Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Gráfelméleti algoritmusok beágyazása D-Wave kvantumszámítógépbe

TDK dolgozat, 2017

Szabó Dániel

Konzulens: dr. Friedl Katalin
Számítástudományi és Információelméleti Tanszék

Tartalomjegyzék

Kivonat

Napjainkban egyre gyakrabban hallani híreket a kvantuminformatika világából, például új kvantumszámítógépekről vagy kvantumkommunikációs áttörésekről. Ezekkel összefüggésben egyre fontosabb szerep juthat a közeljövőben a kvantumalgoritmusoknak.

A kvantuminformatika a kvantummechanikából ismert jelenségeken alapul, amelyek szerint a mikrorészecskék mozgása valószínűségi módon írható le. Ilyen részecskével valósítható meg egy kvantumbit: egyszerre van a két, jól megkülönböztethető klasszikus állapot (0 és 1) szuperpozíciójában, az egyikben p, a másikban $1 - p$ valószínűséggel. Ilyen módon egy n kvantumbites rendszer egyszerre 2^n állapotban van, ennek leírására (azaz az egyes állapotok valószínűségének eltárolására) klasszikus esetben 2^n-nel arányos darabszámú bit kell. Ettől lehet hatékonyabb sok probléma megoldása kvantumszámítógéppel, mint klasszikusan: pl. prímtényezőkre bontás, rendezetlen halmazban keresés[1][2].

A kanadai D-Wave Systems vállalat volt az első a világon, amely kvantumszámítógépet adott el, a legutóbbi számítógépük 2048 kvantumbites[3]. Bár megoszlanak a vélemények arról, hogy valóban kvantumos elven működő számítógépekről van-e szó, ettől függetlenül érdemes foglalkozni azzal a számítási modellel, amely a gépek mögött van. A modell alapja a Kiméra gráf (Chimera graph), amelybe be kell ágyazni a problémákat[4].

A dolgozat célja, hogy segítségül szolgáljon a modell megértéséhez. Ezen kívül elvégeztem néhány NP-teljes gráfelméleti probléma beágyazását (pl. van-e egy gráfban megadott méretű klikk; van-e olyan részgráfja, ami adott méretű teljes páros gráf). Ehhez már megoldott problémák alapötleteit használtam fel[5], amelyeket szintén ismertetek. Mivel nem kevés hétköznapi probléma megfogalmazható gráfelméleti úton (pl. nagy hálózatokban közösségek keresése), remélhetőleg idővel néhány ilyen feladat hatékony megoldásához vezethetnek a leírtak.

Abstract

Nowadays quantum computing is getting more and more publicity. For example, we can hear news about new quantum computers or breakthroughs in quantum communication. These may cause quantum algorithms to become more and more important in the not too distant future.

Quantum computing is based on quantum mechanical effects which describe the movement of microparticles probabilistically. A quantum bit can be realized with a microparticle which can be in a superposition of 0 and 1 at a time with probability p and $1-p$, respectively. So, an n-quantum bit system encodes 2^n states at a time, while in a classical computer the number of bits used to store the probabilities for each state is proportional to 2^n. That is why several problems can be solved more efficiently on a quantum computer than on a classical one. E.g. factorization, searching in unordered sets[1][2].

The Canadian D-Wave Systems was the first company in the world to sell quantum computers. Their latest model is a 2048-quantum bit system[3]. Although, it is controversial that these computers really work by quantum principles, the computational model that they provide is worth to investigate. This model is based on the Chimera graph, in which we should embed our problems[4].

One goal of the work is to help to understand the model. I embedded some NP-hard graph theoretical problems (e.g. find the maximum clique size in a graph, determine the maximum size of a complete bipartite subgraph). These use some ideas from earlier results[5] that I also present. Plenty of everyday problems can be translated to graph theory problems (e.g. community search in large networks), so there is hope that in time this method will lead to efficient solutions of such problems.

I. Bevezetés

A kvantuminformatika területe egyre népszerűbbé válik, ahogyan születnek az áttörésnek számító eredmények ezen a területen. Idén nyáron például arról hallhattunk, hogy Kínában kvantumkommunikációs műholddal kísérleteztek sikeresen, október elején pedig megvalósult a világ első kvantumkulcsszétosztással titkosított videohívása[6]. Ennek az a jelentősége, hogy kvantumkriptográfiával matematikailag bizonyítottan feltörhetetlen kódokat lehet előállítani a biztonságos kommunikációhoz.

A dolgozat II. részében egy rövid kvantuminformatikai ismertető következik, majd az egyes kvantumszámítógép-modellekről, és a D-Wave cégről – amelynek a számítógépeiben használt modellnek a segítségével fogunk megoldani problémákat – lesz szó. A III. rész azokat a beágyazási módszereket ismerteti, amelyeket majd felhasználunk a konkrét problémák beágyazására. Ezek a gyakorlati alkalmazásokban is előforduló alapvető feladatok kerülnek sorra a IV. részben: maximális klikket, maximális teljes páros részgráfot, és minimális domináns halmazt keresünk tetszőleges gráfokban. Ezek mellett egy általam írt programról is szó lesz, amellyel ellenőrizhetjük a beágyazások helyességét. Az V. részben az addigiak összefoglalása, és továbblépési lehetőségek szerepelnek.

II. Kvantuminformatikai bevezető

Az 1970-es évektől kezdve már beszélhetünk kvantuminformatikáról, de ekkor még alig voltak eredmények. A tudományág az 1980-as években kezdett igazán fejlődni (Feynman[7]), bár ekkor is még elméleti szinten maradt. Az 1990-es években készültek el az első (még csak pár kvantumbites) kvantumszámítógépek. Fontos kvantumalgoritmusok is születtek ebben az évtizedben: pl. Shor prímfaktorizáló- és Grover keresőalgoritmusa, amelyek jóval hatékonyabbak klasszikus társaiknál[8].

1. Kvantummechanikai és -informatikai alapok

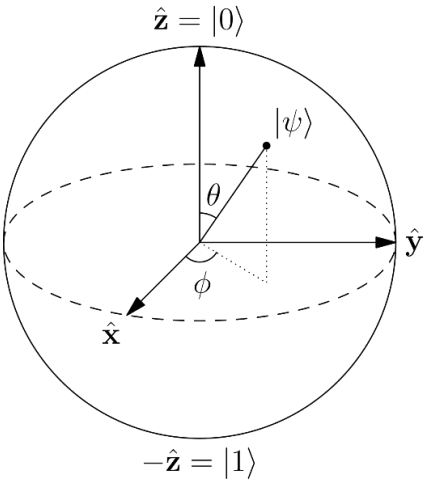
A kvantuminformatika elnevezés onnan származik, hogy olyan számítógépekkel foglalkozik, amelyek működése kvantummechanikai jelenségeken alapul. Ebből a szempontból a legfontosabb jelenség például a kétréses kísérletben figyelhető meg: egy átlátszatlan lemezen két, párhuzamos rés van. Ennek az egyik oldalán található forrásból egyesével bocsátunk ki részecskéket (pl. fotonokat vagy elektronokat). A lemez túlsó oldalán lévő ernyőn interferencia kép alakul ki, tehát a vizsgált részecske egyszerre mindkét résen átmegy, és a fáziseltolódás miatt hol gyengíti, hol erősíti önmagát. A részecske helyzete valószínűségi módon írható le. Ez nem csak a mikrorészecskék pozíciójára igaz, hanem más jellemzőire is, ezt írják le a Heisenberg-féle határozatlansági relációk. További fontos kvantummechanikai jelenségek az összefonódás és az alagúteffektus.

Egy kvantumbit realizálására olyan mikrorészecskét használnak, amelynek valamely jellemzője alapján van két, jól megkülönböztethető állapota (pl. foton két, egymásra merőleges síkú polarizációja). Ez a két állapot lesz a 0 és az 1. A részecske pedig általában ezek szuperpozíciójában van, az egyikben p, a másikban 1-p valószínűséggel, és a mérés az, ami „belekényszeríti” az egyik vagy a másik állapotba (vö. Schrödinger macskája). Pl. az említett kétréses kísérletben, ha a résekhez detektorokat helyezünk, azaz mérést végzünk, akkor már nem alakul ki interferencia kép, csak azon a résen halad át a részecske, amelyik detektora jelez.

A 0 és az 1 egy ortonormált bázist alkotnak, egységvektorokként kezeljük őket. Ezt a két vektort braket-jelöléssel $|0\rangle$ és $|1\rangle$ formában írjuk (kiejtése: „ket null” és „ket egy”). Ebben a bázisban egy egységvektor (a Hilbert-tér egy eleme) egy kvantumbit, ami felírható a $|0\rangle$ és a $|1\rangle$ lineáris kombinációjaként. Az együtthatókat valószínűségi amplitúdóknak nevezzük, általános esetben ezek komplex számok. Így egy kvantumbit felírása:

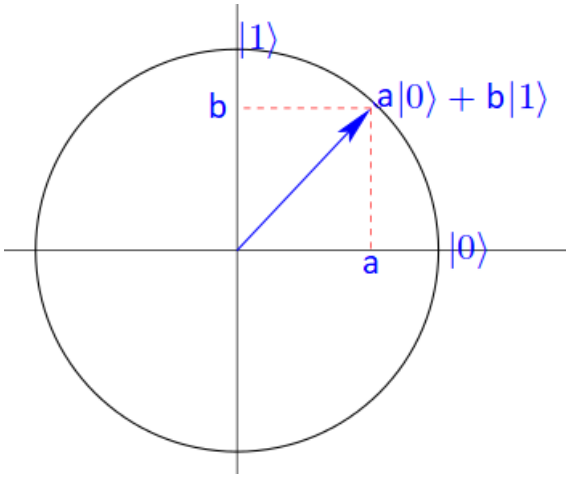
$$a|0\rangle + b|1\rangle,$$

ahol $a, b \in \mathbb{C}$. Ennek vizualizálására a Bloch-gömböt [1. ábra] szokták használni.



1. ábra: A Bloch-gömb vázlata
(forrás: Wikipédia – Bloch sphere)

A mi esetünkben azonban bőven elég a valós együtthatók esetével foglalkozni: ekkor egy síkbeli derékszögű koordinátarendszerben egység hosszú helyvektorokként ábrázolhatjuk a kvantumbiteket [2. ábra]. Az x tengelyen a $|0\rangle$, az y tengelyen a $|1\rangle$ valószínűségi amplitúdóját (rendre a-t és b-t) olvashatjuk le. Mivel egységvektorról beszélünk, teljesül, hogy $a^2 + b^2 = 1$ a Pitagorasz-tétel miatt. Ekkor annak a valószínűsége, hogy a bitet 0-nak mérjük, a^2; annak pedig, hogy 1-nek mérjük, $1 - a^2 = b^2$.



2. ábra: Kvantumbit egyszerűsített ábrázolása
(eredeti kép forrása: TrillionByte – What's a Quantum Bit?)

Több kvantumbit együttese tekinthető egyetlen rendszernek. Két bit esetén négy állapot különböztethető meg, hiszen mindkét bit mérhető 0-nak vagy 1-nek. Egy általános, két kvantumbites rendszer leírásához tehát már négy együttható szükséges:

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$$

Itt látható a kvantuminformatika erőssége: egy n kvantumbites rendszer leírásához klasszikusan 2^n valós (sőt, általános esetben komplex) számot kell eltárolnunk.

Ezért vannak olyan problémák, amelyek kvantumszámítógéppel hatékonyabban oldhatók meg, mint klasszikusan. Például ismert probléma a számok prímtényezőkre bontása, azaz a prímfaktorizáció. A rendkívül elterjedt RSA nyilvános kulcsú titkosítás sem lenne biztonságos, ha a problémára ismert lenne polinomiális algoritmus. Klasszikusan nem ismert, hogy létezne ilyen, azonban Peter Shor 1994-ben megalkotta a hatékony prímfaktorizáló kvantumalgoritmust[9]. Szerencsére azonban egyelőre nem tudunk róla, hogy lenne a jelenleg használt, több száz jegyű prímek szorzatának felbontásához megfelelő kapacitású kvantumszámítógép. Idővel persze lehet, hogy lesz, viszont a kvantuminformatika másik ága, a kvantumkommunikáció segíthet ezen. A bevezetőben említett kvantumkommunikációs hálózat ugyanis a titkos kulcsú titkosítás megvalósításához használható, ami már bizonyítottan feltörhetetlen.

Egy másik példa a rendezetlen halmazban keresés. Egy n elemű halmazban keresünk egy x elemet. Ha a halmaz rendezetlen, nyilvánvaló, hogy (legrosszabb esetben) ehhez minden elemet meg kell vizsgálnunk, hogy egyenlő-e x-szel, azaz klasszikusan a lépésszám n-nel arányos. Lov Grover 1996-ban írta le azt a kvantumalgoritmust, ami $O(\sqrt{n})$ lépéssel megoldja a keresést[10].

Előfordulhat, hogy két kvantumbit olyan állapotban van, amelyben az egyikük bebillenése valamelyik klasszikus állapotba maga után vonja a másikét is. Például az alábbi kétbites rendszerben ½ valószínűséggel mindkét bit 0, és szintén ½ valószínűséggel mindkettő 1. Így, ha az egyiket 0-nak illetve 1-nek mérjük, akkor a másiknak is ugyanilyennek kell lennie, akkor is, ha a mérés pillanatában nagyon távol van egymástól a két kvantumbit. Ezt a jelenséget nevezzük összefonódásnak, a biteket pedig összefonódott bitpárnak.

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

2. Lehűtéses és áramköri modell

Több modell alapján is megvalósíthatók kvantumszámítógépek. Először az áramköri modell alakult ki Deutsch 1989-es cikke[11] nyomán. Ez vált a standard modellé. Ebben kvantum logikai kapuk sorozatából áll elő a számítás. A kvantumbitek működését nehéz összehangolni, ezért egyelőre csak kevés bites áramköri alapú kvantumszámítógépek vannak.

A lehűtéses számítási modell főleg optimalizálási problémák megoldására használatos. Azt használja ki, hogy a fizikai rendszerek mindig az energiaminimumra törekednek. Itt nem mi befolyásoljuk a működést, hanem a probléma betáplálása után hagyjuk, hogy a fizikai törvényszerűségeknek megfelelő változások történjenek[12].

A Hamilton-függvény olyan függvény, amelybe, ha beírjuk a kvantumbitek értékeit, akkor az eredménye ezen állapot energiaszintje lesz. Kezdetben például minden bit

azonos valószínűséggel 0 és 1, ez az állapot könnyen előállítható. Ekkor a paramétereket (csúcs- és élsúlyok) figyelmen kívül hagyó kezdeti Hamilton-függvény számít, ebben az esetben pedig ez a minimális energiájú állapot. Végül a paramétereket figyelembe vevő végső Hamilton-függvény értéke lesz a mérvadó. Az így előálló legalacsonyabb energiaszintű állapot hordozza magában a megoldást. Ekkor már klasszikus bitekről beszélhetünk, ugyanis már nincsenek szuperpozícióban[13].

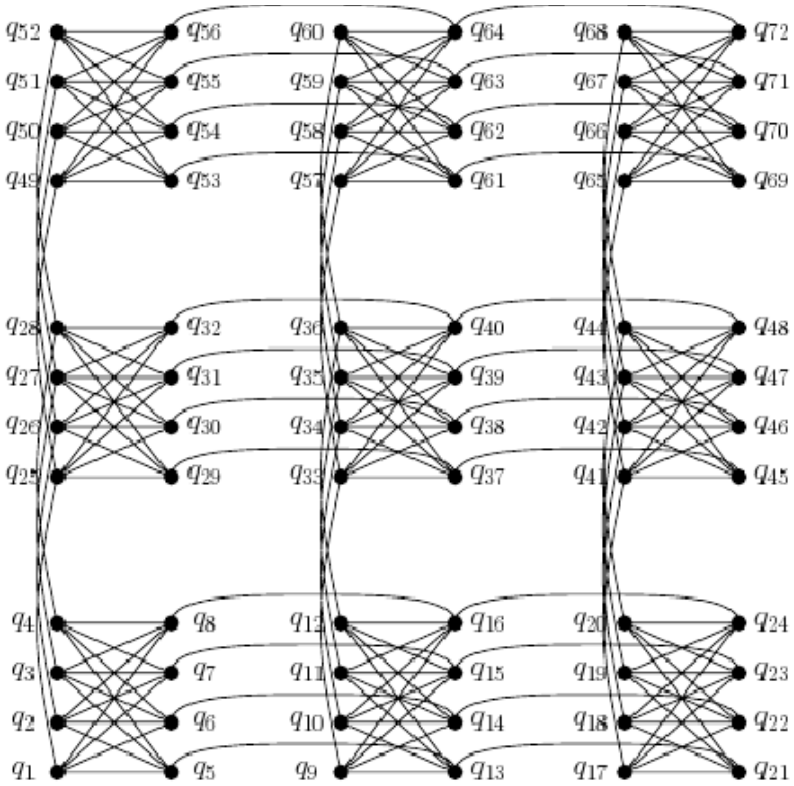
Több megközelítés is létezik még, de ezek ekvivalensnek bizonyultak az áramköri modellel. Az ezeken az alapokon nyugvó számítógépeket nevezzük univerzális kvantumszámítógépeknek. A D-Wave gépei nem univerzálisak[12].

3. D-Wave

A D-Wave egy kanadai cég, amelyet 1999-ben alapítottak, így ez az első kvantuminformatikai vállalat. 2004-ben döntöttek úgy, hogy kvantumszámítógépet építenek. Eleinte magas hőmérsékletű szupravezetőkkel próbáltak megvalósítani kvantumbitet – ezek egyik fatáját hívják d-wave-nek: innen származik a cég neve. 2010-ben adták ki az első kereskedelmi gépüket, a D-Wave One-t, ami 128 kvantumbites volt. A legutóbbi számítógépük a 2017-es D-Wave 2000Q, ami 2048 kvantumbites [3. ábra]. A számítógépeiket használó szervezetek között olyanokat találunk, mint a Volkswagen, a Google vagy a NASA[3][14].

3. ábra: D-Wave 2000Q (Copyright: D-Wave Systems)
(forrás: D-Wave – The D-Wave 2000Q™ System)

A D-Wave gépei lehűtéses kvantumszámítógépek. A számítási modelljük alapja a Kiméra[1] gráf (Chimera graph), amely 4+4 csúcsú teljes páros gráfok ($K_{4,4}$) összekötéséből és rácsba rendezéséből áll [4. ábra]. A kvantumbitek megfeleltethetők a csúcsoknak, az élek pedig az összefonódással állnak kapcsolatban. A bitek fizikai megvalósítása szupravezetőből (nióbium) készült körökkel, és az ezekben folyó áramok által indukált kis mágneses mezőkkel történik[15].



4. ábra: Kiméra gráf
(forrás: [5] FIG. 13)

A csúcsokhoz és az élekhez is rendelhetők súlyok, amelyek minden lehetséges esethez meghatározzák az energiaszintet, és így azt, hogy mit tekintünk a legjobb megoldásnak. A problémák beágyazása egy ilyen gépbe tulajdonképpen ezen súlyok értékeinek meghatározását jelenti. A megoldást valószínűségi alapon kapjuk: a legalacsonyabb energiaszintű állapot előállásának a legnagyobb a valószínűsége. Ezért érdemes többször futtatni, és a kapott eredmények eloszlását figyelni.

[1] Kiméra (Khimaira) a görög mitológia alakja, többek között Szphinx, Kerberosz és a lernéi Hüdra testvére. Jellegzetessége, hogy több állat különböző testrészeiből áll: oroszlán, aminek a hátából kecskefej nő ki, a farka helyén pedig kígyó leng. A gráf elnevezésének oka az lehet, hogy az különálló $K_{4,4}$ gráfokból van összeszerkesztve.
[Forrás: Wikipédia – Chimera (mythology): https://en.wikipedia.org/wiki/Chimera_(mythology) (2017. 10. 09.)]

Már a D-Wave One megjelenésétől vannak, akik szerint ez valójában nem is igazi kvantumszámítógép, ugyanis nem látják bizonyítottnak, hogy tényleg kihasznál-e kvantummechanikai jelenségeket a gép a megoldás meghatározásához. A cég publikált bizonyítékokat, de azok csak közvetve mutatták ki az összefonódás jelenlétét[16]. A kétkedés mértéke 2007 óta csökkent, de nem halt el teljesen[3].

III. Beágyazási módszerek

Ebben a részben néhány általános beágyazási módszer ismertetése következik, amelyeket felhasználtam a konkrét problémákhoz. Ezeket néhol példák is illusztrálják, amelyek segíthetik a megértést.

A D-Wave kvantumszámítógépeinek programozása a hagyományos programozástól jelentősen eltér. Például nincs memóriája, és így állapota sincs, valamint a működése valószínűségi alapú, nem pedig determinisztikus[4]. Olyan módon kell beletáplálni a problémát, hogy meg tudja oldani azt az energiaminimalizálásra törekedve. Az energiaszintet leíró Hamilton-függvény[5]:

$$H(s) = A(s)H_I + B(s)H_P = A(s)H_I + B(s)\left(-\sum_{i=1}^{N} a_i q_i + \sum_{\langle i,j\rangle} b_{ij} q_i q_j\right).$$

Az egyenletben az $s \in [0,1]$ az idő múlását jelzi. H_I a kezdeti, H_P a végső Hamilton-függvény. Az *A(s), B(s)* együtthatók határozzák meg H_I illetve H_P súlyát a pillanatnyi energiaszintben, $A(0) = B(1) = 1$ és $A(1) = B(0) = 0$. A kvantumbitek (avagy a Kiméra gráf csúcsainak) száma *N*, ezek közül az *i*-ediket q_i jelöli, aminek a súlya a_i. A gráf éleinek halmaza $\langle i, j\rangle$; az *i*-edik és a *j*-edik bit (csúcs) közötti él súlya b_{ij}.

Amikor egy potenciális megoldást kapunk, akkor a kvantumbitek már klasszikusnak tekinthetők: bebillentek a 0 vagy az 1 állapotba. Ezért $q_i \in \{0, 1\}$ teljesülni fog, azaz összesen 2^N darab lehetőség van. A cél, hogy ezek közül a „jók" esetén kisebb legyen H_P értéke, mint a kevésbé jók esetén.

1. Közvetlen beágyazás

Közvetlen beágyazás alatt azt értjük, hogy a csúcs- és élsúlyokat direkt módon, a megoldandó problémától függően határozzuk meg. A lehűtési folyamat végén kapott eredményekből (tehát a minimális energiaszinthez tartozó bitértékekből) valahogy ki kell derülnie annak, hogy mi is a megoldás, így ezt kell kódolnunk a bitekbe. A megoldandó probléma meghatároz bizonyos kényszereket, amelyeknek mindenképp teljesülni kell egy megoldásban. Ezeket is be kell táplálnunk a gépbe, erre használjuk a súlyokat.

A későbbiekben érdemes különbséget tenni a logikai és a fizikai kvantumbitek és élek között. A fizikai bitek és élek azok, amik a Kiméra gráfban megjelennek, mint csúcsok, illetve élek. H_P képletében fizikai bitekről van szó. Sok esetben nem elegendők a Kiméra gráf struktúrájából következő fokszámok a bitekhez, ezért vezetjük be a logikai kvantumbitek fogalmát. Egy logikai kvantumbit több fizikaiból állhat, amelyek értékeinek meg kell egyezni. Ennek biztosításához szükséges, hogy az egy logikai bitet alkotó csúcsok összefüggő részgráfot alkossanak a Kiméra gráfban. A logikai bitek között

futnak a logikai élek. Egy logikai él megvalósításához elég, ha a két logikai bit egy-egy fizikai kvantumbitje között fut egy fizikai él.

A következő példa forrása a D-Wave egyik hivatalos kiadványa[4].

A térképszínezés problémáról lesz szó. A feladat az, hogy az egyes területeket kiszínezzük a lehetséges *C* db szín egyikével úgy, hogy azok a területek, amelyeknek van közös határszakaszuk (nem csak különálló pontokban érintkeznek), különböző színt kapjanak. A beágyazás több lépésből fog állni: először a területek színét kell elkódolnunk, majd a szomszédos területekre vonatkozó feltétel teljesülését biztosítjuk.

a. A színek elkódolása

Minden területhez tartozzon *C* db logikai kvantumbit, amelyeket megfeleltetünk a színeknek. (Azért nem lehet fizikai biteket használni, mert a kényszerek biztosításához szeretnénk, ha az egyes színek kvantumbitjei teljes gráfot alkotnának. Ez fizikai bitekkel a Kiméra gráf struktúrája miatt már $C = 3$ esetben sem lehetséges.) Az a logikai bit legyen 1, ami a terület színének felel meg, a többi legyen 0. Ezzel olyan kényszert kaptunk, hogy néhány bit közül egyszerre pontosan egy legyen 1. Elsőként a legkönnyebben megérthető $C = 2$ és $C = 3$ eseteket nézzük, utána pedig az ezekből szerzett tapasztalatok segítségével egy általános megoldást is adunk.

Két bit (azaz két szín: $C = 2$) esetén H_P-nek a most vizsgált kényszerből következő, egy területre vonatkozó része $E = -(a_1 q_1 + a_2 q_2) + b_{12} q_1 q_2$ alakú lehet. Ebben az a_1, a_2, b_{12} súlyokat úgy kell megválasztani, hogy az E függvény értéke kisebb legyen $q_1 \neq q_2$ esetén, mint egyébként. E értékei esetekre bontva:

- ha mindkét bit 0, akkor $E = 0$;
- ha az *i*-edik bit ($i \in \{1, 2\}$) lesz 1, a másik 0, akkor $E = -a_i$;
- ha mindkettő 1, akkor $E = -a_1 - a_2 + b_{12}$.

Például az $a_1 = a_2 = 1$; $b_{12} = 2$ választás esetén a „jó” megoldások energiaszintje -1, a „rosszaké” 0, tehát ez a paraméterezés megfelelő.

A továbbiak egyszerűsítése kedvéért már most is megfogalmazhatjuk egy igényünket. Több bit esetén is az számít jó megoldásnak, ha pontosan egy olyan i létezik, amelyre $q_i = 1$, a többi bit értéke 0. Azt szeretnénk, hogy a különböző jó megoldások azonos eséllyel szerepeljenek. Mivel az esélyt az esetnek megfelelő energiaszint határozza meg, ami a jó megoldások esetén mindig $-a_i$ lesz, ezért feltesszük, hogy $a_1 = a_2 = \cdots = a$. Hasonlóan, szimmetriaokok miatt (két kvantumbit felcserélésével ne változzon meg az eredmény) legyen minden élsúly is azonos: $b_{12} = b_{13} = \cdots = b$.

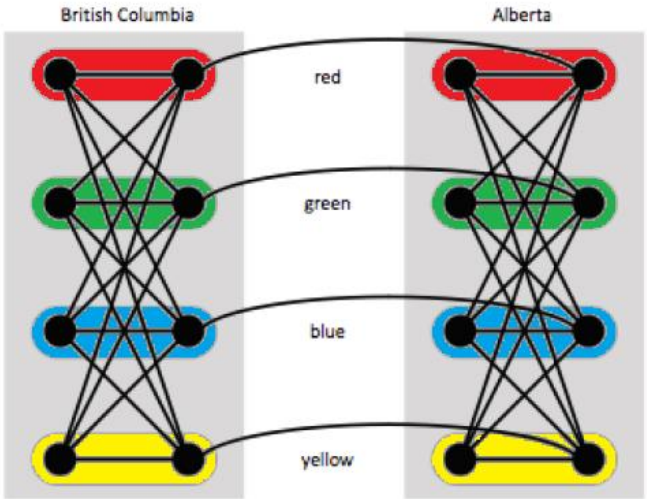
$C = 3$ esetén az ide tartozó energia: $E = -a(q_1 + q_2 + q_2) + b(q_1q_2 + q_1q_3 + q_2q_3)$.

- Ha minden bit 0, akkor $E = 0$;
- ha pontosan egy bit 1, a többi 0, akkor $E = -a$;
- ha pontosan két bit 1, a harmadik 0, akkor $E = -2a + b$;
- ha mindhárom bit 1, akkor $E = -3a + 3b$.

Itt is megfelelő paraméterezés az $a = 1$; $b = 2$: a jó esetekre -1, a rosszakra 0 vagy 3 az energiaszint. Könnyen látszik, hogy a színek számának növelésével, azaz a $C \geq 4$ esetben is ugyanígy eljárva jó megoldást kapunk az $a = 1$; $b = 2$ választással.

Most már beágyazhatjuk a Kiméra gráfba a területek színeit: minden egyes terület feleljen meg a Kiméra gráf egy $K_{4,4}$ teljes páros részgráfjának. A $K_{4,4}$ egy-egy sora (két-két fizikai kvantumbit) feleljen meg az egyes színeknek. (A négyszín-tétel miatt négy szín mindenképp elég egy térkép megfelelő színezéséhez.) Tehát itt több fizikai kvantumbit alkot egy logikait, ezért biztosítanunk kell, hogy ezek a fizikai bitek ne lehessenek eltérő értékűek. A fenti, „1 a 2-ből” mintájára azt látjuk, hogy az $a_1 = a_2 = -1$; $b_{12} = -2$ esetben, ha azonos a két bit, akkor 0, ha különböző, akkor 1 a végső Hamilton-függvény értéke. Ez alapján legyen egy-egy szín logikai bitjén belül a fizikai bitek súlya -1, az ezeket összekötő él súlya -2.

Másrészt szükséges, hogy egy-egy terület $K_{4,4}$ részgráfján belül a négyből csak egy szín legyen kiválasztva. A korábbiak alapján az egyes színek logikai kvantumbitjeinek súlya 1, a köztük futó logikai éleké 2. Ezeket egyenlő arányban osztjuk szét a fizikai megfelelőik között. A logikai bitekhez két fizikai tartozik, ezért a fizikai kvantumbitek ½ súlyt kapnak; és mivel két logikai bit között mindig két fizikai él van, ezek súlya 1 lesz. Ha egy fizikai bit vagy él több ok miatt is kap különböző súlyokat, akkor a Hamilton-függvény linearitása miatt ezek összegét kell venni.



5. ábra: Szomszédos területek leképezése
(forrás: [4] Figure 4)

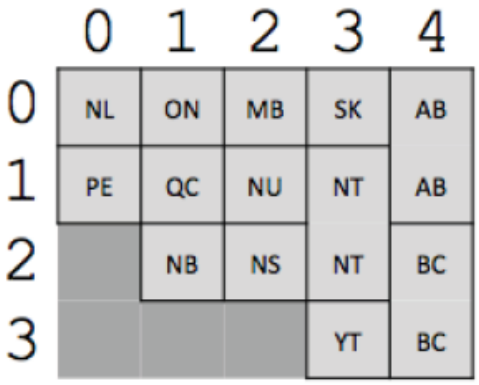
b. A szomszédsági kényszer

Az egyes területek színeit már beágyaztuk, most a szomszédsági kényszert (a szomszédos területek nem lehetnek azonos színűek) fogjuk elkódolni. Legyen két szomszédos területnek megfeleltetett $K_{4,4}$ az A és a B. Ha például A-ban és B-ben a piros szín kvantumbitje egyszerre 1, akkor az energiaszintnek emelkednie kell. Ha azonban mindkét helyen 0, vagy az egyiken 0, a másikon 1 a bit értéke, akkor ne változzon a H_P értéke. A korábbi, „1 a 2-ből” mintájára látjuk, hogy $a_1 = a_2 = 0$; $b_{12} = 1$ jó paraméterezés: a mindkét bit 0 és a pontosan egy bit 0 esetben is 0 az ehhez a kényszerhez tartozó energiaszint, de a mindkét bit 1 esetet büntetjük, ekkor 1-gyel nő a H_P értéke.

Ennek megvalósításához két szomszédos terület azonos színt jelképező logikai kvantumbitjeit összekötő élre van szükség. Az 5. ábrán ívvel jelölve láthatók ezek. Elegendő tehát ezek súlyát 1-re állítani az azonos színű szomszédok büntetéséhez.

Még egy problémával szembesülhetünk: a fent leírt leképezéssel a Kiméra gráf struktúrája miatt minden területnek legfeljebb 4 szomszédja lehet, és három terület nem lehet páronként szomszédos egymással. Ebben segít a területek „klónozása”: egy területnek több $K_{4,4}$-et feleltetünk meg. Ez a több fizikai kvantumbitből álló logikai bitekkel teljesen analóg módon működik, annak megfelelően kell beállítani a súlyokat. Példaként lássuk Kanada térképét [6. ábra], és annak leképezését [7. ábra].

6. ábra: Kanada 13 területegysége
(forrás: [4] Figure 1)



7. ábra: Kanada térképének leképezése
Egy cella egy $K_{4,4}$-nek felel meg
A területek nevét a postai kódokkal rövidítették
(forrás: [4] Figure 5)

Ezek alapján az 5. ábrának megfelelő (tehát klónozás nélküli) esetben a súlyok alakulása:

- a fizikai kvantumbitek esetén a súlyok (kényszerek szerinti bontásban)
  - egy szín logikai bitjén belül megegyező értékű kvantumbitek legyenek: -1
  - egy terület $K_{4,4}$-én belül csak egy szín legyen aktív: 1/2
  - a szomszédsági kényszer miatt: 0
  - ➢ a Hamilton-függvény linearitása miatt ezek összegét kell venni, ami -1/2
- a fizikai élek esetén a súlyok (az élek helyzete szerinti bontásban)
  - egy szín két fizikai bitjét összekötő él: -2
  - egy $K_{4,4}$-en belül különböző színek között futó él: 1
  - két szomszédos terület azonos színt reprezentáló logikai bitjei közötti él: 1

A D-Wave-nél az 512 kvantumbites számítógépbe ágyazták a problémát. A $C = 2, 3, 4$ értékekre futtaták, mindegyikre háromszor, és mindig ezer mintával (azaz ezerféle – nem feltétlenül különböző – kvantumbit-konfigurációval). Az eredményeket mutatja az 1. táblázat. A cellák első értéke a kényszereknek megfelelő jó megoldások száma, a második pedig a kapott különböző eredmények száma. Látható, hogy Kanada térképét két színnel még nem, de hárommal vagy többel már sikerült kiszínezni. Ez megfelel a valóságnak, legkevesebb három szín kell a színezéshez.

A színek száma (C)	Megfelelő színezések/különböző minták
2	0/37, 0/45, 0/34
3	252/417, 228/371, 258/393
4	837/978, 812/947, 801/955

1. táblázat: A futtatások eredménye
(forrás: [4] Table 3)

2. QUBO

A QUBO probléma (Quadratic Unconstrained Binary Optimization problem) a másodfokú, kényszer nélküli, bináris optimalizálási probléma angol rövidítése. Lényege, hogy olyan másodfokú kifejezést akarunk minimalizálni, aminek a változói a {0, 1} halmazból vehetnek fel értéket. A QUBO egy NP-nehéz probléma[17]. Láttuk, hogy a Hamilton-függvény éppen a (végső soron bináris) kvantumbitek másodfokú függvénye, és az értékét minimalizálni szeretnénk. Tehát az energiaszint minimalizálása, és így a kvantumszámítógépnek adott probléma megoldása QUBO problémaként kezelhető.

A fejezet további részének a fő forrása [5].

Az ilyen módon történő beágyazásnak két lépése van: a probléma leképezése QUBO-ra (mapping); és a beágyazás a hardverbe (embedding). A QUBO-ból történő beágyazáshoz használható a D-Wave heurisztikus beágyazó szoftvere, így ezzel a lépéssel itt nem foglalkozunk.

A leképezésre több módszer létezik. Vannak általánosan használhatók (pl. időosztásos, CNF megközelítés), ezek azonban erőforrásigényesek, és így a jelenleg elérhető gépeken csak kis méretű problémák beágyazására alkalmasak. Másik lehetőség, hogy minden problémára külön-külön készítünk QUBO-t, ezt nevezzük közvetlen leképezésnek.

a. Időosztásos módszer

A módszer lényege, hogy lépésekre osztjuk a kezdeti- és a végső állapot közötti intervallumot, és így egyszerre csak egy-egy kisebb változás történik. Valamennyire tudjuk, minek kellene történni, és ami ezzel ellentétes, azt szankcionáljuk. Ezek leírására állapotváltozókat, és akciókat használunk, amelyeket a kvantumbitekbe kódolunk. A büntetések összege adja az energiaszintet.

Például a kezdeti- vagy a végállapotban ismerünk bizonyos feltételeket, amelyeknek teljesülni kell az állapotváltozókra, és ha a bitek között van, ami ennek ellentmond, akkor büntetésképp növeljük az energiaszintet. Hasonlóképpen, ha egy akció végrehajtása előtti és az utána következő állapotban egy bit megváltozott, pedig az akcióból ez nem következik, azt is büntetjük. Valamint az akcióknak lehetnek előfeltételei, és kerülendő, hogy úgy hajtsuk végre azt, hogy ezek nem teljesülnek.

b. CNF megközelítés

A CNF, azaz konjunktív normálforma olyan logikai formula, amelyben a bináris változók *vagy* kapcsolatai szerepelnek zárójelekben, és ezek között *és* kapcsolat van. Ha minden zárójelen belül k db változó van, akkor k-CNF kifejezésről beszélünk. Elérhetők olyan szoftverek (pl. SATPLAN), amelyek elvégzik problémák CNF-ként való megfogalmazását, így abból indulunk ki, hogy a megoldandó probléma CNF-ben adott.

A CNF-ből először PUBO-t gyártunk, ami ugyanaz, mint a QUBO, kivéve, hogy tetszőleges fokú lehet (a P betű a polinom szóra utal). Ezt úgy érjük el, hogy a CNF minden zárójelben lévő kifejezésének egy szorzat felel meg. Ebben a ponált változók helyett 1 és a megfelelő kvantumbit értékének különbsége szerepel, a negált változók helyett pedig a megfelelő kvantumbit. Így például $(a \vee \neg b \vee \neg c \vee d)$ megfelelője $(1-a)bc(1-d)$ lesz. Látható, hogy a kapott kifejezés pontosan akkor lesz 0, ha a neki megfelelő logikai kifejezés igaz, egyébként pozitív az értéke.

Még a fokszámot kell csökkenteni. Az erre használható egyik algoritmus lényege, hogy a szorzatokban leggyakrabban előforduló bitpárokat egyetlen, új kvantumbittel helyettesítjük. Szükséges bevezetni egy büntető tagot, ami azt biztosítja, hogy az új bit pontosan akkor igaz, amikor az eredeti kifejezés is. Ezt addig ismételgetjük, amíg a kapott kifejezés már csak másodfokú.

Erre egy lehetőség a következő: tegyük fel, hogy egy *yzw* tagban az *yz* tényezőt kicseréljük az új *x* változóra. Ekkor elvárjuk, hogy *x* pontosan akkor legyen 1, ha *y* és *z* is 1, egyébként 0 legyen. Ehhez büntető tagnak megfelel az $x(3-2y-2z)+yz$, amit értéktáblázattal ellenőrizhetünk.

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
$x=yz$	1	1	1	0	0	0	0	1
$x(3-2y-2z)+yz$	0	0	0	1	3	1	1	0

2. táblázat: A büntető tag megfelelőségének ellenőrzése

Láthatjuk, hogy a büntető tag pontosan akkor 0, amikor $x=yz$, egyébként pozitív, azaz valóban biztosítja, hogy az új változó értéke ugyanaz legyen, mint az eredeti két változó szorzata. Tehát az *yzw* harmadfokú tag helyett írhatjuk az $xw+x(3-2y-2z)+yz$ másodfokú kifejezést.

Magasabb fokú tagok esetén ugyanezt a lépést többször kell elvégezni, egyszerre mindig csak eggyel csökken a fokszám. Tehát egy n-edfokú tag esetén $n-2$ lépésre – és ugyanennyi új változóra – van szükség, hogy az eredmény másodfokú legyen.

c. Közvetlen leképezés

A közvetlen leképezés nem általános módszer, minden problémára más és más, ezért csak példán keresztül mutatható be. Ez a példa a Hamilton-út keresés lesz (irányítatlan gráfban).

Egy *n* csúcsú gráfhoz vezessünk be n^2 db változót: $x_{ij}=1$ azt jelenti, hogy az *i* jelű csúcs a Hamilton-úton lévő csúcsok sorában a *j*-edik pozícióban szerepel (úgy is fogalmazhatunk, hogy a csúcsokat sorban látogatva az *i* csúcs *j*-edikként kerül sorra);

$i, j \in \{1, 2, \dots, n\}$. Az x_{ij}-k logikai kvantumbitek, ugyanis a (szoftverrel történő) beágyazás után sokan lehetnek hatással egymásra, és így a fizikai bitek közötti élek kevésnek bizonyulnak.

Három megszorítás van, amelyek megszegését büntetni kell.

1) Minden csúcs pontosan egyszer van a Hamilton-úton.
2) Egy pozícióban pontosan egy csúcs van (avagy egyszerre nem látogatunk meg egynél több vagy kevesebb csúcsot).
3) Két, szomszédos pozícióban lévő (egymás után látogatott) csúcs között fut él.

1) Minden *i* csúcsra összegezzük, hogy hányszor látogattuk meg. Ha ez nem egy, büntetünk.

$$\sum_i \left(\left(\sum_j x_{ij} \right) - 1 \right)^2$$

2) Minden *j* pozícióra összegezzük, hogy hány csúcsot látogattunk meg ekkor. Ha ez nem egy, akkor büntetünk.

$$\sum_j \left(\left(\sum_i x_{ij} \right) - 1 \right)^2$$

3) Minden $j < n$ pillanatra összegezzük, ha az *i'* csúcs következik az *i* csúcs után (előbbit a *(j+1).*, utóbbit a *j.* pozícióban látogattuk meg), de nincs köztük él a gráfban.

$$\sum_{j=1}^{n-1} \sum_{i,i' : (i,i') \notin E} x_{ij} x_{i'(j+1)}$$

Látjuk, hogy valóban másodfokú (QUBO) kifejezéseket kaptunk. Ennek a három kifejezésnek az összege lesz a QUBO probléma, amit a megfelelő szoftverrel már be tudunk ágyazni a D-Wave kvantumszámítógépeibe.

3. Ütemezési típusú problémák

A fejezet forrása [5]. Az ún. ütemezési problémák lényege az, hogy az egyes részfeladatok között szétosztjuk a felhasználandó erőforrásokat, és időszeleteket rendelünk hozzájuk. Azon feladatok, amelyek ugyanazt az erőforrást használják, nem futhatnak egyazon időszeletben. Mindeközben teljesülnie kell bizonyos megszorító feltételeknek. Ezek alapján az ütemezési problémák egy része megfeleltethető gráfszínezési problémának. A gráf csúcsai a feladatok, az azonos erőforrást használó feladatok között él fut, eltérő színek jelölik a különböző időszeleteket. Így a kromatikus szám lesz az összes feladat elvégzéséhez minimálisan szükséges időszeletek száma.

Például a gráfszínezési probléma a következőképpen fogalmazható meg ütemezési problémaként. Adott az irányítatlan, n csúcsú G gráf, amelynek csúcsait k db színnel szeretnénk kiszínezni. Feleltessünk meg minden v csúcsnak néhány logikai kvantumbitet:

- k db akció, amelyeket a_v^c-vel jelölünk, jelentése: a v csúcsot c színűre színezzük;
- egy s_v^g állapotváltozó, ami azt jelöli, hogy kiszíneztük-e már v-t;
- k db s_v^c állapotváltozó, ami azt jelöli, hogy a v csúcs c színű-e.

Jelöljük $C(v)$-vel azon csúcsok halmazát, amelyek a bemeneti gráfban szomszédosak v-vel. Minden a_v^c akciónak $|C(v)|+1$ előfeltétele van: egyrészt a v csúcs még ne legyen kiszínezve, azaz s_v^g legyen hamis; másrészt v szomszédai között ne legyen c színű, azaz $\forall\, v_i \in C(v)$ esetén $s_{v_i}^c$ legyen hamis. Minden akciónak van két következménye is: s_v^g és s_v^c igaz lesz.

A kezdeti állapotban minden v-re és minden c-re s_v^g és s_v^c is hamis. A végső állapotban minden csúcs ki kell legyen színezve, azaz minden v-re s_v^g igaz. Egy végrehajtás n db akció sorozatából áll, amelyek mindegyikében egy-egy csúcsot színezünk ki.

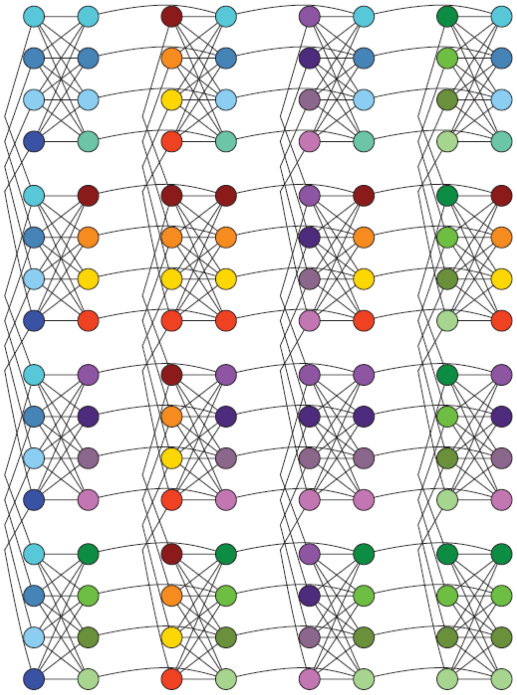
Az általános módszerekkel viszonylag könnyen elvégezhető az ütemezési problémák leképezése QUBO-ra, így, ha egy feladatot meg tudunk fogalmazni ütemezési problémaként, akkor már tekinthetjük úgy, hogy sikerült leképezni (és így – a D-Wave megfelelő szoftverének segítségével – beágyazni is).

4. Teljes gráf beágyazása

Sok esetben nagyon kényelmessé teszi a beágyazást, ha bármely két logikai csúcs között van él, azaz ha mindegyik bit értéke hatással lehet a többire. Ez a Kiméra gráfban a fizikai bitek között sajnos nem teljesül, de ha a megfelelő fizikai kvantumbitekkel valósítjuk meg a logikaiakat, akkor már elérhetjük a célt. A fejezet forrása [18].

A Kiméra gráffal azonos szerkezetű, de általánosabb gráfba fogunk ágyazni. Legyen ez a gráf $K_{4,4}$-ek helyett $K_{c,c}$ gráfokból álló rács. Egy $m \times m$-es rácsba az itt leírt módszerrel be tudunk ágyazni egy $cm + 1$ csúcsú teljes gráfot (azaz K_{cm+1}-et). A fejezet további részében a teljes gráf csúcsaira használjuk a csúcs elnevezést, míg a Kiméra gráf csúcsaira bitekként hivatkozunk – ezek ebben az esetben a fizikai bitek, míg a logikai bitek megfelelnek a teljes gráf csúcsainak.

Az 1×1-es rácsba ($m = 1$ eset), azaz az egyetlen $K_{c,c}$-ből álló gráfba ágyazáshoz legyenek a logikai bitek a $K_{c,c}$ sorai (mint a közvetlen beágyazásnál a színek kódolásánál láttuk), kivéve az utolsó sort, ahol mindkét bit feleljen megy egy-egy új logikai bitnek. Így $c+1$ db logikai bitet kaptunk, és valóban bármely kettő között fut él, azaz a K_{c+1} beágyazását végeztük el.



8. ábra: K_{17} beágyazása 4×4-es rácsba
(forrás: [18] Fig. 6.(b))

Minden további lépésnél eggyel növeljük a rács méretét mindkét irányban, mondjuk jobbra és lefelé. Ekkor az új jobb alsó sarokban lévő $K_{c,c}$ egyes sorainak bitpárjai azonos logikai bithez fognak tartozni, de minden sor másikhoz, és mindegyik a beágyazandó teljes gráf olyan csúcsához, amely az előző méretű rácsban még nem szerepelt (így c db új csúcsot ágyazunk be). A bővítéssel kapott többi $K_{c,c}$ fizikai bitjei annak a beágyazandó csúcsnak a logikai bitjéhez fognak tartozni, amelyikhez közvetlen él köti őket. Mindezt szemlélteti a 8. ábra, ahol a teljes gráf egyes csúcsainak megfeleltetett fizikai bitek (azaz a logikai bitek fizikai bitjei) kaptak azonos színt.

IV. Beágyazások

Ebben a részben kerül sor az általam kigondolt beágyazások ismertetésére. Maximális méretű teljes részgráf, maximális méretű teljes páros részgráf és minimális méretű domináns halmaz keresésével foglalkoztam. Ezen kívül írtam egy egyszerű programot, amellyel a közvetlen beágyazást lehet szimulálni – erről is lesz szó.

1. MAXKLIKK

A MAXKLIKK annak a gráfelméleti problémának a jelölése, amelyben egy bemeneti *G* gráfban a legnagyobb előforduló teljes részgráf (klikk) mérete a kérdés. Ennek az eldöntési változata, amikor *G* mellett egy *k* paraméter is adott, és az a kérdés, hogy van-e a gráfban *k* méretű klikk. A probléma közismerten NP-teljes. Bár a probléma beágyazását már korábban megoldották, én ettől függetlenül jutottam az alábbiakra.

a. Közvetlen beágyazással

Egy *n* csúcsú *G* gráf esetén az *n* csúcsú teljes gráfot ágyazzuk be (az előbbiekben tárgyalt módon) – de csak *G* komplementerének éleit fogjuk használni. *G* csúcsainak súlyai viszonylag kis abszolút értékű pozitív számok legyenek, pl. mindegyik legyen 1. (Mivel pozitívak, a Hamilton-függvényhez negatív értéket fognak adni, hiszen abban a csúcsokra vonatkozó súlyösszeg előtt negatív előjel szerepel.) *G* komplementer éleinek súlyai pedig valamivel nagyobb abszolút értékű pozitív számok, pl. 3; *G* éleinek súlya legyen 0.

Ezek logikai csúcsok és élek súlyai, amelyeket megfelelően szétosztunk a fizikai megfelelők között: egy logikai csúcs fizikai csúcsai között egyenlően osztjuk szét; egy logikai él fizikai megfelelői közül egy megkapja a teljes logikai súlyt, a többi 0 súlyú lesz. (Az eredményen nem változtatna, ha a logikai élek súlyát is szétosztanánk a fizikai megfelelőik között.) A kvantumbitek értéke 1, ha bevesszük a maximális klikkbe a csúcshoz tartozó logikai csúcsot, és 0, ha nem.

Ez valóban megfelelő beágyazás, mert egyrészt, ha nem klikket választunk ki, az nem lehet optimális. Ugyanis ha van olyan csúcs a kiválasztottak között, ami nincs összekötve az összes többi kiválasztottal, akkor ennek a csúcsnak az elhagyásával alacsonyabb energiaszintű állapotba jutnánk. Hiszen a legalább egy darab $\bar{G}$-beli él törlése miatt legalább 3-mal csökken az energia szintje, a csúcs törlése miatt pedig csak 1-gyel nő, más olyan változás pedig nem történik, ami változtatna az energiaszinten.

Másrészt pedig, ha egy kisebb klikket választanánk ki, mint a maximális, akkor kevesebb érték összege lesz a kiválasztott csúcssúlyok összege. A kiválasztott csúcsok között pedig ebben az esetben csak *G*-beli élek vannak, amelyek súlya 0, így ezek nem változtatják az energiát. Mivel a csúcssúlyok összege negatív előjellel számít, nagyobb energiaszintű állapotot eredményez, ha a maximális klikk helyett egy kisebbet választunk ki.

A kvantumszámítógépen való futtatáskor azt figyeljük, hogy a sok minta között van-e olyan, amelyik eredményében a teljes gráf legalább *k* méretű.

b. Ütemezési problémaként

A *G* gráf minden csúcsához tartozik két állapotváltozó: bevesszük-e a klikkbe (x_i), bevehető-e oda (y_i); és egy akció is: hogy bevesszük a klikkbe.

Kezdetben minden *i*-re $x_i = 0$ és $y_i = 1$. Ahhoz, hogy az *i*. csúcsot bevegyük, két feltételnek kell teljesülnie: még ne legyen bevéve ($x_i = 0$), és legyen bevehető ($y_i = 1$). A bevétel következménye, hogy az *i*. csúcs bekerült a klikkbe, és már nem vehető be ($x_i = 1;\ y_i = 0$). Továbbá azok a csúcsok, amelyek nem szomszédosak az *i*. csúccsal, már nem vehetők be, formálisan: ha $(i\ , j) \in E(\bar{G})$, akkor mostantól $y_j = 0$ kell legyen.

Így valóban mindig teljes gráfot alkotnak a bevett csúcsok és a köztük futó élek. A maximális klikk pedig nyilván elérhető, ha az ehhez tartozó csúcsokat vesszük be.

c. QUBO közvetlen leképezéssel

Két, össze nem kötött csúcs egyszerre nem lehet egy klikkben, és ne maradjon bevehető csúcs a végére. Az ezeknek ellentmondó eseteket büntetjük az energiaszint növelésével. Az alábbi képletben az x_i-k és az y_i-k ugyanazt jelölik, mint az előző alfejezetben.

$$\sum_{(i,j)\in E(\bar{G})} x_i x_j + \sum_{\forall i} y_i$$

2. Maximális teljes páros részgráf

Teljes páros gráfon, avagy biklikken olyan páros gráfot értük, amelynek minden olyan csúcspárja között fut él, amelyek közül az egyik csúcs az egyik, a másik pedig a másik csúcsosztályból való. Most azt a problémát vizsgáljuk, hogy egy bemenetként adott tetszőleges *G* gráfban mekkora a legnagyobb méretű feszített teljes páros részgráf. Azaz olyan *A* és *B* diszjunkt részhalmazait keressük a csúcsoknak, amelyekre teljesül, hogy minden *A*-beli csúcs össze van kötve bármely *B*-belivel, de semelyik két *A*-beli, illetve semelyik két *B*-beli csúcs között nincs él. Ennek a méretén a kiválasztott csúcsoknak az $|A| + |B|$ elemszámát értjük. Ennek is az eldöntési változatát tekintjük: van-e *G*-nek olyan feszített teljes páros részgráfja, amelyben a csúcsosztályok méreteinek összege *k*. A probléma NP-teljes[19].

a. Ütemezési problémaként

A tetszőleges *G* gráf minden (*i*-edik) csúcsához tartozik négy állapotváltozó: benne van-e a páros részgráf egyik csúcshalmazában, *A*-ban: x_i^A, benne van-e a másikban, *B*-ben: x_i^B, bevehető-e *A*-ba: y_i^A, bevehető-e *B*-be: y_i^B. Két akció is tartozik a csúcsokhoz: bevétel *A*-ba: a_i^A, bevétel *B*-be: a_i^B.

Kezdetben minden *i*-re $x_i^A = 0;\ x_i^B = 0;\ y_i^A = 1;\ y_i^B = 1$. Az a_i^A előfeltétele, hogy az *i*. csúcs bevehető legyen *A*-ba ($y_i^A = 1$). Következményei, hogy már be van véve ($x_i^A = 1$), már nem vehető be ($y_i^A = 0$). Valamint az összes, még semelyik halmazba be nem vett, de valamelyikbe még bevehető csúcsra ellenőrizni kell, hogy még mindig bevehető-e oda.

Egy csúcs (legyen ez a *j*.) pontosan akkor marad bevehető *B*-be, ha szomszédos az *i*. csúccsal, formálisan: ha $(x_i, x_j) \in E(\bar{G})$, akkor mostantól $y_j^B = 0$ kell legyen. Egy csúcs pontosan akkor marad bevehető *A*-ba, ha nem szomszédos az *i*. csúccsal, formálisan: ha $(x_i, x_j) \in E(G)$, akkor ezentúl $y_j^A = 0$ kell legyen. Az a_i^B akciókkal természetesen hasonló a helyzet, csak az *A*-kat és a *B*-ket kell felcserélni egymással.

Így valóban csak olyan csúcs kerülhet be valamelyik halmazba, amely az adott halmazon belül lévő csúcsok közül egyikkel sem szomszédos, és a másik halmaz csúcsai mind szomszédosak vele, azaz végig biklikket alkotnak a bevett csúcsok. Az is könnyen látszik, hogy ezzel a módszerrel mindenképp elérhető a maximális méretű teljes páros részgráf az ebben benne lévő csúcsok bevételével.

b. QUBO közvetlen leképezéssel

Ha két, szomszédos csúcs egyazon csúcshalmazba került, vagy ha két, nemszomszédos csúcs különböző halmazba került, azt büntetni kell. Valamint ne maradjon egyik halmazhoz hozzáadható csúcs sem. Rendre ezt a három feltételt írja le az alábbi kifejezés három tagja, ahol a változók ez előbbi alfejezetben ismertetetteknek felelnek meg.

$$\sum_{(i,j)\in E(G)} \left(x_i^A x_j^A + x_i^B x_j^B\right) + \sum_{(i,j)\in E(\bar{G})} \left(x_i^A x_j^B + x_i^B x_j^A\right) + \sum_{\forall i} \left(y_i^A + y_i^B\right)$$

Ha nem feszített részgráf a cél, akkor majdnem ugyanez az eredmény, csupán az *A* illetve a *B* halmazon belül futó élek büntetését (azaz az első tagot) kell elhagyni.

3. Domináns halmaz

A domináns halmaz egy *G* gráf csúcsainak olyan *D* részhalmaza, amelyre igaz, hogy minden olyan csúcs, ami nincs benne *D*-ben, legalább egy *D*-beli csúcsnak szomszédja. Az ebből származó optimalizálási probléma az, hogy határozzuk meg a legkisebb elemszámú ilyen halmazt. Eldöntési feladatként megfogalmazva: van-e k méretű domináns halmaz *G*-ben. Ez is NP-teljes probléma[20].

a. Ütemezési problémaként

Az eddigiekhez képest fordítva járunk el: kezdetben a *G* gráf minden csúcsát vegyük be a domináns halmazba, és majd kiveszünk belőle néhányat. Minden csúcshoz tartozik két állapotváltozó: benne van-e a domináns halmazban (x_i), kivehető-e onnan (y_i); és egy akció is: kivétel a halmazból.

Az *i*. csúcs kivételének előfeltétele, hogy kivehető legyen ($y_i = 1$). Következménye, hogy az *i*. csúcs már nincs bevéve a domináns halmazba ($x_i = 0$), már nem is vehető ki ($y_i = 0$). Ezen kívül minden, ekkor még kivehető (*j*.) csúcsra ellenőrizzük, hogy még mindig kivehető-e: ha $\sum_{(x_k, x_j)\in E(G)} x_k = 0$ (azaz a *j*. csúcs egyik szomszédja sincs már bevéve a halmazba), akkor már nem vehető ki, egyébként kivehető marad.

Így valóban csak olyan csúcsok kerülhetnek ki a halmazból, amelyeknek még van szomszédja a bennmaradók között, azaz a bennmaradók mindvégig domináns halmazt fogak alkotni. A legkisebb méretű domináns halmazba nem tartozó csúcsok kivételével pedig nyilván elérhető a minimális méretű domináns halmaz.

b. QUBO a közvetlen leképezéssel kapott PUBO-ból

Azt kell büntetni, ha van olyan csúcs, ami nincs bevéve a domináns halmazba, és ugyanez igaz az összes szomszédjára is. Ezen kívül a végére ne maradjon kivehető csúcs. A következő képletben az x_i-k és az y_i-k jelentése az előző alfejezetben leírtaknak felel meg.

$$\sum_{\forall i}\left((1-x_i)\prod_{(i,j)\in E(G)}(1-x_j)\right)+\sum_{\forall i}y_i$$

Ez a kifejezés nem másodfokú, az első tagban ugyanis eggyel több kvantumbit szorzata szerepel, mint ahány szomszédja van az *i*-edik csúcsnak – ez pedig kettőnél jóval több is lehet. Tehát egy PUBO-t sikerült felírnunk. A CNF megközelítés (III/2/b) alfejezetben írtaknak megfelelően ez már QUBO-vá alakítható.

Megismételjük, hogy a módszer lényege az, hogy egy-egy bitpárt egy új változóval helyettesítünk, és egy (másodfokú) büntető taggal biztosítjuk, hogy az új változó értéke valóban az eredeti kettő szorzata legyen. Ezt addig ismételjük, amíg a kapott kifejezés másodfokú nem lesz.

Nézzük például az $x_1x_2x_3 \dots x_k$ tagot. Először az x_1x_2 helyett vezessük be az y_1 tényezőt. Így az $y_1x_3 \dots x_k + y_1(3-2x_1-2x_2)+x_1x_2$ kifejezést kapjuk. A következő lépésben az $y_2 = y_1x_3$ tényezőt használjuk a fokszám további csökkentésére. A kapott kifejezés: $y_2x_4 \dots x_k + y_1(3-2x_1-2x_2)+x_1x_2+y_2(3-2y_1-2x_3)+y_1x_3$.

Ezt addig folytatjuk, amíg végül az alábbi másodfokú kifejezést nem kapjuk.

$$y_{k-2}x_k + y_1(3-2x_1-2x_2)+x_1x_2+\sum_{i=2}^{k-2}(y_i(3-2y_{i-1}-2x_{i+1})+y_{i-1}x_{i+1})$$

Ezt minden, kettőnél nagyobb fokszámú tagra elvégezzük, és így az egész PUBO-t átalakíthatjuk QUBO-ra.

4. Beágyazást szimuláló program

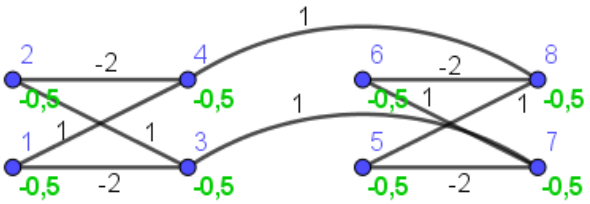
A Java nyelven írt szoftver[21] működése annyiból áll, hogy a bitek összes lehetséges értékére megnézi az energiaértékeket, és ezek közül a minimálisakhoz tartozó konfigurációkat kiírja. Épp ezért nagyobb bemenetekre rendkívül lassúvá válik – éppen az a kvantumszámítógépek jelentősége, hogy azok nem exponenciális futásidejűek az ilyen problémák esetén, szemben a klasszikus társaikkal.

A program bemenetként a beágyazáshoz használt Kiméra gráf méreteit és a súlyokat várja el. Ezért leginkább a közvetlen beágyazáshoz használható segédeszközként. Nem csak a D-Wave-nél használt Kiméra gráfot lehet használni, tetszőleges méretű teljes páros gráfokból állhat, és ezek is tetszőleges méretű rácsot alkothatnak.

A programot kipróbáltam mindkét, a dolgozatban szereplő közvetlen beágyazással megoldott problémára (térképszínezés és MAXKLIKK).

a. Térképszínezés

Az első kísérlet két szomszédos terület két színnel színezése volt. Ehhez egy 1×2-es, $K_{2,2}$ részgráfokból álló rácsba kell ágyazni. A Közvetlen beágyazás fejezetben (III/1) írtaknak megfelelően minden csúcs súlya -0,5; a $K_{2,2}$-n belül az egyes sorok két-két bitjét összekötő élek súlya -2; az összes többi él súlya pedig 1. A 9. ábrán az éleken azok súlya feketével, a csúcsokon azok sorszáma kékkel, a súlyuk pedig zölddel szerepel.



9. ábra: Térképszínezés 2 területtel, 2 színnel

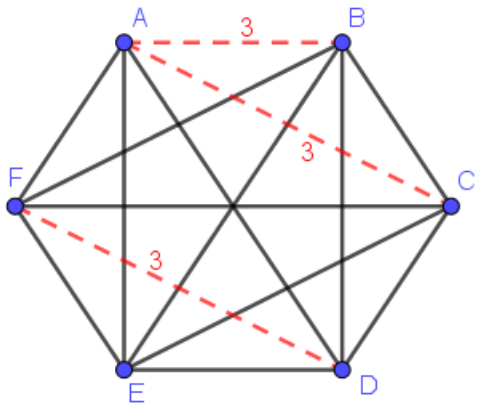
A futás eredménye: akkor lesz minimális az energiaszint, ha a 2, 4, 5 és 7 sorszámú, vagy akkor, ha az 1, 3, 6 és 8 sorszámú bitek értéke 1. Ez megfelel annak, amit várunk: akkor és csak akkor a legalacsonyabb az energia szintje, ha a két terület különböző színt kap.

Három színre is kipróbáltam ugyanezt, ekkor hat esetben kapunk minimális energiaszintet. Valóban, az első terület bármilyen színű lehet a 3 lehetőség közül, de ekkor a második színe már csak a maradék két szín közül választható ki, ami 6 eset. A kapott esetek tényleg a jó megoldásokat írják le.

Olyan esetet is néztem, ahol nincs megoldás (három, páronként szomszédos terület színezése két színnel). Ekkor rengeteg (192 db) olyan eset volt, ahol minimális az energiaszint, és ha egyről ellenőrizzük, hogy az nem megoldás, és feltételezzük, hogy az összes minimális energiaszintű állapot jó megoldás, ha van ilyen, akkor könnyen láthatjuk, hogy ennek a feladatnak nincs megoldása.

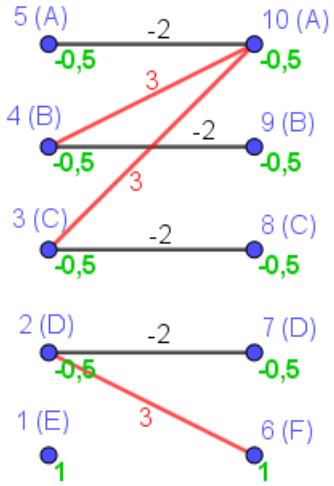
b. MAXKLIKK

A MAXKLIKK probléma esetén először egy hat csúcsú gráffal próbálkoztam [10. ábra]. Az ábrán a gráf élei a fekete színű élek, a piros szaggatottak a gráf komplementerének élei. A MAXKLIKK-et Közvetlen beágyazással megoldó alfejezetben (IV/1/a) írtaknak megfelelően a piros szaggatottal jelölt élek súlya 3, a többié 0. A csúcsok súlya pedig 1.



10. ábra: A MAXKLIKK próbagráfja

A beágyazáshoz Teljes gráf beágyazását használunk (III/4). A D-Wave Kiméra gráfjából hat csúcsú teljes gráf beágyazásához a korábban kifejtett módszerrel 2×2-es rácsra lenne szükség. De én kihasználtam, hogy a programban megadható a Kiméra gráf teljes páros részgráfjainak mérete. Így egyetlen $K_{5,5}$-be ágyaztam a gráfot: minden sor megfelel egy-egy csúcsnak (logikai bit), kivéve az alsó sort: itt mindkét fizikai bit egy-egy önálló logikai bit, azaz a bemeneti gráf egy-egy csúcsa. A 11. ábrán a bitek mellett kékkel a sorszámuk és zárójelben az eredeti, hat csúcsú gráf neki megfelelő csúcsának betűjele szerepel; zölddel pedig a bitek súlya. A nulla súlyú élek nincsenek feltüntetve az ábrán, a nem nulla súlyú élekre azok súlya van írva.



11. ábra: MAXKLIKK beágyazása

A bitek súlya egyrészt a MAXLIKK probléma közvetlen beágyazásából ered: a legtöbb fizikai bit ebből kap 0,5 súlyt, mivel a logikai csúcsok súlya 1, és ez két fizikai bitből áll. Kivétel az 1. és a 6. bit, ezek 1-1 súlyt kapnak. A súlyok másik forrása annak a kényszernek a biztosítása, hogy az egy logikai bithez tartozó fizikai bitek értéke azonos legyen. Ez a Közvetlen beágyazás fejezetben (III/1) írtak alapján minden fizikai bitre, amelyik egy másikkal alkot egy logikai bitet, -1. Így az 1. és a 6. bit súlya marad 1, a többi pedig $0{,}5 - 1 = -0{,}5$ lesz.

Az élek közül a feketék súlya az egy logikai biten belüli fizikai bitek konzisztenciájából adódik: a III/1-ben írtak alapján ez a súly -2. A piros élek a 10. ábra piros éleinek felelnek meg. A beágyazást firtató alfejezetben ugyanis azt írtuk, hogy a logikai bitek között futó fizikai élek közül egyetlen, tetszőleges él kapja meg a teljes súlyt, a többi súlya 0 lesz. Az eredményen nem változtatna, ha az élsúlyt is szétosztanánk a fizikai élek között (pl. a 4. és 10. bit közötti 3 súlyú él helyett egy 4. és 10. közötti 1,5, és egy 5. és 9. közötti 1,5 súlyú és lenne).

A program futásának eredménye: az 1, 3, 4, 6, 8 és 9 vagy az 1, 2, 3, 4, 7, 8 és 9 bitek legyenek 1 értékűek. Azaz a BCEF vagy a BCDE a megoldás: valóban ez a két K_4 részgráf van a bemeneti gráfban, ennél nagyobb klikk pedig nincs.

c. Futásidő

Mivel a szoftver minden lehetséges konfigurációt ellenőriz, a futásidőt legfőképp a bitek száma határozza meg. Kísérleteim alapján egy 28 bites térképszínezés még egy perc alatt lefutott, de 32 bites esetben fél óra sem volt elég a feladat elvégzéséhez. Innen is látszik az exponenciális futásidő, ami miatt ilyen problémák esetén szívesebben fordulunk egy kvantumszámítógéphez – vagy ha ilyen épp nincs kéznél, akkor valamilyen heurisztikus algoritmushoz.

V. Összegzés

A tárgyalt problémák beágyazásának abban áll a jelentősége, hogy valós problémák (vagy ezek bizonyos részei) megfogalmazhatók ilyen módon, azaz valódi problémák hatékony megoldását teszik lehetővé a leírt beágyazások.

A MAXKLIKK és a maximális teljes párosítás probléma például hálózatokban meglévő „közösségek” keresésében segíthet. A domináns halmaz pedig a hálózatoknak a támadásokkal szembeni ellenállóképességének ellenőrzésében válhat hasznossá. (Pl. egy elemű domináns halmaz: SPOF – Single Point Of Failure – egyetlen csomópont kiesése tönkre teheti az egész hálózatot.)

Természetesen sok más, klasszikusan nehéz probléma beágyazása, vagy ezeknek egy másik kvantuminformatikai modell segítségével történő hatékony megoldása hasonlóképpen jól használható lenne. Ezért érdemes foglalkozni a kvantuminformatikával. Szerencsére a mai világban az is motivációt nyújt ehhez, hogy egyre több területen látjuk kézzel fogható jelét e tudományterület jelentőségének.

További munka tárgyát képezheti a leírt leképezések és beágyazások optimalizálási lehetőségeinek vizsgálata. Ez például konkrétan abban valósulhat meg, hogy minél kevesebb fizikai kvantumbitet használjuk egy-egy logikai bit reprezentálására. Ezzel ugyanis elérhetjük, hogy adott méretű Kiméra gráfba nagyobb bemeneti gráf is beágyazható legyen, és így nagyobb méretű problémák hatékony megoldására nyílik lehetőség.

Irodalomjegyzék

[1] Hirvensalo, M. (2004). *Quantum Computing (2. kiadás).* Verlag Berlin Heidelberg: Springer.

[2] Nielsen, M., & Chuang, I. L. (2001). *Quantum Computing and Quantum Information.* Cambridge: Cambridge University Press.

[3] Wikipédia – D-Wave Systems: https://en.wikipedia.org/wiki/D-Wave_Systems (2017. 09. 20.)

[4] Dahl, E. D. (2013). Programming with D-Wave: Map Coloring Problem. https://www.dwavesys.com/sites/default/files/Map%20Coloring%20WP2.pdf

[5] Rieffel, E. G., Venturelli, D., O'Gorman, B., Do, M. B., Prystay, E., & Smelyanskiy, V. N. (2014). A case study in programming a quantum annealer for hard operational planning problems. *arXiv*, 1407.2887v1. https://arxiv.org/pdf/1407.2887.pdf

[6] Nordrum, A. (2017. 10. 03.). China Demonstrates Quantum Encryption By Hosting a Video Call. *IEEE Spectrum*. https://spectrum.ieee.org/tech-talk/telecom/security/china-successfully-demonstrates-quantum-encryption-by-hosting-a-video-call

[7] Feynman, R. P. (1982). Simulating physics with computers. International Journal of Theoretical Physics Vol. 21, Issue 6-7, pp 467–488.

[8] Wikipédia – Timeline of quantum computing: https://en.wikipedia.org/wiki/Timeline_of_quantum_computing (2017. 10. 01.)

[9] Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, pp 124-134. https://dl.acm.org/citation.cfm?id=1399018

[10] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of the 28th annual ACM Symposium on Theory of Computing, pp 212-219. https://dl.acm.org/citation.cfm?id=237866

[11] Deutsch, D. (1989), Quantum computational networks. Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences, Vol. 425, pp 73-90. http://rspa.royalsocietypublishing.org/content/425/1868/73

[12] YouTube – What is Quantum Annealing?: https://youtu.be/zvfkXjzzYOo

[13] YouTube – Physics of Quantum Annealing - Hamiltonian and Eigenspectrum: https://youtu.be/tnikftltqE0

[14] D-Wave – Meet D-Wave: https://www.dwavesys.com/our-company/meet-D-Wave

[15] D-Wave – Introduction to the D-Wave Quantum Hardware: https://www.dwavesys.com/tutorials/background-reading-series/introduction-D-Wave-quantum-hardware

[16] Lanting, T. et al. (2014). Entanglement in a Quantum Annealing Processor. Physical Review X, Vol. 4, 021041. https://journals.aps.org/prx/abstract/10.1103/PhysRevX.4.021041

[17] Wikipédia – Quadratic unconstrained binary optimization: https://en.wikipedia.org/wiki/Quadratic_unconstrained_binary_optimization (2017. 10. 10.)

[18] Klymko, C., Sullivan, B. D., & Humble, T. S. (2012.). Adiabatic Quantum Programming: Minor Embedding With Hard Faults. arXiv, 1210.8395v2. https://arxiv.org/pdf/1210.8395.pdf

[19] Yannakakis, M. (1978). Node-and edge-deletion NP-complete problems. Proceedings of the 10th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, pp 253–264. https://dl.acm.org/citation.cfm?id=804355

[20] Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman & Co.

[21] A program kódja (Java nyelven) használati útmutatóval és példabemenetekkel elérhető a linken: https://drive.google.com/open?id=0BwoN5CdvjrZKaWJ1TjNsTE5mc0U