio n bandymų - iteratyviai yra lyginami įvesto bito paskutinieji simboliai darant AND veiksmą. Kvantinis kompiuteris šį tikslą pasiekia per 1 bandymą. Algoritmo veiksmai, randantys paslėptąjį įvedimą, yra šie [2]:

- Inicializuoti įvesties kubitus į būsenas $|0\rangle$ ir išvesties kubitą į būseną $|1\rangle$.
- Pritaikyti Hadamard operaciją (angl. Hadamard gates) įvesties registrui.
- Sukonstruoti kvantinį orakulą (angl.quantum oracle) - įvykdyti juodosios dėžės funkciją.
- Pritaikyti Hadamard operaciją (angl. Hadamard gates) įvesties registrui.
- Išmatuoti rezultatą.

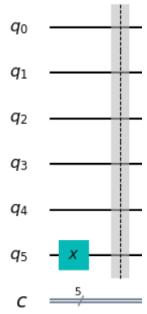
1.3.2. Algoritmo vykdymas

Minėtasis algoritmas [10] buvo įvykdytas naudojantis Qiskit bibliotekos realizacija [2] bei išnagrinėtas veikimas. Eksperimentų vykdymui buvo pateikta $'10001'$ bitų sekos įvestis kaip slaptas raktas, kurį Bernstein-Vazirani algoritmas turėtų surasti. Sekoje yra 5 n simboliai, tai algoritmui reikia 5 ir dar 1 kubito tarpiniams skaičiavimams. Qiskit biblioteka leidžia grafiškai pavaizduoti kvantinės grandinės schemą bei tuo pačiu ir kvantinių algoritmų veikimo principus. Žemiau pateikiama Bernstein-Vazirani algoritmo grandinė Qiskit bibliotekos realizavimu:



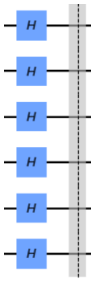
3 pav. Bernstein-Vazirani algoritmo kvantinė grandinė

3 pav. pateikiama kvantinės grandinės (angl. quantum circuit) schema įvedus $'10001'$ kaip įvestį. Grandinėje yra sunumeruoti kubitai q1, q2, ... q5, turintys pradinę būsenas. Pilka linija vadinama barjeru, atskiria grafiškai, vizualiai algoritmo veiksmus bei įtakos jokiems skaičiavimams neturi.



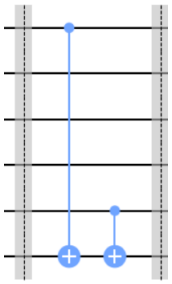
4 pav. Pauli-X operacija (gate)

4 pav. pateikiama Pauli-X operacija, kurios metu paskutiniajam kubitui yra atliekama kubito (angl. bit-flip) apvertimo veiksmas - iš būsenos $|0\rangle$ paverčia kubitą į būseną $|1\rangle$. Šis kubitas bus svarbus tarpiniams skaičiavimams.



5 pav. Hadamard (H) operacija (gate)

5 pav. antrame žingsnyje visiems pirmiems 5 kubitams atskirai yra atliekama Hadamard (H) operacija, kuri reiškia, jog kubitai bus superpozicijoje ir turės vienodas galimybes po lygiai įgyti 0 arba 1.



6 pav. Kvantinio orakulo konstravimas (angl. black-box function)

6 pav. pateikiama kvantinio orakulo konstravimas minėtam algoritmui, kuris atlieka C-NOT (controlled NOT, dar kitaip vadinamas cX) operacijas dviems kubitams. Šis veiksmas, skirtingai nei Hadamard ar Pauli-X operacijos, reikalauja dviejų kubitų - vienas iš jų yra vadinamas kontroliniu kubitu (tai būtų paskutinysis kubitas grandinėje), o kitas vadinamas taikinio kubitu. Iš sekos $'10001'$ reikia pastebėti, jog tai būtų pirmas ir paskutiniai įvesties kubitai, kuriems bus pasiekiamas kvantinis susiejimas.

Sekantis žingsnis yra analogiškas 4 pav. nurodant, jog šiuo atveju kubitams bus atliekama atvirkštinė Hadamard operacija, kuri sugrąžins kubitus į normalizuotą formą, kad būtų galima įvertinti rezultatus išraiška nuo 0 iki 1 (arba kitaip laikant procentine išraiška).

1.3.3. HHL (Harrow-Hassidim-Lloyd) algoritmas

Šis algoritmas yra skirtas išspręsti tiesinių lygčių sistemas. Tiesinių lygčių sistemų sprendimas yra taikomas nuo diferencialinių lygčių skaičiavimų iki įvairių mašininio mokymosi algoritmų [37]. Pats paprasčiausias klasikinis metodas, skirtas išspręsti tiesinių lygčių sistemą, yra vadinamas Gauso eliminavimo metodu, kuris turi $O\left(N^3\right)$ algoritminį sudėtingumą. Yra ir kitų klasikinių algoritmų, skirtų spręsti tiesinių lygčių sistemas - esant tam tikromis sąlygomis matrica yra tridiagonalė Thomas algoritmas [26], dominuoja tik 3 įstrižainės (tridiagonalė matrica) elementai. Jų algoritminis efektyvumas $O\left(N\right)$ arba $O(Ns\kappa\log(1/\epsilon))$. HHL algoritmas turi $O\left(\log(N)s^2\kappa^2/\epsilon\right)$ sudėtingumą, kur κ yra vadinamas sąlygos skaičiumi (angl. condition number), o ϵ yra aproksimacijos tikslumas.

1.3.4. Tiesinės lygčių sistemos pavyzdys

Tiesinės lygčių sistemos sprendimo problemą formuojama taip: rasti sprendinius turint $A\vec{x} = \vec{b}$, kur $A \in \mathbb{C}^{N\times N}$ yra kvadratinė matrica, $\vec{b} \in \mathbb{C}^N$ yra vektorius ir ieškomi sprendiniai žymimi $\vec{x} \in \mathbb{C}^N$ irgi yra vektorius. Taip pat šis algoritmas turi sąlyga, jog A matrica yra simetrinė jungtinė (angl. Hermitian matrix) - $A = A^{\mathrm{H}}$, kur H žymi jungtinį transpozicijos operaciją.

Turima $N = 2$, tada sprendžiama lygčių sistema su tokiomis reikšmėmis:

$$A = \begin{pmatrix} 1 & -1/3 \\ -1/3 & 1 \end{pmatrix}, \quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{ir} \quad \vec{b} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{1.6}$$

Tai šią lygčių sistemą galima suformuluoti kaip $x_1, x_2 \in \mathbb{C}$ radimą, dviejų lygčių sistemą su dviem nežinomaisiais:

$$\begin{cases} x_1 - \frac{x_2}{3} = 1 \\ -\frac{x_1}{3} + x_2 = 0 \end{cases} \tag{1.7}$$

šis HHL algoritmas naudoja 3 registrus, kurie visi priskirti būsenoje $|0\rangle$ vykdymo pradžioje. Vienas registras, kuris pažymimas n_l, yra skirtas atspindėti dvejetainę reprezentaciją A matricos tikrines reikšmes (angl. eigenvalues). Antrasis registras yra žymimas n_b ir skirtas laikyti vektorinį sprendinį. Trečias registras yra skirtas tarpiniams skaičiavimams ir po kiekvieno paskaičiavimo šio registro kubitų būsenų reikšmės atstatomos į $|0\rangle$ būsenas. [23, 2].

Algoritmo veikimas susideda [23, 2] iš šių 6 žingsnių:

1. Užkrauti duomenis $|b\rangle \in \mathbb{C}^N$, t.y. atlikti transformaciją (normalizuoti) - $|0\rangle_{n_b} \mapsto |b\rangle_{n_b}$
2. Atlikti Kvantinės fazės įvertinimą QPE (angl. Quantum Phase Estimation).
3. Pridėti tarpinį kubitą (angl. ancilla qubit) ir įvykdyti pasukimą aplink $|\lambda_j\rangle$.
4. Atlikti atvirkštinį Kvantinės fazės įvertinimą, kuris žymimas $\mathrm{QPE}^\dagger$
5. Išmatuoti tarpinį kubitą skaičiavimų pagrinde ir, jei rezultatas yra lygus 1, registras jau yra tinkamo matavimo būsenoje. Kitu atveju kartoti 2-4 žingsnius.
6. Išmatuoti rezultatą - paskaičiuoti sprendinius.

Qiskit turi jau įgyvendintą bendrinį HHL algoritmą [2] Qiskit Aqua dalyje. Pagal pavyzdį Qiskit [2] teikia tą pačią lygčių sistemą 1.6 praktiniam simuliatoriaus vykdymui. Šiam algoritmui įvykdyti naudojami keli svarbūs parametrai: A - matrica, kuri turi būti jungtinė simetrinė, $\vec{b}$ - reikšmių vektorius ir galima nurodyti kiek kubitų naudoti tarpiniams skaičiavimams. Pateikus tą pačią minėtąją lygtį 1.6 keičiant skirtingą tarpiniams skaičiavimams kubitų kiekį gauti tokie rezultatai:

Kubitų sk. tarp, skaič.	CNOT op. sk.	Bendras op. sk	Kubitų kiekis	Tikslumas	Tikimybė
3	66	129	7	0.999432	0.0562
8	1982	3875	12	1.0	0.000039

1 lentelė. Pavyzdiniai algoritmo parametrų reikšmės [2]

1 lentelėje pateikiami rezultatai keičiant kubitų kiekį naudojamą papildomiems skaičiavimams (angl. ancillae qubits), kuris įgalina pasiekti tikslesnį sprendinio įverčius, tačiau algoritmo grandinė tampa gerokai sudėtingesnė. Tai atskleidžia CNOT operacijų skaičius kartu su bendruoju operacijų skaičiumi, nors iš viso tik 7 ir 12 kubitų bendrai tėra naudojama atitinkamai. Tikslumas (ang, bendrąja prasme, žymi, atstumą (panašumo metrika) tarp dviejų kvantinių būsenų. O tikimybė atspindi, kokia yra tikimybė, kad šie sprendiniai yra šios lygčių sistemos bei algoritmas panaudoja sprendinius, kurie įgyja didžiausią tikimybę.

Kubitų sk. tarpiniams sk.	Sprendinys (x_1, x_2)
3	[1.13586, 0.40896]
8	[1.12499, 0.37498]

2 lentelė. Pavyzdiniai algoritmo sprendinio reikšmės [2]

Iš pateiktos lentelės 2 matoma, jog nurodžius didesnį kubitų kiekį tarpiniams skaičiavimams yra pasiekiamas tikslesnis lygties sprendinys.

Sekantis eksperimentas buvo įvykdytas taikant 4 nežinomųjų tiesinių lygčių sistemą:

$$A = \begin{pmatrix} 3 & 1 & 4 & 3 \\ 1 & 2 & 3 & 2 \\ 4 & 3 & 5 & 1 \\ 3 & 2 & 1 & 1 \end{pmatrix}, \quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad ir \quad \vec{b} = \begin{pmatrix} 6 \\ 3 \\ -5 \\ -5 \end{pmatrix} bei \quad \vec{x} = \begin{pmatrix} -8 \\ -12 \\ 13 \\ 3 \end{pmatrix} \tag{1.8}$$

A matrica parinkta, jog būtų jungtinė simetrinės pavidalo (Hermitian matrix), tačiau įvestis pateikta realiųjų skaičių aibėje, tad užtenka užtikrinti, jog transponuota matrica būtų lygi sau pačiai (simetrija), $\vec{b}$ reikšmės sugeneruotos su skriptu, kuris grąžina sveikojo tipo reikšmių skaičius.

Kubitų sk. tarp, skaič.	CNOT op. sk.	Bendras op. sk	Kubitų skaičius	Tikslumas	Tikimybė
4	122	257	9	0.0885	0.010653
5	172	467	10	0.454402	0.000039
7	1430	2908	12	0.501804	0.000436
10	3366	6376	15	0.276	0.000003

3 lentelė. HHL algoritmo parametrų įverčiai naudojant 1.8 įvestus duomenis

Kubitų sk. tarpiniams sk.	Sprendinys (x_1, x_2, x_3, x_4)
4	[2.13048, 1.73263, 1.54863, -0.2628]
5	[3.19461, 2.48549, 0.17883, -0.2285]
7	[3.14047, 2.71116, 0.0307, -0.1798]
10	[2.15239, 2.47546, 0.92374, -0.09283]

4 lentelė. HHL algoritmo gauti sprendiniai naudojant 1.8 duomenis

3 ir 4 lentelėse pateikiami rezultatai nurodant skirtingą kubitų kiekį tarpiniams skaičiavimams. Iš karto galima pamatyti, jog tikslumo (angl. fidelity) įvertis yra daug prastesnis lyginant su 1 lentelės rezultatu. Taip pat 4 lentelėje matoma, jog sprendiniai yra visiškai nepanašūs į tikrąjį sprendinį, bet 2 lentelėje 2 nežinomųjų lygčių sistemos sprendinys yra aproksimuojančiai artimas. Nepriklausomai nuo to, kokio sudėtingumo tiesinė lygčių sistema yra sukonstruota, jos grandinė yra labai neefektyvi ir perteklinė - palyginimui 3 paveikslėlyje išbandytas Berstein-Vazirani algoritmo schema su 6 kubitais naudoja tik 15 operacijų (kvantinių loginių), bet Qiskit įgyvendinta bendrinė HHL realizacija esant 9 kubitams naudoja nuo 257 bendrų operacijų, o CNOT operacijų 122 iš 3 lentelės. Didelis CNOT operacijų skaičius, didina kvantinės sistemos nepanašumą (angl. fidelity).

1.4. Technologijos ir dabartiniai kompiuteriai

Yra įvairių sprendimų, simuliatorių, kompiuterių, kuriuos galima pasiekti pasinaudojant DK technologijas ir atlikti įvairius skaičiavimus. IBM turi ir siūlo kvantinių skaičiavimų įrankį, vadinamą Qiskit, kuris yra viešai pasiekiamas ir nemokamas. Qiskit yra atviro kodo karkasas [2], vystomas IBM kompanijos ir realizuotas Python programavimo kalba. Šis karkasas leidžia simuliuoti arba atlikti tikrus skaičiavimus IBM kvantiniame kompiuteryje [4] nusiunčiant į eilę. Šiuo metu IBM leidžia pasiekti viešai 10 kvantinių kompiuterių, kuriuos galima laisvai pasiekti ir atlikti įvairius skaičiavimus. Šių kompiuterių atmintis svyruoja nuo 1 iki 32 kubitų - galima išsirinkti norimą kompiuterį, kuris atliktų skaičiavimus,veiksmus. Qiskit karkasu realizuota programa susideda iš kelių bazinių komponentų: registrų, kubitų, bitų, kvantinių loginių operacijų (angl. quantum gates) ir matavimo veiksmo.

Microsoft kompanija savo ruožtu irgi teikia kvantinių skaičiavimų paslaugą, vadinamą Azure Quantum [1]. Microsoft organizacija negamina savo kvantinius kompiuterius, bet bendradarbiauja ir teikia galimybes atlikti skaičiavimus iš kitų įmonių: IQBit, IQNQ, Honeywell, QCI. Microsoft teikia įvairius programų vystymo produktus, QDK (angl. quantum development kit), plėtoja Q# kalbą, kuri panaši sintaksiškai į C# kalbą, bei nereikia specialios programavimo aplinkos - yra tiesiogiai palaikoma Visual Studio programos. Q# kalba galima apsirašyti aukštesniu abstrakcijos lygmeniu algoritmus - nereikia realizuoti techninių kvantinio algoritmo aspektų. Pažymėtina, jog Azure Quantum šiuo metu yra peržiūros versijoje (reikia pildyti paraišką) ir dar nėra viešai pasiekiamos paslaugos.

D-Wave Systems kompanija vysto kitokios architektūros kvantinius kompiuterius. Jų kompiuteriai (įrenginiai) naudoja procesą, vadinamą kvantiniu pakaitinimu (angl. quantum annealing), kurio tikslas yra evoliucionuoti (išvystyti) kvantinę sistemą į žemiausią energiją (angl. zero-point energy arba ground state). Šis procesas bei pats išreiškimas įprastai yra vadinamas Hamiltono (angl. Hamiltonian) sistema. Remiantis adiabatine skaičiavimų teorema, minėtą sistemą galima

formaliai aprašyti taip:

$$H(t) = \left(1 - L\left(\frac{t}{T}\right)\right) H_{\text{prad}} + L\left(\frac{t}{T}\right) H_{\text{gal}} \tag{1.9}$$

Kur $t \in [0, T]$, o L vadinamas Lagranžo funkcija bei sistema idealiomis sąlygomis evoliucionuos iš H_{prad} į H_{gal} žemiausios energijos sistemą.

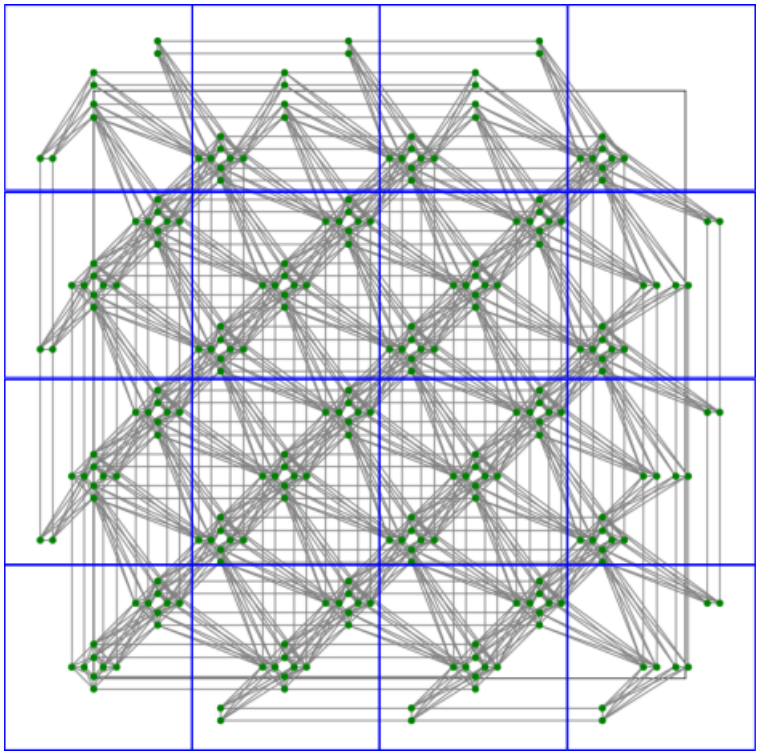
D-Wave kvantiniai kompiuteriai gali išspręsti problemas, suformuotas kaip dvejetainiais kvadratiniais modeliais (angl. Binary Quadratic Models - BQM) arba diskrečiais kvadratiniais modeliais (angl. Discrete Quadratic Models - DQM).

1.4.1. D-Wave išsprendžiami uždaviniai

D-Wave kvantinis kompiuteris sprendžia įvairias kombinatorines optimizavimo problemas, kurios yra išreikštos kaip dvejetainiais arba diskrečiais kvadratiniais modeliais.

D-Wave QPU (angl. Quantum Processing Unit) yra tarpusavyje sujungtų kubitų grotuota struktūra (angl. lattice) [3]. Dalis kubitų yra sujungti tarpusavyje, bet D-Wave QPU kubitai nėra pilnai sujungti. Vietoj to, D-Wave 2000Q ir ankstesni modeliai naudoja specialią topologiją, vadinamą Chimera, o naujesnės kartos Advantage QPU (viešai išleistas tik 2020 metų spalio pradžioje) naudoja topologiją, kuri vadinama Pegasus [11]. D-Wave QPU esanti aibė kubitų ir jungčių, pasiekiama skaičiavimams yra vadinama dirbančiu grafu.

Duotą loginę problemą, apibrėžtą bendriniu grafu (kaip QUBO modeliu), galima pateikti kaip fizinę problemos reprezentaciją į dirbantį grafą naudojant grandis (angl. chains). Ryšys tarp loginės problemos formavimo ir jos pateikimo į fizinę reprezentaciją (D-Wave QPU) yra vadinamas mažuoju įterpimu (angl. minor embedding). D-Wave QPU fizinė kubitų reprezentacija naudoja fizikiniais reiškiniais paremtą Ising modelį (Ising sukinius). Mažojo įterpimo problema yra daug laiko užimanti procedūra, aktyviai tiriami, taikomi įvairūs algoritmai ar metodai [12, 35, 15].



7 pav. Naujos kartos Pegasus architektūros topologinė kubitų išdėstymo schema [18]

7 pav. pateikiamas vienetinių celių grafo išdėstymas, kur žali taškeliai reiškia kubitus, o pilkos linijos - jų sujungimus (angl. couplers). Viena celė Pegasus architektūroje turi 16 kubitų, o čia yra pavaizduota $\mathbf{P_4}$ celių, kas atitinka 4×4 išdėstymo struktūrą. Jungtys, kurios jungia skirtingus kubitus iš kitų celių, vadinamos išorinėmis, o kurios jungia tarpusavio kubitus - vidinėmis (angl. internal and external couplers). Tad galima nesunkiai suvokti, jog turint loginę problemą ją perteikti į fizinę turint ribotą kubitų susijungimą, yra ganėtinai sudėtingas uždavinys.

2. QUBO modeliavimas

2.1. Motyvacija

QUBO - angl. Quadratic Unconstrained Binary Optimization yra matematinis kombinatorinių optimizavimo problemų formulavimas. Šiuo modeliavimo būdu neseniai yra nustatyta, jog QUBO problemas yra galima tiesiogiai pateikti spręsti D-Wave kvantiniam kompiuteriui [3]. Kaip įvairūs autoriai skelbia [24, 21, 19], QUBO modeliai apima, bet neapsiriboja šiomis svarbiomis optimizavimo problemomis: biudžeto paskyrimo, daugelio krepšelių, sąlygos patenkinimo, aibių išskaidymo, grafų spalvinimo, sandėlio prekių paskirstymo, maksimalaus atskyrimo ir kitomis problemomis. Visų šių problemų vienijantis bruožas, jog jie yra optimizavimo uždaviniai [32] ir jie neturi algoritmų, galinčių rasti sprendinius per polinominį laiką, o yra žinomi tik aproksimuojantys, euristiniai klasikiniu atžvilgiu algoritmai.

Formalus ir bendrinis QUBO modelių pavidalas yra išreiškiamas taip:

$$\text{QUBO: minimizuoti } y = x^t Q x \tag{2.1}$$

kur x dvejetaines reikšmes įgyjantis kintamųjų vektorius $x \in [0, 1]$, o Q yra kvadratinė ir simetrinė (pagr. įstrižainei) matrica. Tikslas - yra rasti mažiausią įmanomą sprendinio reikšmę. QUBO modeliai priklauso NP-sunkiems (angl. NP-hard) problemų klasei. Kitaip tariant klasikiniai sprendimų algoritmai (angl. solvers) pateikia tikslų sprendinį esant mažai problemos apimčiai - turint didesnės apimties kintamųjų ir apribojimų rinkinį sprendinio neišskaičiuojama ir po keletos dienų. Suformulavus problemą QUBO modeliu įvairūs autoriai (pacituoti) pateikia aukštos kokybės, bet nevisada optimalius sprendinius per geresnį laiko tarpą.

Egzistuoja keletas svarbių problemų, kurių formulavimas ir sprendimas natūraliai išplaukia į QUBO modelio formą. Viena iš šių problemų yra vadinama skaičių padalinimo uždaviniu, kurio tikslas yra turint aibę $S = \{s_1, s_2, s_3, \ldots, s_m\}$ skaičių, reikia juos išskaidyti į dvi grupes taip, kad šių dviejų grupių narių sumos skirtumas būtų kuo mažesnis. Laikoma, jog $x_j = 1$ jei s_j yra priskiriamas grupei 1, kitu atveju jeigu nepriskiriamas - 0. Poaibės 1 suma išreiškiama $\text{suma}_1 = \sum_{j=1}^{m} s_j x_j$, ir poaibės 2 suma yra lygi $\text{suma}_2 = \sum_{j=1}^{m} s_j - \sum_{j=1}^{m} s_j x_j$. Tikslas yra rasti skirtumą lygų:

$$\text{skirtumas}^2 = \left\{ c - 2 \sum_{j=1}^{m} s_j x_j \right\}^2 = c^2 + 4 x^t Q x \tag{2.2}$$

Kur galutinis pavidalas bei Q matricos forma išreiškiama $q_{ii} = s_i (s_i - c) \quad q_{ij} = q_{ji} = s_i s_j$, kur c randamas paskaičiavus visų elementų sumą. Tai visa sistema susiveda į pavidalą minimizuoti $y = x^t Q x$. Bendras formalus QUBO minimizavimo apibrėžimas teigia, jog reikia rasti, tokias $x \in [0, 1]$ vektoriaus reikšmes, kuriomis sudauginus transponuotą vekrorių x su Q matrica ir vėl padauginus x sprendinys įgytų mažiausią reikšmę. Svarbu paminėti, jog tai yra kombinatorinis perrinkimo uždavinys ir yra galimas ne vienas sprendinys, kuris tenkintų lygties sąlygą, bet optimaliausias - turintis mažiausią reikšmę yra laikomas geriausiu sprendiniu. Nagrinėjant įvairius optimizavimo modelius ir jų funkcijas ši y reikšmę įprastai vadinama kainos funkcija bei remiantis šia reikšme yra lyginami modelių, algoritmų efektyvumo rezultatai.

2.2. Grafo viršūnių spalvinimo algoritmas

Turint grafą $G(V, E)$ kur $V(G) = \{1, 2, \ldots, n\}$ ir $E(G) = \{e_{ij}\}$ laikoma, jei $e_{ij} = 1$ spalva j priskiriama viršūnei i. Iš šios sąlygos grafo k-spalvinimo problemos pirmasis apribojimas

išreiškiamas taip [21, 32]:

$$\sum_{i,j=1} x_{ij} = 1, \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, k \tag{2.3}$$

Iš lygties 2.3 išreiškiama, jog viršūnė i gali (privalo) įgyti tik vieną spalvą j. Sekanti sąlyga grafo spalvinime teigia, jog gretimos viršūnės - turinčios susietą briauną - negali būti nuspalvintos ta pačia spalva j. Šis apribojimas žymimas taip [21, 32]:

$$x_{ip} + x_{jp} \leq 1, \quad p = 1, 2, \ldots, k \tag{2.4}$$

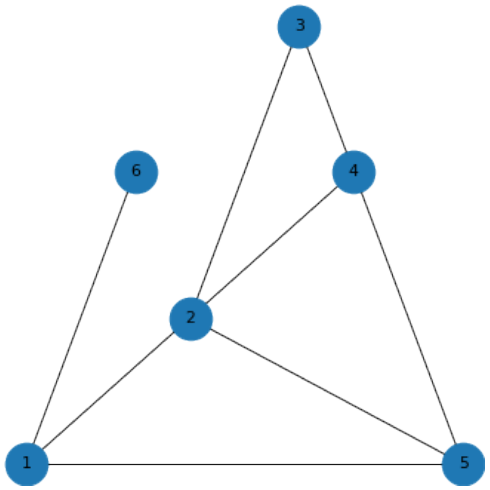
Remiantis lygčių 2.3 ir 2.4 apribojimais QUBO modelio formulavimas yra toks:

$$\sum_{ijk} \left(P \left(x_{ij} - 1 \right)^2 + P x_{ik} \right) \tag{2.5}$$

Kur x_{ij} žymi i viršūnę ir jai yra priskirta spalva j, o x_{ik} žymi viršūnes, kurios yra gretimos (susietos briauna), bet jos turėtų būti nuspalvintos skirtingomis spalvomis.

$$(x_1, x_2, x_3, \ldots, x_{n \times k}) = (x_{11}, x_{12}, \ldots, x_{1k}, x_{21}, \ldots, x_{n1}, \ldots, x_{nk}) \tag{2.6}$$

Kur iš lygties 2.6 n žymi bendrą viršūnių kiekį, o k reiškia bendrą spalvų kiekį. Kaip matoma iš pažymėjimo, reikalingas kubitų kiekis šios problemos išsprendimui yra $n \times k$.



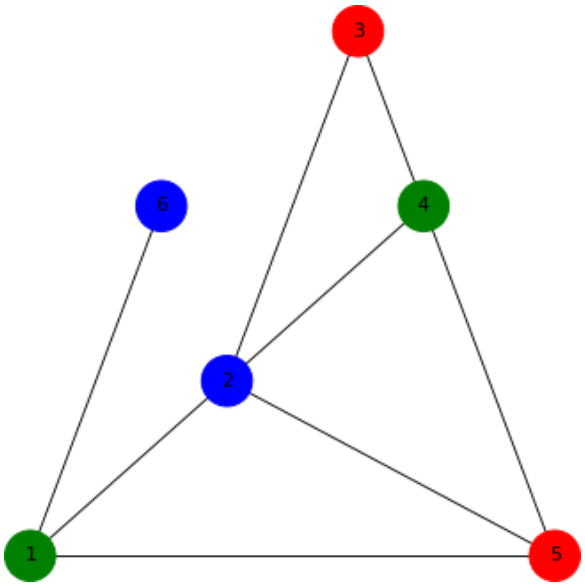
8 pav. Grafo viršūnių spalvinimo pavyzdys

8 pav. pateikiamas viršūnių spalvinimo grafo uždavinys, kuris, kaip pateikta, turi $n = 6$ viršūnes, siekiama panaudoti $k = 3$ skirtingas spalvas viršūnėms ir grafas iš viso turi 8 briaunas. Pagal QUBO modelio formuluotę 2.6 šis pateiktas uždavinys turės iš viso 30 apribojimo sąlygų (angl. constraints) - 6 iš pirmojo, nes kiekviena viršūnė turi įgyti vieną galimą spalvą, 24 iš antrosios sąlygos, nes yra $8 \times k$ kas reiškia, jog viršūnės, susietos briauna, turi įgyti skirtingas spalvas. Q

matricos pavidalo lygtis gauta pagal suformuluotas sąlygas pateikiama žemiau:

$$\begin{pmatrix}
-6 & 6 & 6 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\
6 & -6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 \\
6 & 6 & -6 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 \\
2 & 0 & 0 & -6 & 6 & 6 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 6 & -6 & 6 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 6 & 6 & -6 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & -6 & 6 & 6 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 6 & -6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 6 & 6 & -6 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & -6 & 6 & 6 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 6 & -6 & 6 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 6 & 6 & -6 & 0 & 0 & 2 & 0 & 0 & 0 \\
2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & -6 & 6 & 6 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 6 & -6 & 6 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 6 & 6 & -6 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 & 6 & 6 \\
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & -6 & 6 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 6 & -6
\end{pmatrix} \tag{2.7}$$

Iš 2.7 lygties iš karto svarbu pastebėti, kad Q matrica yra simetrinė palei pagrindinę įstrižainę, tačiau beveik visada yra prasminga siunčiant spręsti kvantiniam kompiuteriui matricą paversti į viršutinę trikampę formą, nes gaunama mažesnio dydžio elementų aibę. Taip pat galima pastebėti tam tikrus dėsningumus, kurie išplaukia iš grafo viršūnių spalvinimo formuluotės. Palei įstrižainę yra formuojamas pirmasis apribojimas išreikštas 2.3 lygtimi, ir pirmi trys elementai atspindi vieną grafo viršūnę, kuri turėtų įgyti tik vieną spalvą iš k galimų. Sekančią sąlygą galima pastebėti iš matricos Q, kurioje matomi elementai, turintys reikšmes 2 - tai parenkama $P = 2$ iš lygties 2.6 antrojo dėmens nario. Kadangi pats QUBO modelis yra neapribotos sąlygų formos, yra taikomi baudos reikšmės (angl. penalty value) ir jis įprastai žymimas P. Naudojant šį koeficientą siekiama, jog suformuluoto modelio sąlyginiai ribojimai atitinkamai atsispindėtų nuo sprendžiamos problemos dalykinės srities - ar tai būtinasis reikalavimas (angl. hard constraint), ar tai yra lengvasis reikalavimas (angl. soft constraint). P parinkimo reikšmė paprastai priklauso nuo dalykinės srities žinojimo, ankstesnių bandymų įvertinimo ar bendrai sprendžiamos problemos formulavimo.



9 pav. Grafo viršūnių nuspalvinimas pritaikius QUBO modelio sprendimą

9 pav. pateikiamas gautas rezultatas išsiuntus anksčiau aptartą Q matricos suformuluotą uždavinį į D-Wave kvantinį kompiuterį.

3. Tvarkaraščio sudarymo modelis taikant hibridinį kvantinį grafo spalvinimo algoritmą

3.1. Motyvacija

Tvarkaraščio sudarymas yra aktuali bei kelianti iššūkių problema. Ši problema yra daugiamatė, turinti daug iškeltų apribojimų ir uždavinių. Tvarkaraščio sudarymas yra resursų, jėgos paskirstymas laiko erdvėje, atsižvelgiant į apribojimus, taip kad būtų tenkinamos kiek įmanoma visos sąlygos. Keletas tvarkaraščio variantų - mokyklos, universiteto, egzaminų, sporto varžybų, transporto ir pan. Šiame darbe siekiama sumodeliuoti mokyklos tvarkaraščio sudarymo problemą įtraukiant QUBO modelio formulavimą. Edukaciniai, mokyklų, universitetų tvarkaraščio sudarymo modeliai reikalauja suplanuoti užsiėmimus išsitenkant į apribotą pamokų, valandų skaičių. Iš mokyklos tvarkaraščio sudarymo formuluotės išplaukia du esminiai apribojimai - mokytojas vienu metu negali mokyti (būti) dviem skirtingoms klasėms, o klasė negali vienu metu (toje pačioje pamokoje) dalyvauti 2 skirtinguose užsiėmimuose (pamokose). Šie du formulavimai įprastai vadinami būtinaisiais (angl. hard constraints) apribojimais ir yra siekiama, jog jie visada būtų atitinkamai įgyvendinti. Taip pat egzistuoja kitoks formulavimas, kuriam norima pridėti sąlygą, jog klasės turetų kuo mažiau vadinamų tarpų tarp sekančios pamokos - šis apribojimas įprastai vadinamas minkštuoju. Svarbu paminėti, jog minkštieji apribojimai dažnai yra priklausomi nuo organizacijos, mokymosi įstaigos specifikos, reikalavimų.

Tvarkaraščio sudarymas priklauso kombinatorikos optimizavimo problemų sričiai, yra aktyviai tyrinėjama problema, dėl jos aktualumo ir reikšmingumo. Tvarkaraščio sudarymo problema ir jos variantai yra įvairiais metodais ir algoritmais: gerus rezultatus demonstruoja pastaraisiais metais genetiniai evoliuciniai algoritmai, [34, 33], meta-euristinė Tabu paieška [25], taikoma grafo spalvinimo būdai [20], naudojamas vadinamasis simuliacinis atkaitinimas (angl. simulated annealing) egzaminų tvarkaraščio sudarymui [27], bei kiti dekompozicijos metodai [39]

3.2. Modelio formulavimas

Šiame skyrelyje pateikiamas mokyklos tvarkaraščio sudarymo problemos formulavimo modelį, pateikiant visus reikalingus apibrėžimus. Tvarkaraščio sudarymo modelis yra padalytas į 2 fazės [9] - pirmosios fazės tikslas yra užtikrinti, jog sugeneruotas tvarkaraščio išdėstymo grafikas tenkina sąlygas, o antrosios fazės tikslas - gautą dieninį grafiką išspręsti - teikiant problemą į kvantinį kompiuterį.

Laikoma, jog yra duota $C = \{c_1, c_2, \ldots, c_m\}$, kuriame yra išviso m klasių, $T = \{t_1, t_2, \ldots, t_n\}$ yra n mokytojų, $D = \{d_1, d_2, \ldots, d_u\}$ - dienų aibė ir įprastai yra 5 darbo dienos (kintamasis u), $P = \{p_1, p_2, \ldots, p_q\}$ - pamokų (valandų) kiekis per dieną (kintamasis q). Jeigu kintamasis x_{ijkl} atspindi susitikimą tarp klasės c_i ir mokytojo t_j kažkurią dieną d_k konkrečioje pamokoje p_l, tai:

$$x_{ijkl} = \begin{cases} 1 & \text{jei klasė } c_i \text{ ir mokytojas } t_j \text{ susitinka dieną } d_k \text{ pamokoje } p_l \\ 0 & \text{kitu atveju} \end{cases} \tag{3.1}$$

Iš šio formuluotės galima iš karto įvertinti, jog yra duota keturių kintamųjų (dimensijų) aibė. Jeigu laikoma, kad TL pažymi bendrą pamokų skaičių, kitaip sakant bendrą susitikimų sumą tarp

mokytojų ir klasių per visas darbo dienas ir valandų išdėstymą, tai formulavimas atrodo štai:

$$TL = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{u}\sum_{l=1}^{q} x_{ijkl} \tag{3.2}$$

Laikoma, kad $R = [r_{ij}]_{m\times n}$ yra savaitinė reikalavimų matrica, kur m klasių skaičius, o n - mokytojų. Kiekvienas matricos elementas r_{ij} žymi susitikimų skaičių tarp klasės c_i ir mokytojo t_j. $CD = [cd_{ik}]_{m\times u}$ yra klasės-dienos matrica, kur cd_{ik} yra pamokų skaičius klasei c_i per dieną d_k. $TD = [td_{jk}]_{n\times u}$ yra mokytojo-dienos matrica, kur td_{jk} atspindi turimų pamokų skaičių mokytojui t_j per atitinkamą dieną d_k. $CT^k = \left[ct_{ij}^k\right]_{m\times n}$ yra dieninė klasės-mokytojo reikalavimų matrica atspindinti pamokų skaičių tarp klasės c_i ir mokytojo t_j atitinkamą dieną d_k. $CP^k = \left[cp_{il}^k\right]_{m\times q}$ yra klasės-pamokos matrica, kurioje $cp_{il}^k = j$ jeigu klasė c_i yra pamokoje p_l atitinkamą dieną d_k su mokytoju t_j, kitu atveju 0. Modelio bendri apribojimai yra šie:

- Mokytojas gali mokinti daugiausiai tik vieną klasę tam tikru laiko momentu.

$$\sum_{i=1}^{m} x_{ijkl} \leq 1, j = 1, 2, \ldots, n, k = 1, 2, \ldots, u, l = 1, 2, \ldots, q \tag{3.3}$$

- Klasė gali neturėti pamokos (egzistuoja vadinamasis "langas") su tam tikru mokytoju ir atitinkamu metu.

$$\sum_{j=1}^{n} x_{ijkl} \leq 1, i = 1, 2, \ldots, m, k = 1, 2, \ldots, u, l = 1, 2, \ldots, q \tag{3.4}$$

- Mokytojas negali turėti daugiau negu 2 pamokas per dieną su ta pačia klase.

$$\sum_{l=1}^{q} x_{ijkl} \leq 2, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n, k = 1, 2, \ldots, u \tag{3.5}$$

Bendras pamokų skaičius TL privalo tenkinti šias sąlygas:

- TL kiek įmanoma yra tolygiai išskaidytas per *u* darbo dienas:

$$\left\lfloor \frac{TL}{u} \right\rfloor \leq noLect_k \leq \left\lceil \frac{TL}{u} \right\rceil, k = 1, 2, \ldots, u \tag{3.6}$$

- TL kiek įmanoma yra tolygiai išskaidytas per *m* klasių:

$$\left\lfloor \frac{TL}{m} \right\rfloor \leq Tc_i \leq \left\lceil \frac{TL}{m} \right\rceil, i = 1, 2, \ldots, m \tag{3.7}$$

- Bendras pamokų skaičius yra tolygiai padalintas per *n* mokytojų:

$$\left\lfloor \frac{TL}{n} \right\rfloor \leq Tt_j \leq \left\lceil \frac{TL}{n} \right\rceil, j = 1, 2, \ldots, n \tag{3.8}$$

- Kiekvienas reikalavimų matricos R elementas privalo tenkinti šias sąlygas:

$$\sum_{k=1}^{u}\sum_{l=1}^{q} x_{ijkl} = r_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n \tag{3.9}$$

- Kiekvienas klasės-dienos matricos CD narys privalo tenkinti šią sąlygas:

$$\left\lfloor \frac{TL}{m \times u} \right\rfloor \leq cd_{ik} \leq \left\lceil \frac{TL}{m \times u} \right\rceil, i = 1, 2, \ldots, m, k = 1, 2, \ldots, u \tag{3.10}$$

- Kiekvienas mokytojo-dienos matricos TD narys privalo tenkinti šias sąlygas:

$$\left\lfloor \frac{TL}{n \times u} \right\rfloor \leq td_{jk} \leq \left\lceil \frac{TL}{n \times u} \right\rceil, j = 1, 2, \ldots, n, k = 1, 2, \ldots, u \tag{3.11}$$

Svarbu paminėti, tvarkaraščio išdėstymo sprendinys egzistuoja tik tada ir tada, kai:

$$TL \leq \min(m, n) \times u \times q \tag{3.12}$$

Remiantis 3.6 - 3.11 sąlygomis reikalavimų, klasės-dienos, mokytojo dienos, klasės-mokytojo matricos gaunamos išsprendžiant 4 lygčių sistemas su apibrėžtomis, galimomis reikšmių aibėmis.

Mokytojų reikalavimų matricos R tiesinės lygčių sistemos formulavimas išreikštas taip [9]:

$$\left.\begin{array}{ll} \sum_{j=1}^{n} r_{ij} & = Tc_i, i = 1, 2, \ldots, m \\ \sum_{i=1}^{m} r_{ij} & = Tt_j, j = 1, 2, \ldots, n \\ & 0 \leq r_{ij} \leq g \\ & g \geq \left\lceil \frac{TL}{m \times n} \right\rceil \end{array}\right\} \tag{3.13}$$

Kur m reiškia bendrą klasių skaičių, o n reiškia bendrą mokytojų skaičių. Svarbus apribojimas yra g rėžio parinkimas ir įprastai jis neturėtų būti didesnis už 10, nes tai reikštų, jog klasė turėtų daugiau nei 2 pamokas per dieną su tam tikru mokytoju esant 5 darbo dienoms per savaitę. Ši tiesinė lygčių sistema priklauso vadinamajai apribojimų įgyvendinimo problemų grupei (angl. constraint satisfaction problem CSP), bei visi minėti ribojimai privalo būti įgyvendinti. Reikalavimų matricos dimensijos $m \times n$ yra m eilučių bei n stulpelių. Kitos matricos yra analogiškai išskaičiuojamos pagal [9] formuluotes ir pagrindinis tikslas yra rasti $CT^k = \left[ct_{ij}^k\right]_{m \times n}$ su kiekviena k diena turint iš viso u dienų. Pilna CT matrica turės iš viso $m \times n \times u$ elementų, bei kiekvienam elementui galioja apribojimo rėžiai - $0 \leq ct_{ij}^k \leq 2$. Kartais egzistuoja bendras CT matricos sprendinys, kuriame užtenka parinkti tik 1 arba 0 pamokų su tam tikra klase i ir mokytoju j dieną k. Šis sprendinio supaprastinimas leidžia lengviau išskaičiuoti sekančios fazės etapus.

Sekantis žingsnis yra - grafo briaunų spalvinimo formulavimas. Išskaičiavus $CT^k = \left[ct_{ij}^k\right]_{m \times n}$ klasės-mokytojo dienines reikalavimų matricas palei kiekvieną dieną suformuluojamas dvidalis grafas, kurio viršūnės padalinamos į klasių bei mokytojų grupės ir formuojama briauna, kai egzistuoja pamoka tarp konkrečios klasės i ir mokytojo j. Kadangi galimų pamokų skaičius tam tikrą dieną k yra išreiškiamas $0 \leq ct_{ij}^k \leq 2$, tai siūloma tvarkaraščio sudarymo problemos grafą skaidyti į 2 dvidalius grafus (angl. bipartite graph)

3.3. Modelio pirmos fazės realizavimas

Pirmosios modelio fazės realizavimas pirmiausiai susideda iš pradinės sąlygos įvertinimo: $TL \leq \min(m, n) \times u \times q$. Tai yra pateikiamas bendras pamokų skaičius TL, mokinių skaičiaus m ir mokytojų skaičiaus n, o kiti parametrai - darbo dienų skaičius u yra lygus 5 ir pamokų (periodų, valandų) skaičius q yra lygus 7. Svarbu paminėti, jog esant 5 darbo dienoms ir turint daugiausiai 7 pamokas, klasė bei mokytojas iš viso daugiausiai tegali turėti 35 valandinius užsiėmimus per savaitę. Sekantis žingsnis yra išskaidyti bendrą pamokų skaičių tarp m klasių, tarp n

mokytojų ir n dėstytojų. Šie kintamieji atitinkamai žymimi $noLect, Tt, Tc$, kur $noLect_k$ atspindės kiek pamokų iš viso vyks tam tikrą dieną k, o Tt_j atspindės kiek pamokų iš viso per darbo savaitę turės mokytojas j ir Tc_i atspindės kiek pamokų per savaitę turės klasė i. Paskutinis žingsnis yra reikalavimų matricos $R = [r_{ij}]_{m\times n}$, klasės dienos matricos $CD = [cd_{ik}]_{m\times u}$, mokytojo-dienos matricos $TD = [td_{jk}]_{n\times u}$ išskaičiavimai ir šie objektai paskaičiuojami pagal anksčiau paminėtus apribojimus. Turint paskaičiuotus šiuos tris elementus galima surasti $CT^k = \left[ct_{ij}^k\right]_{m\times n}$ matricą sulig kiekviena diena išreikšta k. Šios matricos palei dienas randamos sprendžiant tiesinę lygčių sistemą, kuri įtraukia savaitinę reikalavimų, klasės dienos ir mokytojo dienos matricas. Jos formulavimas yra užrašomas taip [9]:

$$\left.\begin{array}{c} \sum_{j=1}^{n} ct_{ij}^k = cd_{ik}, i = 1,2,\ldots,m; k = 1,2,\ldots,u \\ \sum_{i=1}^{m} ct_{ij}^k = td_{jk}, j = 1,2,\ldots,n; k = 1,2,\ldots,u \\ \sum_{k=1}^{u} ct_{ij}^k = r_{ij}, i = 1,2,\ldots,m; j = 1,2,\ldots,n \\ 0 \leq ct_{ij}^k \leq 2, i = 1,2,\ldots,m; j = 1,2,\ldots,n; k = 1,2,\ldots,u. \end{array}\right\} \tag{3.14}$$

Pirmoji sumavimo lygybės sąlyga nusako, jog kiekvieną dieną klasė i turi turėti bendrai tiek pamokų, kiek yra išskaičiuota iš cd_{ik} per atitinkamą dieną k. Antroji lygybė nurodo, jog kiekvieną mokytojas klasė j turi turėti bendrai tiek pamokų (valandų), kiek yra išskaičiuota iš td_{jk} per atitinkamą dieną k. O trečioji lygybė nusako, jog imant iš matricos R kiekvieno nario r_{ij}, susimuotųsi per visas u darbo dienas su kiekvienų k. Taip pat visada yra prasminga įvertinti $0 \leq ct_{ij}^k \leq 2$ rėžius ir galbūt egzistuoja sprendinys turintis siauresnį reikšmių rėžį - $0 \leq ct_{ij}^k \leq 1$, kuris paprastai yra galimas, kai kiekvienas savaitinės reikalavimų matricos narys išsitenka tokiame intervale - $0 \leq r_{ij}^k \leq 5$. Šį reikšminį intervalą galima suprasti, jog mokytojas per savaitę neturi daugiau kaip 5 pamokinius užsiėmimus su klase i per darbo savaitę.

Šiuos visus skaičiavimo etapus galima apibendrinti taip:

1. Užtikrinti, jog pagal duotus dydžius įmanoma surasti tvarkaraščio išdėstymą.
2. Surasti savaitinės reikalavimų, klasės dienos, mokytojo dienos matricas.
3. Pagal išskaičiuotus duomenis, surasti klasės mokytojo matricas su kiekviena k diena.

Šios, pirmosios fazės programiniam realizavimui pasirinkta naudoti Python programavimo kalbą ir įrankius (bibliotekas), kurie leidžia modeliuoti ir aprašyti kombinatorinius optimizavimo problemas. Viena iš šių bibliotekų, vadinama *python-constraint*, pateikė prastus rezultatus siekiant surasti anksčiau aptartų tiesinių lygčių sprendinius - nesugebėta surasti ypač CT matricos sprendinius per pusvalandį arba greičiau turint gana nedidelę kintamųjų paieškos erdvę. Tad buvo nuspręsta rinktis Google kompanijos vystomą kombinatorinių optimizavimo problemų modeliavimo įrankį pavadinimu *OR-Tools*. Šis įrankis leidžia apsirašyti modelius 4 programavimo kalbomis - Python, Java, C++ ir .NET. *OR-Tools* biblioteka pateikė sprendinius per žymiai greitesnius laiko tarpus. *OR-Tools* ir *python-constraint* įrankių vykdymo laiką galima apibendrinti taip:

m	n	TL	OR-Tools (s)	python-constraint (s)
6	5	130	0.05	6.9
8	6	200	0.07	-
10	10	260	0.4	-
15	16	340	0.7	-
20	20	600	2	-
40	40	1200	6.7	-
80	100	2400	29	-

5 lentelė. Modeliavimo įrankių vykdymo laiko rezultatų palyginimas

5 lentelėje pavaizduotas minėtų įrankių vykdymo laikas sekundėmis ieškant 3.14 lygčių sistemos sprendinius, kur nurodyti šie parametrai - m klasių skaičius, n mokytojų skaičius bei TL žymi bendrą pamokų skaičių per visas darbo dienas. Iš karto svarbu pastebėti, jog *python-constraint* įrankis nesugeba surasti sprendinio per 10 minučių, kai kintamųjų aibė yra išreikšta $m \times n \times u$. Taip pat verta paminėti, kad šios ir kitų anksčiau išvardintų tiesinių lygčių sistemų sprendimai neturi tikslinės funkcijos (angl. objective function) - kitaip sakant nėra išreiškiamas joms maksimizacijos ar minimizacijos siektinumas. Tai reiškia, kad išsprendėjai (angl. solvers) turi grąžinti tikslius sprendinius - visi apribojimai yra išreiškiami kaip būtinieji (angl. hard constraints).

Pirmasis žingsnis yra pagal duotus 4 dydžius - klasių, mokytojų, dienų skaičių išdalinti bendrą pamokų skaičių TL į tas minėtas grupes. Surandamas $noLect$, kuris žymi pamokų skaičių su kiekviena diena k turint (įvedant) u dienas, randamas Tt, kuris atspindi kiek įmanoma tolygiai išdalintą pamokų skaičių kiekvienam j mokytojui turint iš viso n mokytojų bei surandamas Tc, kuris atitinkamai atspindi, kiek įmanoma tolygiai išdalintą pamokų skaičių kiekvienai i klasei turint iš viso m klasių. Žemiau pateikiami keli pavyzdiniai aptartų dydžių paskaičiuoti rezultatai:

$$
\begin{aligned}
&m = 6, n = 5, TL = 165, u = 5,\\
&noLect = (33, 33, 33, 33, 33),\\
&Tc = (33, 33, 33, 33, 33),\\
&Tt = (27, 27, 27, 28, 28, 28).
\end{aligned}
$$

$$
\begin{aligned}
&m = 9, n = 6, TL = 210, u = 5,\\
&noLect = (42, 42, 42, 42, 42),\\
&Tc = (23, 23, 23, 23, 23, 23, 24, 24, 24),\\
&Tt = (35, 35, 35, 35, 35, 35).
\end{aligned}
\tag{3.15}
$$

$$
\begin{aligned}
&m = 12, n = 12, TL = 420, u = 5,\\
&noLect = (84, 84, 84, 84, 84),\\
&Tc = (35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35),\\
&Tt = (35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35).
\end{aligned}
$$

Kaip matoma iš pateiktų sprendinių 3.15 pagal duotus parametrus $noLect$ žymi, kiek iš viso k dieną turės vykti pamokinių užsiėmimų, Tc nurodys, kiek kiekviena i klasė turės pamokų per savaitę ir Tt žymi, kiek iš viso pamokinių užsiėmimų turės kiekvienas j mokytojas. Kaip matoma galima pastebėti, kad šių vektorių išskaičiavimas yra gana tiesmukiškas, bet svarbu įvertinti, jog bet kuri klasė ar mokytojas neturėtų daugiau nei 35 pamokinių užsiemimų, nes kitu atvejų tai reikštų, kad reikėtų turėti daugiau nei 7 pamokas per dieną.

Įvertinus anksčiau aptartų pradinių reikšmių skaičiavimo principus galima surasti savaitines reikalavimų R, klasės-dienos CD ir mokytojų-dienos TD sprendinius. Šie sprendiniai, kaip minėta, randami naudojantis *OR-Tools* įrankiu. Šių matricų radimo formas galima alternatyviai užrašyti šiais pavidalais:

$$\begin{cases} Rx = Tc \\ R^{\mathbf{T}}x = Tt \\ 0 \leq r_{ij} \leq g \\ g \geq \left\lceil \frac{TL}{m \times n} \right\rceil \end{cases} \tag{3.16}$$

$$\begin{cases} CDx = Tc \\ CD^{\mathbf{T}}x = noLect \\ 0 \leq r_{ij} \leq g \\ \left\lfloor \frac{TL}{m \times u} \right\rfloor \leq cd_{ik} \leq \left\lceil \frac{TL}{m \times u} \right\rceil \end{cases} \tag{3.17}$$

$$\begin{cases} TDx = Tt \\ TD^{\mathbf{T}}x = noLect \\ 0 \leq r_{ij} \leq g \\ \left\lfloor \frac{TL}{n \times u} \right\rfloor \leq cd_{ik} \leq \left\lceil \frac{TL}{n \times u} \right\rceil \end{cases} \tag{3.18}$$

Kaip pateikta 3.16 - 3.18 lygčių sistemose x reikšmes galima interpretuoti kaip vektorius, kurių visi elementai yra lygūs 1, o reikia rasti R, CD, TD sprendinius su duotais reikšminiais intervalų rėžiais. Nustatyta, kad g parametro parinkimas yra ganėtinai laisvas, todėl galimos įvairios nustatymo strategijos, kurios tenkina bendrą apribojimo sąlyga. Kiti reikšminiai įverčiai yra lengvai apibrėžiami - naudojant matematines grindų ir lubų funkcijas (angl. floor, ceiling). Turint pateiktus pavyzdinius duomenis gautus 3.15 rezultatuose galima surasti R, CD, TD matricas, kurios atrodo taip;

$$R = \begin{pmatrix} 0 & 7 & 7 & 6 & 7 \\ 7 & 0 & 7 & 6 & 7 \\ 5 & 5 & 5 & 7 & 5 \\ 7 & 7 & 7 & 7 & 0 \\ 7 & 7 & 7 & 0 & 7 \\ 7 & 7 & 0 & 7 & 7 \end{pmatrix} \quad CD = \begin{pmatrix} 6 & 5 & 5 & 5 & 6 \\ 5 & 6 & 5 & 6 & 5 \\ 5 & 6 & 5 & 6 & 5 \\ 6 & 5 & 6 & 5 & 6 \\ 5 & 6 & 6 & 6 & 5 \\ 6 & 5 & 6 & 5 & 6 \end{pmatrix}, \quad TD = \begin{pmatrix} 6 & 7 & 6 & 7 & 7 \\ 7 & 6 & 7 & 7 & 6 \\ 7 & 7 & 7 & 6 & 6 \\ 7 & 6 & 6 & 7 & 7 \\ 6 & 7 & 7 & 6 & 7 \end{pmatrix} \tag{3.19}$$

$$R = \begin{pmatrix} 3 & 0 & 5 & 5 & 5 & 5 \\ 5 & 3 & 5 & 5 & 0 & 5 \\ 5 & 5 & 0 & 3 & 5 & 5 \\ 5 & 5 & 5 & 3 & 5 & 0 \\ 0 & 5 & 5 & 3 & 5 & 5 \\ 3 & 3 & 5 & 4 & 3 & 5 \\ 5 & 5 & 5 & 5 & 4 & 0 \\ 5 & 5 & 0 & 4 & 5 & 5 \\ 4 & 4 & 5 & 3 & 3 & 5 \end{pmatrix}, \quad CD = \begin{pmatrix} 5 & 5 & 4 & 4 & 5 \\ 5 & 5 & 4 & 4 & 5 \\ 4 & 5 & 4 & 5 & 5 \\ 5 & 4 & 5 & 5 & 4 \\ 5 & 4 & 5 & 5 & 4 \\ 4 & 5 & 5 & 4 & 5 \\ 5 & 4 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 4 \end{pmatrix}, \quad TD = \begin{pmatrix} 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \end{pmatrix} \tag{3.20}$$

$$R = \begin{pmatrix} 4 & 4 & 4 & 3 & 4 & 4 & 4 & 4 & 0 & 0 & 0 & 4 \\ 0 & 4 & 4 & 0 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 0 \\ 4 & 4 & 0 & 4 & 0 & 3 & 4 & 0 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 0 & 0 & 0 & 4 \\ 4 & 4 & 0 & 4 & 0 & 4 & 4 & 0 & 4 & 4 & 3 & 4 \\ 4 & 0 & 4 & 0 & 4 & 4 & 3 & 4 & 0 & 4 & 4 & 4 \\ 4 & 0 & 4 & 4 & 4 & 4 & 0 & 3 & 4 & 4 & 4 & 0 \\ 4 & 4 & 0 & 4 & 0 & 4 & 3 & 0 & 4 & 4 & 4 & 4 \\ 4 & 3 & 3 & 4 & 3 & 4 & 2 & 4 & 3 & 0 & 2 & 3 \\ 0 & 4 & 4 & 4 & 4 & 0 & 4 & 4 & 4 & 4 & 3 & 0 \\ 3 & 0 & 4 & 0 & 4 & 0 & 4 & 4 & 4 & 4 & 4 & 4 \\ 0 & 4 & 4 & 4 & 4 & 0 & 0 & 4 & 4 & 4 & 3 & 4 \end{pmatrix},$$

$$CD = \begin{pmatrix} 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \end{pmatrix}, \quad TD = \begin{pmatrix} 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \end{pmatrix} \tag{3.21}$$

3.19, 3.20 ir 3.21 pateikiami gauti sprendiniai, kur R sprendinio stulpeliai (angl. columns atitinka j mokytoją ir bendrai turi gauti Tt_j pamokų, analogiškai eilutės (angl. rows) atitinka klasę i ir turi bendrai gauti Tc_i pamokų per savaitę. CD matricos eilutės žymi i klasę, o stulpelių yra 5, nes laikoma, kad savaitėje yra 5 darbo dienos, ir k stulpelis reiškia atitinkamą dieną. TD matricos elementai yra sukonstruoti analogiškai kaip ir ankstesnė, tik joje nurodoma, jog k dieną mokytojas j turės td_jk pamokų. Verta paminėti, jog 3.19 R sprendinyje tam tikra klasė ir mokytojas turės 7 pamokinius užsiėmimus per savaitę, o tai reiškia, kad nebus įmanoma išdalinti šias pamokas tik po viena valandine pamoka. Taip pat 3.21 CD, TD sprendiniai reiškia, jog visos klasės ir visi mokytojai turės maksimalų krūvį, išsitenkinantį per 7 dieninių pamokų apribojimą, nes parinktas didžiausias galimas bendras pamokų skaičius TL.

Turint šiuos išskaičiavimus galima surasti svarbiausius elementus - klasės-mokytojų dieninės reikalavimų sprendinius. Šiuos sprendinius pirmiausiai suprantama kaip R elementų išskaidymą per u dienas nurodant, kad mokytojas gali turėti ne daugiau kaip 2 pamokas su atitinkama klase k dieną. O CD sprendinio vaidmenį galima suvokti kaip teigimą, jog atitinkama klasė privalo tiksliai turėti tiek pamokų atitinkamai, kiek yra paskaičiuota, tam tikrą duotą dieną. Galiausiai, TD sprendinių vaidmenį galima suprasti analogiškai - atitinkamas j mokytojas privalo turėti bendrai pamokų, kiek yra išskaičiuota su kiekviena k diena. Žemiau pateikiami iš 3.20 gauto rezultato paskaičiuotos dieninės reikalavimų matricos pirmų trijų dienų sprendiniai:

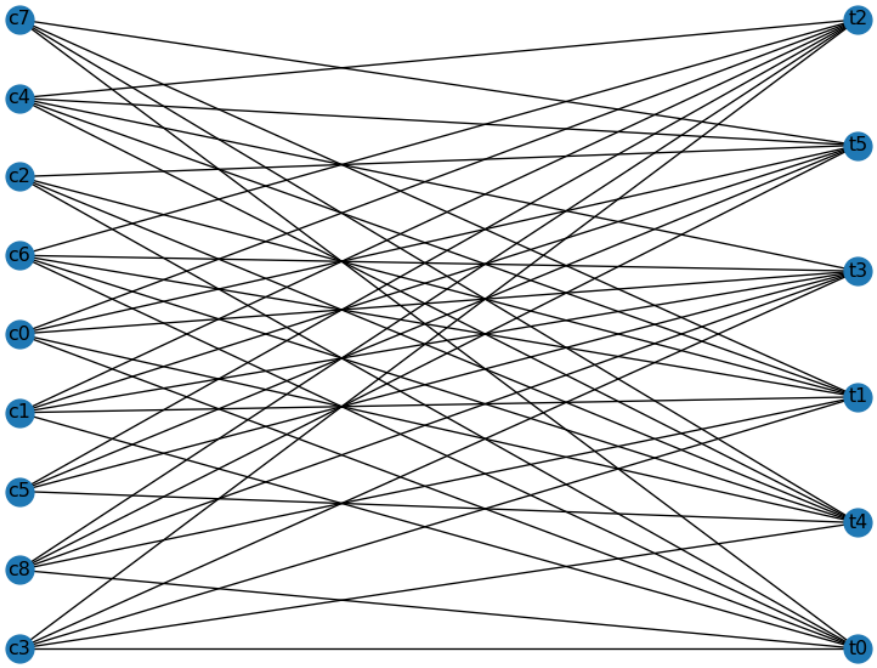
$$CT^1 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}, \quad CT^2 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad CT^3 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \quad (3.22)$$

3.22 pateikiami sprendiniai norint surasti pamokų išdėstymą su šia sąlyga - $0 \leq ct_{ij}^k \leq 1$, bet tik tuo atveju, jei žinoma, kad kiekvienas R elementas yra $0 \leq r_{ij} \leq 5$ rėžiuose. $CT^k = \left[ct_{ij}^k\right]_{m \times n}$ sprendiniai su kiekviena k diena turi tokias pačias dimensijas kaip ir savaitinė reikalavimų matrica R. Nesunkiai galima numatyti, jog egzistuoja sprendinys turint ir tokią galimų reikšmių aibę - $0 \leq ct_{ij}^k \leq 2$, nes tai įtraukia šiek tiek didesnį galimų reikšmių parinkimą. Kiekvienas ct_{ij}^k žymi, kiek pamokų konkrečią dieną atitinkama klasė turės su atitinkamu mokytoju - kur 0 reiškia - neturės, 1 - turės tiksliai vieną pamoką, o 2 - turės 2 pamokas kažkurią dieną. Visas CT skaičiavimų realizavimas yra pateikiamas A priede. Bendrą $CT^k = \left[ct_{ij}^k\right]_{m \times n}$ radimo formulavimą realizuojant $OR - Tools$ galima apibendrinti taip:

- Su kiekvienu CD sprendiniu užtikrinti, jog klasė i turės tiksliai pamokų tam tikrą k dieną, kiek yra nurodyta cd_{ik}.
- Su kiekvienu TD sprendiniu užtikrinti, jog mokytojas j turės tiksliai pamokinių užsiėmimų atitinkamą k dieną, kiek yra nurodyta td_{jk}.
- Su kiekvienu R sprendiniu, padalinti klasės i ir mokytojo j pamokas per u darbo dienas, taip kad tenkintų $0 \leq ct_{ij}^k \leq 2$ sąlygą su kiekviena k diena.

3.4. Modelio antros fazės realizavimas

Antrosios fazės tikslas - turint rastą $CT^k = \left[ct_{ij}^k\right]_{m \times n}$ sprendimą gauti kiekvienos dienos tvarkaraštį atsižvelgiant į tai, kad mokytojas negali vienu metu mokyti daugiau nei 1 klasei ir, analogiškai, klasė negali dalyvauti 2 pamokose vienu metu. Šio tikslo išsprendimui yra naudojamas grafo briaunų viršūnių spalvinimo algoritmas. Yra formuojamas dvidalis grafas iš ct_{ij}^k sprendinių, kurio viena pusė atspindi m klasių viršūnių aibę, o kita puse yra n mokytojų viršūnių aibė. Klasės ir mokytojo viršūnei suformuojama briauna, kai egzistuoja pamoka. Toks sukurtas dieninis dvidalis grafas G yra paverčiamas į briauninį $L(G)$, kurį galima išspręsti kaip QUBO arba diskrečiuoju kvadratiniu modeliu DQM pasinaudojus D-Wave teikiamais skaičiavimų išsprendėjais (angl. solvers) ir spalvinimo algoritmas anksčiau aprašytas 2.2 skyrelyje.



10 pav. Dvidalio grafo konstrukcijos pavaizdavimas iš 3.22 pirmojo sprendimo

Iš 10 pav. pateikiamas pavyzdinis dvidalio grafo formavimas, kuriame yra reikalinga nuspalvinti briaunas taip, kad einančios iš tos pačios viršūnės gautų skirtingas spalvas. Minimalus reikalingas spalvų kiekis šiame grafe, kaip aptarta analizėje, yra vadinamas chromatiniu grafo indeksu, o jis visada yra lygus maksimaliam grafo laipsniui. Dvidalio maksimalų laipsnį $\Delta(G)$ galima surasti nesudėtingai - jis visada turėtų būti ne didesnis už 7 ,remiantis modelio formuluote, arba nustatomas iš CD ar TD sprendinių iš klasės c_i bei mokytojo t_j su atitinkama diena (palei stulpelį). Tai 10 pav. grafo chromatinis indeksas bus lygus - 7, kuris nusako, jog mažiausiai 7 spalvų reikalinga norint dvidalio grafo briaunas. Nuspalvinto dvidalio grafo briaunos žymės pamokas - jų išdėstymą, tenkinantį reikalavimų sąlygas. Įvertinus grafo G reikalingą spalvų kiekį (chromatinį indeksą) galima nesunkiai surasti tinkamą briaunų spalvinimą.

Suformuotas dvidalis grafas yra paverčiamas į briauninį grafą $L(G)$, kurio viršūnės atitinka būtent briaunų spalvinimo tikslą turint G. Praktiškai išbandyti 3 modelio formulavimai, kuriuos gali išspręsti kvantinis D-Wave kompiuteris ir jo išsprendimo būdai: vadinamuoju kvantinis pakaitinimo simuliatoriumi (angl. quantum annealing simulator), naujuoju diskrečiojo kvadratinio modelio išsprendėju ir anksčiau aptartame 1.4.1 skyrelyje D-Wave QPU paverčiant uždavinį į QUBO formuluotę.

Yra galimybė suformuluotą QUBO modelį pagal 2.2 skyrelyje aprašytą ir realizuotą algoritmą spręsti atliekant simuliavimą 2 būdais - vienas iš jų siekia atkartoti papildomai taip pat ir mažojo įtraukimo problemą, o antrasis tiesiog simuliuoja vadinamąjį kvantinio pakaitinimo procesą. Pirmasis simuliavimo principas yra ganėtinai neefektyvus, nes loginės problemos pateikimas į fizinės realizacijos kubitų struktūrą yra, dėl techninių priežasčių, sudėtinga problema. Antrasis simuliatorius veikia analogiškai išskyrus, jis neatkartoja techninės D-Wave QPU realizacijos aspektų - neturi mažojo įterpimo problemos (angl. minor embedding).

Siunčiant uždavinį tiesiogiai į D-Wave QPU visada iškyla mažojo įterpimo problema ir reikalinga ją išpręsti iš anksto arba taikyti alternatyvų pasirinkimą - naudoti hibridinį kvantinį algoritmo išsprendimo variantą, kurio metu pati D-Wave sistema automatiškai apsprendžia, kurią problemos dalį spręsti tiesiogiai naudojant QPU.

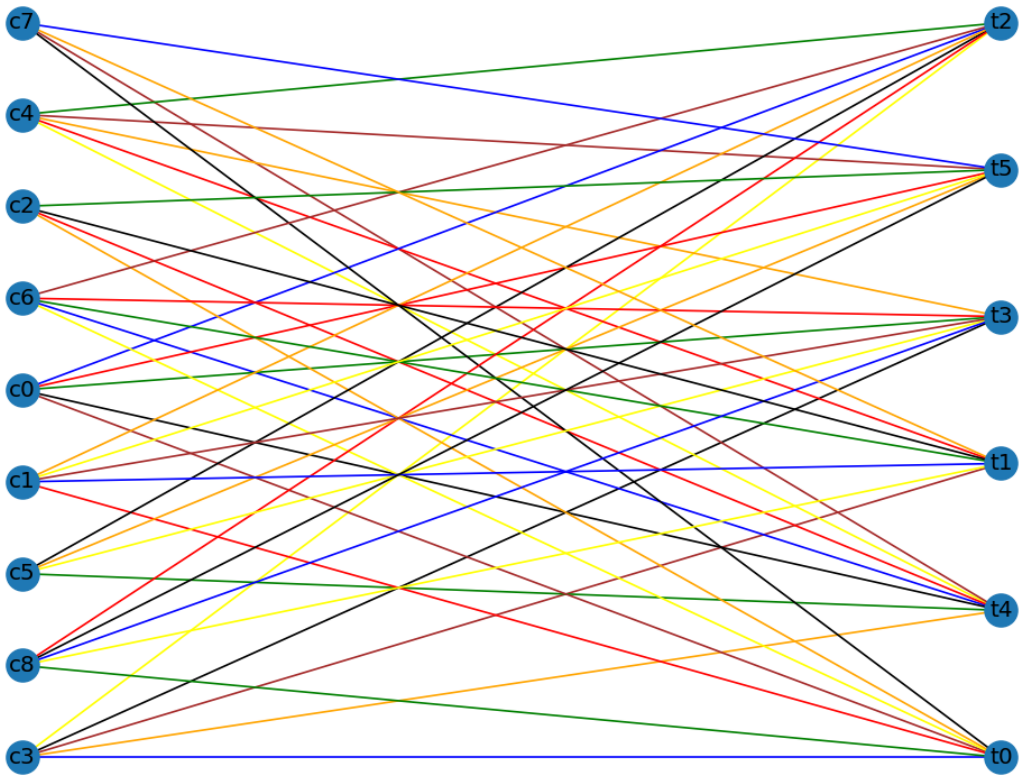
Sekantis išbandytas metodas grafo spalvinimo algoritmui buvo pritaikytas vadinamuoju diskrečiuoju kvadratiniu modeliu. Šio modelio esminis skirtumas yra toks, jog ieškomi nežinomieji dydžiai yra pažymėti galimomis diskrečiomis reikšmėmis:

$$E(\mathbf{v}) = \sum_{\mathbf{i=1}} \mathbf{a_i v_i} + \sum_{\mathbf{i<j}} \mathbf{b_{i,j} v_i v_j} + \mathbf{c} \quad \mathbf{v_i} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots\}.$$

Kur $\{A, B, C, \ldots\}$ yra reikšmės, užkoduotos $\{3.2, 67, \ldots\}$. Šis lengvesnis formulavimas leidžia išspręsti įvairesnes problemas bei yra tinkamas išspręsti grafo spalvinimo uždavinį. Grafo viršūnių spalvinimo algoritmas naudojant DQM formulavimą susideda iš šių realizavimo dalių:

- Inicializuoti n nežinomųjų, kiek yra grafo viršūnių.
- Kiekviena G viršūnė, gali įgyti vieną iš $\{0, 1, 2, \ldots, 6\}$ reikšmių, kurios atspindi galimas k-spalvinimo reikšmes.
- Su kiekviena G briauna suformuoti kvadratinius baudos atvejus, neleidžiant įgyti vienodas spalvines reikšmes bendroms viršūnems.
- Išsiųsti į D-Wave suformuotą uždavinį ir įvertinti gautą rezultatą.

Žemiau pateikiamas gautas iš 10 pav. grafo briaunų spalvinimo rezultatas:



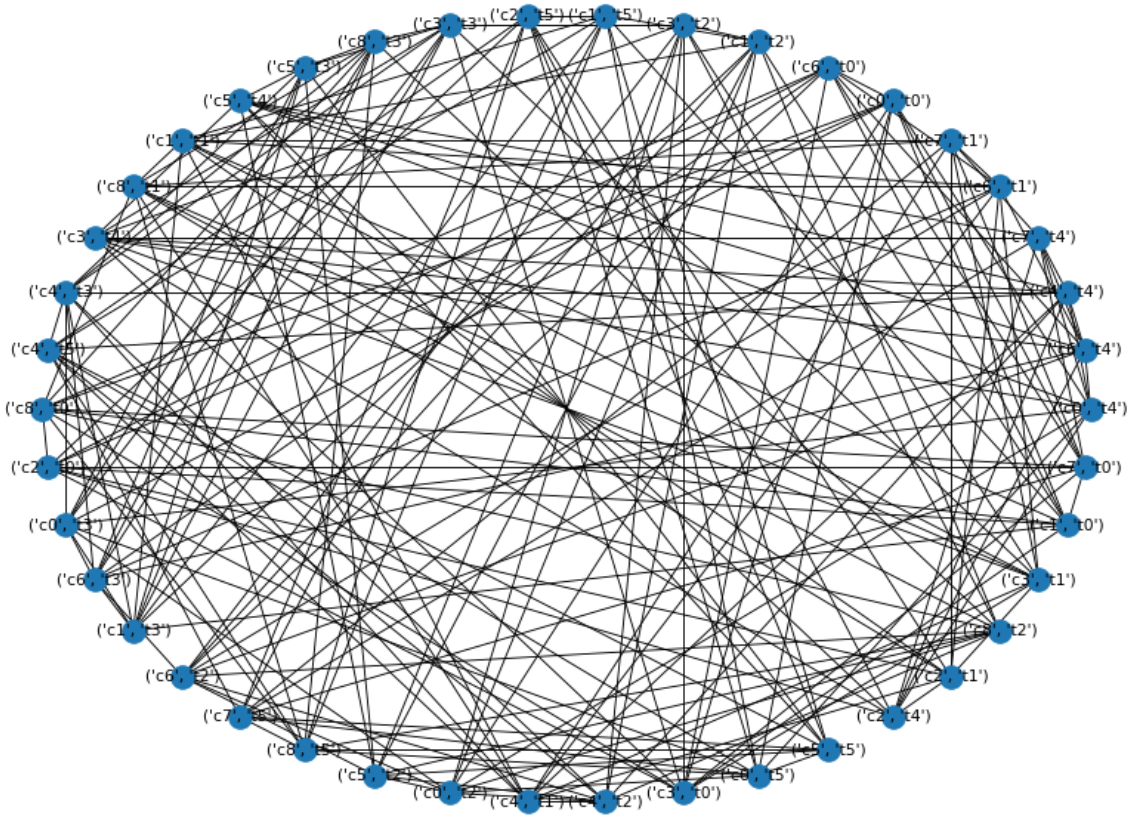
11 pav. Grafo briaunų spalvinimo rezultatas iš 10 pav. gauto ir suformulavus DQM uždaviniu

	p1	p2	p3	p4	p5	p6	p7
c0	t3	t5	t2	0	0	t4	t0
c1	0	t0	t1	t5	t2	0	t3
c2	t5	t4	0	0	t0	t1	0
c3	0	0	t0	t2	t4	t3	t1
c4	t2	t1	0	t4	t3	0	t5
c5	t4	0	0	t3	t5	t2	0
c6	t1	t3	t4	t0	0	0	t2
c7	0	0	t5	0	t1	t0	t4
c8	t0	t2	t3	t1	0	t5	0

12 pav. Vienos dienos pamokų tvarkaraštis gautas iš 11 pav. briaunų spalvinimo rezultato

11 pav. pateiktas vienos dienos dvidalio grafo briaunų rezultatas. Šiam grafui, kaip minėta, reikalinga mažiausiai 7 spalvų, kad sprendinys būtų korektiškas ir kiekviena skirtinga spalva atitinka skirtingą pamoką, laiko tarpą (angl. period). Galima iš karto pastebėti, kad briaunų spalvinimas kartojasi, tačiau tai nepažeidžia būtino reikalavimo ir galima suprasti, kad tuo pačiu metu (toje pačioje pamokoje) skirtingos klasės c_i su skirtingu mokytoju t_j neturės konfliktuojančių užsiėmimų. O 12 pav. pateikiamas glaustesnis, bet atitinkantis analogišką grafo vizualizuotą rezultatą, kuriame galima paprasčiau įvertinti ar nėra pažeidžiamas, konfliktas - mokytojas negali vienu metu dalyvauti daugiau nei 1 pamokoje. Šį apribojimą galimą nesunkiai įvertinti žvelgiant į 12 pav. tvarkaraštį ir ieškant mokytojo t_j pasikartojimų (dubliavimų) per stulpelius - pamokas. 0 reikšme pažymėtas įrašas reiškia, kad atitinkamą pamoką p_q (kur l pamokų skaičius) tarp klasės c_i ir mokytojo t_j nevyks joks užsiėmimas.

Svarbu paminėti, kad šis vienos dienos tvarkaraščio rezultatas yra sėkmingai gautas iš D-Wave hibridinio diskrečiojo kvadratinio modelio išsprendėjo (angl. hybrid DQM sampler) suformuoto briauninio grafo uždavinio. Šiame pateiktame 11 pav. yra būtent 42 briaunos - pavertus į briauninį grafą jos taps 42 viršūnėmis ir turės 204 briaunas minėtasis $L(G)$.



13 pav. Briauninis grafas gautas iš 10 pav. surasto dvidalio grafo

13 pav. pateikiamas gautas briauninis grafas, kuriame reikia nuspalvinti viršūnes, o jos atitiks dvidalio grafo briaunų spalvinimo problemą. Kaip galima įvertinti grafas yra ganėtinai tankus, jis turės tiek viršūnių, kiek yra išskaičiuota pamokų $noLect_k$ per atitinkamą dieną ir tokio dydžio problemą D-Wave hibridinis išsprendėjas randa lengvai.

Dažnai galima rasti CT^k sprendinius, kuriuose gaunama, jog mokytojas j turės su klase i 2 pamokas tam tikrą k dieną. Tokiu atveju gautą dieninį sprendinį siūloma išskaidyti į dvi grupes - išrinkti pamokas taip, kad maksimalus pirmojo dvidalio grafo laipsnis būtų nedidesnis už 4, o antrojo - ne didesnis už, tokiu būdų yra galima be konfliktų surasti tvarkaraščio išdėstymą.

4. Rezultatai

Žemiau pateikiami įvairių dieninių tvarkaraščių sugeneruoti rezultatai iš D-Wave hibridinio išsprendėjo.

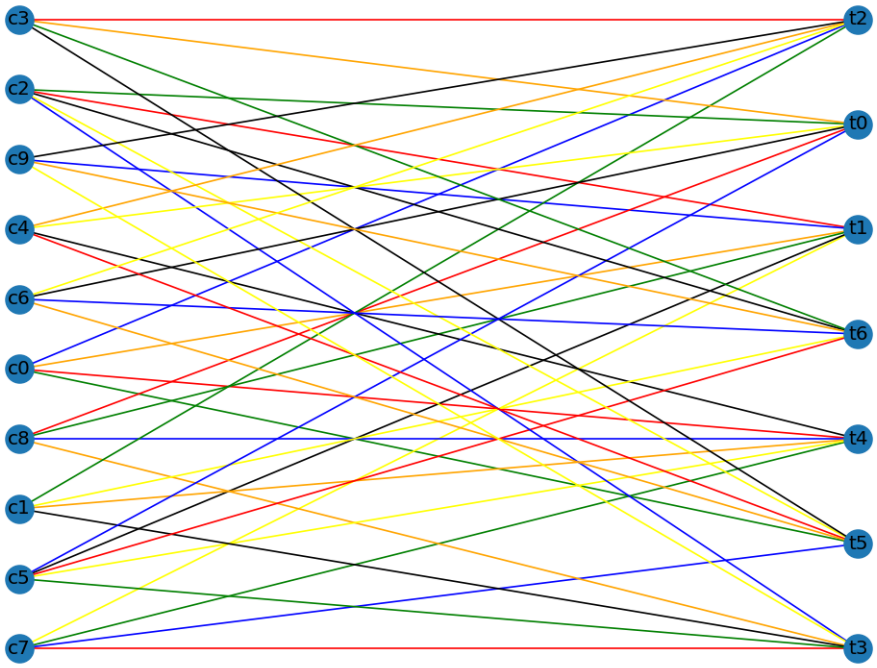
	p1	**p2**	**p3**	**p4**	**p5**	**p6**	**p7**
c0	t21	t0	t18	t1	t13	t26	0
c1	t35	t27	t29	0	t26	t2	t16
c2	t19	t11	t23	t32	t31	t9	0
c3	t0	t19	0	t23	t22	t11	t3
c4	t20	t1	t10	t2	t28	0	t15
c5	t25	t3	t12	t7	t23	t0	0
c6	0	t35	t32	t10	t2	t7	t5
c7	t8	t29	t13	t15	t33	t4	t1
c8	t33	t10	t35	t4	t15	t18	t2
c9	t14	t26	t31	0	t19	t24	t23
c10	t18	t33	t27	t24	t8	t34	0
c11	t29	t34	t28	t8	0	t20	t12
c12	t2	t4	t5	t13	t29	t1	t8
c13	0	t20	t11	t31	t35	t21	t34
...	„	...	...	...	...	...	...
c32	t31	t7	t1	t35	t21	t5	t33
c33	t3	t32	t9	t6	t14	t16	t11
c34	t11	0	t30	t19	t16	t32	t0
c35	t10	t8	0	t26	t5	t35	t20
c36	t15	t13	t33	t21	t12	t31	t28
c37	t32	t23	t17	0	t20	t19	t25
c38	t34	t22	t19	t3	t7	t6	t4
c39	0	t2	t4	t34	t10	t15	t21

6 lentelė. Vienos dienos sugeneruotas tvarkaraštis su 40 klasių ir 36 mokytojais - nurodant bendrą pamokų skaičių lygų 252

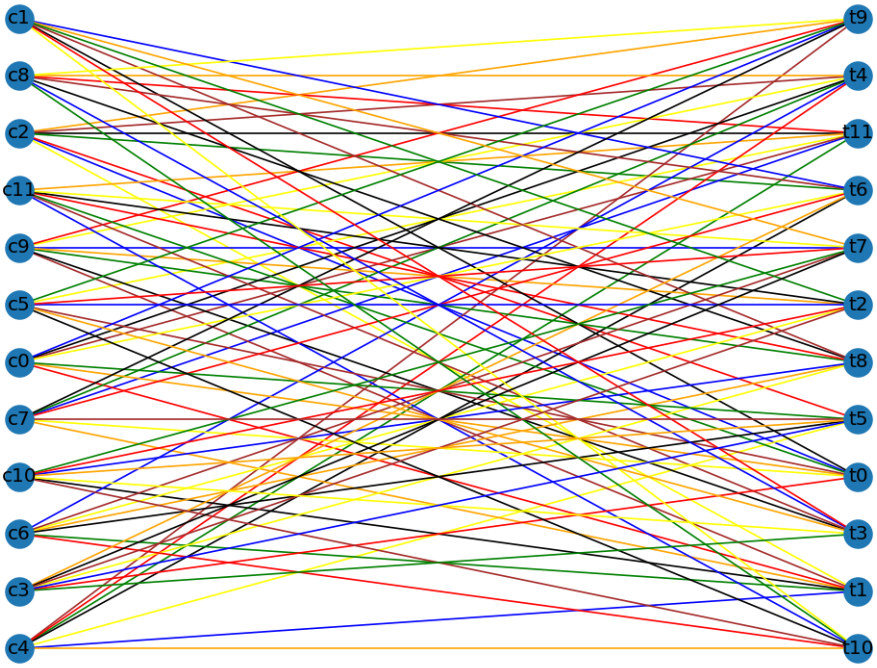
	p1	p2	p3	p4	p5	p6	p7
c0	t53	0	t29	t42	t11	t48	t16
c1	t60	t53	t35	t26	0	t66	t20
c2	t25	t2	t7	t39	0	t47	t44
c3	t33	t28	t60	t9	t53	0	t10
c4	t6	t11	t66	0	t33	0	t69
c5	t20	t1	t12	t32	t58	0	t18
c6	t69	t24	t13	0	t48	t54	t29
c7	t11	t54	t19	t28	t69	0	t21
c8	t17	t44	t31	t56	t1	t20	0
c9	t28	t33	t26	0	t61	t19	t48
c10	t65	t43	t46	t21	t44	t71	0
...	...	...	...	..	...	...	...
c74	t23	t15	t47	0	t29	t49	t53
c75	t35	t51	t39	0	t41	t10	t66
c76	0	t12	t59	t5	t35	t32	t60
c77	t56	0	t38	t20	t4	t65	t52
c78	0	t59	t71	t46	t57	t68	t24
c79	t63	t55	0	t67	t21	t7	t9
c80	t41	t52	t3	t48	t25	t14	0
c81	t22	t5	t68	t8	t56	0	t54
c82	t45	t18	t8	t31	0	t27	t12
c83	t39	t56	t23	0	t27	t41	t70
c84	t68	t35	0	t58	t36	t39	t37

7 lentelė. Vienos dienos sugeneruotas korektiškas tvarkaraštis su 85 klasėmis ir 72 mokytojais - nurodant bendrą pamokų skaičių lygų 504

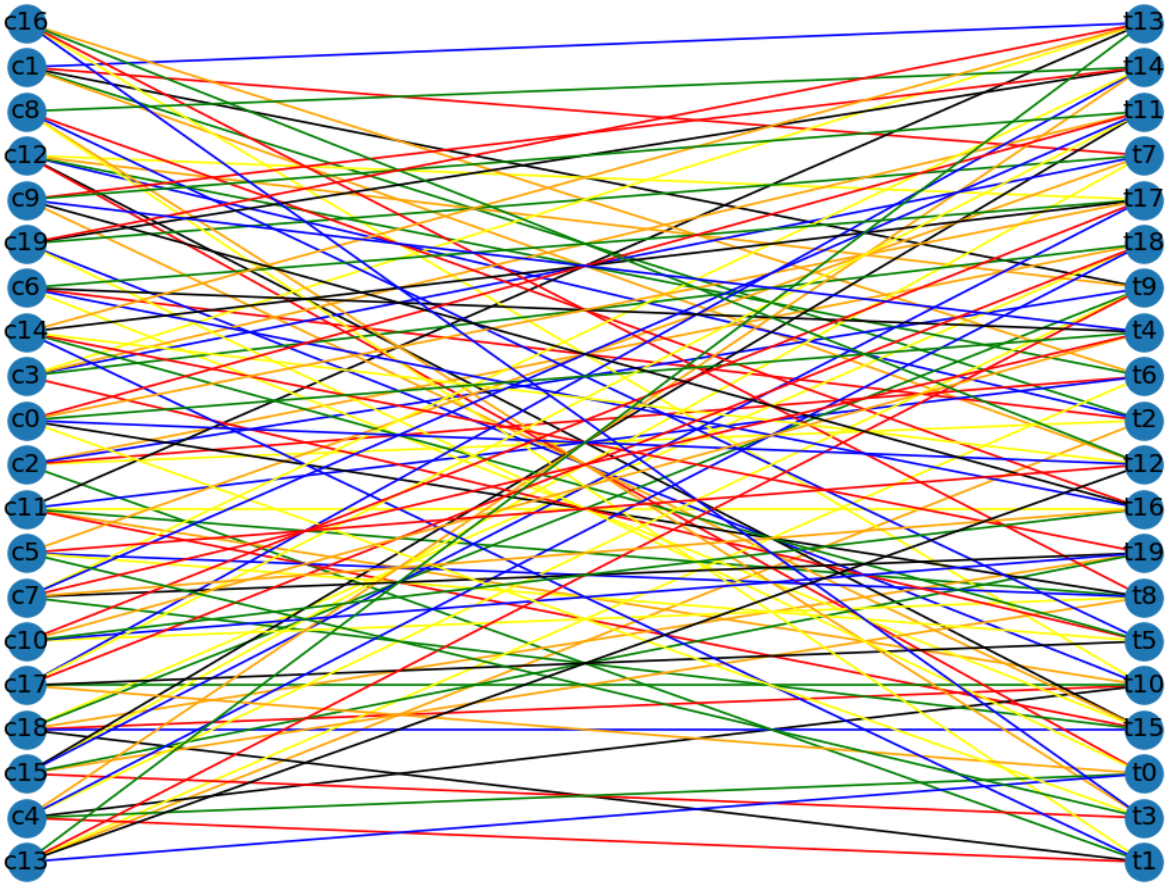
7 ir 6 lentelėse pateikiami gauti vienos dienos tvarkaraščiai, kurių korektiškumą įvertinama iteratyviai ieškant per stulpelius (pamokas) pasikartojančių to paties mokytojo t_j elementų ir, jeigu nustatomas toks atvejis - laikoma, kad sprendinys yra neteisingas. Paminėtina, kad ekvivalentūs dvidaliai susitikimų grafai turėjo tiek briaunų, kiek yra pamokų tarp visų klasių ir mokytojų atitinkamą dieną. Tai savo ruožtų reiškia. jog į D-Wave hibridinį išsprendėją buvo suformuoti briauniniai $L(G)$ grafai iš tų pastarųjų dvidalių grafų. Įvertinta, jog 504 viršūnių spalvinimo uždavinys hibridiniui išsprendėjui yra ganėtinai nesudėtingas ir daug laiko neužimantis veiksmas. Tokio dydžio problemai D-Wave duoda laiko sprendimui iki 5 sekundžių, bet negarantuoja, kad atsakymas bus validus.



14 pav. Grafo briaunų nuspalvinimas turint 10 klasių ir 7 mokytojų



15 pav. Grafo briaunų nuspalvinimo sprendinys turint 12 klasių, mokytojų bei 84 pamokų problemai



16 pav. Grafo briaunų nuspalvinimo sprendinys turint 20 klasių, mokytojų bei 110 pamokų problemai

Pateiktuose 15, 14, 16 pav. pateikiami įvairių problemos dydžių išspręsti, naudojantis hibridiniu išsprendėju, vienos k tvarkaraščiai. Visuose pažymėtuose pavyzdiniuose rezultatuose buvo siekiama gauti maksimaliai įmanomą krūvį arba klasėms, arba mokytojams - jeigu laikoma, kad per dieną galima turėti 7 pamokas, tai išskaičiuojamas nesunkiai didžiausias galimas bendras pamokų kiekis per visą savaitę TL, bet tai yra laisvai keičiami išankstiniai dydžiai.

Pažymėtina, kad atlikus įvairius skaičiavimus, naudojantis D-Wave kvantiniu hibridiniu išsprendėju, nustatyti šie aspektai: grąžinamas rezultatas nėra garantuojama, kad bus korektiškas, šis išsprendėjas neturi simuliacinės bibliotekos ar įrankio atitikmens, D-Wave paslaugos yra mokamos. Turint omenyje šiuos minusus, nebuvo galima pilnai ištirti ir įvertinti D-Wave hibridinio išsprendėjo galimybes, dėl jo riboto pasiekiamumo ir simuliatoriaus neturėjimo.

Išvados ir rekomendacijos

Mokyklos tvarkaraščio sudarymo modeliavimas yra ganėtinai sudėtingas uždavinys. Formuluojant tvarkaraščio sudarymo problemą buvo siekta įvertinti ir nustatyti, kaip efektyviai, patikimai dalį problemos gali išspręsti kvantinis kompiuteris. Bendrąja prasme šis tvarkaraščio sudarymas taip pat ir kiti planavimo uždaviniai priklauso prie kombinatorinių optimizavimo problemų sričių. Tai reiškia, kad šioms problemos nėra žinomų efektyvių ir tikslių išsprendimo algoritmų. D-Wave vystomi kvantiniai kompiuteriai nėra laikomi universaliais kvantiniais kompiuteriais, bet jie naudoja procesą, vadinamą kvantiniu pakaitinimu (angl. quantum annealing). Šio proceso tikslas - išvystyti sistemą (sprendinį) į žemiausios energijos būseną. Išnaudojant šio proceso savybes galima puikiai panaudoti performuluotas optimizavimo problemas į dvejetainius kvadratinius modelius arba naudoti hibridinius išsprendėjus, kurie darbo metu pademonstravo gana gerus rezultatus ir teikia neblogas perspektyvas. Tačiau yra svarbu įsivertinti rezultatų teisingumą, nes vien šis veiksmas gali būti brangus skaičiavimų prasme.

Darbe buvo sėkmingai realizuotas tvarkaraščio sudarymo modelis ir antrosios problemos dalies pamokų išdėstymas, rastas naudojantis hibridiniu kvantiniu išsprendėju, kuris pateikė sąlyginai gerus rezultatus.

Įvertinus skirtingo dydžio siunčiamus uždavinius į hibridinį D-Wave išsprendėją, galima daryti šias išvadas:

- Išsprendėjas negarantuoja, jog pateiktas sprendimas bus validus, bet galima patikrinti korektiškumą. Jeigu nustatoma, kad gautas nekorektiškas rezultatas, galima modifikuoti algoritmo parametrus ir siųsti spręsti dar kartą. Toks sprendimo būdas kainuoja daug resursų, nes išsprendėjas, kol kas neturi simuliacinės aplinkos.
- Grafo spalvinimo problema yra nesudėtingai pritaikoma kitiems sunkiai išsprendžiamiems uždaviniams - komunikacinių stotelių ir jų dažnių priskyrimui, egzaminų tvarkaraščio sudarymui ar kitoms optimizavimo problemoms.
- Yra tinkamas (su tam tikromis išlygomis) spręsti didelės apimties problemas - uždavinius, turinčius tūkstančius kintamųjų.

Pažyminėtina, kad tvarkaraščio sudarymo modelis nėra pilnai išspręstas, turint omenyje, jeigu dieninėje klasės mokytojo reikalavimų matricoje turime 2 pamokas tarp mokytojo ir konkrečios klasės kažkurią dieną. Kad būtų patenkinama ir minėtoji sąlyga, būtų prasminga numatyti strategiją arba algoritmą, kuris leistų išspręsti ir šį atvejį. Vienas iš tokių galimų sprendimų - dvidalio grafo skaidymas į 2 mažesnius grafus arba siūlymas taikyti kitokį grafo spalvinimo algoritmą, kuris žymėtų briaunas, turinčias svorį.

Ateities tyrimų planas

Realizavus tvarkaraščio sudarymo modelį ir jį išsprendus hibridiniu sprendėju, būtų prasminga ir naudinga išplėsti modelį šiais scenarijais:

- Reikalinga performuluoti arba išskaidyti dvidalio grafo formulavimą taip, kad būtų galima išspręsti turint 2 pamokas per dieną tam tikrai klasei su tam tikru mokytoju.
- Nustatyti ir įvertinti parametrus, kurie leistų patikimai išspręsti didesnės apimties problemą, naudojantis hibridiniu išsprendėju.
- Palyginti D-Wave hibridinio išsprendėjo efektyvumą arba spartą prieš kitus klasikinius ir egzistuojančius išsprendimo algoritmus.

Literatūros šaltiniai

[1] Azure quantum. https://azure.microsoft.com/en-us/services/quantum/. Pasiektas: 2020-04-03.

[2] Qiskit dokumentacija. https://qiskit.org/documentation/. Pasiektas: 2020-06-14.

[3] *DWave Quantum Computing*, 2020 (pasiekta 2020-12-23).

[4] *IBM Q Quantum Computers*, 2020 (pasiekta 2020-12-23).

[5] *IonQ Quantum Computing*, 2020 (pasiekta 2020-12-23).

[6] *Rigetti Quantum Computing*, 2020 (pasiekta 2020-12-23).

[7] Andreas Atter, Martin Grötschel, and Arie Koster. Frequency planning and ramifications of coloring. *Discussiones Mathematicae Graph Theory*, 22, 02 1970.

[8] Dave Bacon and Wim van Dam. Recent progress in quantum algorithms. *Commun. ACM*, 53(2):84–93, February 2010.

[9] Rakesh Badoni and D. Gupta. A graph edge coloring approach for school timetabling problems. *International Journal of Mathematics in Operational Research*, 01 2013.

[10] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 11–20, New York, NY, USA, 1993. Association for Computing Machinery.

[11] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors, 2020.

[12] Tomas Boothby, Andrew D. King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *CoRR*, abs/1507.04774, 2015.

[13] Florent Bouchez, Alain Darte, Christophe Guillon, and Fabrice Rastello. Register allocation: What does the NP-completeness proof of chaitin et al. really prove? or revisiting register allocation: Why and how. In *Languages and Compilers for Parallel Computing*, pages 283–298. Springer Berlin Heidelberg, 2007.

[14] Benjamin J. Brown. A fault-tolerant non-cli ord gate for the surface code in two dimensions. *Science Advances*, 6(21), 2020.

[15] J. Cai, W. Macready, and Aidan Roy. A practical heuristic for finding graph minors. *ArXiv*, abs/1406.2741, 2014.

[16] Patrick J. Coles, Stephan J. Eidenbenz, Scott Pakin, Adetokunbo Adedoyin, John Ambrosiano, Petr M. Anisimov, William Casper, Gopinath Chennupati, Carleton Co rin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Andrey Y. Lokhov, Alexander Malyzhenkov, David Dennis Lee Mascarenas, Susan M. Mniszewski, Balu Nadiga, Dan O'Malley, Diane Oyen, Lakshman Prasad, Randy Roberts, Philip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter Swart, Marc Vu ray, Jim Wendelberger, Boram Yoon, Richard J. Zamora, and Wei Zhu. Quantum algorithm implementations for beginners. *CoRR*, abs/1804.03719, 2018.

[17] D. Coppersmith. An approximate fourier transform useful in quantum factoring, 2002.

[18] D-Wave The Quantum Computing Company. *D-Wave System Documentation*, 2021.

[19] Nike Dattani. Quadratization in discrete optimization and quantum mechanics, 2019.

[20] Runa Ganguli and S. Roy. A study on course timetable scheduling using graph coloring approach. 2017.

[21] Fred W. Glover and Gary A. Kochenberger. A tutorial on formulating QUBO models. *CoRR*, abs/1811.11538, 2018.

[22] Arne L. Grimsmo, Joshua Combes, and Ben Q. Baragiola. Quantum computing with rotation-symmetric bosonic codes. *Phys. Rev. X*, 10:011058, Mar 2020.

[23] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[24] Gary A. Kochenberger and Fred Glover. A unified framework for modeling and solving combinatorial optimization problems: A tutorial. In *Multiscale Optimization Methods and Applications*, pages 101–124. Kluwer Academic Publishers.

[25] Zhipeng Lü and Jin-Kao Hao. Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200:235–244, 01 2010.

[26] WT Lee. Tridiagonal matrices: Thomas algorithm. *MS6021, Scientific Computation, University of Limerick*, 2011.

[27] Nuno Leite, Fernando Melicio, and Agostinho Rosa. A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122, 12 2018.

[28] Hongyan Li, Xianfeng Ding, Jiang Lin, and Jingyu Zhou. Study on coloring method of airport flight-gate allocation problem. *Journal of Mathematics in Industry*, 9(1), September 2019.

[29] Andrew Lyons. Acyclic and star colorings of cographs. *Metal Finishing*, 159, 03 2011.

[30] Yunseong Nam, Yuan Su, and Dmitri Maslov. Approximate quantum fourier transform with o(n log(n)) t gates. *npj Quantum Information*, 6(1), March 2020.

[31] Engineering National Academies of Sciences and Medicine. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC, 2019.

[32] Young-Hyun Oh, Hamed Mohammadbagherpoor, Patrick Dreher, Anand Singh, Xianqing Yu, and Andy J. Rindos. Solving multi-coloring combinatorial optimization problems using hybrid quantum algorithms, 2019.

[33] R. Raghavjee and N. Pillay. An application of genetic algorithms to the school timetabling problem. volume 338, pages 193–199, 01 2008.

[34] Rushil Raghavjee and Nelishia Pillay. A study of genetic algorithms to solve the school timetabling problem. In Félix Castro, Alexander Gelbukh, and Miguel González, editors, *Advances in Soft Computing and Its Applications*, pages 64–80, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[35] E. Rie el, Davide Venturelli, Bryan O'Gorman, Minh Do, E. Prystay, and V. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14:1–36, 2015.

[36] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.

[37] Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

[38] Swamit S. Tannu and Moinuddin K. Qureshi. Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 987–999, New York, NY, USA, 2019. Association for Computing Machinery.

[39] Christos Valouxis, Christos G Gogos, and P. Alefragis. Decomposing the high school timetable problem. 2012.

Priedai

Dokumentą sudaro du priedai: A priede pateikiamas visas $CT^k = \left[ct_{ij}^k\right]_{m\times n}$ skaičiavimo išeities kodas

A. CT sprendimo išeities kodas

```
from collections import defaultdict
import numpy as np
import pandas as pd
from ortools.sat.python import cp_model


def rastiCTMatricas(m, n, u, R, CD, TD, galimosReiksmes = [0, 1, 2]):
    # Sukurti modeli.
    model = cp_model.CpModel()


    salyguSkaicius = 0
    # Sukurti kintamuosius.
    x = [ model.NewIntVarFromDomain(cp_model.Domain.FromValues(
            galimosReiksmes), f'x{i}') for i in range(n*m*u)]
    for i in range(m):
        for k in range(u):
            v = [n*i + j + k*(n*m) for j in range(n) ]
            model.Add(sum([x[el] for el in v]) == CD[i, k])
            salyguSkaicius += 1


    for j in range(n):
        for k in range(u):
            indeksas = j*u + k
            v = [ j + n*i + (n*m)*k for i in range(m) ]
            model.Add(sum([x[el] for el in v]) == TD[j, k])
            salyguSkaicius += 1


    for i in range(m):
        for j in range(n):
            indeksas = i*n + j
            v = [ indeksas + k*(n*m) for k in range(u) ]
            model.Add(sum([x[el] for el in v]) == R[i, j])
            salyguSkaicius += 1

    # Sukurti išsprendeja
    solver = cp_model.CpSolver()

    solver.parameters.max_time_in_seconds = 60.0
    status = solver.Solve(model)
    if status == cp_model.OPTIMAL:
        reiksmes = []
```

```
        for i in range(n*m*u):
            reiksmes.append(solver.Value(x[i]))

        susmulkintiGabalai = np.reshape(reiksmes, (m*u, n))
        CT = blockshaped(np.asarray(susmulkintiGabalai), m, n)
        print(f"Sąlygų skaičius - {salyguSkaicius}")
        return CT
    else:
        print("Neegzistuoja sprendinys")
        return None;
```