Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

Kvantumalgoritmusok véletlen bolyongással

Diplomaterv

Készítette
Kabódi László

Konzulens
dr. Friedl Katalin

2014

Tartalomjegyzék

Kivonat **4**

Abstract **6**

1. A kvantumszámításról **8**
1.1. A kezdetek 8
1.2. A qubit 8
1.3. Több qubit 9
1.4. A mérés 10
1.5. Műveletek a qubiteken 10
1.6. Kvantum teleportálás 12
1.7. Nem-klónozhatósági tétel 13

2. Grover keresőalgoritmusa **15**

3. Véletlen séták **20**
3.1. Klasszikus eset 20
3.2. Kvantum eset 22

4. Markov-láncok kvantum kiterjesztése **26**
4.1. Kvantum Markov-lánc 26
4.2. MNRS-keresés 27

5. A QuIDD rendszer **32**
5.1. A QuIDD adatszerkezet 32
5.2. A Grover-féle keresés lépésszáma QuIDD használatával 36

6. Egy lehetséges felhasználás: 3-SAT **41**
6.1. A 3-SAT probléma 41
6.2. Klasszikus megoldás 41
6.3. Kvantum megoldás 42

7. Összefoglaló **47**

Irodalomjegyzék **47**

HALLGATÓI NYILATKOZAT

Alulírott *Kabódi László*, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2014. december 29.

Kabódi László
hallgató

Kivonat

Jól ismert, hogy egyes klasszikus algoritmusok gyorsíthatóak a véletlen felhasználásával. Ennek egyik eltejedt módja a véletlen séta. Mikor a kvantumalgoritmusok kutatása kezdett elterjedni, rögtön adódott, hogy a kvantum világban is megvizsgálják a véletlen séta alkalmazhatóságát. A kutatások eredménye az, hogy lehetséges a kiterjesztés, és a Markov-láncokat felhasználva hatékony keresőalgoritmus készíthető. A klasszikus megközelítéshez képest a kvantum változat négyzetes gyorsulást is hozhat általános esetekre.

A dolgozat a kvantumalgoritmusok alapjainak ismertetésével indul, kiemelve a fontosabb különbségeket a klasszikus és a kvantumos megközelítés között. Továbbá bemutatja a kvantumvilág néhány érdekes tulajdonságát is. Nevezetesen a kvantum teleportálást, amivel kvantumállapotot lehet kvantumcsatorna nélkül közvetíteni, és a nem-klónozhatósági tételt, ami kimondja, hogy kvantumállapotot másolni nem lehet.

A második fejezet a keresést felgyorsító Grover-algoritmust mutatja be. Az eredeti algoritmus ugyan nem használ véletlen bolyongást, de megmutatható, hogy lehet kvantum Markov-lánccal úgy szimulálni, hogy a hatékonysága ne változzon. Azért is érdemes ezzel az algoritmussal foglalkozni, mert bár Grover algoritmusa nem ad exponenciális gyorsulást, mint Shor híres prímtényezőkre bontó eljárása, de a keresést, mint részfeladatot, sok algoritmus használja. Valamint az ebből általánosított amplitúdó amplifikációs eljárás sok későbbi kvantumalgoritmus alapját képzi.

A harmadik fejezet röviden ismerteti a klasszikus véletlen sétát, és ennek kvantum kiterjesztését. Mivel a kvantumalgoritmusok a kvantumvilágban rejlő rejtett párhuzamosságot használják ki, ez a kiterjesztés nem magától adódó. Viszont általános esetben négyzetes gyorsulás érhető el, valamint néhány speciális esetben akár exponenciális is.

A negyedik fejezet a Markov-lánc kvantum kiterjesztését felhasználó Frédéric Magniez, Ashwin Nayak, Jérémie Roland és Miklos Santha által kidolgozott keresőalgoritmust mutat be [11]. Ez az algoritmus egy Markov-láncból készített kvantum séta operátort és egy orákulumot használ. Az iterációk folyamán a megfelelő elemek amplitúdóját növeli, így az algoritmus végén a méréssel nagy valószínűséggel egy keresett elemet kapunk. Bizonyítható, hogy az eljárás ekvivalens a Grover-kereséssel.

Az ötödik fejezet egy, a kvantumáramkörök szimulációjára kitalált adatszerkezetet mutat be [18], amit a QuIDDPro program is felhasznál [17]. Az eredeti cikkben a lépésszám

bizonyítása elég elnagyoltan szerepelt, itt egy részletes, általam kidolgozott érvelést ismertetek. Ez után megvizsgálom, hogy tudnám-e valahogy optimalizálni a szimulációt. Konkrétan Grover algoritmusán vizsgálom, hogy ha az operátorokat más sorrendben szorzom össze, javul-e a program sebessége.

A témában való elmélyedés és az algoritmus jobb megértése érdekében végigkövettem a korábban ismertetett algoritmus működését egy adott, kis méretű bemenetre. Ezt tartalmazza a hatodik fejezet, amelyben szimuláció futtatásához az előző fejezetben levő QuIDD adatszerkezetet felhasználó QuIDDPro programot használtam. Ehhez nem elegendő a szokásos vázlatos megadása a kvantumalgoritmusban előforduló operátoroknak, ezeket meg kellett alkotnom, pontosan össze kellett állítanom. A 3-SAT problémát választottam szemléltetésre, a feladat ismertetése után az algoritmus lépéseit követem végig egy adott 3-CNF-en.

Abstract

It is well known, that some classical algorithms can be accelerated by using randomization. One common method for this is via random walk. When the research of the quantum algorithms began to spread, it was obvious to study the application of the random walk in the quantum world. It was discovered, that this extension is possible and an efficient search algorithm can be constructed using Markov chains. Compared to the classical approach, a quadratic speed up can be achieved in the general cases.

This paper starts with explaining the basics of the quantum algorithms, highlighting the main differences between the classical and the quantum approaches. Furthermore, it presents some interesting properties of the quantum world. In particular the quantum teleportation, which allows transmission of quantum states without a quantum channel and the no-cloning theorem which states that a quantum state can not be copied.

The second chapter presents Grover's search algorithm. The original algorithm does not use random walk, but it can be shown, that is can be simulated with a quantum Markov chain without losing its efficiency. Although Grover's algorithm does not provide exponential speed up, like Shor's famous algorithm for integer factorization, search can be found in many algorithms as subtask. Also the amplitude amplification, the generalization of Grover's algorithm, forms the basis of many quantum algorithms.

The third chapter briefly describes the classic random walk, and it's quantum extension. This is not trivial, because the quantum algorithms use the hidden parallelism in the quantum world. In the general cases, quadratic speed ups can be achieved, and in some special cases even exponential.

The fourth chapter shows a search algorithm designed by Frédéric Magniez, Ashwin Nayak, Jérémie Roland and Miklos Santha, based on the quantum Markov chains [11]. This algorithm uses a quantum operator based on a Markov chain, and an oracle. During the iterations it amplifies the amplitude of the marked elements, so at the end of the search we obtain a marked element, with a high probability. It can be shown, that this algorithm equivalent to the Grover-search.

The fifth chapter presents a data structure designed to simulate quantum circuits [18], used by the QuIDDPro program [17]. In the original paper, the proof of the complexity was not detailed, in the chapter I show a detailed reasoning I developed. I also examine

whether using the associative property of the operators can optimize the simulation of Grover's algorithm.

In order to get a better understanding of the algorithm, I traced the workings of the algorithm on a small input. The sixth chapter shows this with the help of the QuIDDPro program, which uses the QuIDD data structure, described in the previous chapter. In order to do this, it wasn't sufficient to describe the operators in the usual way, I had to construct them from the basic building blocks. I chose the 3-SAT problem to do this. After describing the problem, I follow the algorithm step-by-step on a given 3-CNF.

1. fejezet

A kvantumszámításról

1.1. A kezdetek

A számítógépek egyik legkorábbi feladata a kódfejtés mellett a fizikai szimulációk végrehajtása volt. Ugyanis a fizikusok az elméleteik ellenőrzésére gyakran használnak a mai napig számítógépes szimulációkat. A megalkotott modell alapján kiszámolják a várt eredményt, amit majd egyeztetnek a kísérletekkel. Viszont a kvantumfizika terjedésével problémába ütközött ezen módszer, ugyanis míg a klasszikus fizikai rendszerek a méretük növekedésével lineárisan, de legfeljebb polinomiálisan több számítást igényelnek, a kvantumrendszerek számítási igénye exponenciálisan növekedik. Ez a probléma Feynmannak is feltűnt, és egy beszédében [5] felvetette, hogy egy kvantumfizikai alapokra épülő számítógép talán megoldhatná ezt a problémát. A következő 10 évben, elsősorban fizikusok, elkezdték vizsgálni egy lehetséges számítási modell tulajdonságait, a benne rejlő lehetőségeket. A következő nagyobb ugrás a kvantumszámítás történetében Shor 1994-es faktorizációs algoritmusa volt, ami egy kvantumszámítógépen polinomiális várható időben tud egy számot prímtényezőkre bontani [15]. Ez a probléma azért nagyon fontos, mert a legelterjedtebb nyílt kulcsú titkosító algoritmus, az RSA biztonsága azon a feltételezésen alapszik, hogy ez a probléma bonyolult, nem oldható meg hatékonyan. Tehát az RSA-t használhatatlanná tenné, ha létezne használható, megfelelő méretű kvantumszámítógép.

1.2. A qubit

Egy kvantumszámítógép az adatokat nem a klasszikusan megszokott biteken, hanem kvantum biteken, azaz qubiteken tárolná. Ennek legelterjedtebb jelölésmódja a Dirac által 1939-ben bevezetett bra-ket jelölés. Ekkor a bra ($\langle x|$) a sorvektorokat jelöli, míg a ket ($|y\rangle$) az oszlopvektorokat. Nagy előnye a rendszernek, hogy ránézésre látható, hogy mely vektorok szorzata lesz skalár ($\langle x|y\rangle$), és melyeké mátrix ($|y\rangle \langle x|$). Nyilvánvaló, hogy a rendszer csak akkor használható, ha minden vektor azonos dimenziójú.

Egy kvantum bit alapvetően máshogy viselkedik, mint egy klasszikus bit. Míg egy klasszikus bit az általa felvehető két állapot egyikében lehet csak, a qubit ezek tetszőleges szuperpozíciójában is. A hagyományos állapotoknak a $|0\rangle$ és $|1\rangle$ ortonormált vektorok felelnek meg. Az általános qubit jelölése az előbb bevezetett bra-ket formalizmussal $\alpha|0\rangle + \beta|1\rangle$, ahol α és β komplex számok, és igaz rájuk, hogy $|\alpha|^2 + |\beta|^2 = 1$. Az α és β együtthatók neve amplitúdó, és azt fejezik ki, hogy mennyire található az adott kvantum rendszer az adott állapotban. A bázis általában $|0\rangle$ és $|1\rangle$ szokott lenni, de előfordulnak még a $|\uparrow\rangle$ és $|\downarrow\rangle$ valamint a $|+\rangle$ és $|-\rangle$ jelölések is.

1.3. Több qubit

Több qubit együtt egy nagyobb rendszert alkot. Ezt $|x\rangle \otimes |y\rangle$ módon, egy tenzorszorzással jelöljük. Azonban a tenzorszorzás jelét gyakran elhagyjuk, így a következő jelölést szoktuk inkább használni: $|x\rangle|y\rangle$. Az egyszerűség kedvéért azonban gyakran egy közös ket-be írjuk ezeket: $|xy\rangle$. Ekkor is igaz, hogy az együtthatók abszolút értékeinek négyzetösszege egy, vagyis az n qubites $\alpha_1|x_1\rangle + \alpha_2|x_2\rangle + \ldots + \alpha_n|x_n\rangle$ rendszer esetén $|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_n|^2 = 1$. Itt adódik egy remek lehetőség annak megmutatására, hogy miért is növekszik exponenciálisan a számítási igény a rendszer növekedésével. Ha n qubitből előállítunk egy rendszert, azt a következőképp írhatjuk le: $\gamma_1|00\ldots0\rangle + \gamma_2|00\ldots1\rangle + \ldots + \gamma_{2^n}|11\ldots1\rangle$, amit n hosszú $0-1$ sorozatok súlyozott összegeként is felfoghatunk. Mivel ezekből pont 2^n darab van, ennyi különböző amplitúdót kell eltárolni egy n qubites rendszer klasszikus reprezentációjához. Ezen kívül bevezethetünk egy újabb jelölést is. Az n hosszú $0-1$ sorozatot tekinthetjük egy n hosszú bináris számnak is, így jelölhetjük annak tízes számrendszerbeli alakjával, azaz $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$.

Egy másik, klasszikustól eltérő tulajdonsága a kvantum rendszereknek, hogy létezik összefonódott állapot. Mint azt az előbb is láttuk, több qubitből lehet egy nagyobb rendszert alkotni. Viszont nem minden nagyobb rendszer áll elő kisebb rendszerek tenzorszorzataként. Például a $\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$ rendszer nem állítható elő semmilyen két, egy qubitet tartalmazó rendszerből. Ugyanis ekkor az

$$(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|1\rangle$$

egyenletnek kéne teljesülnie. Az α és β változókról ekkor a következőket tudjuk:

- Mivel nincsenek vegyes tagok, ezért $\alpha_0\beta_1 = \alpha_1\beta_0 = 0$
- Az azonos tagok miatt $\alpha_0\beta_0 = \alpha_1\beta_1 = \frac{1}{\sqrt{2}}$

Az első megállapításból következik, hogy $\alpha_0 = 0$ vagy $\beta_1 = 0$, a másodikból pedig az, hogy $\alpha_0 \neq 0$ és $\beta_1 \neq 0$. ami ellentmondás. Azaz két egyqubites állapot tenzorszorzataként nem

állhat elő a fent említett rendszer. Az ilyen állapotokat nevezzük összefonódott állapotnak, azokat pedig, amelyek előállnak kisebb rendszerekből, szeparábilisnak.

1.4. A mérés

További, klasszikus szemszögből nézve furcsa tulajdonsága a kvantum rendszereknek, hogy bármiféle külső vizsgálat elrontja a szuperpozíciót. Azaz mérés hatására a rendszer az állapotok szuperpozíciója helyett egy konkrét állapotot fog felvenni. Azaz egy $\alpha|0\rangle+\beta|1\rangle$ qubit állapota egy mérés hatására vagy $|0\rangle$ vagy $|1\rangle$ lesz, méghozzá az amplitúdójukkal arányosan. Pontosabban $|\alpha|^2$ valószínűséggel $|0\rangle$, valamint $|\beta|^2$ valószínűséggel $|1\rangle$. Használhatunk a méréshez a $|0\rangle$ és $|1\rangle$ helyett más ortonormált bázist is. Ekkor, ha a bázisvektorok $|\varphi_0\rangle$ és $|\varphi_1\rangle$, a mérendő qubitet fel lehet írni ebben a bázisban $\alpha'|\varphi_0\rangle+\beta'|\varphi_1\rangle$ alakban. Ilyenkor a mérés eredménye $|\alpha'|^2$ illetve $|\beta'|^2$ valószínűséggel $|\varphi_0\rangle$ illetve $|\varphi_1\rangle$ lesz. Ez több qubites rendszer esetén is teljesen ugyanígy működik. Viszont több qubites rendszernél előfordulhat, hogy csak az egyik qubitet mérjük meg. Ekkor a többi szuperpozícióban marad, csak a különböző állapotok amplitúdója változik, azt ugyanis le kell normálni az eredmény valószínűségével, hogy az amplitúdó négyzetek továbbra is valószínűséget adjanak. Például két qubit esetén, ha a kiindulási rendszerünk $\alpha_0|00\rangle+\alpha_1|01\rangle+\alpha_2|10\rangle+\alpha_3|11\rangle$, és az első qubitjét mérjük, akkor $|\alpha_0|^2+|\alpha_1|^2$ valószínűséggel a

$$|0\rangle\left(\frac{\alpha_0}{\sqrt{|\alpha_0|^2+|\alpha_1|^2}}|0\rangle+\frac{\alpha_1}{\sqrt{|\alpha_0|^2+|\alpha_1|^2}}|1\rangle\right)$$

állapotba, $|\alpha_2|^2+|\alpha_3|^2$ valószínűséggel pedig az

$$|1\rangle\left(\frac{\alpha_2}{\sqrt{|\alpha_2|^2+|\alpha_3|^2}}|0\rangle+\frac{\alpha_3}{\sqrt{|\alpha_2|^2+|\alpha_3|^2}}|1\rangle\right)$$

állapotba kerülünk.

1.5. Műveletek a qubiteken

A kvantumszámítás lépéseit unitér mátrixokkal lehet reprezentálni. Egy mátrix unitér, ha a transzponált konjugáltja megegyezik az inverzével. Mivel minden unitér mátrix invertálható, így minden kvantumművelet invertálható kell, hogy legyen. Mivel minden unitér mátrix négyzetes, ezért minden művelet n qubitből n qubitet csinál. Valamint mivel az unitér transzformációk megőrzik a hosszt, a kvantumállapotok mindig normáltak maradnak, azaz az amplitúdók a műveletek után is mérési valószínűségeket fejeznek ki. Ezek a megkötések igen szigorúnak tűnnek, különösen az invertálhatóság. Ugyanis sok klasszikus művelet, például a logikai „vagy” ($\vee$) és a logikai „és” ($\wedge$) sem invertálható, mégis megmutatható, hogy így is minden klasszikus művelet elvégezhető kvantumbiteken is.

Az egyik legfontosabb ilyen művelet az Hadamard-Walsh-transzformáció, amit gyakran neveznek egyszerűen Hadamard-transzformációnak is. Ez szemléletesen egy tiszta állapotból egy leginkább kevert állapotot hoz létre, épp ezért sok algoritmus kezdődik ennek a transzformációnak az alkalmazásával. Egy qubitre a mátrix a következőképp néz ki:

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Ennek hatása a szokásos bázisvektorokra:

- $H_1 |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
- $H_1 |1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Valamint a mátrix önmaga inverze, azaz $H_1(H_1 |0\rangle) = |0\rangle$ és $H_1(H_1 |1\rangle) = |1\rangle$. Itt könnyen megmutatható, hogy a méréssel a qubiteknél vigyázni kell. Ugyanis ha a H_1 két alkalmazása között megmérjük a qubitet, az Hadamard-kapu nem lesz önmaga inverze. Például $|0\rangle$-ra $H_1 |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Ha itt az eredmény qubiten végzünk egy mérést, akkor vagy $|0\rangle$, vagy $|1\rangle$ lesz a qubitünk állapota. Ha erre hattatjuk az Hadamard-operátort, akkor eredményül újra $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$-et vagy $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$-et kapunk, a mérés eredményétől függően.

Viszont a bemenetek ritkán állnak egy qubitből. Szerencsére egy k qubiten ható operátorból egyszerűen lehet egy $k+1$ qubiten hatót csinálni úgy, hogy egy 1 qubitessel tenzorszorozzuk. Tehát például $H_2 = H_1 \otimes H_1$, és általánosan $H_{k+1} = H_k \otimes H_1$.

Egy másik fontos transzformáció a c-NOT, azaz irányított negálás. Itt két qubites a bemenet, és az első irányítja, hogy a másodikat negálja-e a kapu. Azaz a négy bázisvektorra a kapu hatása

- $M_{C-NOT} |00\rangle = |00\rangle$
- $M_{C-NOT} |01\rangle = |01\rangle$
- $M_{C-NOT} |10\rangle = |11\rangle$
- $M_{C-NOT} |11\rangle = |10\rangle$

Ennek is felírhatjuk a mátrixát, ami a következő alakot ölti:

$$M_{C-NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Végül nézzük meg a fázistoló transzformációt, ami a bemenet fázisát forgatja el θ szöggel.

Ennek hatása $M_\theta |0\rangle = |0\rangle$, valamint $M_\theta |1\rangle = e^{i\theta} |1\rangle$. Ennek mátrixa a következő:

$$M_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

Ezen három transzformáció segítségével bármilyen unitér transzformáció előállítható. Tehát ha ezekre a transzformációkra, mint kapukra tekintünk, akkor ezen három kapu együtt egy teljes rendszert alkot, hasonlóan a klasszikus NAND kapuhoz.

1.6. Kvantum teleportálás

A mérésnek az összefonódott állapotra nagyon izgalmas hatása van. Vegyük például a már említett $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$ állapotot. Ezt nevezik teljesen összefonódott állapotnak, vagy EPR-párnak is, Einstein, Podolski és Rosen után. Ha ennek a rendszernek az egyik qubitjén elvégzünk egy mérést, akkor a másik qubit is azonnal úgy viselkedik, mintha elvégeztük volna rajta a mérést. Azaz a szuperpozíció helyett konkrét értéket fog felvenni, akármilyen távol is vannak egymástól.

Ez a különleges viselkedés felhasználható arra, hogy egy kvantumállapotot „teleportáljunk", azaz kvantumcsatorna nélkül egyik pontból a másikba juttassunk. Legyen adott Alice-nak egy ismeretlen $\alpha |0\rangle + \beta |1\rangle$ kvantumállapota, és ezt ő el szeretné küldeni Bobnak. Tegyük föl, hogy ehhez csak egy klasszikus csatorna áll rendelkezésére, azaz egy olyan csatorna, ami csak klasszikus információt tud küldeni. Mivel Alice nem ismeri a nála levő kvantumállapotot, ezért nem tudja az amplitúdókat elküldeni, hogy azután Bob valahogy előállíthassa ugyanazt az állapotot. Megmérni a valószínűségeket pedig nem lehet, mivel egy mérés után elveszti a rendszer a szuperpozícióját. Ráadásul, még ha sok azonos állapotú qubitje is lenne Alice-nak, végtelen sok mérést kellene végeznie, hogy pontosan tudja a kvantumállapotot. Tehát valami más módszert kell kitalálni. Ilyenkor lehet kihasználni az EPR-párok furcsa tulajdonságát. Ugyanis ha Alice és Bob megosztottak egy teljesen összefonódott kvantumpárt, egy azon elvégzett méréssel át tudja küldeni Alice a saját qubitjének állapotát. Azért nevezik teleportációnak, mert ez pillanatszerűen lezajlik, a távolságtól függetlenül. Ám, amint azt később látni fogjuk, ez nem tesz lehetővé fénynél gyorsabb információtovábbítást.

Kezdetben Alice-nak és Bobnak van egy megosztott EPR-párjuk. Ehhez hozzávesszük Alice ismeretlen qubitjét. Ekkor a következő állapotot kapjuk:

$$(\alpha |0\rangle + \beta |1\rangle)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{\alpha}{\sqrt{2}} |000\rangle + \frac{\alpha}{\sqrt{2}} |011\rangle + \frac{\beta}{\sqrt{2}} |100\rangle + \frac{\beta}{\sqrt{2}} |111\rangle$$

(Fontos, hogy a következőkben a három qubit sorrendje végig Alice ismeretlen qubitje, az EPR-pár Alice-nál levő része, az EPR-pár Bobnál levő része lesz.) Ez átírható a következő

alakba:

$$\frac{1}{2}\Big(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)(\alpha\,|0\rangle + \beta\,|1\rangle) + \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)(\alpha\,|0\rangle - \beta\,|1\rangle) + \\ + \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)(\alpha\,|1\rangle + \beta\,|0\rangle) + \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle)(-\alpha\,|1\rangle + \beta\,|0\rangle)\Big)$$

Ez az átírás ugyan elsőre bonyolultabbnak tűnik, mint az előző alak, de az Alice-nál levő qubit pár itt végig az egyik úgynevezett Bell állapotban van. Ezek az állapotok:

- $B_{00} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$
- $B_{01} = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$
- $B_{10} = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $B_{11} = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle)$

Ha most Alice ezen állapotok mentén végez mérést a nála levő két qubiten, a Bobnál levő qubitek állapota is ismert lesz számára. Már csak a mérés eredményét kell elküldenie Bobnak, aki ennek függvényében a következőket teszi:

- B_{00}: Bob állapota $\alpha\,|0\rangle + \beta\,|1\rangle$, vagyis Bobnál a kívánt állapot van.
- B_{01}: Bob állapota $\alpha\,|1\rangle + \beta\,|0\rangle$, vagyis Bob a negálást hajtja végre a qubitjén.
- B_{10}: Bob állapota $\alpha\,|0\rangle - \beta\,|1\rangle$, ekkor a fáziscsere műveletet kell végrehajtania.
- B_{11}: Bob állapota $\alpha\,|1\rangle - \beta\,|0\rangle$, így Bobnak mindkét fenti műveletet végre kell hajtania

Ez az utolsó lépés megköveteli, hogy Alice Bobnak klasszikus információt küldjön, így hiába változott meg a nála levő qubit állapota a mérés elvégeztével azonnal, Bob ekkor még nem tudhatja milyen állapot is van nála. Ahhoz, hogy biztosan az Alice által elküldeni kívánt állapot legyen nála, meg kell kapnia Alice mérésének eredményét, és az ennek megfelelő transzformációkat végre kell hajtania. És mivel klasszikus információt jelenleg nem tudunk a fénynél gyorsabban továbbítani, így Bob sem kapta meg az Alice-nál levő kvantum bitet a fénynél gyorsabban.

1.7. Nem-klónozhatósági tétel

Nagy különbség a klasszikus és a kvantumvilág között, hogy míg egy klasszikus bitet minden gond nélkül le lehet másolni, ezt egy kvantumbittel nem tehetjük meg. Ez abból következik, hogy minden kvantumbiten végzett művelet invertálható, nem létezhet olyan művelet, ami egy ismert állapotot az ismeretlen bemenetté változtat. Pontosabban nincsen olyan U unitér transzformáció, hogy minden φ kvantumállapotra $U(|\varphi\rangle\,|x\rangle) = |\varphi\rangle\,|\varphi\rangle$.

Bizonyítás. Tegyük fel, hogy létezik egy ilyen U transzformáció. Legyen $|\alpha_1\rangle$ és $|\alpha_2\rangle$ két ortogonális állapot. Ekkor U másoló tulajdonsága miatt

$$U\Big(\frac{1}{\sqrt{2}}(|\alpha_1\rangle + |\alpha_2\rangle)\,|x\rangle\Big) = \Big(\frac{1}{\sqrt{2}}(|\alpha_1\rangle + |\alpha_2\rangle)\Big)\Big(\frac{1}{\sqrt{2}}(|\alpha_1\rangle + |\alpha_2\rangle)\Big) = \\ = \frac{1}{2}\Big(|\alpha_1\rangle\,|\alpha_1\rangle + |\alpha_1\rangle\,|\alpha_2\rangle + |\alpha_2\rangle\,|\alpha_1\rangle + |\alpha_2\rangle\,|\alpha_2\rangle\Big)$$

Viszont ha U linearitását kihasználva másképpen írjuk fel ugyan ezt:

$$U\Big(\frac{1}{\sqrt{2}}(|\alpha_1\rangle + |\alpha_2\rangle)\,|x\rangle\Big) = \frac{1}{\sqrt{2}}U(|\alpha_1\rangle\,|x\rangle) + \frac{1}{\sqrt{2}}U(|\alpha_2\rangle\,|x\rangle) = \\ = \frac{1}{\sqrt{2}}\,|\alpha_1\rangle\,|\alpha_1\rangle + \frac{1}{\sqrt{2}}\,|\alpha_2\rangle\,|\alpha_2\rangle$$

Látható, hogy a két megközelítésből adódó eredmény, noha ugyanannak kellene lennie, nem egyezik. Tehát ellentmondásra jutottunk, így a feltevés nem teljesülhet. □

2. fejezet

Grover keresőalgoritmusa

Az első híres, és nagy gyorsulást elérő kvantum algoritmus Shor faktorizációs algoritmusa volt [15]. Ez az algoritmus exponenciális gyorsulást hoz a klasszikus algoritmusokhoz képest a számok prímtényezőkre bontásában. A kvantumalgoritmusok történetében ez egy nagyon fontos lépés volt, de a használata úgy látszik egy elég jól körülhatárolható problémakörre terjed ki.

Ilyen szempontból sokkal általánosabb Lov Grover 1996-os keresőalgoritmusa [6]. Ez a keresőalgoritmus az orákulumos keresést valósítja meg rendezetlen adathalmazon, és a klasszikushoz képest négyzetes gyorsulást eredményez. Ugyanis klasszikus esetben egy ilyen keresés lépésszáma $\Theta(N)$, ahol N a halmazban levő elemek száma. Determinisztikus algoritmusnak, ha nincs keresett elem, a halmaz minden elemét végig kell néznie, máshogy nem tudná kizárni, hogy létezik keresett elem. Nagyságrendileg egy véletlen algoritmus sem tud ezen segíteni, ugyanis annak is várhatóan $\frac{N}{2}$ lépésre van szüksége. Ezzel szemben Grover algoritmusa $\Theta(\sqrt{N})$ lépést használ, és nagy valószínűséggel helyes eredményt ad.

A dolgozatban vizsgált kvantumalgoritmusok mind nagy valószínűséggel adnak helyes eredményt. Azonban ez a valószínűség az algoritmus megismétlésével tetszőlegesen növelhető. Továbbá a hibás válasz csak hamis negatív lehet, ha nincs keresett elem, akkor az algoritmus mindenképpen megfelelő kimenetet fog adni.

Grover algoritmusának a kvantumalgoritmusok fejlődésére tett hatását az indokolja, hogy a keresés nagyon sok problémában előfordul, mint részfeladat. Továbbá az algoritmus által használt amplitúdó amplifikáció egy más problémákhoz is igen jól használható módszert adott. Az alábbiakban ezt a keresést fogom ismertetni [7] alapján.

Az algoritmus egy orákulumos keresést valósít meg. Jelölje az elemek halmazát X, és legyen a keresett elemek halmaza $T \subset X$. Azt, hogy egy elem a keresettekhez tartozik-e, vagy sem, egy fekete dobozként működő orákulum tudja. Legyen ez egy adott

$$f_T(x) = \begin{cases} 1 & \text{, ha } x \in T \\ 0 & \text{, különben} \end{cases}$$

függvény. Azaz ez a függvény minden elemről eldönti, hogy az a T halmazban van, vagy sem.

Az algoritmus lényege, hogy egy kezdeti eloszlásból egy olyan állapotot állítson elő, amit ha megmérünk, nagy valószínűséggel egy keresett elemet kapunk. A kezdeti eloszlás az az állapot, ahol mindegyik kimenetnek azonos a valószínűsége, azaz $\sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle$. Ehhez $n = \log_2 N$ qubitre lesz szükségünk. Az egyszerűség kedvéért legyen N kettőhatvány, azaz $N = 2^n$. Ha ez nem teljesül, a hiányzó elemeket fel lehet tölteni nem keresett elemekkel, ez nem befolyásolja az algoritmust, az alaphalmaz mérete csak kevesebb, mint kétszeresére nő. A kiindulási állapot könnyen előállítható az n qubites $|0\rangle$ állapotból, ugyanis egy $H_n = H_1 \otimes H_1 \otimes \cdots \otimes H_1$-et, azaz n qubiten ható Hadamard-operátort alkalmazva pont a kívánt állapotot kapjuk.

Ez után a keresett elemek amplitúdóját megfordítjuk, azaz (-1)-gyel szorozzuk. Ehhez a lépéshez van szükségünk az orákulumra, itt kódoljuk az állapotba, hogy melyik elemek a keresettek. A következő lépés pedig minden elem amplitúdóját tükrözi az amplitúdók átlaga által meghatározott egyenesre. Ez az átlag kevés keresett elem mellett közel lesz $\frac{1}{\sqrt{N}}$-hez, de annál valamivel kisebb értéket fog felvenni, mert az amplitúdók nagy része $\frac{1}{\sqrt{N}}$, és csak néhány ennek (-1)-szerese. Emiatt a nem keresett elemek amplitúdója valamivel $\frac{1}{\sqrt{N}}$ alá csökken, a keresetteké pedig $-\frac{1}{\sqrt{N}}$-ről majdnem $\frac{3}{\sqrt{N}}$-re nő. Majd ismét negáljuk a keresett elemek amplitúdóját, és ismét tükrözünk. Az algoritmus ezt a két lépést ismételgeti meghatározott ideig, így a végén a keresett elemek amplitúdója, és ezért mérésének a valószínűsége jóval meghaladja a többiét. A részletesebb számolást hamarosan látni fogjuk.

Nézzük most meg, milyen operátorok szükségesek ehhez. Nyilvánvaló, hogy szükséges egy olyan, ami a keresett elemek amplitúdóját megfordítja. Legyen ez V_f, így

$$V_f |x\rangle = (-1)^{f(x)} |x\rangle = \begin{cases} -|x\rangle & \text{ha } x \in T \\ |x\rangle & \text{különben} \end{cases}$$

Ez az operátor az állapot előjelébe kódolja az $f(x)$ függvény értékét. Azaz ez maga az orákulum a keresésben.

Ezen kívül szükségünk van még arra az operátorra, ami az átlagra tükrözi az amplitúdókat. Ezt a $(-H_n R_n H_n)$ szorzat teszi meg, ahol

$$R_n |x\rangle = \begin{cases} -|x\rangle & \text{ha } x = 0 \\ |x\rangle & \text{különben} \end{cases}$$

Vagyis a $|0\rangle$ állapotot szorozza (-1)-gyel, a többit pedig változatlanul hagyja. Érdemes megfigyelni, hogy $H_n R_n H_n = I - 2P$, ahol I az $2^n \times 2^n$-es egységmátrix, P pedig egy vetítés mátrixa, aminek minden eleme $\frac{1}{2^n}$. Ez abba az egydimenziós altérbe vetít, melyet

a $\psi = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$ vektor generál.

A $(-H_n R_n H_n)$ transzformáció ekkor tényleg az átlag körül tükröz. Ugyanis vegyünk egy tetszőleges $\sum c_x |x\rangle$ állapott. Itt c_x a $|x\rangle$ vektor amplitúdóját jelöli. Legyen $A = \frac{1}{2^n} \sum c_x$ az átlagos amplitúdó. Ekkor $\sum c_x |x\rangle = A \sum |x\rangle + \sum (c_x - A) |x\rangle$ az eredeti állapotunk felbontása két merőleges altérbe eső vektorra. Két vektor akkor merőleges egymásra, ha a belső szorzatuk nulla. Ez a két vektor tényleg merőleges egymásra, ugyanis

$$\begin{aligned}(A \sum_{x \in X} \langle x|)(\sum_{y \in X} (c_y - A) |y\rangle) &= (A \sum_{x \in X} \langle x| \sum_{y \in X} c_y |y\rangle) - (A \sum_{x \in X} \langle x| \sum_{y \in X} A |y\rangle) = \\ &= (A \sum_{x \in X} \sum_{y \in X} c_y \langle x| |y\rangle) - (A^2 \sum_{x \in X} \sum_{y \in X} \langle x| |y\rangle) = \\ &= A \sum_{y \in X} c_y - 2^n A^2 = 2^n A^2 - 2^n A^2 = 0\end{aligned}$$

Mivel A egy konstans, ezért a szummán kívülre lehet hozni. Valamint

$$\langle x| |y\rangle = \langle x|y\rangle = \delta_{xy} = \begin{cases} 1 & \text{ha } x = y \\ 0 & \text{különben} \end{cases}$$

mivel a $|x\rangle$, $x \in X$ állapotok egy ortonormált bázist alkotnak. Így $\sum_x \sum_y \langle x| |y\rangle = 2^n$ adódik, továbbá $\sum_x c_x = 2^n A$. Tehát a két vektor belső szorzata nulla, vagyis a felbontás tényleg két merőleges altérre bontja fel az állapotot.

A P vetítés definíciójából $P \sum c_x |x\rangle = A \sum |x\rangle$, ahonnan

$$(-H_n R_n H_n) \sum c_x |x\rangle = (2P - I) \sum c_x |x\rangle = \sum (2A - c_x) |x\rangle$$

Tehát minden amplitúdót a (-1)-szeresére változtat, majd az átlag kétszereséhez ad. Vagyis az átlagra tükrözi az amplitúdókat.

Ez tényleg növeli a keresett elemek mérésének valószínűségét. Legyen $|T| = k$ a keresett elemek száma. Vegyük az $\frac{1}{\sqrt{2^n}} \sum |x\rangle$ állapotot, aminek egyenletes az amplitúdóeloszlása. Ha ekkor V_f a keresett elemeknek megfordítja az előjelét, az átlag

$$A = \frac{1}{2^n}((2^n - k)\frac{1}{\sqrt{2^n}} - k\frac{1}{\sqrt{2^n}}) = \frac{1}{\sqrt{2^n}}(1 - \frac{2k}{2^n})$$

lesz. Látható, hogy ha az elemek kis része megjelölt, akkor ez csak kicsit lesz a nem megjelölt elemek amplitúdója alatt. Viszont ha ezen végrehajtjuk a $(-H_n R_n H_n)$ transzformációt, a keresett elemek amplitúdója $\frac{1}{\sqrt{2^n}}(3 - \frac{4k}{2^n})$ lesz, míg a többié $\frac{1}{\sqrt{2^n}}(1 - \frac{4k}{2^n})$-re csökken.

A Grover-algoritmust ezen két tükrözés egymás utáni alkalmazása alkotja, tehát a keresés egy lépését a $G_n = -H_n R_n H_n V_f$ operátor adja meg. Azt kell már csak megvizsgálni, hogy

hányszor kell ismételni ennek az alkalmazását. Az eredmény az lesz, hogy G_n ismételt alkalmazásával egy ideig nő a keresett elemek amplitúdója, majd csökkenni kezd, és ezt a változást periodikusan ismétli. Pontosabban egy szinusz görbének megfelelően változik a keresett elemek mérésének a valószínűsége. Ez elsőre furcsa lehet, ugyanis általában a véletlenen alapuló algoritmusok a hosszabb futtatás, azaz több mintavételezés hatására pontosabb eredményt adnak. Például a keresésre használt véletlen algoritmus esetén, ha egy elemet csak egyszer vizsgálunk, N ismétlés után biztosan helyes megoldást kapunk. Ezzel szemben Grover algoritmusát nem szabad tetszőleges ideig futtatni, mert az eredmény egy idő után romlani kezd. Illetve ilyenkor tovább kell futtatni, amíg újra el nem éri a valószínűség csúcsát.

A továbbiakban legyen $F = X \setminus T$ a nem-megoldások halmaza, így $|F| = 2^n - k$. Legyen $G_n = (-H_n R_n H_n V_f)$ operátor r-szeri alkalmazása után a következő állapotunk:

$$t_r \sum_{x \in T} |x\rangle + f_r \sum_{x \in F} |x\rangle$$

Ha most erre alkalmazzuk V_f-et, az a megjelölt elemek amplitúdóját negálja, azaz a

$$-t_r \sum_{x \in T} |x\rangle + f_r \sum_{x \in F} |x\rangle$$

állapotba jutunk. Az átlagos amplitúdó ekkor $A = \frac{1}{2^n}((2^n - k)f_r - kt_r)$. A következő lépés a $(-H_n R_n H_n)$ operátor, ez a következő állapotot hozza létre:

$$t_{r+1} \sum_{x \in T} |x\rangle + f_{r+1} \sum_{x \in F} |x\rangle$$

ahol

$$\begin{cases} t_{r+1} = 2A - t_r = (1 - \frac{2k}{2^n})t_r + (2 - \frac{2k}{2^n})f_r \\ f_{r+1} = 2A - f_r = -\frac{2k}{2^n}t_r + (1 - \frac{2k}{2^n})f_r \end{cases}$$

Mivel egyenletes amplitúdóeloszlásból indulunk ki, ezért $t_0 = f_0 = \frac{1}{\sqrt{2^n}}$ kezdeti feltételeket használjuk. Továbbá tudjuk, hogy $kt_r^2 + (2^n - k)f_r^2 = 1$-nek minden r-re teljesülnie kell, ugyanis minden állapotban az amplitúdónégyzetek összegének egynek kell lennie. Vagyis (t_r, f_r)-et egy ellipszis pontjai határozzák meg. Azaz felírhatjuk, hogy

$$\begin{cases} t_r = \frac{1}{\sqrt{k}} \sin\theta_r \\ f_r = \frac{1}{\sqrt{2^n - k}} \cos\theta_r \end{cases}$$

valamilyen θ_r-re. Ezt behelyettesítve a rekurzióba a következő összefüggéseket kapjuk:

$$\begin{cases} \sin\theta_{r+1} = (1-\frac{2k}{2^n})\sin\theta_r + \frac{2}{2^n}\sqrt{2^n-k}\sqrt{k}\cos\theta_r \\ \cos\theta_{r+1} = -\frac{2}{2^n}\sqrt{2^n-k}\sqrt{k}\sin\theta_r + (1-\frac{2k}{2^n})\cos\theta_r \end{cases}$$

Mivel $k \leq n$, ezért $1-\frac{2k}{2^n} \in [-1,1]$. Így választhatunk egy $\omega \in [0,\pi]$-t, hogy $\cos\omega = 1-\frac{2k}{2^n}$. Ekkor $\sin\omega = \frac{2}{2^n}\sqrt{2^n-k}\sqrt{k}$. Ezzel az előző egyenletek egy kellemesebb alakba írhatóak:

$$\begin{cases} \sin\theta_{r+1} = \sin(\theta_r+\omega) \\ \cos\theta_{r+1} = \cos(\theta_r+\omega) \end{cases}$$

A kezdeti feltételekből megkapjuk, hogy $\sin^2\theta_0 = \frac{k}{2^n}$, ami θ_0 meghatározásához használható: $\theta_0 \approx \sqrt{\frac{k}{2^n}}$. Így a rekurzió megoldása a következő:

$$\begin{cases} t_r = \frac{1}{\sqrt{k}}\sin((2r+1)\theta_0) \\ f_r = \frac{1}{\sqrt{2^n-k}}\cos((2r+1)\theta_0) \end{cases}$$

Mivel k megoldásunk van, ezért egy megoldás megtalálásának valószínűsége

$$kt_r^2 = \sin^2\theta_r = \sin^2((2r+1)\theta_0)$$

Ehhez szeretnénk olyan, minél kisebb r-et találni, hogy a valószínűség maximális legyen, tehát $(2r+1)\theta_0$ legyen a legközelebb $\frac{\pi}{2}$-höz. Ez $r \approx \frac{\pi}{4\theta_0}$-nál lesz. Annak a valószínűsége, hogy $\left\lfloor \frac{\pi}{4\theta_0} \right\rfloor$ iteráció után megtalálunk egy megoldást, legalább $\frac{1}{4}$.

Az algoritmus összefoglalva:

1. Ha $k > \frac{3}{4}2^n$, akkor válasszunk egy véletlenszerű y-t, az jó megoldás lesz.
2. Különben számoljuk ki $r = \left\lfloor \frac{\pi}{4\theta_0} \right\rfloor$ értéket, ahol $\theta_0 \in [0,\pi/3]$ és $\sin^2\theta_0 = \frac{k}{2^n}$-ből számolható.
3. Állítsuk elő az $\frac{1}{\sqrt{2^n}}\sum|x\rangle$ állapotot az Hadamard transzformációval.
4. Alkalmazzuk a $G_n = -H_nR_nH_nV_f$ operátort r-szer.
5. Mérjünk, hogy megkapjuk y-t.

Ekkor az $r = \left\lfloor \frac{\pi}{4\theta_0} \right\rfloor \approx \frac{\pi\sqrt{N}}{4\sqrt{k}}$ iteráció után eredményül kapott y legalább $\frac{1}{4}$ valószínűséggel jó megoldás lesz. Ha ennél nagyobb valószínűséggel szeretnénk a helyes megoldást megkapni, az algoritmus megismétlésével ez a valószínűség tetszőlegesen növelhető.

3. fejezet

Véletlen séták

3.1. Klasszikus eset

Sok esetben gyorsabb algoritmust lehet készíteni, ha véletlent is használunk, mintha az algoritmusunk determinisztikusan futna. Egy nagy ága a véletlent használó algoritmusoknak a Monte-Carlo-módszert használó algoritmusok. Ennek lényege, hogy a feladatot csak néhány, véletlenszerűen kiválasztott pontban végezzük el, ezekből vonunk le következtetéseket a teljes problémára. Ha megfelelő számú és eloszlású véletlen számokat használunk, az eredmény megfelelő pontosságúvá tehető. A módszert Stanisław Ulam találta ki, miközben a Manhattan Programon dolgozott és a neutron láncreakciót kutatta fissziós eszközökben [4]. A módszer alapjait később sok más célra is felhasználták, például a prímtesztelésre is.

Sokáig nem volt ismert, hogy egy számról eldönthető-e polinom időben, hogy prím-e. Azonban már több, mint 40 éve léteznek hatékony, véletlent használó algoritmusok, melyek nagy valószínűséggel helyes eredményt adnak (például a Miller-Rabin-tesz [13][10]). Bár ma már ismert, hogy van determinisztikus polinom idejű algoritmus is (AKS-teszt [1]), de a véletlent használó eljárások a gyakorlatban még mindig hatékonyabbak. Egy potenciális kvantumszámítógépen azonban nem csak tesztelni lehetne prímeket polinom időben, hanem faktorizálni is, Shor algoritmusával.

Az egyik gyakran használt, ám alapjaiban igen egyszerű ilyen módszer a véletlen séta. Ez szemléletesen annak felel meg, hogy egy gráfban minden él egy valószínűséget jelöl, és az aktuális csúcsból a valószínűségeknek megfelelően lépünk tovább. A véletlen sétát gyakran használják például a fizikában, a Brown-mozgás egyszerűbb modellezésére. A Brown-mozgás a folyadékban lebegő folyékony vagy gáz halmazállapotú molekulák mozgását írja le. Ez egy folytonos probléma, de lehet közelíteni a diszkrét véletlen sétával.

Egy X halmazon vett véletlen séta valószínűségei leírhatók egy $|X|\times|X|$-es P mátrixszal, aminek p_{xy} eleme azt mondja meg, hogy az $x \in X$ pontból az $y \in X$ pontba mekkora valószínűséggel jutunk el. Ekkor P egy sztochasztikus mátrix lesz, azaz minden eleme nemnegatív, és minden sorösszege 1. Egy $x \in X$ pontból a következő pontok eloszlása

könnyen megkapható, ha egy olyan vektorral szorozzuk meg P-t, aminek az x. eleme 1, a többi nulla. Legyen P aperiodikus, azaz olyan, hogy létezik i, hogy P^i minden eleme pozitív. Ekkor a Frobenius-Perron tétel miatt [12] az 1 a mátrix sajátértéke, és minden más sajátérték abszolútértéke kisebb, mint 1. Legyen továbbá λ_1 a legnagyobb abszolútértékű, egytől különböző sajátérték. Ekkor $1 - |\lambda_1| = \delta$ a spektrális rés. Ha $p_{xy} > 0 \Rightarrow p_{yx} > 0$, akkor megfordítható véletlen sétáról beszélünk.

Az egyik legegyszerűbb ezt felhasználó keresési algoritmus a következő: legyen adott a P mátrix, valamint egy, a jelölt elemeket tartalmazó $M \subset X$ halmaz.

1. Vegyünk egy tetszőleges $x \in X$ elemet.
2. Ismételjük t-szer
   (a) Ha $x \in M$, akkor visszaadjuk és STOP.
   (b) Különben P alapján kiválasztjuk a következő x elemet.
3. Ha a t ismétlés alatt nem találtunk megfelelő elemet, STOP, és adjuk vissza, hogy nincs megfelelő elem.

Ha a kezdeti adatok olyanok, hogy egy M-beli elem kiválasztásának a valószínűsége ε, akkor az algoritmus várhatóan $t = \Theta(\frac{1}{\delta\varepsilon})$ választás mellett jó eredményt ad.

Ha valami okból kifolyólag egy elem M-beliségének ellenőrzése nehéz feladat, az algoritmus egy módosított változata jobb lehet:

1. Vegyünk egy tetszőleges $x \in X$ elemet.
2. Ismételjük t_2-ször
   (a) Ha $x \in M$, akkor visszaadjuk és STOP.
   (b) Különben P alapján szimuláljunk t_1 lépést, így megkapjuk az új x-et.
3. Ha eddig nem találtunk megfelelő elemet, STOP, és adjuk vissza, hogy nincs megfelelő elem.

Ha $t_1 = \Theta(\frac{1}{\delta})$ és $t_2 = \Theta(\frac{1}{\varepsilon})$, az algoritmus nagy valószínűséggel jó eredményt ad.

Vizsgáljuk meg az algoritmusok lépésszámát [11]. Ehhez három különböző lépésszámot használunk.

- Inicializálási költség (I): a kezdeti adatszerkezet felépítése.
- Frissítési költség (F): egy lépés végrehajtásának költsége. Azaz egy tetszőleges x állapotból egy azt követő állapotba lépés P alapján.
- Ellenőrzési költség (E): az aktuális x állapot ellenőrzése, azaz $x \in M$ vizsgálata.

Állítás: Legyen P egy szimmetrikus, aperiodikus Markov-lánc, és $\delta > 0$ a spektrális rése egy n elemű X halmazon. Legyen továbbá $M \subset X$ a keresett elemek halmaza, és $\frac{|M|}{|X|} \geq \varepsilon > 0$ ha M nem üres. Ekkor egyenletes eloszlásból indulva

1. az első algoritmus nem üres M halmaz esetén nagy valószínűséggel talál egy M-beli elemet, ha $t = O(\frac{1}{\delta\varepsilon})$, és ehhez $O(I + \frac{1}{\delta\varepsilon}(F + E))$ lépést használ.

2. a második algoritmus nem üres M halmaz esetén nagy valószínűséggel talál egy M-beli elemet, ha $t_1 = O(\frac{1}{\delta})$ és $t_2 = O(\frac{1}{\varepsilon})$, és ehhez $O(I + \frac{1}{\varepsilon}(\frac{1}{\delta}F + E))$ lépést használ.

Tehát a második változatban az elsőhöz képest az inicializálás és a frissítés költsége nem változik, de az ellenőrzésé $\frac{1}{\delta\varepsilon}E$ helyett csak $\frac{1}{\varepsilon}E$ lesz.

3.2. Kvantum eset

A kvantumalgoritmusok kutatásakor felmerült a kérdés, hogy hogyan lehetne a klasszikus algoritmusok között megismert véletlen sétát a kvantumalgoritmusok között hasznosítani. Valamint, ami még fontosabb, lehet-e kvantum sétával gyorsulást elérni a klasszikus algoritmusokhoz képest.

A számegyenesen vett diszkrét véletlen sétához viszonylag könnyű szemléletes kvantum megfelelőt találni. Vegyünk két Hilbert-teret, legyen az egyik $\mathcal{H}_P$, ami a séta helyét kódolja a számegyenesen, a másik pedig $\mathcal{H}_C$, ami a lépés irányát jelzi. Legyenek $\mathcal{H}_P$ állapotai: $|i\rangle$, $i \in \mathbb{Z}$, azaz a számegyenes számai, valamint $\mathcal{H}_C$ állapotai: $|\uparrow\rangle$ és $|\downarrow\rangle$. $\mathcal{H}_C$ egy érmedobást fog szimulálni, és a séta ennek függvényében fog pozitív vagy negatív irányba lépni.

Legyen a séta tere egy $\mathcal{H}$ Hilbert-tér úgy, hogy $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$. Ekkor a következő unitér operátor az első regiszter függvényében jobbra, illetve balra lép:

$$S = \Big(|\uparrow\rangle\langle\uparrow|\Big) \otimes \Big(\sum_i |i+1\rangle\langle i|\Big) + \Big(|\downarrow\rangle\langle\downarrow|\Big) \otimes \Big(\sum_i |i-1\rangle\langle i|\Big)$$

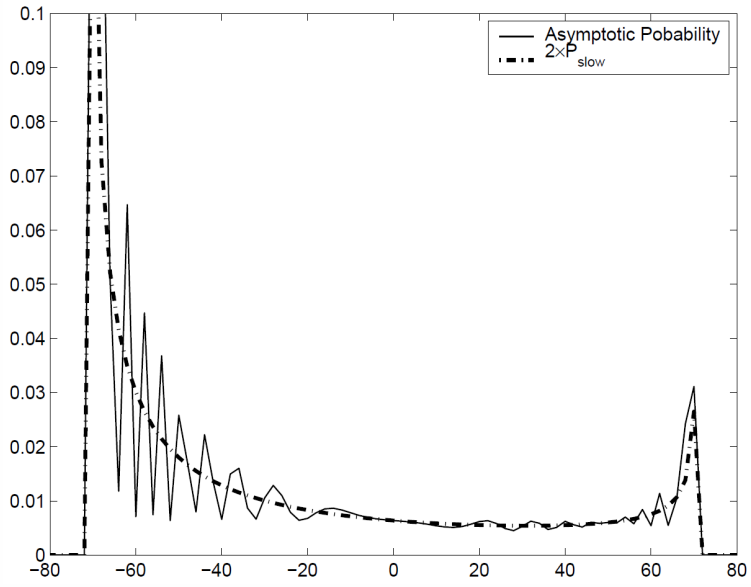
Ennek hatása a bázisvektorokra:

- $S|\uparrow\rangle|i\rangle = |\uparrow\rangle|i+1\rangle$
- $S|\downarrow\rangle|i\rangle = |\downarrow\rangle|i-1\rangle$

Ha először egy C unitér transzformációt hajtunk végre a $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_P$ első regiszterén, majd az S operátort, akkor egy, a C-től függő $U = S \cdot (C \otimes I)$ véletlen sétát kapunk a számegyenesen. Egy ilyen C operátor lehet például a már megismert H_1 Hadamard-operátor.

Érdekes megfigyelni, hogy a H_1 operátort használva C helyén az $U = S \cdot (H_1 \otimes I)$ séta nem lesz azonos a $|\downarrow\rangle|0\rangle$ és a $|\uparrow\rangle|0\rangle$ kezdőállapotokra. Azaz a klasszikus sétával ellentétben nem mindegy, hogy kezdetben jobbra vagy balra lépünk. Erre megoldás lehet az, hogy

$\frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle) \otimes |0\rangle$-ből indítjuk a sétát. Vagy használhatunk egy másik C operátort, például az $Y = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$-t. Ennek hatására U bármelyik állapotból indítva szimmetrikus lesz.

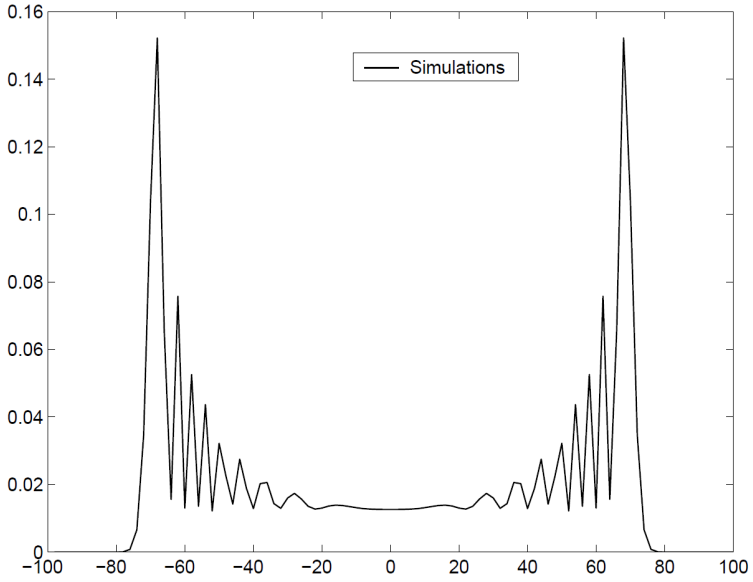


3.1. ábra. *A $|\downarrow\rangle \otimes |0\rangle$ állapotból indított séta valószínűségeloszlása, $C = H_1$ operátorral, $T = 100$ lépés után [8]. Jól látszik, hogy az eloszlás nem szimmetrikus.*

Ha megfigyeljük a séta valószínűségi eloszlását, feltűnik az első lényeges különbség a klasszikus véletlen sétával szemben. Klasszikus esetben T lépés után egy Gauss görbét kapunk, ha megnézzük a pontokra annak a valószínűségét, hogy a séta épp ott áll. Ezzel szemben a kvantum sétánál egy kétcsúcsú eloszlást kapunk. Megmutatható, hogy klasszikus sétánál $\sigma = \sqrt{T}$ a szórás, azaz ekkora a várható legnagyobb távolság a kiindulási ponttól. Ezzel szemben kvantum séta esetén $\sigma \sim T$, azaz a kvantum séta négyzetesen gyorsabban távolodik a kezdőállapottól.

A 3.2. ábrán a szimuláció $T = 100$ lépésig futott. Mivel ebben sétában páros helyről páratlanra, és páratlanról párosra lépünk, a nullából kiindulva páros számú lépésben csak a páros helyeken lehetünk. Ezért az ábrán csak a páros helyeken felvett értékek szerepelnek, ugyanis a páratlan helyeken a valószínűség ebben a lépésben nulla. Jól látható, hogy az eloszlás tényleg nem a klasszikus sétáknál megszokott Gauss-görbe lesz, hanem egy két maximummal rendelkező eloszlás.

Nagyon jól látható ezen a példán a kvantumalgoritmusok sajátossága, hogy az állapotok szuperpozícióját használja ki. Ugyanis maga az operátor teljesen determinisztikus, nincs benne véletlen, az eredmény mégis az lesz. Viszont ez a véletlen csak a méréssel kerül a rendszerbe. Ugyanis a mérés elvégzése előttig minden, azonos helyről indított séta azonos kvantumállapotban lesz.



3.2. ábra. *Szimmetrikus kezdőállapotból indított, $C = H_1$ operátort használó séta velószínűségeloszlása [8].*

Ezt, a számegyenesen való véletlen sétát, egyszerűen kiterjeszthetjük egy gráfon való véletlen sétára. Egy d maximális fokszámú gráf esetén az első regisztert lecseréljük egy d-dimenziós Hilbert-térre. Legyen ennek a jelentése az adott csúcsból kimenő élek, egy adott sorrendben számozva. Ekkor $|j\rangle \otimes |v\rangle$ a v csúcsot és a j. élét jelenti. Így

$$S\,|j\rangle \otimes |v\rangle = \begin{cases} |j\rangle \otimes |w\rangle & \text{ha } v\text{-ből a } j.\text{ élen } w\text{-be jutunk} \\ 0 & \text{különben} \end{cases}$$

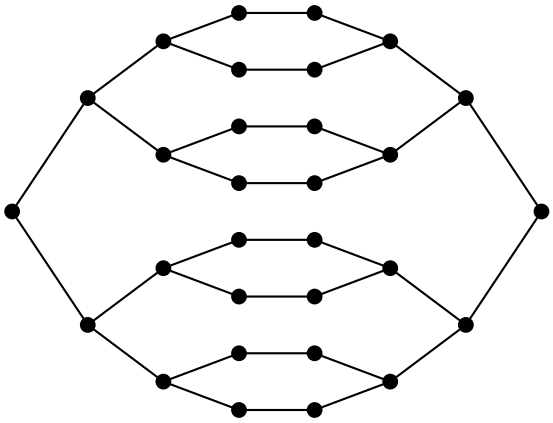
választással meg tudjuk oldani a lépést. Nyilván ehhez a C-t is le kell cserélni egy d dimenziós transzformációra. (Ha a gráf nem d-reguláris, akkor a maradék éleket például hurokélnek véve d-regulárissá tehető.) Ilyen C lehet például az Hadamard-operátor kiterjesztése, a diszkrét Fourier-transzformáció operátora, ami a következő formában írható fel mártixként:

$$DFT = \frac{1}{\sqrt{d}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{d-1} \\ & & \dots & & \\ 1 & \omega^{d-1} & \omega^{2(d-1)} & \dots & \omega^{(d-1)(d-1)} \end{pmatrix}$$

ahol $\omega = e^{\frac{2\pi i}{d}}$ a d-ik komplex egységgyök.

Egy példa arra az esetre, ahol a gráfon való véletlen séta esetén exponenciálisan gyorsabb a kvantum bejárás a klasszikusnál egy olyan gráf, ahol két teljes bináris fa van a leveleiknél összekötve, és az egyik gyökérből szeretnénk a másikba eljutni. Ugyanis ekkor klasszikus esetben amíg az induló csúcstól a gráf közepéig érünk, annak a valószínűsége, hogy egy

szinttel lejjebb jutunk, duplája annak, hogy egy szinttel visszalépünk. Azonban miután átértünk a második félgráfba, ez megfordul. Annak a valószínűsége, hogy a cél felé megyünk, fele lesz annak, hogy visszafelé lépünk. Ezzel szemben megmutatható, hogy a kvantum eset várhatóan lineáris időben eljut a célba. Ám ez a példa azon kívül, hogy jól illusztrálja a két világ közti különbséget, másra eddig nem volt jó, ugyanis sajnos valóséletbeli problémák nem használják.



3.3. ábra. *Két 8 levelű teljes bináris fa, a leveleiknél összekötve. Ezen a fán az egyik eredeti gyökérből indulva a másik gyökérig a kvantum séta exponenciális gyorsulást hoz a klasszikussal szemben.*

4. fejezet

Markov-láncok kvantum kiterjesztése

4.1. Kvantum Markov-lánc

A véletlen séták tekinthetőek Markov-láncoknak is. Ugyanis egy Markov-láncban minden lépés csak az aktuális állapottól függ, a korábbi lépésektől független. Épp ezért a kvantum séták vizsgálatakor adódott, hogy a Markov-láncok kvantum változatát is elkezdjék vizsgálni. Klasszikusan a Markov-láncokat igen sok problémakörben alkalmazzák, de sokszor csak a határeloszlása érdekes a láncoknak. Például ilyen probléma a telefonközpontok tervezése a kiszolgálók és igények függvényében, vagy az internetes kereséshez használt algoritmusok. A PageRank algoritmus alapját például egy olyan internetező képzi, aki az oldalon található linkekre adott valószínűséggel kattint. Ha elakad, vagy egy körkörös hivatkozásba botlik, akkor pedig egy véletlenszerű oldalról kezdi újra a böngészést. Ám itt is csak a határeloszlás határozza meg az oldalak sorrendjét. Az ilyen problémákra a kvantumalgoritmusokkal eddig nem találtak gyorsulást.

Azonban a Markov-láncok keresésre is használhatóak. Például ha egy gráf egy adott csúcsát keressük, akkor egy csúcsból kiindulva a gráfon véletlen sétát folytatva pont egy Markov-láncot kapunk. Ha ebben a sétában minden meglátogatott csúcsot megvizsgálunk, egy kereső algoritmust kapunk. Az ilyen problémák már gyorsíthatók kvantumalgoritmusokat felhasználva. Egy ilyen keresőalgoritmus a szerzők alapján elnevezett MNRS-keresés [11].

A cél egy n elemű X halmaz elemei közül egy megjelölt megtalálása. Legyen a megjelöltek halmaza $M \subseteq X$. Egy klasszikus megoldás erre a korábban említett Markov-lánc szimulálása. Most ennek a hatékony kvantum változatát vizsgáljuk meg. Ehhez először néhány jelölést kell bevezetnünk.

Egy Markov-láncot legegyszerűbben a $P = (p_{xy})$ állapot-átmeneti mátrixával lehet leírni. Ennek határeloszlása $\pi = (\pi_x)$, amire $\pi P = \pi$. (Vagyis egy $\lambda = 1$ sajátértékű baloldali sajátvektor, amiből a Perron-Frobenius tétel értelmében irreducibilis Markov-lánc esetén csak egy van.) Ez az az eloszlás, amit a lánc nem változtat meg. Időmegfordított Markov-

lánc alatt azt a $P^* = (p^*_{yx})$ mátrixszal megadott láncot értjük, amire $\pi_x p_{xy} = \pi_y p^*_{yx}$.

Jelölje $\Pi_\psi = |\psi\rangle\langle\psi|$ a $|\psi\rangle \in \mathcal{H}$ által feszített altérre való vetítést, és legyen $\mathrm{ref}(\psi) = 2\Pi_\psi - I$ a $|\psi\rangle$-re történő tükrözés, ahol I az identitást jelöli. Könnyen belátható, hogy a Π_ψ tényleg egy vetítést határoz meg. Ugyanis vegyünk egy ortonormált bázist, aminek $|\psi\rangle$ az egyik eleme. Akkor ha ezt a bázist vesszük, a $|\psi\rangle$-t a Π_ψ operátor helyben hagyja, a többi bázisvektort pedig kinullázza. Mivel minden vektor felírható a bázisvektorok lineáris kombinációjaként, minden vektornak csak a $|\psi\rangle$ irányú vetületét hagyja meg. Tükrözés esetén $\mathrm{ref}(|\psi\rangle)$ a $|\psi\rangle$ bázisvektort önmagában hagyja, ugyanis $2\Pi_\psi |\psi\rangle - I|\psi\rangle = 2|\psi\rangle - |\psi\rangle = |\psi\rangle$, a többi bázisvektort pedig az ellentétébe viszi, ugyanis bármely $|\psi\rangle$-re merőleges $|\varphi\rangle$-re $2\Pi_\psi |\varphi\rangle - I|\varphi\rangle = 0 - |\varphi\rangle = -|\varphi\rangle$.

Azonban nem csak bizonyos állapotokra lehet vetíteni vagy tükrözni, hanem alterekre is. Minden $\mathcal{K}$ altérre, melyet a $|\psi_i\rangle : i \in I$ ortonormált állapotok feszítenek, legyen $\Pi_\mathcal{K} = \sum_{i\in I} \Pi_{\psi_i}$ a $\mathcal{K}$-ra történő vetítés, és $\mathrm{ref}(\mathcal{K}) = 2\Pi_\mathcal{K} - I$ a $\mathcal{K}$-ra tükrözés.

Vezessük be az $\mathcal{A} = \mathrm{Span}(|x\rangle|p_x\rangle : x \in X)$ valamint a $\mathcal{B} = \mathrm{Span}(|p^*_y\rangle|y\rangle : y \in X)$ jelöléseket a $\mathcal{H} = \mathbb{C}^{X\times X}$ megfelelő altereire, ahol

$$|p_x\rangle = \sum_{y\in X} \sqrt{p_{xy}}\,|y\rangle \quad \text{és} \quad |p^*_y\rangle = \sum_{x\in X} \sqrt{p^*_{yx}}\,|x\rangle$$

Itt az $\mathcal{A}$ altérben csak azon állapotok amplitúdója lehet nemnulla, melyeknél x-ből y-ba el lehet jutni. Ekkor ez az amplitúdó az átmenet valószínűségével lesz arányos. Pontosabban ennek a valószínűségnek a gyöke lesz. Mivel a P mátrix sztochasztikus, így ezekkel a szorzókkal az állapotok pont normálva lesznek.

Ezeket a jelöléseket felhasználva definiálhatjuk a kvantum séta unitér operátorát a P mátrix által meghatározott Markov-lánchoz: $W(P) = \mathrm{ref}(\mathcal{B})\mathrm{ref}(\mathcal{A})$.

Felírhatjuk a $W(P)$ diszkriminánsmátrixát, ami $D(P)_{xy} = (\sqrt{p_{xy}p^*_{yx}})$. Mivel $\sqrt{p_{xy}p^*_{yx}} = \frac{\sqrt{\pi_x}}{\sqrt{\pi_y}}p_{xy}$, ezért ez felírható $D(P) = \mathrm{diag}(\pi)^{1/2} P \mathrm{diag}(\pi)^{-1/2}$ alakban, ahol $\mathrm{diag}(\pi)$ egy olyan diagonális mátrix, aminek főátlójában a π eloszlás elemei találhatók. Mivel P egy Markov-lánc mátrixa, ezért $D(P)$ szinguláris értékei mind a $[0,1]$ intervallumba esnek, tehát felírhatók valamely $\theta \in [0, \frac{\pi}{2}]$ szám koszinuszaként, azaz $\cos\theta$-ként.

Legyen $\Delta(P)$ a $W(P)$ fázishézagja (phase gap) 2θ, ahol θ a legkisebb $(0, \frac{\pi}{2})$-be eső szög, amire $\cos\theta$ a $D(P)$ szinguláris értéke.

4.2. MNRS-keresés

Az algoritmust tekinthetjük a klasszikus véletlen keresés kvantum megfelelőjének. Ezért a kiindulási állapot a határeloszlás megfelelője lesz. Ez a következőképp írható fel:

$$|\pi\rangle = \sum_{x\in X} \sqrt{\pi_x}\,|x\rangle\,|p_x\rangle = \sum_{y\in X} \sqrt{\pi_y}\,|p^*_y\rangle\,|y\rangle$$

A keresés felfogható úgy is, hogy egy olyan altérre szeretnénk vetíteni, ami a megjelölt elemeket tartalmazza. Legyen ez az altér $\mathcal{M} = \mathbb{C}^{M \times X}$. Vagyis az $|x\rangle |y\rangle$ első regisztere egy megjelölt elemet tartalmaz. Tehát a cél egy olyan vetítést készíteni, ami erre az altérre vetít. Ha ez sikerülne, egy olyan normalizált $|\mu\rangle$ vektort kapnánk, ami

$$|\mu\rangle = \frac{\Pi_{\mathcal{M}} |\pi\rangle}{||\Pi_{\mathcal{M}} |\pi\rangle||} = \frac{1}{\sqrt{p_M}} \sum_{x \in M} \sqrt{\pi_x} |x\rangle |p_x\rangle$$

ahol $p_M = ||\Pi_{\mathcal{M}} |\pi\rangle||^2 = \sum_{x \in M} \pi_x$ a normáláshoz szükséges tag. Ennek jelentése az, hogy mekkora valószínűséggel jelölt egy elem a határeloszlásban.

A $|\pi\rangle$ kiindulási állapotból a $|\mu\rangle$ állapotba eljuthatunk tükrözésekkel is. Ebben az esetben a $\mathrm{ref}(\pi)\mathrm{ref}(\mu^\perp)$ tükrözések többszöri megismétlése pont jó eredmény adna. (Itt $|\mu^\perp\rangle$ a $|\mu\rangle$-re merőleges állapotokat jelöli.) Ugyanis a $|\pi\rangle$ és a $|\mu^\perp\rangle$ közötti φ szögre igaz, hogy $\sin(\varphi) = \langle\mu|\pi\rangle = \sqrt{p_M}$, így a két tükrözés egymás utáni végrehajtása egy 2φ szögű forgatásnak felel meg. Tehát összességében $O(\frac{1}{\varphi}) = O(\frac{1}{\sqrt{p_m}})$ forgatással $|\mu\rangle$ egy becslését kapjuk.

Azonban $\mathrm{ref}(\mu^\perp)$ egy számításigényes művelet, ezért inkább egy becslését végezzük csak el. Nézzük az $\mathcal{S} = \mathrm{Span}(|\pi\rangle, |\mu\rangle)$ alteret. Ebben a két dimenziós altérben $\mathrm{ref}(\mu^\perp)$ és $\mathrm{ref}(\mathcal{M})$ egymással megegyezik. Ugyanis $\mathrm{ref}(\mathcal{M}) = \mathrm{ref}(\mu)$, valamint $\mathrm{ref}(\mu)$ és $\mathrm{ref}(\mu^\perp)$ hatása között pont egy (-1)-es szorzó van. Vagyis ha az algoritmus folyamán az állapotok ebbe az altérbe esnek, $(-\mathrm{ref}(\mathcal{M}))$-et kell csak megvalósítani. Ennek érdekében vizsgálnunk kell, hogy az első regiszter $\mathcal{M}$-be esik-e.

A $\mathrm{ref}(|\pi\rangle)$ megvalósításához az algoritmus $W(P)$-t használja fel. Ugyanis az $\mathcal{A} + \mathcal{B}$ altér tartalmazza a $|\pi\rangle$ és $|\mu\rangle$ vektorokat, így az $\mathcal{S} = \mathrm{Span}(|\pi\rangle, |\mu\rangle)$ alteret, valamint $W(P) = \mathrm{ref}(\mathcal{B})\mathrm{ref}(\mathcal{A})$-nak $|\pi\rangle$ az egyetlen 1 sajátértékű sajátvektora. Ugyanis P ilyen irányú tulajdonsága kiterjeszthető $W(P)$-re.

Tétel (Spektrális Lemma [16]): Legyen P egy irreducibilis Markov-lánc, és legyenek $\cos\theta_1, \ldots, \cos\theta_l$ a $D(P)$ mátrix $(0, 1)$-be eső szinguláris értékei. Ekkor:

1. $\mathcal{A} + \mathcal{B}$-n $W(P)$ azon sajátértékei, melyeknek képzetes része nem nulla, pontosan az $e^{\pm 2i\theta_1}, \ldots, e^{\pm 2i\theta_l}$ komplex egységgyökök, azonos multiplicitással.

2. $\mathcal{A} \cap \mathcal{B}$-n a $W(P)$ operátor úgy hat, mint az identitás. Az $\mathcal{A} \cap \mathcal{B}$ alteret a $D(P)$ baloldali (és jobboldali) egy sajátértékű szinguláris vektorai feszítik

3. $\mathcal{A} \cap \mathcal{B}^\perp$-en és $\mathcal{A}^\perp \cap \mathcal{B}$-n a $W(P)$ operátor $(-I)$-ként hat. Ezeket az altereket $D(P)$ azon, rendre bal- illetve jobboldali sajátvektorai feszítik, melyek sajátértéke 0.

4. $W(P)$-nek $\mathcal{A} + \mathcal{B}$-n ezeken kívül nincsen más sajátértéke. $\mathcal{A}^\perp \cap \mathcal{B}^\perp$-en $W(P)$ identitásként hat.

Ha feltesszük, hogy P ergodikus és megfordítható, akkor $D(P)$ szimmetrikus lesz, és a szinguláris értékei megegyeznek P sajátértékeinek abszolútértékeivel. Tehát $W(P)$ egyetlen

1 sajátértékhez tartozó sajátvektora $|\pi\rangle$. Emiatt, a következő tételben szereplő fázisbecslést használva a $|\pi\rangle$-re való tükrözést megvalósíthatjuk.

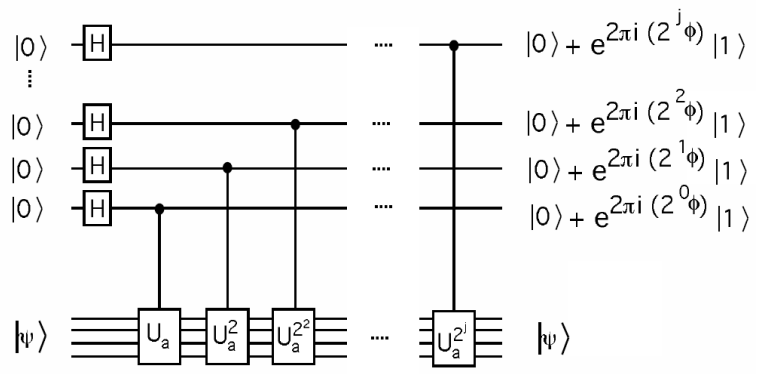
Tétel (Fázisbecslő tétel [3]): Minden $m, s \geq 1$ egész párra és $2^m \times 2^m$-es U unitér operátorra létezik egy $C(U)$ kvantum áramkör, ami $m+s$ qubiten hat, és teljesíti a következő tulajdonságokat:

1. A $C(U)$ áramkör $2s$ Hadamard-kaput és $O(s^2)$ vezérelt fázisforgatást használ, valamint 2^{s+1}-szer hívja a vezérelt unitér $c-U$ operátort.

2. U minden 1 sajátértékű $|\psi\rangle$ sajátvektorához $C(U)\,|\psi\rangle\,|0^s\rangle = |\psi\rangle\,|0^s\rangle$

3. Ha $U\,|\psi\rangle = e^{2i\theta}\,|\psi\rangle$, ahol $\theta \in (0, \pi)$, akkor $C(U)\,|\psi\rangle\,|0^s\rangle = |\psi\rangle\,|\omega\rangle$, ahol $|\omega\rangle$ egy s-qubites állapot, melyre $|\langle 0^s|\omega\rangle| = \frac{\sin(2\theta)}{2\sin\theta}$

Azért nevezzük fázisbecslő tételnek a tételt, mert $|\omega\rangle$-t megmérve $\frac{\theta}{\pi}$ egy becslését kapjuk. Az áramkört a következő ábra mutatja meg:



4.1. ábra. *A $C(U)$ fázisbecslő áramkör [3]*

Ezt a fázisbecslő áramkört felhasználva megalkotható egy $R(P)$ operátor, ami jól közelíti a ref(π)-t. A 4.1. ábrán a jelölések kicsit eltérnek az itt használtaktól. A H kapu az 1 qubiten ható H_1 Hadamard-kapu, az U_a operátor az itt U-nak nevezett kapu, aminek a helyébe a keresés során a $W(P)$ kvantum séta operátort tesszük. Az itt használt s qubites $|\omega\rangle$ állapot az ábrán $j+1$ qubites, és az áramkörből jobb oldalt kijövő qubiteket tartalmazza, a legalsó sor nélkül. Azaz

$$|\omega\rangle = (|0\rangle + e^{2\pi i 2^{(s-1)}\theta}\,|1\rangle)(|0\rangle + e^{2\pi i 2^{(s-2)}\theta}\,|1\rangle)\cdots(|0\rangle + e^{2\pi i 2^0\theta}\,|1\rangle) = \sum_{y=0}^{2^m-1} e^{2\pi i\theta y}\,|y\rangle$$

Tétel: Legyen P egy ergodikus Markov-lánc egy $n \geq 2$ állapotú állapottéren. Ekkor bármely k egészre létezik $R(P)$ kvantum áramkör, amely $2\lceil \log_2 n \rceil + ks$ qubiten hat, ahol $s = \log_2\left(\frac{1}{\Delta(P)}\right) + O(1)$, és teljesíti a következő tulajdonságokat:

1. Az $R(P)$ áramkör $2ks$ Hadamard-kaput és $O(ks^2)$ vezérelt fázisforgatást használ, valamint $k2^{s+1}$-szer hívja a vezérelt $c{-}W(P)$ kvantum-séta operátort, és inverzét, a $c{-}W(P)^+$ operátort.

2. $W(P)$ minden 1 sajátértékű $|\pi\rangle$ sajátvektorához $R(P)\,|\pi\rangle\,|0^{ks}\rangle = |\pi\rangle\,|0^{ks}\rangle$

3. Ha $|\psi\rangle$ az $\mathcal{A}+\mathcal{B}$ egy $|\pi\rangle$-re merőleges alterébe esik, akkor $\big|\big|(R(P)+Id)\,|\psi\rangle\,|0^{ks}\rangle\big|\big| \leq 2^{1-k}$

Bizonyítás. Írjuk le $R(P)$ működését. Legyen $m = n^2$ és $s = \lceil \log_2\big(\frac{2\pi}{\Delta(P)}\big)\rceil$. Először alkalmazzuk a $C(U)$ fázisbecslő áramkört a $W(P)$ kvantum sétára. A becslés pontosítása érdekében $C(U)$-t k-szor hajtjuk végre. Mivel $C(U)$ nem módosítja $W(P)$ sajátvektorait, ezért elég csak a második regisztert k-szorozni, a $|\psi\rangle$-ből nem kellenek további másolatok.

A második regiszter itt k-szor fogja a becsült fázist, azaz a $|\omega\rangle$-t tartalmazni. Így a második regisztert s qubites szakaszokra bontva azon k-szor lehet $|\omega\rangle$-t megmérni.

Ez a művelet bármely $\mathcal{A}+\mathcal{B}$-beli $|\psi\rangle$-t felbont $W(P)$ sajátvektorainak megfelelően, a hozzájuk tartozó becslésekkel együtt. Ez után (-1)-gyel megszorozzuk az összes, legalább egy nemnulla becslést tartalmazó bázist. Ennek célja, hogy $|\pi\rangle$ kivételével az összes sajátvektor fázisát megfordítsuk. Majd végrehajtjuk a fázisbecslés inverzét.

Ekkor $|\pi\rangle\,|0^{ks}\rangle$ nem változik $R(P)$ hatására. Hogyha $|\psi\rangle$ merőleges $|\pi\rangle$-re, akkor $W(P)$ olyan sajátvektorainak lineáris kombinációjaként áll elő, melyek $e^{\pm 2i\theta}$ sajátértékekhez tartoznak, ahol $\frac{\Delta(P)}{2} \leq \theta < \frac{\pi}{2}$. Mivel k-szor ismételtük meg a fázisbecslést, $|\psi\rangle\,|0^{ks}\rangle$-t felbonthatjuk $|\psi_0\rangle + |\psi_1\rangle$-re úgy, hogy $|\psi_0\rangle$ esetén minden $|\omega\rangle$ segédregiszter értéke nulla, és $|\psi_1\rangle$-nek legalább egy segédregiszterében nemnulla van, valamint $||\psi_0|| \leq 2^{-k}$. Így $R(P)\,|\psi\rangle\,|0^{ks}\rangle = |\psi_0\rangle - |\psi_1\rangle$, és $(R(P)+I)\,|\psi\rangle\,|0^{ks}\rangle = 2\,|\psi_0\rangle$.

□

Ezeket felhasználva már felírhatjuk magát az algoritmust:

1. Ötször ismételjük meg a következőt:
   (a) Vegyünk egy x-et véletlenszerűen P-nek a π határeloszlása alapján
   (b) Ha $x \in M$, akkor STOP, és visszaadjuk x-et

2. Legyen T véletlenszerű $[0, \frac{1}{\sqrt{\varepsilon}})]$ intervallumon, legyen $k \in \log_2(T) + O(1)$, valamint legyen s az előző tétel szerinti

3. Ismételjük meg T-szer a következőt:
   (a) Minden $|x\rangle\,|y\rangle\,|z\rangle$ bázisvektorra ($|x\rangle\,|y\rangle = |\psi\rangle$ és $|z\rangle = |\omega\rangle$ az előbbi jelölések szerint), ha $x \in M$, fordítsuk meg a fázisát, azaz

$$|x\rangle\,|y\rangle\,|z\rangle \to \begin{cases} -\,|x\rangle\,|y\rangle\,|z\rangle & \text{ha } x \in M \\ |x\rangle\,|y\rangle\,|z\rangle & \text{különben} \end{cases}$$

(b) Alkalmazzuk az előbbi tétel $R(P)$ áramkörét a fenti k-val, minden iterációban új $|0^{ks}\rangle$ segédqubiteket használva.

4. Mérjük meg az első regisztert

5. Ha ez az $x \in M$, akkor adjuk vissza x-et, különben „nincs megjelölt elem"

Állítás: Ez az algoritmus $D(P) \geq \delta > 0$ esetén megfordítható, ergodikus Markov-láncokra nagy valószínűséggel talál egy megjelölt elemet, ha van olyan.

Bizonyítás. Hogyha M üres, akkor az algoritmus nem talál megjelölt elemet. Tehát feltehetjük, hogy M nem üres. Amennyiben $p_M > \frac{1}{4}$, azaz annak a valószínűsége, hogy egy véletlen elem M-beli nagyobb, mint $\frac{1}{4}$, az első lépés $1 - (\frac{3}{4})^5 > \frac{3}{4}$ valószínűséggel talál egy megjelölt elemet. Tehát feltehetjük, hogy $p_M \leq \frac{1}{4}$. Ekkor megmutatjuk, hogy ez az algoritmus a véletlen Grover-algoritmust szimulálja.

Legyen $\mathcal{S} = \mathrm{Span}(|\pi\rangle, |\mu\rangle)$. A Grover-algoritmus T darab $\mathrm{ref}\,|\pi\rangle\,\mathrm{ref}\,|\mu^{\perp}\rangle$ iterációból áll, ahol T-t véletlenszerűen választjuk $[0, \frac{1}{\sqrt{\varepsilon}}]$ intervallumból. Mivel $\varepsilon \leq p_M \leq \frac{1}{4}$, ezért a Grover-algoritmus konstans valószínűséggel a $|\pi\rangle$ vektort úgy forgatja, hogy a $|\mu\rangle$-vel vett belső szorzata konstans legyen.

Legyen $|\phi_i\rangle$ az az állapot, ahova i Grover iteráció $|\pi\rangle$-t viszi, $|\psi_i\rangle$ pedig az, ahová az előző algoritmus. Indukcióval megmutatjuk, hogy $|||\psi_i\rangle - |\phi_i\rangle|| \leq i2^{1-k}$. Tegyük fel, hogy i-re igaz. $|\psi_i\rangle$ felírható $|\phi_i\rangle + (|\psi_i\rangle - |\phi_i\rangle)$ alakban. Mivel $|\phi_i\rangle \in \mathcal{S}$, ezért $\mathrm{ref}(\mu^{\perp})$ és $(-\mathrm{ref}\mathcal{M})$ hatása megegyezik rajta.

Legyen $|\tau\rangle = |\phi_{i+1}\rangle - R(P) \cdot \mathrm{ref}(\mathcal{M})\,|\phi_i\rangle$. Itt $|\phi_{i+1}\rangle$ a Grover-algoritmus $(i+1)$. lépése, $R(P) \cdot \mathrm{ref}(\mathcal{M})\,|\phi_i\rangle$ pedig a Grover-algoritmus i. lépésére az MNRS-keresés egy lépésének eredménye. Vagyis $|\tau\rangle$ az MNRS-keresés egy lépésének hibája Grover kereséséhez képest. Mivel $\mathrm{ref}(\mathcal{M})\,|\phi_i\rangle \in \mathcal{S}$, és $\mathcal{S} \subseteq \mathcal{A} + \mathcal{B}$, az előbbi tétel harmadik következménye miatt $||\tau|| \leq 2^{1-k}$. Ekkor $|||\phi_{i+1}\rangle - |\psi_{i+1}\rangle|| \leq |||\tau\rangle|| + |||\phi_i\rangle - |\psi_i\rangle||$ teljesül, ugyanis az MNRS keresés hibája $i+1$ lépés után nem lehet nagyobb, mint az i lépés után levő hiba és az $(i+1)$. lépésben elkövetett hiba összege. Ebből pedig az állítás már következik, ugyanis

$$|||\phi_{i+1}\rangle - |\psi_{i+1}\rangle|| \leq |||\tau\rangle|| + |||\phi_i\rangle - |\psi_i\rangle|| \leq 2^{1-k} + i2^{1-k} = (i+1)2^{1-k}$$

Ekkor $k = \log_2(T) + c$ esetén, ahol c konstans, $|||\psi_T\rangle - |\phi_T\rangle|| \leq 2^{1-c}$, ami megfelelően nagy c választása esetén tetszőlegesen kicsivé tehető.

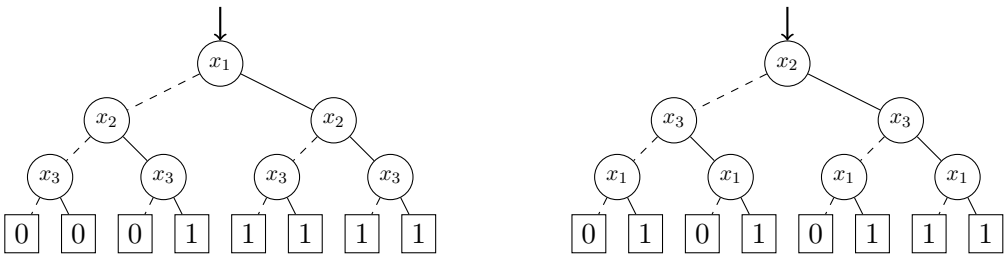
□

5. fejezet

A QuIDD rendszer

5.1. A QuIDD adatszerkezet

A kvantumalgoritmusok vizsgálatánál komoly nehézségeket okoz, hogy tudomásunk szerint nincsen működő kvantumszámítógép, ezért az algoritmusok teszteléséhez klasszikus számítógépen kell szimulálni a qubiteket és a rajtuk végzett műveleteket. Ehhez a legkézenfekvőbb módszer az, ha a qubiteket vektorokkal, az operátorokat pedig mátrixokkal reprezentáljuk. Ekkor a kvantumszámítás lépései egyszerű mátrixszorzások lesznek. Azonban egy n qubites állapot tárolásához 2^n klasszikus bit kell, és az ezen ható operátorok $2^n \times 2^n$-es mátrixok lesznek. Szerencsére ezek az operátorok általában mutatnak valami struktúrát, amit kihasználva lehetőség nyílik arra, hogy tömörebb formában tároljuk a mátrixokat, és a számolás is tömörebb lehessen. Erre ad egy adatszerkezetet a kvantuminformációs döntési diagram (Quantum Information Decision Diagram, QuIDD) [18].



5.1. ábra. *Egy három változós logikai függvény BDD-je x_1, x_2, x_3 valamint x_2, x_3, x_1 sorrendben. A szaggatott él jelöli a* 0 *vagy HAMIS értéket, a folytonos pedig az* 1 *vagy IGAZ értéket.*

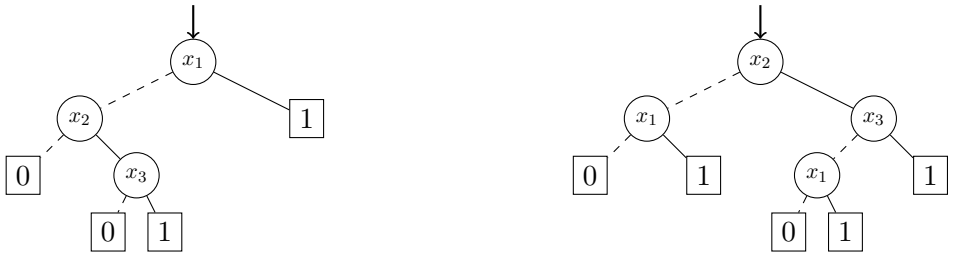
Az adatszerkezet alapja egy bináris döntési diagram (Binary Decision Diagram, BDD). Ezt először logikai függvények ábrázolására használták, ahol a gráf csúcsai egy-egy változót jeleztek, és egy csúcs két gyereke a változó igaz illetve hamis értéke esetén vizsgálandó következő változó volt. A fa levelei a függvény értékeit tárolják olyan változóértékek mellett, amilyenekkel oda eljutottunk. Azonban ez a szerkezet nem tömöríti az adatokat, így n változó esetén 2^n levele lesz a fának. Azaz ha ezt használnánk a qubitek tárolására, a

probléma továbbra is exponenciális maradna. Ezt a hiányosságát pótolja a redukált, rendezett bináris döntési diagram (Reduced Ordered BDD, ROBDD). Ez két egyszerűsítési szabályt vesz a BDD-hez, melyek a következők:

1. Nincs olyan v és v' csomópont, hogy a v és a v' gyökerű részgráfok izomorfak.

2. Nincsenek belső csúcsok, melyeknek mind a két gyereke azonos.

Az 5.1. ábrán az $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$ logikai függvényhez tartozó döntési fát ábrázoltuk a változók két sorrendezését felhasználva. Megfigyelhető, hogy a BDD miért bánik bizonyos helyzetekben pazarlóan a tárhellyel. Az x_1, x_2, x_3 sorrendnél látható, hogy a teljes jobb oldali részfa redundáns, $x_1 = 1$-ből látszik, hogy a kifejezés kielégíthető. A másik, x_2, x_3, x_1 sorrendnél pedig megfigyelhető, hogy a bal oldali x_3 két részfája azonos, az az x_3 csúcs elhagyható.

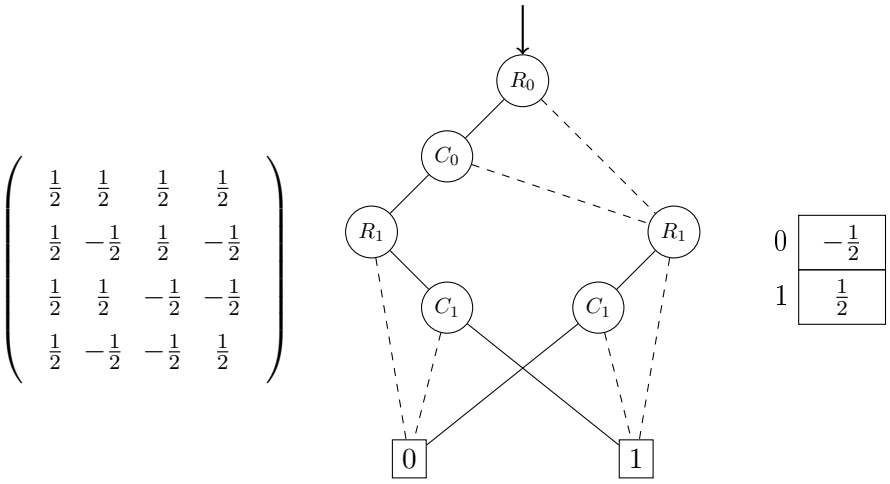


5.2. ábra. *Az előbbi döntési fákból készített ROBDD-k. Látható, hogy a sorrend függvényében más lett a két fa mérete.*

Könnyen belátható, hogy a változók sorrendezése, azaz az, hogy a változók a fában milyen sorrendben követik egymást, nagy hatással lehet a kialakult fa méretére. Ráadásul az optimális sorrend megtalálása NP-nehéz probléma. A gyakorlatban használt esetek egy részében azonban ezzel a módszerrel a változóknak már csak polinomiális függvénye lesz a fa mérete.

A QuIDD tárolási módszere ezen az ROBDD elrendezésen alapul. A csomópontok itt a mátrix sorait illetve oszlopait jelentik, a levelek pedig az út által meghatározott helyen levő értéket tárolják. Pontosabban a csomópontok a sorok illetve oszlopok sorszámának bináris ábrázolásának bitjeit jelentik, azaz „A keresett elem sorának az x. bitje 1?” szerű kérdéseket jelentenek. Tehát egy 2^n sorú mátrixban legfeljebb n csomópont határozza meg az aktuális sort.

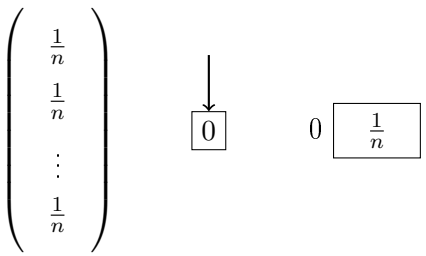
Azonban csak a tárolás nem elég ahhoz, hogy a számítások sebessége növekedhessen, szükség van ezeken a fákon végrehajtható műveletekre is. Ugyanis ha minden művelet elvégzéséhez a fából újra elő kellene állítani a mátrixot, a műveletek elvégzése továbbra is exponenciális időt használna. Az *Apply* művelet azonban kiküszöböli ezt a problémát. Ez rekurzívan bejárja mindkét fát, és a megfelelő csomópontok között végrehajtja a megadott műveletet.



5.3. ábra. *H_2, a két qubiten ható Hadamard kapu, és QuIDD reprezentációja*

Az *Apply* rekurzívan bejárja a két fát, és minden lépésnél megvizsgálja a csomópontot, ahol épp áll. Legyen az egyik fában az aktuálisan vizsgált csomópont v_f, a másikban pedig v_g. Ekkor ha v_f előbb van a fákhoz használt sorrendezésben, mint v_g, az új fába v_f kerül, és a művelet meghívja magát rekurzívan v_f egyik gyerekére és v_g-re, valamint v_f másik gyerekére és v_g-re. Azaz ha $T(v_f)$ jelöli az egyik gyereket, és $E(v_f)$ a másikat, akkor $Apply(v_f, v_g, op)$ az $Apply(T(v_f), v_g, op)$ és az $Apply(E(v_f), v_g, op)$ függvényeket hívja.

Hogyha a v_g van előrébb, akkor ugyan ez történik, csak épp v_g gyerekeivel. Míg ha a két csomópont azonos változóról szól, az $Apply(T(v_f), T(v_g), op)$ és az $Apply(E(v_f), E(v_g), op)$ függvények indulnak el. A rekurzió akkor ér véget, ha mindkét csomópont egy levél. Ekkor az *Apply* végrehajtja a két levélben tárolt értéken az *op* műveletet. Látható, hogy ennek a műveletnek a lépésszáma $O(ab)$, ahol a az egyik fa csomópontjainak a száma, b pedig a másiké.



5.4. ábra. *Egy egyenletesen elosztott kezdővektor, és QuIDD reprezentációja*

A QuIDD is ezt az *Apply* algoritmust használja a műveletek végrehajtásához.

A levelekben nem a konkrét értékeket tárolja, hanem egy mutatót egy tömb megfelelő elemére, ami az értéket tárolja. Továbbá szükséges, hogy ezek az értékek komplex számok legyenek. A változók sorrendje általában olyan, hogy sorok és oszlopok egymást felváltva követik, ugyanis a kvantumalgoritmusokban használt operátorokat így lehet hatékonyan

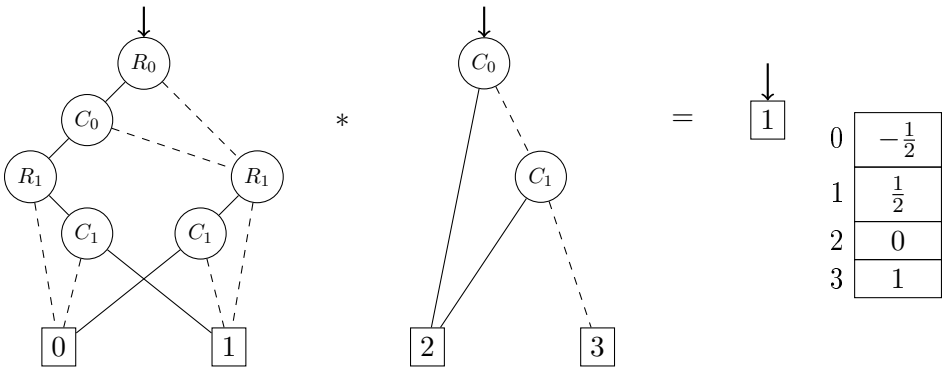
tárolni. Ugyanis ez a blokkos szerkezeteket helyezi előtérbe, ami igen gyakori a tenzorszorzással előállított mátrixoknál.

A tenzorszorzáshoz először át kell alakítani a sorrendezést. Pontosabban a második mátrix változóit el kell tolni az első változói után. Így az első fa minden levele után fejtődik ki a második fa, ami pont a tenzorszorzásnak felel meg. Ugyanis a második mátrix minden elemét megszorozza az első mátrix elemeivel.

A tenzorszorzás akár nagy méretnövekedést is okozhat. Ugyanis egy $n \times n$-es mátrixot egy másik $n \times n$-es mátrixszal szorozva a keletkező mátrix $n^2 \times n^2$-es lesz. Azonban ez a növekedés az adatstruktúrában bizonyos esetben nem fog bekövetkezni.

Lemma: Adott Q_i QuIDD-ek esetén ezek $\otimes_{i=1}^{n} Q_i$ tenzorszorzatának QuIDD-je $|In(Q_1)| + \sum_{i=2}^{n} |In(Q_i)| \left|Term(\otimes_{j=1}^{i-1} Q_j)\right| + |Term(\otimes_{i=1}^{n} Q_i)|$ csomópontot tartalmaz. Itt $In(Q_i)$ a QuIDD belső csúcsait jelenti, $Term(Q_i)$ pedig a leveleket.

Ha a tenzorszorzat végeredményében a levelek, azaz az előforduló különböző elemek száma nem növekszik, akkor az egész reprezentáció mérete n-ben lineáris marad. Ezért ha egy operátor előáll kisebb operátorok tenzorszorzataként, és nem tartalmaz sok különböző elemet, annak mérete $O(n)$ lesz.



5.5. ábra. $H_2 |00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$, *mátrixszorzás QuIDD formában*

A mátrixszorzás végrehajtása felírható skalárszorzások sorozataként is. Azonban a skaláris szorzáshoz nem elég csak simán az *Apply*-t használni, ezért Bahar et al. mátrixszorzó algoritmusát [2] használja a QuIDD. Ennek a legnagyobb előnye, hogy magán a tömörített QuIDD szerkezeten is működik, nem kell visszaállítani belőle az eredeti mátrixot. Az algoritmusnak a lépésszáma $O((ab)^2)$, ahol a és b a két mátrix QuIDD reprezentációjának a csomópontjainak a száma. Valamint, mint az összes ilyen fán végrehajtott algoritmus esetében, a létrehozott új fa mérete is ebben a nagyságrendben van, azaz $O((ab)^2)$. Azonban, ahogy az 5.5. ábrán is látható, a szorzás után az új mátrix reprezentációjának mérete akár csökkenhet is.

5.2. A Grover-féle keresés lépésszáma QuIDD használatával

A QuIDD adatszerkezetet leíró cikkben a polinomiális lépésszám megmutatására Grover híres keresőalgoritmusát használták. Erre egy igen durva, $O(|A|^{16}\, n^{14})$ felső becslést adtak, ahol $|A|$ a V_f mátrix, azaz az orákulum QuIDD reprezentációjában a csomópontok száma, n pedig az állapotvektor mérete, azaz 2^n darab elem között folyik a keresés. A cikkben nem részletezik, hogy hogyan jött ki ez a becslés, ezért itt levezetem.

A mátrixszorzás lépésszáma QuIDD reprezentációval $O(|A|^2\, |B|^2)$, ahol $|A|$ illetve $|B|$ a két mátrix reprezentációjának csomópontszáma. Az algoritmus első lépése az állapotvektor inicializálása, azaz n qubit nullába állítása. Ez után a $H_n\,|x\rangle$ szorzást kell végrehajtani, hogy az új $|x\rangle$ az összes állapot egyenletes szuperpozíciójában legyen. Mivel az Hadamard mátrix $O(n)$ csomóponttal ábrázolható, $|x\rangle$ pedig 1 csomópont, a szorzás lépésszáma $O(n^2)$. Eddig tartott az algoritmus inicializáló része, ami bár $O(n^2)$ lépés, viszont az eredmény tárolható egy csomópontban, mivel az állapotvektor minden eleme azonos.

Az algoritmus ismétlendő része leírható $(-H_nR_nH_nV_f\,|x\rangle)$ alakban, ahol $|x\rangle$ az előző lépésben előállított vektor, azaz egy csomóponttal tárolható. V_f mérete legyen $|A| = |V_f|$, hogy a végeredmény a cikkben levővel azonos alakban legyen. Ekkor az első szorzás lépésszáma $O(|A|^2)$. Ez lesz a következő szorzás jobb oldalán levő elem mérete. A H_n-nel való szorzás $O((n\,|A|^2)^2) = O(|A|^4\, n^2)$ lépéssel megoldható.

Állítás: R_n szintén megadható $O(n)$ csomóponttal.

Bizonyítás. R_n egy diagonális mátrix, melynek a bal felső eleme (-1), a főátló többi eleme pedig 1. Könnyen látható, hogy egy azonos méretű egységmátrix kisebb egységmátrixok tenzorszorzataként előállítható. Ráadásul mindegyik QuIDD-nek két levele van. Valamint egy olyan mátrix, ami a jobb felső sarokban (-2)-t tartalmaz, a többi helyen pedig 0-t, négy csomóponttal tárolható. Egy a sort, egy az oszlopot kódolja, a további kettő pedig a 0 és a (-2) értékeket tárolja. Majd az egységmátrixhoz hozzáadva ezt a mátrixot megkapjuk R_n-et. Az összeadás lépésszáma $O(ab)$, ahol a az egységmátrix csomópontjainak száma, ami $O(n)$, b pedig a (-2)-t tartalmazó mátrixé, ami 4. Ugyanis az *Apply*-t kell használni *op* helyére $+$-t téve. Tehát R_n mérete $O(4n) = O(n)$. □

A kereső algoritmus következő lépése az R_n-nel való szorzás, amihez emiatt elég $O((n\,|A|^4\, n^2)^2) = O(|A|^8\, n^6)$ lépés.

Végül ismét H_n-nel kell szorozni, ami továbbra is $O(n)$ méretű, így a lépésszáma az ismétlődő részben $O((n\,|A|^8\, n^6)^2) = O(|A|^{16}\, n^{14})$ lesz, ahogy a cikkben is írták.

Állítás: A Grover algoritmus egy iterációjának lépésszáma $O(|A|^{16}\, n^{14})$, azaz polinomiális a qubitek számában.

Bizonyítás. Az algoritmus inicializálása tehát $O(1)$ lépéssel megoldható az eredmény struktúrája miatt. Ezután az egy iterációhoz szükséges ($-H_n R_n H_n V_f |x\rangle$) kiszámítása az előbbieket felhasználva $O(|A|^{16} n^{14})$. □

Állítás: Az $|x\rangle$ állapotvektor leírásához szükséges állapotok száma nem változik egy Grover-iteráció után.

Bizonyítás. Az algoritmus működése során a keresett elemekhez tartozó értéket növeli, a többihez tartozót pedig ennek megfelelően csökkenti minden iteráció. Azonban minden keresetthez tartozó érték megegyezik, és minden nem keresetthez tartozó is megegyezik. □

Az $O(|A|^{16} n^{14})$-es becslés nagyon tág, ezért megvizsgálom, hogy lehetne finomítani. Egy kézenfekvő ötlet, amit sok helyen, például a számítógépes grafikában már rég óta széleskörűen használnak, hogy a mátrixszorzásokat elég egyszer, előre elvégezni. Ugyanis a mátrixok szorzása asszociatív, így nem számít, hogy hogyan zárójelezzük őket. Továbbá mivel a keresés folyamán a szükséges operátorok nem változnak, ezt megtehetjük. H_n egy bemenettől független transzformáció, tehát az biztosan nem fog változni. R_n szintén változatlan, ugyanis a $|0\rangle$ állapotot negálja, függetlenül a keresett elemtől. Az egyedüli mátrix, ami függ a konkrét kereséstől, az a V_f. De az is csak magától a keresendő elemtől függ, ami a keresés folyamán nem változik, ezért az algoritmus indításakor lekérdezhető az orákulumtól, és a végrehajtás folyamán nem fog változni. Tehát a $H_n R_n H_n V_f$ szorzat előre kiszámolható.

Állítás: A $H_n R_n H_n$ mátrix $O(n)$ csomóponttal tárolható.

Bizonyítás. Mint azt már korábban láthattuk, $H_n R_n H_n = I - 2P$, azaz egy olyan mátrix, ahol a főátló elemei azonosak, és a főátlón kívüli elemek is megegyeznek. Tehát összesen két féle elem található a mátrixban, méghozzá egy diagonális elrendezésben. A főátlóban levő elemek $1 - \frac{2}{2^n}$, a többi elem pedig $-\frac{2}{2^n}$. Legegyszerűbben ez a mátrix $I - 2P$ alakban állítható elő. Egy $n \times n$-es egységmátrixot elő tudunk állítani tenzorszorzatként, így az $O(n)$ csomóponttal tárolható. A P mátrix előállításához olyan 2×2-es mátrixból indulunk ki, aminek minden eleme $\frac{1}{2}$. Ezt önmagával tenzorszorozzuk addig, míg a megfelelő méretű mátrixot nem kapjuk, aminek az elemei pont $\frac{1}{2^n}$ lesznek. Ez a mátrix egy csomóponttal tárolható, ugyanis minden eleme azonos. Két mátrix különbségét az *Apply* algoritmus minusz operátorral való meghívásával számolhatjuk ki, aminek a lépésszáma $O(ab)$, azaz ebben az esetben $O(n \cdot 1)$. Mivel $(I - 2P)$-t számolunk, a P mátrix kétszeresét kell kivonni az egységmátrixból. Egy mátrix kétszeresének előállítása pedig megoldható az értékeket tároló tömb minden elemének 2-vel szorzásával, ami nyilvánvalóan nem módosítja a tárolásához szükséges csomópontok számát. Azaz $(I - 2P)$-t kiszámolni $O(n)$ lépés. □

Állítás: V_f mátrix eltárolható $O(n)$ csomóponttal.

Bizonyítás. V_f egy olyan mátrix, ami csak a főátlóban tartalmaz nullától különböző elemeket. Azon belül is mindenhol 1, kivéve a keresett elem helyén, ugyanis ott (-1). Az egységmátrix továbbra is tárolható $O(n)$ csomóponttal, továbbá egy olyan mátrix, aminek egy eleme (-2), a többi pedig nulla, négy csomóponttal tárolható. Így V_f mérete $O(n \cdot 4) = O(n)$ lesz. □

Ekkor $H_nR_nH_n$-t jobbról megszorozzuk a V_f mátrixszal, ami $O((ab)^2)$ lépés, ami jelen esetben $O((n \cdot n)^2) = O(n^4)$ lépés, ami azt jelenti, hogy $H_nR_nH_nV_f$ tárolásához is elegendő ennyi csomópont.

Állítás: $H_nR_nH_nV_f$ eltárolható $O(n)$ csomóponttal.

Bizonyítás. A négy mátrix szorzata egy olyan mátrix lesz, ami abban különbözik az $I-2P$ mátrixtól, hogy a keresett elem oszlopa (-1)-gyel meg van szorozva. Tehát ha veszünk egy X mátrixot, amiben a keresett elem oszlopa $\frac{4}{2^n}$, és ezt kivonjuk az $I-2P$-ből, az már majdnem jó. Ekkor ugyanis a főátlóbeli elem még nem lesz megfelelő, de a többi már igen. A főátlóban a keresett elem oszlopban levő elemnek $(-1+\frac{2}{2^n})$-nek kell lennie, de így $1-\frac{2}{2^n}-\frac{4}{2^n}$ lesz. Tehát egy olyan mátrixot adunk hozzá, amiben a főátló megfelelő eleme $(-2+\frac{8}{2^n})$, mindenhol máshol nulla. Ez négy csomóponttal leírható. Az X mátrix három csomóponttal adható meg, ugyanis ki kell választani a megfelelő oszlopot, az egy értéket vesz fel, az összes többi elem értéke nulla. Ezekből a mátrixokból V_f előállítható $O(n \cdot 4)$ és $O(n \cdot 3)$ lépéssel, tehát $O(n)$ méretű lesz az eredmény. □

Tehát az iteráció egy lépése egy $O(n)$ méretű mátrixszal való szorzás, ami $O((n \cdot 3)^2)$ lépés, ugyanis az állapotvektor egy kivételével minden eleme megegyezik. A keresett elem helyén álló érték különbözik csak, így egy belső csúcs és két levél elég a tárolásához.

Érdekes lehet megvizsgálni, hogy ennek a szorzatnak az előállítása mekkora számításigényű. Ehhez a $H_nR_nH_nV_f$ szorzat kiszámításának lépésszámát kell vizsgálni. Mindegyik mátrix $O(n)$ méretű, így a művelet lépésszáma $O((((n \cdot n)^2 \cdot n)^2 \cdot n)^2) = O(n^{22})$. Innen adódik a következő állítás.

Állítás: Ha a Grover-féle keresőalgoritmusban az iterációs lépésben a $H_nR_nH_nV_f$ mátrixot előre kiszámoljuk, és csak ezzel szorzunk, QuIDD reprezentáció használata esetén $O(n^{22})$ lépés az inicializálás, valamint $O(n^2)$ lépés az iterációs szakasz. □

Azonban a mátrixszorzás asszociativitása miatt megvizsgálhatunk más felbontást is. Vizsgáljuk a $(H_nR_nH_n) \cdot (V_f) \cdot |x\rangle$ alakot. Ekkor az inicializáló szakaszban csak a $H_nR_nH_n$ szorzatot számoljuk ki. Ez $O(((n \cdot n)^2 \cdot n)^2) = O(n^{10})$ lépés. Viszont az algoritmus magjában

ekkor két szorzást kell végrehajtani. Először a $V_f |x\rangle$ szorzást kell végrehajtani, ami $O(n^2)$ lépés.

Állítás: A $V_f |x\rangle$ szorzat három csomóponttal tárolható.

Bizonyítás. $|x\rangle$ állapotvektor három csomóponttal leírható, ugyanis csak a keresett elem helyén álló érték különbözik a többitől. V_f pedig ennek az elemnek fordítja meg az előjelét, vagyis a vektor felépítése nem változik. Ezért továbbra is három csomópont elegendő a tárolásához, amiből egy belső csomópont, kettő pedig levél. □

Állítás: Ha a Grover-féle keresőalgoritmusban az iterációs lépésben a $(H_n R_n H_n)$ mátrixot előre kiszámoljuk, és $(H_n R_n H_n) \cdot V_f |x\rangle$ zárójelezéssel számoljuk, QuIDD reprezentáció használata esetén $O(n^{10})$ lépés az inicializálás, valamint $2 \cdot O(n^2)$ lépés az iterációs szakasz.

Bizonyítás. Az inicializáló szakaszban most csak a $H_n R_n H_n$ szorzást kell elvégeznünk, ami $O(((n \cdot n)^2 \cdot n)^2) = O(n^{10})$ lépéssel elvégezhető.

Az iteráció lépésszáma a két szorzás lépésszámának összege lesz. Az első a $V_f |X\rangle$, ami $O((n \cdot 3)^2)$ lépés, a második pedig ennek megszorzása a $H_n R_n H_n$ mátrixszal. Mivel az előző szorzás eredménye három csomóponttal tárolható, a második szorzás lépésszáma szintén $O((n \cdot 3)^2)$ lépés lesz. Ezek összege $2 \cdot O(n^2)$ lépés. □

Megnézhetjük azt is, hogy milyen eredményt kapnánk, ha az eredeti becslést finomítanánk a szorzások után kapott struktúra elemzésével, azaz ha a négy mátrixszal egymás után szoroznánk. Azonban ez az út nem célravezető. Ugyanis az Hadamard mátrixot épp azért használjuk, mert, szemléletesen szólva, elkeni az állapotvektort, így az azzal való szorzás csak az $O(n)$ csomópontú becslést használhatnánk. Ugyan a cikkben levő becslésnél ez is jobb lenne, mivel ott csak a polinomiális kapcsolat bizonyítása volt a cél, az előző kettő módszerhez képest nagyon rossz eredményt kapnánk.

A három különböző zárójelezést a QuIDDPro programmal ki is próbáltam 100 qubitre, egy iteráción. Ekkor $2^{100} \approx 10^{30}$ elem között keresünk. A teljes algoritmushoz ekkor $r \approx 8,8 \cdot 10^{14}$ iterációt kellene futtatni.

Tehát az eredmények egy táblázatban összefoglalva:

	lépésszám		futási idő	
módszer	inicializálás	szorzás	inicializálás (s)	egy iteráció (s)
eredeti	$O(n)$	$O(\|A\|^{16} n^{14})$	0,083	1,38
$(H_n R_n H_n V_f) \cdot \|x\rangle$	$O(n^{22})$	$O(n^2)$	1,257	0,3
$(H_n R_n H_n) \cdot (V_f) \cdot \|x\rangle$	$O(n^{10})$	$2 \cdot O(n^2)$	0,234	0,303

5.1. táblázat. *Lépésszámok és futási idők egy keresett elem esetén*

Az egész iterációt $R = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor$-szer kell elvégezni, ahol jelen esetben $M = 1$ a keresett elemek száma és N a bemeneti adatok száma, azaz $n = \log(N)$. Ezért az iterációk száma n-ben exponenciális. Vagyis az inicializációban levő lépések száma összességében elhanyagolható az egész algoritmushoz képest, sokkal inkább az iterációban résztvevő feladatok lépésszámát éri meg csökkenteni. Emiatt úgy tűnik ideálisabb, ha az egész $H_n R_n H_n V_f$ szorzatot előre kiszámoljuk.

6. fejezet

Egy lehetséges felhasználás: 3-SAT

6.1. A 3-SAT probléma

Ebben a fejezetben egy konkrét példán keresztül vizsgáljuk meg az MNRS keresőalgoritmus működését. Ez a példa a 3-SAT probléma lesz egy konkrét bemeneten.

A 3-SAT probléma bemenete egy 3-konjunktív normálformára (3-CNF) rendezett logikai változókból álló kifejezést, és a kimenete az, hogy ez a kifejezés kielégíthető-e, azaz létezik-e olyan helyettesítési értéke a logikai változóknak, amire a kifejezés értéke igaz.

A CNF a logikai változók egy olyan elrendezése, amiben a logikai változók, vagy azok negáltjai diszjunkciók konjunkciójaként szerepelnek, azaz vagyok ése. Egy képletben összefoglalva: $\bigwedge_i \bigvee_j (\neg) x_{ij}$. A 3-CNF annyiban különbözik ettől, hogy minden egyes vagyban három logikai változó vesz részt.

A logikai változókat szokás atomoknak, az atomokat és negáltjaikat közös néven literáloknak, ezek diszjunkcióit pedig klózoknak nevezni.

Azért a 3-SAT a választott probléma, mert ez elég kicsi, ám algoritmikusan nehéz probléma. Ugyanis a 2-SAT, ami 2-CNF alakú bemenetet kap, egy P-beli probléma, azaz létezik rá polinom idejű algoritmus. Ellentétben a 3-SAT-tal, ami NP-teljes, azaz jelen ismereteink szerint nem létezik rá polinom idejű algoritmus.

A továbbiakban vizsgált példa a következő 3-CNF lesz:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Ebben a példában x_1, x_2 és x_3 az atomok, és $(x_1 \vee x_2 \vee \neg x_3)$, $(x_1 \vee \neg x_2 \vee x_3)$ valamint $(x_1 \vee x_2 \vee x_3)$ a klózok.

6.2. Klasszikus megoldás

A 2-SAT problémára létezik determinisztikus, polinom időben futó megoldás [9]. Ennek az alapja az, hogy $(x \vee y)$ és $(\neg y \vee z)$ alakú klózokból $(x \vee z)$ alakúakat készítünk. Ha ennek

az átalakításnak az ismételt alkalmazásával nem kapunk egy változóra mind $(x \vee x)$, mind $(\neg x \vee \neg x)$ alakú közt, akkor a kifejezés kielégíthető. Az algoritmus $O(n^4)$ lépést használ.

Ezzel szemben az egyik leggyorsabb klasszikus 3-SAT megoldó algoritmus a véletlent használja, és a futásideje exponenciális. Az algoritmus maga Schöning nevéhez fűződik [14]. Mivel az algoritmus véletlent használ, a megoldása csak nagy valószínűséggel lesz helyes megoldás, de az algoritmus többszöri futtatásával ez a valószínűség tetszőlegesre növelhető.

Az algoritmus bemenete egy k-CNF, n változóval. Az algoritmus a következő:

1. Válasszunk tetszőleges kezdeti értékeket a változóknak $\{0, 1\}$-ből.
2. Ismételjük $3n$-szer a következőt:
   (a) Ha a formula igaz, STOP, és visszaadjuk a változók aktuális értékét.
   (b) Válasszuk ki az egyik hamis klózt, és az egyik benne szereplő literál értékét állítsuk az ellentettjére.
3. Ha nem találtunk olyan választást, ami kielégítené a bemenetet, nem kielégíthetőt adunk eredményül.

Az algoritmus lépésszáma 3-CNF-ek esetén $\left(\frac{3}{4}\right)^n$.

A módszer használható a 2-SAT probléma megoldására is. Ekkor minden csere $\frac{1}{2}$ valószínűséggel helyes lesz, ugyanis ha a kifejezés kielégíthető, az adott hamis klózban legalább az egyik literált negálni kell. Ha a változók $0-1$ sorozatát egy számnak fogjuk fel, akkor a számegyenesen vett véletlen bolyongást kapunk, ami várhatóan $O(n^2)$ lépésben jut el n távolságra, ami a helyes megoldás maximális távolsága a kiindulási állapottól.

Ezzel szemben a 3-SAT probléma esetén a jó lépés valószínűsége csak $\frac{1}{3}$. Azaz az esetek nagyobb részében rossz irányba lép, ezért előfordulhat az is, hogy sokáig rossz irányba megy a séta, és jobban megéri újrakezdeni a kiindulási állapotból, mint megvárni, hogy a séta visszataláljon. Megmutatható, hogy az algoritmus $\Theta(2^n)$-es lépésszámú.

Az algoritmus felfogható egy Markov-láncnak is, amiben egy hiperkocka élei mentén lépkedünk. Ugyanis a változók egy konkrét értékadását leírhatjuk egy n karakter hosszú $0-1$ sorozattal. Ha ezeket a szavakat vesszük a Markov-lánc elemeinek, akkor pontosan az egy Hamming-távolságú szavak lesznek szomszédosak, azaz azok a szavak, amik egy karakterben különböznek egymástól.

6.3. Kvantum megoldás

Az algoritmus kvantum változatában nem maga a 3-CNF a bemenet, hanem egy orákulum. Ez az orákulum egy operátorként van megadva. Legyen X az összes lehetséges megoldások halmaza, amit n bites $0-1$ sorozatokként reprezentálhatunk. Ekkor az i. bit 0 értéke az x_i

logikai változó hamis értékét jelöli, az 1 értéke pedig annak az igaz értékét. Legyen továbbá $M \subset X$ a formulát kielégítő megoldások halmaza. Ekkor az orákulum hatása a következő lesz:

$$|x\rangle \mapsto \begin{cases} -|x\rangle & \text{ha } x \in M \\ |x\rangle & \text{különben} \end{cases}$$

Ebben a konkrét példában az orákulum a következőképp néz ki:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Ha a három logikai változót $x_3x_2x_1$ sorrendben írjuk le, az így kapott $0-1$ sorozatot bináris számnak véve egy $0 \leq x \leq 7$ számot kapunk. Ez adja meg a mátrix megfelelő sorát, ha a számozást nullával kezdjük. Tehát például az $x_1 = \text{IGAZ}, x_2 = \text{HAMIS}, x_3 = \text{HAMIS}$ értékeket behelyettesítve a 001 sorozatot kapjuk, ami a második sort illetve oszlopot jelenti. Ez az eleme az orákulumnak -1, azaz ez a behelyettesítés kielégíti az adott kifejezést.

Ahhoz, hogy a kvantumalgoritmust tudjuk alkalmazni a konkrét példára, előbb meg kell konstruálnunk az operátort. Ehhez azonban tudnunk kell a Markov-lánc átmeneti mátrixát. Az előző, klasszikus megoldásból kiindulva itt is olyan sétát vizsgálunk, ahol minden lépésben egy Hamming-távolságú szavak között lépünk. Ekkor a P mátrix a következőképp fog kinézni:

$$P = \frac{1}{3} \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(Az $\frac{1}{3}$ szorzó ez elején a normálás miatt van, hogy a mátrix tényleg valószínűségeket adjon.) Mivel minden szóból a tőle egy Hamming-távolságra levőkbe tudunk menni, ezért az időmegfordított lánc mátrixa megegyezik a simáéval, tehát $P^* = P$ és $p^*_{yx} = p_{xy}$.

A következő lépés a kvantum séta $W(P)$ operátorának az előállítása. Ezt a definíció

szerint $W(P) = \mathrm{ref}(\mathcal{B}) \cdot \mathrm{ref}(\mathcal{A})$ formában valósítjuk meg, ahol $\mathcal{A} = \mathrm{Span}(|x\rangle \sqrt{p_{xy}} |y\rangle)$ valamint $\mathcal{B} = \mathrm{Span}(\sqrt{p^*_{yx}} |x\rangle |y\rangle)$, tehát minden állapothoz össze kell szedni, hogy melyik állapotba léphet, és az ezek által generált alterekre vivő vetítéseket kell összeadni, majd abból egy tükrözést készíteni. A három változónak nyolc állapota van, amit három qubiten lehet tárolni. Azonban az itt szükséges tér $|x\rangle |y\rangle$ vektorokból áll, ami hat qubites, mivel mind $|x\rangle$, mind $|y\rangle$ az állapotokat kódolja.

Példaképp nézzük meg, hogy a $|x\rangle = |000\rangle$ esetben mit kell csinálni. A P mátrix alapján három állapotba léphetünk $|000\rangle$-ból, így az ekkor készítendő alteret a

$$|\psi_0\rangle = \frac{1}{\sqrt{3}} |000\rangle (|001\rangle + |010\rangle + |100\rangle)$$

vektor generálja. Az altérre való vetítést a $\Pi_{\psi_0} = |\psi_0\rangle \langle\psi_0|$ projekciós mátrixszal végezhetjük el. Miután mind a nyolc állapotra elkészítettük a vetítési mátrixokat, az $\mathcal{A}$ altérre való vetítést a $\Pi_{\mathcal{A}} = \sum_{i=1}^{8} \Pi_{\psi_i}$ mátrix adja. Innen az altérre való tükrözés mátrixa könnyen megkapható, ugyanis $\mathrm{ref}(\mathcal{A}) = 2\Pi_{\mathcal{A}} - I$.

A $\mathcal{B}$ altérre való tükrözés teljesen hasonló módon készíthető el. Ekkor a $|y\rangle = |000\rangle$-hoz tartozó altér a következő lesz:

$$|\varphi_0\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle) |000\rangle)$$

Innen a vetítés erre az altérre $\Pi_{\varphi_0} = |\varphi_0\rangle \langle\varphi_0|$, a teljes $\mathcal{B}$ altérre pedig $\Pi_{\mathcal{B}} = \sum_{i=1}^{8} \Pi_{\varphi_i}$. Az előzővel megegyező módon a tükrözés $\mathrm{ref}(\mathcal{B}) = 2\Pi_{\mathcal{B}} - I$.

Miután ezt a két tükrözést kiszámítottuk, a kvantum séta operátorát ezek összeszorzásával kaphatjuk meg: $W(P) = \mathrm{ref}(\mathcal{B}) \cdot \mathrm{ref}(\mathcal{A})$. Ez ebben az esetben egy $2^6 \times 2^6$-os, azaz 64×64-es mátrix lesz.

Ez után a $C(U)$ fázisbecslő áramkört, illetve az azt végrehajtó operátort kell megalkotnunk a $W(P)$ kvantum sétára. Ennek az egyik paramétere egy s változó, ami a becslés pontosságát határozza meg. Az algoritmus erre $s = \lceil \log_2(\frac{2\pi}{\Delta(P)}) \rceil$-t használja, ahol $\Delta(P)$ a $D(P) = \mathrm{diag}(\pi)^{1/2} P \mathrm{diag}(\pi)^{-1/2}$ mátrixban 2θ, ahol θ a legkisebb $(0, \frac{\pi}{2})$ közötti szög, amire $\cos\theta$ a $D(P)$ mátrix szinguláris értéke. Mivel jelen esetben $P = D(P)$, ezért elég P-t vizsgálni. Továbbá mivel P szimmetrikus, tehát normális, a szinguláris értékek megegyeznek a sajátértékek abszolútértékével. Így most $\cos\theta = \frac{1}{3}$ lesz, ahonnan $s = 2$.

Az áramkört megvalósító operátort az algoritmus leírását tartalmazó fejezetben levő ábra alapján készítjük el. Ehhez először a $W(P)$ operátornak elkészítjük egy vezérelt változatát, $c{-}W(P)$-t. Ez $W(P)$-nél eggyel több, azaz 7 qubiten hat. A plusz egy qubit az engedélyező qubit. Ha az 1, akkor $c - W(P)$ a maradék qubiten úgy hat, mint $W(P)$, ha pedig 0, akkor identitásként viselkedik. Manuálisan úgy lehet egy ilyen kaput megalkotni (feltéve, hogy az első qubit a vezérlő qubit), hogy a $W(P)$-t reprezentáló mátrixszal azonos méretű

egységmátrixot veszünk, és a következő blokkmátrixot készítjük el:

$$c - W(P) = \begin{bmatrix} I & 0 \\ 0 & W(P) \end{bmatrix}$$

ahol I a $W(P)$-vel megegyező méretű egységmátrix.

Ez után $W(P)$ megfelelő hatványaival ismételjük meg ugyanezt. A konkrét esetben csak $W(P)^2$ szükséges $C(U)$ megalkotásához. Majd szükséges ezek inverze, de mivel ezek unitér mátrixok, ez transzponálással és konjugálással könnyen előállítható. Ezek a műveletek a QuIDD adatstruktúrán könnyen elvégezhetőek. A transzponáláshoz minden csomópontnál fel kell cserélni az oszlopokat sorokra, és viszont, a konjugáláshoz pedig az értékeket tároló tömb minden elemét konjugálni kell. Mivel ekkor mind a belső csomópontokat (sor-oszlop csere), mind a leveleket (konjugálás) csak egyszer kell változtatni, ez $O(a)$ lépésben elvégezhető, ahol a a QuIDD reprezentáció csomópontjainak a száma.

Az algoritmus kiindulási állapota a $|\pi\rangle = \sum_{x \in X} \sqrt{\pi_x} |x\rangle |p_x\rangle$. Itt π_x a Markov-lánc stacionárius eloszlásának x. elemét jelenti. Mivel a konkrét lánc 8 állapotú, így π egy 8 elemű vektor, $\pi = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. Így a kiindulási állapot a következő lesz:

$$\begin{aligned} |\pi\rangle = \sqrt{\frac{1}{8}}\sqrt{\frac{1}{3}}(& |000\rangle (|001\rangle + |010\rangle + |100\rangle) + |001\rangle (|000\rangle + |011\rangle + |101\rangle) + \\ & + |010\rangle (|011\rangle + |000\rangle + |110\rangle) + |011\rangle (|010\rangle + |001\rangle + |111\rangle) + \\ & + |100\rangle (|101\rangle + |110\rangle + |000\rangle) + |101\rangle (|100\rangle + |111\rangle + |001\rangle) + \\ & + |110\rangle (|111\rangle + |100\rangle + |010\rangle) + |111\rangle (|110\rangle + |101\rangle + |011\rangle)) \end{aligned}$$

Az algoritmus első lépése szerint öt elemet véletlenszerűen megvizsgálunk, hogy az megoldás-e. Ezt a lépést a példában kihagyjuk, mert erre csak amiatt van szükség, hogy a bizonyításban fel lehessen tenni, hogy a megjelölt elem megtalálásának valószínűsége $p_m \leq \frac{1}{4}$.

Ez után kiválasztunk egy T számot, ahányszor az algoritmus a fő ciklust ismételni fogja. Ez egy $[0, 1/\sqrt{\varepsilon}]$ közötti egyenletes eloszlásból adódó szám. Ebben a példában a jó elemek aránya $\frac{5}{8}$, így T most $[0, 2]$ intervallumból származik. Itt választunk egy k-t is, ami a fázis keresés pontosítása miatt szükséges, ugyanis az iterációban a fáziskeresést k-szor végezzük el egyszerre, hogy az eredmény pontosabb legyen. A mostani példában $T = 2$ és $k = 1$ választással élek.

A következő lépés a kezdeti állapotot állítja elő. Majd következik maga az iteráció. Ebben először az orákulummal ellenőrzi, hogy melyik elem megfelelő. Majd az eredményre alkalmazza az $R(P)$ operátort. Ez először a $C(U)$ áramkört hajtja végre vezérelt $W(P)$ operátorokkal, majd minden fázist megfordít (megszoroz (-1)-gyel), végül a fázisbecslés inverzét alkalmazza. Az algoritmus számára csak a nem nulla fázisú állapotok fázisát kell

megfordítani, de mivel a nulla fázis (-1)-szerese szintén nulla, sokkal egyszerűbb, ha minden fázist megfordítunk. Az eredmény ugyan az lesz, viszont ezt egy egyszerű fázisfordító kapuval meg lehet csinálni, nem kell valahogy megjelölni azokat, amiket meg szeretnénk fordítani.

A kvantumalgoritmus futtatásának szimulációját a QuIDD programmal végeztem, és a megfelelő operátorokat is ott állítottam össze. A futtatás végén a következőt kaptam az eredményre:

- Az eredmény első qubitje (x_3 logikai változó) 51%-ban 0
- Az eredmény második qubitje (x_2 logikai változó) 51%-ban 0
- Az eredmény harmadik qubitje (x_1 logikai változó) 72%-ban 1

Ha egy valódi kvantumszámítógépen futott volna az algoritmus, most a mérés egy bizonyos bitsorozatot adott volna, és azt ellenőrizni kéne, hogy helyes-s. De a szimuláció előnye, hogy a konkrét százalékokat is megkaptuk. Az első két bit értéke igen változó, de a harmadik nagy valószínűséggel 1. Ami jó megoldásnak tűnik, ugyanis x_1 a megoldások $\frac{4}{5}$-ében IGAZ értéket vesz föl, míg x_2 és x_3 csak a megoldások $\frac{3}{5}$-ében tesz így. Ráadásul, ha $x_1 = 1$, akkor az eredmény biztosan igaz.

7. fejezet

Összefoglaló

A dolgozat célja egy általános kvantumalgoritmikus ismertető után a véletlen séta kvantum megfelelőjének vizsgálata volt. Ennek érdekében a használt jelölésrendszer ismertetése után először Grover keresőalgoritmusát mutattam be. Ez $O(\sqrt{n})$ lépésben keres megjelölt elemeket egy rendezetlen halmazban. Ez után ismertettem a klasszikus véletlen bolyongást, és annak kvantum megfelelőjét.

A klasszikus és a kvantum bolyongások ismeretében a kettő közötti különbségeket és hasonlóságokat is megvizsgáltam. Majd a kvantum véletlen bolyongás egy másféle megközelítéséből megvizsgáltam a kvantum Markov-láncokat. Ezt felhasználva egy hatékony keresőalgoritmus készíthető, ami Grover korábban ismertetett algoritmusának megfelelője.

Végül egy ismert NP-teljes probléma megoldására használtam a Markov-láncon alapuló algoritmust. Ez a 3-SAT probléma volt, amin végigkövettem az algoritmus futását. Ehhez vettem egy kis méretű 3-CNF-et, és azon elvégeztem az algoritmus lépéseit. A kis méret miatt követhetőek lettek az állapotok az algoritmus futása közben. Ehhez felhasználtam az ismertetett QuIDD adatszerkezetet, ami a kvantumáramkörök szimulációjához hatékony struktúrát biztosít. Itt Grover algoritmusának lépésszámát is vizsgáltam, majd ennek egy optimalizálását elemeztem. A QuIDD adatszerkezetet felhasználó QuIDDPro programban egy kis példán megvizsgáltam a 3-SAT megoldásának futását.

A téma további vizsgálatának egyik iránya lehetne nagyobb, valós életben előforduló 3-SAT probléma szimulálása, valamint az ismert klasszikus algoritmusokkal való összevetés. Érdekes lehet még megvizsgálni, hogy a QuIDDPro szimulációs programban Grover keresőalgoritmusa vagy a Markov-láncon alapuló megoldás a hatékonyabb.

Irodalomjegyzék

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math*, 2:781–793, 2002.

[2] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications, 1993.

[3] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[4] Roger Eckhardt. Stan Ulam, John von Neumann, and the Monte Carlo Method. *Los Alamos Science*, pages 131–143, 1987.

[5] Richard ER Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, June 1982.

[6] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[7] Mika Hirvensalo. *Quantum Computing*. Natural Computing Series. Springer, 2004.

[8] Julia Kempe. Quantum random walks - an introductory overview, 2003.

[9] Melven R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.

[10] Rónyai Lajos, Ivanyos Gábor, and Szabó Réka. *Algoritmusok*. TypoTEX, 2000.

[11] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via quantum walk, 2011.

[12] Rózsa Pál. *Bevezetés a mátrixelméletbe*. TypoTEX, 2009.

[13] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[14] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, FOCS'99*, pages 410–414, 1999.

[15] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124–134, Washington, DC, USA, 1994. IEEE Computer Society.

[16] Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 32–41, Oct 2004.

[17] George F. Viamontes, Igor L. Markov, and John P. Hayes. QuIDDPro: High-performance quantum circuit simulation. `http://vlsicad.eecs.umich.edu/Quantum/qp/`.

[18] George F. Viamontes, Igor L. Markov, and John P. Hayes. Improving gate-level simulation of quantum circuits, 2003.