DE 15-PUZZEL IN HET KWANTUMREKENEN

Bachelorproject

Simon Knuijver, s1368915, s.p.knuijver@alumnus.rug.nl

Samenvatting: De vraag die in deze these onderzocht wordt is of het zinvol is om van kwantumalgoritmen gebruik te maken bij het bepalen van de oplosbaarheid en het oplossen van de 15 puzzel. Kwantumalgoritmen worden bestudeerd in samenhang met de onderliggende abstracte algebra van de oplosbaarheid en het oplossen van de 15-puzzel. Als resultaat worden de complexiteit van een geïmplementeerde simulatie van een oplossend algoritme en ideeën voor een efficiënt oplossend kwantumalgoritme gegeven.

1. Inleiding

Om met een kwantumalgoritme een exponentiële complexiteitswinst te boeken t.o.v. een klassieke oplossing moet het probleem dat wordt beschouwd, geformuleerd worden als een instantiëring van het probleem van het vinden van een verborgen Abelse subgroep [2][4][21][22] [29, par.5.4.3]. We kunnen dus stellen dat het uitvinden van een efficiënt kwantumalgoritme draait om het begrijpen van wat die vereiste inhoudt. Wetenschappelijk gezien is het ook nuttiger om de algebra en groepentheorie verder te bestuderen om te zien of de kracht van het kwantumrekenen nog voorbij het Abels subgroepprobleem veralgemeniseerd kan worden. Anderzijds is het vertalen van een probleem naar een abelse subgroepprobleem verre van eenvoudig en terdege van nut wanneer dat probleem een klasse van problemen vertegenwoordigd.

In veel KI algoritmen ligt een adaptieve gesloten kern besloten binnen een niet-gesloten systeem. Die kern kan dankzij zijn relatieve geslotenheid gemakkelijk geëxtraheerd worden en computationeel (beter: complexiteits theoretisch) worden onderzocht. In een deel van de adaptieve algoritmen treffen we dan problemen aan die zijn te formaliseren als graafproblemen en het zoeken naar of leren vinden van oplossingen daarvoor. De vraag die derhalve in deze these onderzocht wordt is of het zinvol is om van kwantumalgoritmen gebruik te maken bij het bepalen van de oplosbaarheid en het oplossen van de 15 puzzel. De zinvolheid komt in de eerste plaats tot uitdrukking in de complexiteitstheoretische klasse die de puzzel representeert. De 15 puzzel vormt als zodanig weliswaar slechts een benchmark voor de vergelijking van de complexiteit van klassieke en kwantumalgoritmen, maar met name in de algebra die bij oplosbaarheid wordt gebruikt, wordt de 15 puzzel in de tweede plaats ook juist zelf erg inspirerend. We kunnen de puzzel namelijk gebruiken om het abelse subgroepprobleem beter te begrijpen. Ten derde komen we bij het beschrijven van de kwantumcomputationele variant van klassieke zoekalgoritmen het Grover zoekalgoritme te bestuderen. Grovers algoritme voor het doorzoeken van een ongesorteerde database biedt door gebruik te maken van amplitudebewerking van een kwantumsysteem subexponentiële tijdswinst.

De 15 puzzel

De 15 puzzel is een schuifblokpuzzel op een bord van 4 x 4 vakjes, met stenen op 15 van de vakjes en één leeg.

De standaard formulering is als volgt [30]:

- Toestand: Een toestand specificeert de locatie van elk van de 15 blokjes en het lege vlakje in de 15 velden.
- Begintoestand: iedere toestand kan worden aangewezen als begintoestand.
- Opvolgerfunctie: genereert de toegestane toestanden die resulteren uit het proberen van de vier verschuivingen (leeg gaat naar Links, Rechts, Onder, Boven).
- Eindtoestand: de gewenste toestand die bereikt moet worden door herhaaldelijke toepassing van de opvolgerfunctie op de begintoestand.
- Oplossingstest: test of een toestand overeenkomt met de eindtoestand.
- Padkosten: Elke stap kost 1, dus de padkosten zijn het aantal stappen in het pad.

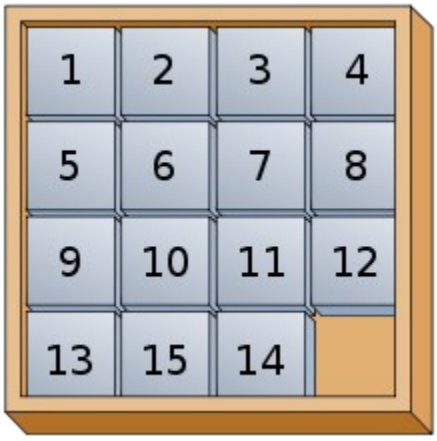
- Bord: het bord is de enumeratie van de 4 x 4 velden.

Naar aanleiding van de discussie hieronder zal deze definitie aangevuld kunnen worden met:

- Permutatie: is het product van transposities, wanneer het product bestaat uit een even aantal transposities, is de permutatie even en anders oneven.
- Transposities is de verwisseling van twee willekeurige blokjes, alsof de blokjes uit het bord worden genomen en verwisseld. De andere blokjes blijven op hun plek.
- Oplossing: een oplossing is een pad dat bestaat uit een reeks transposities, verschuivingen of toestanden of een combinatie van de drie waarmee de begintoestand wordt veranderd in de eindtoestand.

Oplosbaarheidsbewijs

Voor het oplosbaarheidsbewijs wordt meestal de puzzel van Lloyd gebruikt, hierin zijn de 14 en 15 omgewisseld en zit het lege vlak op plek 16 (figuur 1). Het lege vlak legt een pad over het bord af en eindigt daar waar het begon. De route gevolgd kan bestaan uit paden gevolgd en teruggelopen en uit gesloten paden die cyclische permutaties moeten zijn van een oneven aantal stenen. Geen andere beweging is mogelijk [8][17] [18].



Figuur 1. Lloyds onoplosbare probleem.

Door de blokken op te tillen en te wisselen als in: die twee wisselen, of: 1 op de plek van twee, twee op de plek van drie en drie op de plek van 1, krijgen we een cykel, de pariteit van de lengten van deze cykels zijn bepalend voor de oplossende permutatie en daarmee voor de oplosbaarheid van de puzzel als geheel. De som van de pariteiten van de cykels in de oplossende permutatie is even dan wel oneven, daarmee staat het vervolgens vast dat omgekeerd, oneven permutaties niet tot een oplossing kunnen leiden. Het omgekeerde geldt overigens voor problemen met een oneven oplossende permutatie: deze kunnen niet middels even permutaties opgelost worden. Het gestelde probleem kent dus twee klassen, een even en een oneven klasse (subgroepen) en daarmee zijn er onoplosbare problemen te stellen. In het bijzonder heeft Lloyd's 14-15 puzzel geen oplossing (omdat deze correspondeert met de transpositie $\sigma=(14,15)\notin A15$, zie hoofdstuk 2 voor de algebra.)

Had je de puzzel op mogen lossen door de blokjes eruit te tillen en te verwisselen dan zou de verzameling cykels bijna een oplossend algoritme hebben gegeven.

Klassieke oplossende algoritmen

In principe biedt de aaneensluiting van getransponeerde toestanden een leidraad voor het zoeken. De fysieke constraints van de puzzel geven als het ware een index voor mogelijke opvolgende toestanden. Uit alle mogelijke transposities tussen alle permutaties van de puzzel kan door de aaneensluiting een selectie gevormd worden, die zich dan uit als graaf.

Om er zeker van te zijn dat een kortste pad gevonden is, is het noodzakelijk om de probleemruimte, nu gerepresenteerd als een graaf G = {V,E} volledig te doorzoeken. Veelal wordt tijdens het zoeken de inverse van transponeren verwijderd en een gerichte graaf gebruikt. Ervoor zorgen dat twee puzzeltoestanden niet twee keer uitgebreid worden verschilt vervolgens van algoritme tot algoritme. Verder zijn de algoritmen makkelijker te begrijpen in pseudocode. De klassieke algoritmen zijn onder te verdelen in seriële en parallelle varianten. Daarbinnen kan er wel of niet gebruik worden gemaakt van een heuristiek.

BFS [30]

1. neem de begintoestand als eerste knoop x
2. voeg x toe aan een lijst
3. haal de laatst toegevoegde knoop uit de lijst
4. voor elke mogelijke transpositie;

- voeg de resulterende toestand toe aan de lijst
– zorg ervoor dat je niet rond gaat cirkelen!

Parallelle BFS [5]
Het dataparallellisme van BFS dat ontstaat doordat een hele laag knopen tegelijkertijd kan worden uitgebreid (Map) is door Reinefeld et al. [5] gebruikt voor een parallelle implementatie van BFS. De intersecties in de verzameling toestanden die voortkomen uit uitbreiding wordt eveneens parallel opgelost (Reduce) in de vorm van MapReduce (appendix 2). Op die manier hebben zij al eerste de hele zoekboom en alle oplossingen van de 15-puzzel gegenereerd en opgeslagen voor verder onderzoek.

DFS [30] breidt iedere onuitgebreide knoop uit zodra die wordt tegengekomen
DFS:
1. Plaats de root knoop op een stapel samen met de kant 0
2. haal een knoop van de stapel en vergelijk met de eindtoestanden
3. als het de eindtoestand is, geef dan de vertices terug
4. anders, voer de successorfunctie uit en plaats alle (onbezochte) kanten en knopen (toestanden) op de stapel
5. Als de stapel leeg is is de hele zoekboom uitgebreid en geef de waarde 0 terug.
6. Als de stapel niet leeg is herhaal vanaf 2

IDFS [30]: Voer DFS herhaaldelijk uit tot diepte d, bij iedere iteratie neem je d = d+1

Parallelle DFS [28] DFS kent een parallelle implementatie analoog aan MapReduce, waarbij er een intersectietabel (transpositietabel) wordt bijgehouden terwijl diverse processoren depth first zoeken.

A* [30] kan worden afgeleid uit BFS door het toevoegen van een heuristische kostenschatting tot de eindtoestand h(n), zodat f(n) = g(n) + h(n) en in plaats van in de diepte uit breiden, alle knopen uit te breiden die lage kosten hebben.
admissible → nonoverestimating want moet tegengesteld zijn aan de + in f = g+h

IDA* [17] is een combinatie van het blinde iterative depth first zoeken met A*.

De groep van transposities
Het zou echter ook volstaan om slechts de elementen van de oplossing te leren kennen (de kanten) of als dat zou kunnen een (groot) deel van de graaf sneller te doorzoeken. De volgorde van de kanten die deel uitmaken van de oplossing draagt weinig bij aan de complexiteit zodra de ongeordende verzameling kanten waaruit de oplossing bestaat bekend is. Zouden we deze groep kunnen definiëren in lineaire tijd en de gezochte symbolen eruit kunnen selecteren, dan is de ordening van de geselecteerde transposities niet van invloed op de asymptotische complexiteit.

complexiteit
De complexiteit van de n-puzzel heeft een aantal aspecten. Reinefeld et al. [5] geven precieze aantallen voor Lloyd's 14-15 puzzel. In het algemeen zijn er n!/2 permutaties die een eindtoestand kunnen vormen die via een oplossing bereikbaar is voor een begintoestand (zie hierna). Tussen elk van die n!/2 posities is er een pad oplossing van diepte d transposities te vormen. Dat zijn dus n!/2 x n!/2 problemen en dus (2(n-1!))^2 oplossingen van lengte d. Gezien de successorfunctie bestaat er geen transpositie tussen ieder paar transposities, eerder worden de n!/2 permutaties gegenereerd door de 3-cykels. Iedere bordtoestand komt voort uit ruwweg 4 transposities die inverteerbaar zijn en bovendien is de puzzel natuurlijk spiegel- en draaisymmetrisch. De complexiteit voor een oplossend algoritme komt voornamelijk voort uit het feit dat er maar een beperkt aantal transposities per toestand is en er dus een sequentie van toestand-aansluitende transposities gevonden moet worden voor de oplossing.
Zoals boven opgemerkt draagt de precieze volgorde van de transposities in de oplossing niet asymptotisch bij aan de complexiteit als die verzameling transposities bekend zou zijn.
Meestal wordt de volledige zoekruimte beschreven als die bereikbare helft van de permutaties [17],[5]. Het is echter zo dat de oplossingen niet gekend zijn wanneer alleen de transposities {R,L,O,B} in de oplossing bewaard worden. Daarvoor zouden ook de door de verschuiving bewerkte toestanden bewaard moeten worden. Het selecteren daarvan gebeurt

constructief vanuit de begintoestand door de klassieke zoekalgoritmen. Die algoritmen selecteren die transposities uit de verzameling van n!*n!/2 transposities of n!*n!/4 zonder inverse dankzij de indexatie op linker deelnemende toestand van de transposities in tijd O(l).
All-pairs shortest paths oplossingen die ook alle transposities weergeeft, bovenop de beperkte gebruikelijke weergave van enkel de lengte van de kortste paden zijn klassiek van O(V^3) [31].
Rheinefeld et al. hadden in 2009 voor het opslaan van de volledige probleemruimte gegenereerd vanuit één begintoestand 80 terabyte nodig [5]. Zij gebruiken een hashing om uit de tabel de bijbehorende oplossingen in O(d) terug te zoeken.
De nxn puzzel is intractible. Het bewijs dat Ratner en Warmuth [22] daarvoor geven bestaat uit een afbeelding van de puzzel op een 4/4/2 vervulbaarheidsprobleem waarvan bekend is dat het NP-compleet is.
Oplosbaarheid is van orde O(n^2) [32]. Oplosbaarheid voor arbitrair gevormde objecten (in de puzzel zijn ze vierkant) is al PSPACE compleet.
Let op dat het ook zou volstaan om een groot deel van de zoekboom te kunnen doorzoeken in polynomiale tijd om een efficiënt algoritme te verkrijgen.

Deze weergave van de complexiteit leidt tot de onderzoeksvraag en methode. Nu is het aan de ene kant interessant om dieper in te gaan op de algebraïsche theorie waarmee dit probleem en allerlei variaties geformaliseerd worden, in het bijzonder is de link met de klasse van eerder aangekaarte kwantumalgoritmen voor dit werk interessant. Anderzijds zullen we ook willen gaan kijken naar het probleem zoals dat waarvoor de blokjes niet uit de puzzel genomen mogen worden maar het daadwerkelijk verschuiven nodig is: wat is dan het kortste pad naar een oplossing? Door deze twee aspecten in het hierna volgende achterelkaar te behandelen, worden de bekendste kwantumalgoritmen belicht en meteen in verband gebracht met één van de belangrijke gebieden van de KI, zoekalgoritmen (en Machine Learning). Behalve dat de klassieke algoritmen een kwantumversie krijgen, worden ze ook gebruikt voor het verifiëren van kwantumoplossingen.

2. methode: quantumcircuits, verborgen subgroepen & Grover-iteraties

We bespreken hier kort het kwantumcircuit model van Deutsch [6]. Het voordeel van dit model is dat het zowel de complexiteit inzichtelijk maakt als een weergave biedt voor de evolutie van een kwantummechanisch systeem door discrete toestanden en daarmee de gebruikelijke Hamiltonians en Heisenberg vergelijkingen van de experimentele natuurkunde vervangt en toegankelijker maakt. Het model is analoog aan het model voor klassieke elektrische schakelingen. In het kwantumrekenen bestaan er oneindig veel verschillende poorten. Dit zijn in dit geval unitaire operatoren. Zoals te lezen is in Nielsen en Chuang [29] zijn die operatoren equivalent met matrices uit de lineaire algebra. We kunnen dus zeer formeel kijken naar natuurkundige experimenten en de bewerkingen uit de experimenten gebruiken om m.b.v lineaire algebra hypothetische experimenten te plannen, de kwantumcircuits. Vervolgens kan er door het samenbrengen van een paar (klassieke en kwantum) circuits een algoritme gevormd worden.

In het kwantumrekenen speelt namelijk enerzijds superpositie een belangrijke rol in wat we klassiek ruimte complexiteit noemen. Een kwantumbit (qubit) register kan alle toestanden die een klassiek geheugenregister kan bevatten ook bevatten maar tegelijkertijd. Het is algemeen bekend dat bijvoorbeeld de positie en het momentum van een deeltje gerepresenteerd moeten worden door een complex getal

$$\alpha|0\rangle + \beta|1\rangle \qquad (1)$$

voorafgaand aan de observatie van één van beide grootheden. Deze toestand, die superpositie genoemd wordt kan als geheugen register gebruikt worden []. Daarbij zij meteen opgemerkt dat een meting of observatie van het register ervoor zorgt dat het register samenvouwt tot één toestand uit de computationele basis (|0> óf |1>): bij meting interfereren de basistoestanden in (1) en vouwen die zich volgens de gekwadrateerde alpha en

bèta kansen samen tot een discrete (vandaar kwantum) toestand [29]. Een kwantum algoritme moet zich dus niet vergissen in het parallellisme en alleen globale eigenschappen van een functie proberen te bepalen zoals of de functie even of oneven is.

De algebra van de n-puzzel

In de wiskunde is de (symmetrische) groep S_n de groep van permutaties voor n symbolen waarin de groepsoperator compositie is [1]. Omdat er n! mogelijke permutaties zijn voor n symbolen, is de orde van de groep (het aantal elementen) n!. De permutaties worden als bijectieve functies van de verzameling symbolen naar zichzelf gezien. De 15puzzel is echter ten eerste een groepoïde want er is slechts een gedeeltelijk gedefinieerde compositie [26]. Ten tweede zijn de permutaties die bereikbaar zijn vanuit een gegeven beginpositie die van de alternerende groep A_15.

De eerste eigenschap is vooral van belang voor het bepalen van een oplossing. Althans, elke permutatie kan geschreven worden als een product van transposities. De transpositie kan in cykelnotatie geschreven worden als (ab) waarbij deze operator a en b omwisselt en de rest van de symbolen in de permutatie (bordtoestand) onberoerd laat. De groepoïde beschrijft de reeksen transposities.

De tweede eigenschap is vooral belangrijk voor het bepalen van oplosbaarheid. Het produkt van transposities dat nodig is om een permutatie te beschrijven is niet uniek maar het aantal ervan is wel altijd even of oneven. In het bewijzen van de oplosbaarheid van de n-puzzel wordt er een afbeelding van de puzzel gemaakt op de symmetrische groep. Dan wordt er in het bewijs gebruik gemaakt van en benadrukt dat er voor elke begintoestand de helft van de mogelijke permutaties uit de groep niet bereikbaar is omdat die een oneven aantal transposities behoeven terwijl er alleen even transposities beschikbaar zijn. De puzzel vormt daarom een abelse subgroep [25].

$$G\_\sigma\tau \rightarrow S\_16 \rightarrow A\_15 \qquad (2)$$

G_στ :

Een groepoïde is een veralgemenisering van de groep bestaande uit een verzameling G met een partieel gedefinieerde binaire functie (g,h) → g h en een overal gedefinieerde inverse en voldoet aan associativiteit: a*(b*c)=(a*b)*c als de termen a, b en c zijn gedefinieerd. Verder is, als a*b is gedefinieerd, a−1*a*b =b en a*b*b−1=a en tenslotte zijn alle a−1*a en a*a−1 gedefineerd (maar dit kunnen verschillende elementen zijn).

Dus:

1. (a*b) *c=a* (b*c);
2. ∀a ∃a−1*a∃a*a−1;
3. a*b*b−1=a en a−1*a*b=b.

Schrijven we de bordtoestand $\tau \in S\_16$ als τ. De transposities verwisselen het lege veld (label 16) met een blokje i (als i een horizontale of verticale aanligger is van het lege vlak) en worden geschreven als (16,i). Itereren we transposities, dan is een toegestaan pad van een toestand τ een reeks transposities geschreven van rechts naar links in cykelnotatie: (16,ik)···(16,i2)(16,i1).

De regel voor berekening van de positie die we verkrijgen uit een gegeven toestand door toepassing van een reeks transposities is dus de volgende. Vermenigvuldig de reeks met de toestand in de groep S_16 om de code voor de resulterende toestand te verkrijgen [27]. In symbolen (bv.): (16,2)(16,6)(16,7)(16,11)(16,12)() = (16,2)(16,6)(16,7)(16,11)(16,12) = (16,12,11,7,6,2), als transpositiereeks op (), de identiteit die de toestand niet verandert.

We nemen als onze elementen alle mogelijke combinaties στ waar τ de code is van een bereikbare positie en σ = (16,il)···(16,i1) een toegestane reeks transposities is vanuit die toestand.

Op deze set elementen στ hebben we maar een gedeeltelijk gedefinieerde compositieregel want we kunnen alleen compositie van transposities σ1τ1*σ2τ2=σ1σ2τ2 interpreteren, gegeven dat τ1 de code is voor de toestand die wordt bereikt vanuit τ2 na toepassing van de reeks transposities σ2, m.a.w. de vermenigvuldiging is gedefinieerd iff τ1=σ2τ2 in S16.

Elk element heeft verder ook een inverse (σ τ)^{−1} = σ^{−1}ω waar ω=σ τ in S_16. Aan alle voorwaarden voor een groepoïde wordt dus voldaan [33].

Het zal blijken de vraag te zijn of deze groepoïde een verborgen subgroep(oïde) heeft bij het

onderzoeken van een oplossend algoritme voor de 15-puzzel. Eventueel kijken we daar naar de verzameling van álle transposities ∑ om te weten te komen of er daar een patroon in zit dat overeenkomt met de oplossing voor een gegeven probleem.
Mogelijk is er ook een afbeelding van de bovengenoemde groepoïde op een torus die wel gewoon een groep vormt (appendix 1). De groep van alle permutaties van de 16 elementen van de 15-puzzel vormen de groep S_{16}.

S_{16}
Kijken we nu enkel naar alle n! permutaties ofwel toestanden. Anders dan transposities én permutaties kent de groep maar één object, in dit geval dus de permutaties (toestanden).
Groep: een geordend paar (S, *) waar S is een verzameling is en * een binaire bewerking op S. Voldoet aan:
Closure: a, b ∈ S ⇒ (a * b) ∈ S
Associativiteit: a, b, c ∈ S ⇒ (a* b)* c = a * (b* c)
Identiteit: ∃ e ∈ S s.t. ∀ a ∈ S a * e = e*a = a
Inverse: ∀ a ∈ S ∃ a-1∈ S s.t. a*a-1 = a-1*a = e

Om van een Abelse groep te kunnen spreken moet er verder aan de volgende voorwaarde zijn voldaan:
Commutativiteit : ∀ a, b ∈ S a* b = b * a)

Subgroep: (H, *) is een subgroep van (S, *) iff H is een groep tov * en H ⊆ S

A_{15} is een Abelse subgroep: Een subgroep is abels als deze als groep abels is. De alternerende groep op een eindige verzameling M is de groep van even permutaties op de verzameling M aangegeven met A_M. Als M ={1...n}, dan wordt deze groep aangegeven met A_n. Dit is een normale subgroep (invariant onder conjugatie door leden van de groep) van de symmetrische groep en voor n=15 is het een eenvoudige (alleen triviale normale subgroepen) groep. [34]

Voorts komt er verderop in dit paper de volgende algebra voor [32]:
Orde van een groep |G| = Het aantal elementen in the groep.
Orde van een groepselement |g| = Het kleinste aantal keer dat de binaire bewerking wordt toegepast op g voordat het identiteitselement e wordt verkregen |g| = k als gk = e
Modulo rekenen: x = y mod n als y de rest is van x na deling door n, als n*f = x dan is y de rest van x*n^-1
Een Coset is een (kopie van een) subgroep vermenigvuldigd met een element van de oudergroep.
Generator: Een verzameling T ⊆ S genereert de groep (S,*) als elk element in S kan worden gegenereerd door een eindig product van de elementen in T. Als T een enkel element is, wordt het een generator van de groep genoemd.

Verborgen subgroepprobleem: Zij f een functie van een eindig gegenereerde groep G naar een eindige verzameling X zodat f constant is op de cosets van een subgroep K, en verschillend op elke coset. Gegeven een quantum black box [] voor het uitvoeren van de unitaire transformatie U|g|h = |g|h⊕f(g), voor g ∈ G, h ∈ X, en ⊕ een gekozen geschikte binaire bewerking op X: vindt een genererende verzameling voor K [N&C,pp 240].
Moore en Narayanan [15] hebben geprobeerd de oplosbaarheid als verborgen subgroepprobleem te formuleren. Wij zullen zoiets proberen voor de oplossingen.
Het is vaak nuttig om data in kwantumtoestanden voor te stellen in een wat abstractere vorm. Bijvoorbeeld, gegeven een groep G, zouden we |g> als basistoestand kunnen nemen, corresponderend met het groepselement g ∈ G, en

$$|\phi\rangle = \sum_{\{g\in G\}} b_g |g\rangle \qquad (3)$$

als een willekeurige superpositie over de groep. Aangenomen dat de elementen in de groep te representeren zijn als bitreeksen is het vaak niet nodig de codering expliciet te vermelden. |.> is een ket, een kolomvector. Als een kwantumregister de toestanden |psi> en |phi> bevat, dan is de toestand van het register als geheel het tensorproduct van de twee toestanden:

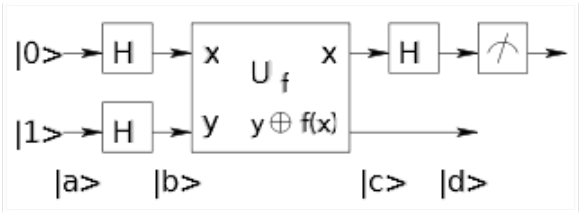
$$|\psi\rangle \otimes |\phi\rangle = |\psi\rangle|\phi\rangle = |\psi, \phi\rangle. \qquad (4)$$

De toegestane operatoren op de computationele (basis)toestanden maken een afbeelding van genormaliseerde naar genormaliseerde toestanden, dus unitaire operatoren U, als in

UU† = U†U = 1, U^{−1} = U†. Waarin het zwaard het Hermitische transponeren is [24]. We behandelen het kwantumrekenen verder in hoofdstuk twee, alwaar we ook dieper ingaan op de unitaire operatoren als poorten in kwantumcircuits.

Circuit van Deutsch

We gebruiken de notatie [α β]' voor de toestand $\alpha|0\rangle + \beta|1\rangle$ waarbij de som van de genormaliseerde gekwadrateerde coëfficiënten optellen tot één en de observatiekansen van de computationele basistoestanden in de kets aangeven. In principe kan een kwantumalgoritme gerepresenteerd worden in een circuitmodel waarin kwantumpoorten, die zijn gedefinieerd door unitaire (normaliserende) matrices, de ingevoerde basistoestanden bewerken [13]. De Hadamard poort wordt bijvoorbeeld gedefinieerd als H = 1/sqrt(2)[[11], [1-1]], toegepast op (1) geeft deze poort H[α β]' = H(α0>+β|1>)=|α/√(2)(|0>+|1>)+β/√(2)(|0>-|1>).
De superpositie kan bewerkt worden om een globale eigenschap van een functie te bepalen. Neem bijvoorbeeld het volgende circuit:



Figuur 2. Het Deutsch circuit, illustreert quantum circuits.

Deutsch toonde hiermee aan dat bepaald kan worden of een functie even of oneven is [29]:
De invoer

$$|a\rangle = |01\rangle \qquad (5)$$

wordt door twee Hadamard poorten gestuurd:

$$|b\rangle = [(|0\rangle + |1\rangle)/\sqrt{(2)}]\ [(|0\rangle - |1\rangle)/\sqrt{(2)}] \qquad (6)$$

2. Toepassen van U_f op |ψ1> geeft:

$$|c\rangle = |+/-[(|0\rangle + |1\rangle)/\sqrt{(2)}]\ [(|0\rangle - |1\rangle)/\ \mathrm{sqrt}(2)]$$
if f(0) = f(1)
$$|+/-[(|0\rangle - |1\rangle)/\sqrt{(2)}]\ [(|0\rangle - |1\rangle)/\ \mathrm{sqrt}(2)]$$ if f(0) != f(1). (7)

3. De laatste Hadamard poort op de eerste qubit geeft:

$$|d\rangle = |+/-|0\rangle\ [(|0\rangle - |1\rangle)/\sqrt{(2)}]$$ if f(0) = f(1)
$$|+/-|1\rangle\ [(|0\rangle - |1\rangle)/\sqrt{(2)}]$$ if f(0) != f(1). (8)

Aangezien f(0) ⊕ f(1) (= f(0) plus f(1) modulo2) gelijk 0 is als f(0) = f(1) en anders gelijk 1, kunnen we dit resultaat schrijven als:

$$|\psi\rangle = +/-|f(0) \oplus f(1)\rangle|1\rangle\ [(|0\rangle - |1\rangle)/\sqrt{(2)}] \qquad (9)$$

Dus door meting van het eerste qubit kunnen we f(0) ⊕ f(1) bepalen. De globale eigenschap van f(x), f(0) ⊕ f(1) is duidelijk gemaakt met één evaluatie van f(X). Een klassieke computer zou hier minstens twee evaluaties voor nodig hebben.

Quantum Fourier transformatie

De belangrijkste unitaire transformatie voor het kwantumrekenen is tot op heden de discrete kwantum Fourier transformatie (QFT) [2],[29]. De QFT over een abelse groep G is:

$$F_G := 1/|\sqrt{\{G\}}| \sum_{\{x \in G\}} \sum_{\{y \in \hat{G}\}} \chi_{\{y\}}(x)\ |y\rangle\langle x| \qquad (10)$$

waarin $\hat{G}$ een volledige verzameling symbolen (comp. basistoestanden) is van G, en $\chi_{\{y\}}(x)$ het y de element van G geëvalueerd in x aangeeft. De meest eenvoudige QFT over een familie van groepen is de QFT over $\mathbb{Z}^n_2$:

$$F_{\mathbb{Z}n2} = 1\sqrt{2^n} \sum_{x,y \in \mathbb{Z}n2} (-1)^{x \cdot y} |y\rangle\langle x| = H^{\otimes n}. \qquad (11)$$

Deze Fourier transformatie wordt in de oplossing van Simon's probleem gebruikt [29].)

Abels subgroepprobleem

In het verborgen subgroepprobleem hebben we een black box functie f : G → S, waar de groep G bekend is en S een eindige verzameling. De functie belooft f(x) = f(y) te vervullen desda x −1 y ∈ H i.e., y = xh voor een h ∈ H voor een onbekende subgroep H ≤ G. We zeggen dat zo'n functie H verbergt. Het doel van het verborgen subgroepprobleem is H te leren kennen (zeg, in termen van een genererende verzameling) gebruik makend van queries aan f [29].

Het is duidelijk dat H in principe gereconstrueerd kan worden als we de gehele waarheidstabel van f kennen.

Shor

Het algoritme van Shor [2] is eigenlijk een instantiering van het verborgen subgroepprobleem [20]. In dit algoritme zien we echter een mooi voorbeeld van een afbeelding van een probleem op een periodieke groep van de natuurlijke getallen. Omdat dit vooralsnog de enige manier is om efficiënte kwantumalgoritmen te ontwerpen, is het zinvol dit te bestuderen en na te denken over een afbeelding of analogie van het puzzelprobleem op het factoriserings- of discrete logaritme probleem.

Met Shor's algoritme kan je de periode van een functie vinden, een getal dat het produkt van twee priemgetallen gedeeld door twee is. Het probleem is als volgt:

Gegeven een periodieke reeks integers, b.v. de machten van twee 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, zijn modulo 15: 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, een periodieke reeks met periode 4. Euler's totient functie is een regel die de perioden voorspelt. Als N een product is van twee priemgetallen, p en q, en gegeven de reeks x mod N, x2mod N, x3mod N, x4mod N.

Dan, mits x is deelbaar is door p of q, zal de voorgestelde reeks zich herhalen met een periode die (p-1)(q-1) in tweeën deelt. Dus als we de periode van de reeks x mod N, x2mod N, x3mod N, x4mod N, kunnen vinden dan komen we wat te weten over de priemfactoren van N. We komen namelijk een deler van (p-1)(q-1) te weten komen.

Maar zelfs al zal x mod N, x2mod N, x3mod N, x4mod N, zich uiteindelijk herhalen, dan nog is het aantal stappen voordat dat gebeurt mogelijk erg groot. Dat is de reden dat dit factoriseren klassiek een moeilijk probleem is.

Het probleem komt dus neer op het beantwoorden van twee vragen:

1. Kunnen we m.b.v. een kwantum computer, efficiënt een superpositie creëren over x mod N, x2mod N, x3mod N, ..? → Dat kan m.b.v. herhaaldelijk kwadrateren.
2. Gesteld dat we die superpositie hebben gecreeerd, hoe vinden we dan de periode?

Het algoritme van Shor voor het factoriseren van een getal ziet er als volgt uit [29]:

1. Kies een random getal a < N
2. Bereken de grootste gemene deler van a en N.
3. Als de ggd(a, N) ≠ 1, dan is er een niet-triviale factor van N, return.
4. Anders, gebruik de periode-vindende routine hieronder, om r, de periode van de volgende functie te berekenen: $f(x) = a^x \bmod N$. dus de orde r van a in Z_n^x , dat de kleinste positive integer is waarvoor f(x + r) = f(x).
5. Als r oneven is, ga terug naar stap 1.
6. Als a^r/2 ≡ -1 (modN), ga terug naar stap 1.
7. ggd(ar/2 ± 1, N) is een niet-triviale factor van N. Het resultaat.

De periode van een functie vinden is voor klassieke computers een moeilijk probleem. Een kwantum computer kan dit efficiënt oplossen door een reeks spectrale argumenten voor de Fourier transformatie in superpositie te brengen en m.b.v. entanglement en interferentie de gewenste uitkomst met grote zekerheid te doen resulteren. Op zich is het niet heel zinvol om de pseudocode met formules hier te reproduceren zonder nader in te gaan op de operatoren maar ook die vergen nadere inspectie en studie.

Periode-vindende kwantum subroutine [29]

De kwantum circuits gebruikt voor dit algoritme worden op maat ontworpen voor elke keuze van N en de random a gebruikt in de f (x) = ax mod N. Het hier weergegeven algoritme is ontleend aan Nielsen en Chuang [29]*. Gegeven N, vind Q = 2^q zodanig dat $N^{\wedge 2}$ <= Q <= $2N^{\wedge 2}$, wat impliceert dat Q / r > N. De invoer en uitvoer qubit registers moeten superposities bevatten van waarden van 0 tot Q-1, en dus elk uit q qubits bestaan. Gebruik van twee keer zoveel qubits garandeert dat er op zijn minst N verschillende x zijn die dezelfde f(x) te produceren, zelfs als de periode r, N/2 benadert.

Invoer:

(1) Een *black box* met een *controlled*-Uj bewerking, voor geheel getal j,

(2) een eigentoestand |u> van U met eigenwaarde $e^{2\pi i\varphi\_u}$, en

(3) $t = n + [\log(2 + 1/2\varepsilon)$ qubits geïnitialiseerd als $|0\rangle$.
Uitvoer: een n-bit benadering φ_u van φ_u.
Runtime: $O(t^2)$ bewerkingen en één aanrope van de controlled-Uj black box. Succeskans $> 1 - \varepsilon$.

Procedure:

1. $|0\rangle|u\rangle$ initial state
2. $\rightarrow 1/\sqrt{(2^t)}\sum_{j=0}^{2t-1}|j\rangle|u\rangle$ create superposition
3. $\rightarrow 1/\sqrt{2^t}\sum_{j=0}^{2t-1}|j\rangle U^j|u\rangle$ apply black box
   $= 1/\sqrt{2^t}\sum_{j=0}^{2t-1} e^{2\pi ij\varphi_u}|j\rangle|u\rangle$ result of black box
3. $\rightarrow|\varphi_u\rangle|u\rangle$ inverse Fourier transform
4. $\rightarrow \varphi_u$ measure first register

Algoritme: kwantum fase schatting.*

*** Nielsen en Chuang pp.225, formulering in termen van G zoals () wordt verwacht.**

Oplossingen
Kijken we nu opnieuw naar de puzzel. Niet elke transpositie is een geldig pad, maar in de transposities van S_{16} is het gemakkelijk je voor te stellen dat er geldige paden zitten in de n!(n!-1) transposities en dat die geldige paden ongeveer om de b/n!-1 transposities optreden. Nu is het zaak die geldige paden te vinden in subexponentiele tijd- en ruimtecomplexiteit. De aaneensluitingsconstraint willen we dus vertalen naar periodiciteit. Het probleem is echter wel dat het moeilijk is om van tevoren een ordening in de transposities van S_{16} aan te brengen zodat die periodiciteit ook naar voren zal komen uit de op een Fouriertransformatie gebaseerde kwantum periode vindende subroutine.
Een andere mogelijkheid zou zijn alle permutaties van alle transposities voor S_{16} voor alle padlengten aan te bieden aan een orakel dat aangeeft of het een pad is dat bestaat uit toelaatbare c.q. aaneensluitende transposities bestaat. Gegeven dat elke geldige transpositie dan maar hooguit één keer in zo'n pad voorkomt, zou je kunnen beginnen bij zeg de paden van lengte 50 en eindigen bij paden van lengte 100 (als die er nog zijn) en zo die paden die klassiek erg veel tijd zouden kosten om op te sommen, indexeren en op te slaan. Dan zou het circuit wel efficiënt moeten zijn.

Grover
Er zijn ook al wel kwantumvarianten van een aantal graafalgoritmen beschreven die niet exponentieel efficiënter zijn dan hun klassieke evenknie. Deze algoritmen zijn voornamelijk gebaseerd op het algoritme van Grover [9][10]. Ik zal hier het algoritme van Grover en de het kwantum DFS algoritme beschrijven en daarna een poging doen zelf ook kwantumvarianten van IDDFS te geven of te onderzoeken.
We hebben een unitaire operator, $U\omega$, die op een subroutine of quantum black box werkt en database records vergelijkt volgens een bepaald zoekcriterium. Het algoritme specificeert niet hoe deze subroutine werkt, maar het moet een kwantum subroutine zijn die werkt met superposities van toestanden. Verder moet het in het bijzonder werken op een van de eigentoestanden, die corresponderen met het database record dat overeenkomt met het zoekciterium. Ons doel is deze eigentoestand aan te wijzen of gelijkwaardig, de eigenwaarde ω, waar $U\omega$ speciaal op werkt. De stappen in Grover's algoritme zijn als volgt gegeven [29]. Laat er een uniforme verdeling zijn over alle toestanden $|s\rangle = 1/\sqrt{N}\sum_{x=1}^{N}|x\rangle$. Dan wordt de operator $U_s = 2|s\rangle\langle s| - I$ de Grover diffusie operator genoemd.
Hier is het algoritme:

1. Initialiseer het systeem in de toestand $|s\rangle = 1/\sqrt{N}\sum_{x=1}^{N}|x\rangle$
2. Voer de volgende "Grover iteratie" r(N) keer uit. (De functie r(N), gaat asymptotisch naar $O(N^{1/2})$).
1. Pas de operator $U\omega$ toe.
2. Pas de operator Us toe.
3. Doe de observatie Ω. Het meetresultaat zal $\lambda\omega$ zijn met kans $\rightarrow 1$ voor $N \gg 1$. Uit $\lambda\omega$, kan ω worden verkregen.

Algoritme: Grover's zoekalgoritme [9].

Een klassieke simulatie van de Grover iteratie wordt gegeven in hoofdstuk 3. Dit algoritme is bedoeld voor het doorzoeken van de graaf $G = \{V,E\}$.

Quantum (versie van klassieke) algoritmen
Depth-first search: in sectie 3.2's stap 2b, roepen we findsol aan. Dit is een subroutine die is

ontleend aan een artikel van Furrow [34]. Furrow baseert deze subroutine op twee implementaties van Grover's zoekalgoritme. Voor de volledigheid geef ik hier de subroutine (uit het Engels) en de twee gerefereerde implementaties.

Findsol [34]
Het hier gebruikte principe is het volgende. Eerst wordt het snellere BBHT benut, voor het geval er wel oplossingen zijn. Dan, als we geen oplossing vinden, wordt er gekeken naar een oplossing met BCWZ om zeker te zijn. De asymptotische complexiteit wordt O(p N/M) aanroepen voor het vinden van een x-waarde zodat F(x) = 1.
1. Draai BBHT tot r keer. Indien een van de runs een resultaat geeft dat voldoet aan F, geef dat resultaat onmiddellijk terug.
2. Draai BCWZ met parameter ϱ-1. Als het een resultaat geeft dat aan F voldoet, retourneer dat resultaat, anders return false.

BBHT [36]
Initialize m = 1 and set $\lambda = 8/7$.
2. While $m \leq 2\sqrt{N}$, repeat the following unless instructed to return:
(a) Choose an integer j uniformly at random such that $0 \leq j < m$.
(b) Execute Grover's original algorithm, using j Grover iterations. Let the outcome
be called i.
(c) If F(i) = 1, return i; otherwise, set m to λm.
3. Return *false*..

BCWZ [35]
BCWZ: The BCWZ search algorithm is passed a parameter ϱ−1 and returns a random solution to F after O(pN lg ϱ−1) calls, provided that such a solution exists. There is a probability ϱ that it will fail, in which case it returns false. If M = 0, it returns false in O(pN log ϱ−1) calls to F.

kwantumcomplexiteit
We zeggen dat een algoritme voor een probleem efficiënt is als het is te beschrijven als een circuit dat een aantal poorten bevat dat is polynoom in het aantal bits die nodig zijn voor het bevatten van de input. Bijvoorbeeld, als de input een getal modulo N is, is de invoergrootte log2 N.
Met een quantum computer kan het zijn dat het uiteindelijke resultaat van een berekening niet met zekerheid correct is. In plaats daarvan, zijn we meestal tevreden met een algoritme dat het juiste antwoord kan produceren met een kans die hoog genoeg is (voor een beslis probleem, hoger dan 1/2; voor een niet-beslissing probleem zo dat we een juiste oplossing in tijd Ω (1) kunnen controleren). Door de berekening vele malen te herhalen, kunnen we de kans voor het berekenen van een onjuiste antwoord willekeurig klein maken.
Naast expliciet computationele problemen, waar de invoer een string is, wordt ook gebruik gemaakt van het concept query complexiteit. Hier is de invoer een black box transformatie, en is ons doel om een of andere eigenschap van de transformatie te ontdekken door het maken van zo min mogelijk queries.
Bijvoorbeeld, in het probleem van Simon, krijgen we een transformatie f: Z^{n}_{2} → S dat voldoet aan f (x) = f (y) desda y = x ⊕ t voor een onbekende t ∈ Z^{n}_{2} en is het doel om t te weten te komen. Het belangrijkste voordeel van querycomplexiteit is dat het ons in staat stelt om ondergrenzen te bewijzen voor het aantal benodigde queries voor het oplossen van een bepaald probleem. Bovendien, als we een efficiënt algoritme voor een probleem vinden in query complexiteit, dan hebben we als we een expliciete circuit het realiseren van de black-box transformatie krijgen, een efficiënt algoritme voor een expliciet computationeel probleem. Soms willen we niet alleen de grootte van een circuit voor de uitvoering van een bepaalde unitaire operatie hebben, maar ook over de breedte, het maximum aantal poorten op een pad van een input naar een uitgang. De breedte van een circuit vertelt ons hoe lang het duurt om het te implementeren als we poorten parallel kunnen laten opereren.

3. resultaten: quantumoplosbaarheid & -oplossingen

oplosbaarheid
In het voorgaande heb ik geprobeerd inzicht te krijgen in het verborgen subgroepprobleem. Een vertaling van de oplosbaarheidsbepaling is geprobeerd door Moore en Narayanan [15].
We weten dat de mogelijke bewerkingen op de puzzel de Abelse subgroep A_15 vormen maar:

a) kunnen we daarmee een verborgen subgroepprobleem reconstrueren?
b) is dat voldoende voor oplosbaarheidsbepaling?

Efficiënte oplossingen

De resultaten van het maken van een afbeelding analoog aan Shor's algoritme op het verborgen subgroepprobleem en het eventuele complexiteitstheoretische resultaat daarvan kan ik hier nog niet presenteren. Enige suggesties in die richting zullen moeten volstaan.
Een formulering in termen van een abelse subgroepprobleem lijkt uitgesloten als de periodiciteit onderbroken lijkt te worden door het maar gedeeltelijk gedefinieerd zijn van de compositie. Als dat echter geen probleem is zou je periodiciteit kunnen zoeken in de volledige verzameling {transposities, bijbehorende toestanden}, die niet beperkt wordt door de compositieregel van de groepoïde.
De subgroep is dan natuurlijk de groep die hoort bij de begin x eindtoestand. Het is de vraag of en hoe deze te extraheren is uit de groep. Enerzijds is dat misschien te verifiëren met een volledige graaf van de drie- of acht-puzzel. Anderzijds biedt het bewijs van intractability [22] weinig hoop.
Voor de drie puzzel:
* Voor de 3-puzzel zijn er in totaal 24 transposities in de groepoïde en 24 x 23 in de verzameling van transposities, de eerste kan je zo ordenen dat je elke keer dat je één van de cirkels (1/d+d) keer rondgaat er een transpositie is die bij de oplossing van je probleem hoort. Voor de verzameling transposities is een afbeelding op het subgroepprobleem dat niet onmiddellijk duidelijk is wat de binaire bewerking zou kunnen zijn. Mogelijk gaat het om "aansluiting" maar dat was nou juist de operator van de groepoïde.
Misschien moet je wel naar de verzameling 3-permutaties kijken die de puzzel genereren.

Quantum IDDFS1

Noem de edges factoren, zoals in factorisatie, je moet ze dan zien te selecteren. Een pad is een verzameling transposities (of 3-cykels?).

1. Neem alle mogelijke paden van lengte d

2. Selecteer hieruit de transposities die in de oplossing horen.

Door de ordening van de transposities zijn de te selecteren transposities periodiek.

Of

2. selecteer hieruit de toestanden die in de oplossing gepasseerd worden

Door de ordening van de permutaties zijn de te selecteren toestanden periodiek.

3. Als de begin en eindtoestand geen deel uitmakenvan de gevonden transposities, dan is er geen oplossing van lengte d. Herhaal met lengte d = d+1.

Het ordenen van de gevonden transposities kost O(d) tijd voor de 15 puzzel.

Een groot deel van de graaf efficiënt doorzoeken zou mogelijk zijn als we het grote deel kunnen formuleren in termen van een abelse subgroepprobleem. Gezien het feit dat onregelmatigheden in de puzzel vooral op lijken te treden door de aanwezigheid van de bordranden, zou je bijvoorbeeld meer regelmatigheid en periodiciteit verwachten in een deel van de graaf. Maar ook de specifieke begin en eindtoestanden dragen bij aan onregelmatigheid. Ook enkel de transposities met de bijbehorende configuraties/toestanden vinden zou volstaan als zoekalgoritme want het sorteren van geïndexeerde transposities draagt niet bij aan de asymptotische complexiteit. Mogelijk zijn er hier ook klassieke algoritmen voor te ontwerpen.
Mocht er geen subgroep te vinden zijn dan kunnen we in navolging van de algoritmen gepresenteerd in [34] en [35] proberen een algoritme voor de graaf te ontwerpen dat gebaseerd is op de Grover-iteratie.

Algoritme Quantum IDDFS2 is gebaseerd in navolging van [34] en [35] gebaseerd op de Grover-iteratie:

DFS
Het volgende algoritme DFS voert een *depth-first search* uit door een graaf G=(V,E).

1. Noem de begintoestand va. Laat er een boolean-array v zijn van de grootte V, met entries v[i] = 0.
2. Roep DFS-BODY (va, d) aan. Verhoog d en herhaal.

Functie DFS-BODY (vertex vk, diepte d):
(a) Bezoek vk. Stel vi[k] = true.
(b) Gebruik findsol om een buurman van vk te vinden die nog niet is bezocht, vi.
(c) Is er een dergelijke vi:
i. Recursief aanroepen van DFS-BODY (vi,d-1).
ii. Na terugkeer uit de recursieve aanroep, ga terug naar stap 2b.
3. Return.

Ik heb de grover iteratie die als subroutine dient in dit algoritme klassiek gesimuleerd. De grover iteratie is weergegeven in pseudocode in figuur .

Oplossing plek x = 2

```
%Grover iteratie
neem n = 4
for ii=1:maxit     |x> = 1/2(|1>+|2>+|3>+|4>)
   % oracle toepassen
   x(s)=-x(s);       |x> = 1/2(|1>-|2>+|3>+|4>)
   % QFT toepassen
   y=fft(x);         F|x> = 1/2(|1>+|2>+|3>+|4>)
   % Tekens van alle termen omkeren behalve de
eerste
   y([2:length(y)])=-y([2:length(y)]);
   % inverse QFT toepassen
   x=ifft(y);                              |x> = 2
   % kans op meten van de oplossing bijhouden
   p(ii)=abs(x(s)^2);
end
```

figuur 3. de Grover iteratie

COMPLEXITEIT: *worst case* complexiteiten

	matrix model	**edge list model**	**Klassiek**
BFS:	O(sqrt(V^3 lgV))	O(√V E lg V)	O(b^d)
DFS	O(sqrt(V^3 lgV)	O(√V E lg V)	O(b^d)
APSP	O(√V^5 lg V)	O(√V^3E lg V + V^2 lg3 V)	Dijkstra's algorithm O(V2) O(E + V√log L) Ahuja et al. 1990

4. Conclusies

We hebben in dit paper wat initiële resultaten kunnen presenteren, maar moeten nog verder onderzoek doen alvorens gegronde uitspraken te kunnen doen over de validiteit van de voorstellen. Voor het kwantumrekenen hebben we de normale behandeling van de 15-puzzel moeten heroverwegen. Hoewel de normale behandeling en het vinden van een oplossing met behulp van een kwantum versie van een graaf gebaseerd algoritme enige complexiteits-winst zou kunnen bieden, bestuderen we liever *iterative depth first* de dan oneindig grote verzameling paden of de toroïdale afbeelding van de puzzel. De reden daarvoor is de mogelijk exponentiële winst aan efficiëntie voor een kwantumalgoritme als deze als verborgen subgroepprobleem geformuleerd kan worden. Zo'n formulering zou echter nog niet onbedingd tot succes leiden [3] en is ook nog niet tot stand gekomen in deze these. Het is niet eenvoudig een zinnige compositieregel te verzinnen die met de niet-aaneensluitende transposities toch een groep vormt. Het is misschien een afweging tussen domweg een niet fysiek interpreteerbare functie en het verlaten van de constraint dat die regel er moet zijn of, wanneer we paden als object van de groep nemen, een redelijke functie op paden die de aaneensluiting vervangt door periodiciteit. Tenslotte is de natuurlijke volgorde door de aaneensluiting van transposities die de puzzel tot een graaf vormt misschien wel de beste m.b.t. de complexiteit. Verder onderzoek zal ook dat moeten uitwijzen.
Het belangrijkste resultaat van dit onderzoek is misschien wel het volgende. Dat is het besef dat een algoritme dat het grootste deel van de zoekboom efficiënt doorzoekt zou kunnen volstaan. Ook het besef dat de verzameling kanten mogelijk doorzocht kan worden zonder vast te houden aan het graafformalisme is iets wat een minder algebraïsche behandeling van het probleem waarschijnlijk niet aan de orde had gebracht. Ik denk zelf dat de gezochte periodiciteit wel in de verzameling paden kan zitten, alleen zou het vinden of aanbrengen van die periodiciteit weleens neer kunnen komen op een algoritme met exponentiële complexiteit.

Anderzijds zou er misschien ook een tegenbewijs te leveren kunnen zijn.
Groepoïden hebben voorzover ik nu weet geen subgroepen en dat is waarschijnlijk niet handig mbt quantumrekenen Dat is echter niet uitgesloten, veel huidig onderzoek in het kwantumrekenen is gericht op uitbreiding voorbij het verborgen subgroepprobleem.
MapReduce heb ik nog niet kunnen gebruiken voor het ontwerp van een kwantumalgoritme. We hebben in dit paper ook nog niet gekeken naar heuristieken om het zoeken te versnellen, iets wat in het licht van het voorgaande, alsmede voor praktische toepassingen toch wel zeer nuttig zou zijn.
Toekomstig werk zou er ook uit bestaan het bewijs van Ratner en Warmuth [22] goed te bestuderen en een oplossing te implementeren volgens hun voorschrift: eerste een bijna optimale oplossing bepalen en die dan lokaal optimaliseren, misschien dat dat ook binnen het kader van A* past.
Ook quantum walks, die de kern van de kwantumgraafalgoritmen bleken te vormen heb ik nog niet bestudeerd.
In iedere geval is er tot op heden nooit een bruikbaar patroon in de puzzel gevonden dat een verborgen structuur aangeeft, zoals dat bij oplosbaarheid nog wel het geval is. Ik ben er niet zeker van dat die structuur van de verborgen subgroep onbekend mag zijn bij het ontwerpen van een algoritme. In Shor's algoritme lijkt die middels Euler's totient functie al vooraf duidelijk te zijn. Shor's methode is door Kitaev in het algemene geplaatst door diens bewijs dat Shor een Abelse subgroepprobleem oplost. Het is dan ook niet verwonderlijk dat Moore en Narayanan probeerden om ook het Abelse subgroepprobleem zoals we dat bij de oplosbaarheid zijn tegengekomen op een zelfde manier proberen aan te pakken [15]. Bij hen bleef er een leemte bestaan tussen idee en fysica.

Referenties

1. Stevenhagen , P. (2011). ALGEBRA I. College-dictaat ULeiden, TU Delft. Retrieved online at websites.math.leidenuniv.nl/algebra/algebra1.pdf

2. Shor, P.W. 1994. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, Nov. 20–22, 1994, IEEE Computer Society Press, pp. 124–134. http://arxiv.org/abs/quant-ph/9508027v2

3. Moore, C., Russell, A. and Schulman, L. J. The symmetric group defies strong Fourier sampling: Part I. http://arxiv.org/abs/quant-ph/0501056

4. Jozsa, R. 2001. Quantum factoring, discrete logarithms and the hidden subgroup problem Computing in Science & Engineering
Mar/Apr 2001 Volume: 3 Issue:2

5. Reinefeld, A, Schütt, T. 2009. Out-of-core parallel frontier search with MapReduce. in
Mewhort, Douglas J. K. (ed.) et al., High performance computing systems and applications. 23rd international symposium, Berlin: Springer. Lecture Notes in Computer Science 5976, 323-336 (2010).

6. Deutsch, D. 1985. Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London A 400, pp. 97–117 (1985)

7. Law, E. 27.03.2007. Group Theory. Lecture sheets. Document Retrieved online at www.cs.cmu.edu/~elaw/files/grouptheory.pdf

8. Spitznagel, E.L.Jr. 1967.A New Look at the Fifteen Puzzle. Mathematics Magazine, Vol. 40, No. 4 (Sep., 1967), pp. 171-174. Mathematical Association of America

9. Grover L.K. 1996. A fast quantum mechanical algorithm for database

search. Proceedings, 28th Annual ACM Symposium on the Theory of Computing, p. 212

10. Grover L.K. 2001. From Schrödinger's equation to quantum search algorithm. American Journal of Physics, 69(7): 769-777.

11. Allender, E., Loui, M.C., Regan, K.W. Complexity Classes. State University of New York at Buffalo. Book chapter, retrieved online at: ftp.cs.rutgers.edu/pub/allender/ALRch33.pdf

12. Rieffel, E., Polak W. 2000. An Introduction to Quantum Computing for Non-Physicists. FX Palo Alto Labratory. Document retrieved onlne: arXiv:quant-ph/9809016v2

13. Barenco, A., Bennett, C.H.,Cleve, R., DiVincenzo, D.P. Margolus, N., Shor, P. Sleator, T., Smolin, J. Weinfurter, H. 1995. Elementary gates for quantum computation submitted to Physical Review A, March 22, 1995 arXiv:quant-ph/9503016v1

14. Quantum computing for beginners. Narayanan, A. undated. Department of Computer Science University of Exeter

15. Moore, M, Narayanan, A. undated. Quantum inspired computing. Dept. of CS Univ. of Exeter

16. Archer, A.F. A Modern Treatment of the 15 Puzzle

17. Korf, R.E. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search* Artificial Intelligence 27, 97-109 Elsevier Science Publishers B.V. (North-Holland)

18. Woolsey Johnson W. and Story, W.E. 1879. Notes on the "15" Puzzle. 1897 American Journal of Mathematics, Vol. 2, No. 4 (Dec., 1879), pp. 397-404

19. Lomont, C. 2004. The Hidden Subgroup Problem - Review and Open Problems. http://arxiv.org/abs/quant-ph/0411037v1

20. Kitaev, A. Yu. Quantum measurements and the Abelian Stabilizer Problem http://lanl.arxiv.org/abs/quant-ph/9511026v1

21. Richard Jozsa (Dec 2000) Quantum factoring, discrete logarithms and the hidden subgroup problem http://arxiv.org/abs/quant-ph/0012084v1

22. Ratner, D. and Warmuth, M. 1990 The (n2-1)-Puzzle and Related Relocation Problems. J. Symbolic Computation (1990) 10, 111-137

23. Erickson, J. Algorithms Course Materials APSP January 2011 http://www.cs.uiuc.edu /~jeffe/teaching/algorithms/

24. Kreyszig, E. 2006. Advanced Engineering Mathematics. by John Wiley & Sons, Inc.

25. Brown, R. 1987. From Groups to Groupoids: A Brief Survey, Bull. London Math. Soc. 19 (1987) 113-134

26. Weinstein, A. Groupoids: Unifying Internal and External Symmetry A Tour through Some Examples. Notices of the AMS, Vol. 43- 7.

27. Cannas da Silva, A., Weinstein, A. 1998. Geometric Models for Noncommutative Algebras. University of California at Berkeley December 1,

28. Romein, J.W., Bal, H.E., Schaeffer, J. and Plaat, A. 2002. A Performance Analysis of Transposition-Table-Driven Work Scheduling in Distributed Search. IEEE Transactions on Parallel and Distributed Systems, VOL. 13, NO. 5.

29. Nielsen, M.A. Chuang, I.L. 2010. Quantum Computation and Quantum Information 10th Anniversary Edition. ISBN: 9781107002173

30. Russel, S. & Norvig, P. 2010. Artificial Intelligence: A Modern Approach, 3rd Edition. Prentice Hall, Upper Saddle River, NJ, USA.

31. Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271.

32. McCoy, N.H.,Berger, T.R. 1971. Algebra: Groups, Rings and Other Topics. Voston: Allynn and Bacon.

33. A Panoply of Quantum Algorithms. Bartholomew Furrow

34. Buhrman, H., Cleve, R., de Wolf, R., Zalka, C. 1996. Bounds for Small-Error and Zero-Error Quantum Algorithms. PhysComp.

35. Tight bounds on quantum searching. Boyer, M., Brassard, G., Høyer, P., Tapp, A.

36. Seibel, K., Cull, P. 1994.The Knight's Tour on the Cylinder and Torus. REU 1994 Proceedings. http://www.math.oregonstate.edu/~math_reu/REU_Proceedings/Proceedings1994/7_Seibel94.pdf

37. Lieven, L.B. 2007. http://www.neverendingbooks.org/index.php/the-15-puzzle-groupoid-1.html

Appendix 1, toroïdale afbeelding van de puzzel

Is het mogelijk om een 1:1 afbeelding van de toegestane toestanden van de puzzle z'n permutatie groep te maken op een groep die een groepoïde formalisme onnodig maken? en misschien meer perspectief bieden op een verborgen subgroep.

Sleutel tot de aanpak is een toroïdale afbeelding van het bord [36]. De kolommen en rijen worden zodanig op elkaar aangesloten dat het verplaatsen van vier posities in elke kardinale richting een kardinaal je terug brengt naar de oorspronkelijke positie. Een toestand van de puzzel kan worden omgezet in een toestand met het lege vlakje (label 16) in vakje 16, door de kolommen en rijen rechts en naar beneden te rollen.

bijvoorbeeld:

15 12 7 1 ⇒ 1 15 12 7 ⇒ 9 2 6 11 14 10 16 5 ⇒ 5 14 10 16 ⇒ 13 3 4 8 2 6 11 9 ⇒ 9 2 6 11 ⇒ 1 15 12 7 3 4 8 13 ⇒ 13 3 4 8 ⇒ 5 14 10 16

Gegeven een toroïdale afbeelding het bord, kan de bovenstaande herschikking worden gezien als een eenvoudige vertaling van de oorsprong. Als zodanig blijft de essentiële indexering van de posities van de blokjes behouden. Blokjes 1 tot en met 15 in het herschikte tableau te definiëren een permutatie. In cykelnotatie: (1,9) (3,6) (4,11,12,7) (5,13) (10,15)

Deze permutatie geeft de rangschikking van de blokjes weer ten opzichte van de lege vlakje. De voorstelling wordt aangevuld met twee cyclische permutatie groepen van orde 4. Deze kunnen eenvoudig worden gezien als tellers die het aantal posities naar links en boven geven dat de oorsprong is verschoven.

(1,9)(3,6)(4,11,12,7)(5,13)(10,15) * (16,18)(17,19) * (20,21,22,23)

waar naar 1 boven = (16,17,18,19) en 1 naar links = (20,21,22,23)

Het is goed te zien dat elke rangschikking van blokjes in de vijftien puzzel uniek en eenduidig vertaald kan worden in een permutatie van deze vorm. De toegestane verschuivingen kunnen worden gemodelleerd door toepassing van een van de volgende permutaties op de toestand.

L := (1,4,3,2)(5,8,7,6)(9,12,11,10)(13,15,14)(20,21,22,23);

R := (1,2,3,4)(5,6,7,8)(9,10,11,12)(13,14,15)(20,23,22,21);

U := (1,13,9,5)(2,14,10,6)(3,15,11,7)(4,12,8)(16,17,18,19);

D := (1,5,9,13)(2,6,10,14)(3,7,11,15)(4,8,12)(16,19,18,17);

waar L het lege vlakje wisselt met de token naar links, etc.

Gezien deze bewering (waar na inspectie), kunnen alle toestanden van de puzzel gemodelleerd worden door producten van de bovenstaande generatoren.
Het resterende probleem is dat de bovengenoemde generatoren kunnen worden toegepast op een manier die niet geschikt voor de fysieke puzzel.
R toegepast op de identiteit positie wisselt het lege vlakje met het blokje in positie 13:

1	2	3	4
5	6	7	8
9	10	11	12
16	14	15	13

Er kan worden aangetoond dat al deze swaps buiten de randen van de puzzel kunnen worden bereikt met toegestane transposities. Bijvoorbeeld, de bovengenoemde positie kan worden bereikt door de reeks, LLLURDRRULLLDRRURDLL L. Dus, alle producten gevormd met de bovenstaande generatoren vertegenwoordigen toegestane puzzel toestanden.
En daarom is de groep gevormd uit de bovenstaande generatoren een goed gevormd model van de15 puzzel.

Appendix 2 mapreduce

Functional programming (e.g., Lisp)

– map function applies a function to each value of a sequence

(define (square n)

(n n))*

(map square '(1 2 3 4 5))

outputs (1 4 9 16 25)

– reduce function combines all elements of a sequence using a binary operator

(define (plus a b)

(+ a b))

(reduce plus 0 '(1 2 3 4 5))

outputs 15

(((((0+1)+2)+3)+4)+5)