Kvantealgoritmer

Ulrich B. Hoff, Kvantify

Kvantecomputeren er en spirende teknologi mange efterhånden har hørt om. Mindre udbredt er kendskabet til, hvordan de opnår deres potentiale til at løse beregningsproblemer, der hidtil har været uden for vores rækkevidde. Hardwaren er naturligvis en stor del af forklaringen, idet den funktionaliserer kvantefysikkens fænomener og gør dem til en kontrollerbar ressource, men derefter er det kvantealgoritmerne, der tager over og omsætter de ressourcer til effektive løsninger – i bedste fald med en eksponentiel speed-up over klassiske computere, men sandsynligvis kun for en afgrænset mængde af problemer.

"Ingen arme, ingen kage!" Denne essentielle læresætning gælder i høj grad også inden for kvantecomputing. Vi vil så umådeligt gerne have fingrene ned i kagedåsen med alle de gode løsninger på de allersværeste beregningsproblemer, som i den grad vil kunne forsøde vores tilværelse, men dåsen er desværre gemt godt af vejen et sted højt oppe på øverste hylde. Vi skal derfor virkelig yde en indsats; vi skal udvikle den nødvendige hardware, der kan få os op i højden, og vi skal på en krævende rekognosceringsmission rundt i de mørke strabadserende afkroge af matematikken og datalogien for at identificere og analysere de eftertragtede hårde problemer. Derefter venter arbejdet med at få lirket dåsen op, og det er her kvantealgoritmerne kommer i spil.

Beregnelighed

Hele computerbegrebet og den fundamentale forståelse af dens funktion og formåen skylder vi den britiske matematiker Alan Turing. I 1930'erne udviklede han en model for, hvad en beregning er i termer af nogle helt basale processer og operationer – en konstruktion, der senere er blevet kendt som en *Turingmaskine*. Med andre ord, så formaliserede Turing altså, hvad de grundlæggende ingredienser i en algoritme er. Sammen med den amerikanske matematiker Alonzo Church gik han endda så vidt som at opstille en hypotese om, at alt hvad der er realistisk beregneligt kan brydes ned i operationer udført af en Turingmaskine. Den påstand er kendt som *Church-Turing-tesen*.

Idéen om at få maskiner til at udføre beregninger går dog noget længere tilbage end Alan Turings formalisering. Et af de tidlige eksempler er den engelske polyhistor Charles Babbages mekaniske computer "the analytical engine", og det var netop til denne maskine, at Ada Lovelace i 1848 skrev verdenshistoriens første computerprogram til udregning af Bernoullital. Lovelace grundlagde dermed computerprogrammering og softwareudvikling, og hun havde store visioner for, hvad der kunne opnås i samspillet mellem algoritme, programmering og maskine [1]:

> "Supposing that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."
>
> *–Ada Lovelace*

Det med kompleksiteten skal vi se nærmere på, for selvom det kan synes som om computere er i stand til at simulere og beregne hvad som helst, så vokser træerne ikke ind i himlen.

Figur 1. Alan Turing (t.v.) lagde ikke blot fundamentet for computeren, men spillede også en central rolle for de alliere-des arbejde med at bryde tyskernes Enigmakryptering. Ada Lovelace (t.h.), datter af Lord Byron, arbejdede i midten af 1800-tallet sammen med Charles Babbage og skrev verdens første computerprogram til hans maskine.

Kompleksitet

Efter udviklingen af den transistorbaserede computer er udviklingen af regnekraft gået stærkt. I 1965 forudsagde Gordon Moore, at antallet af transistorer i CPUer ville blive fordoblet ca. hvert andet år, og den forudsigelse har holdt stik med bemærkelsesværdig præcision. Siden 1999 har CPUer kunnet processere data med mere end en milliard operationer i sekundet og de bliver kun hurtigere. Det ville iøvrigt ikke have overrasket Alan Turing, da han allerede i 1950 forudsagde, at computerteknologien 50 år senere ville være på netop det niveau.

Nogle problemer er imidlertid af en sådan karakter, at ligegyldig, hvor hurtige vi gør processorerne, så vil det tage eksponentielt lang tid at nå en løsning og derfor være praktisk umuligt. Hvis computeren først leverer sit output efter flere år eller årtier, så vil svaret nok være af meget beskeden praktisk relevans. I de tilfælde må man ty til heuristiske løsningsmetoder, som gør det muligt at

finde en tilnærmet løsning inden for overskuelig tid – men uden garanti for, at den er optimal.

I datalogien klassificeres problemerne efter deres kompleksitet, dvs. hvordan det nødvendige antal ressourcer/operationer skalerer med størrelsen af problemet. Det er netop skaleringen der afgør, om vi kan løse problemet, og det element er flettet ind i den såkaldte *stærke Church-Turing-tese*, også kendt som den kompleksitetsteoretiske Church-Turing-tese (her i John Preskills udlægning) [2]:

> "A problem can be solved efficiently if the number of steps on a Turing machine scales like a polynomial in the size of the input to the problem".

Med "effektiv løsning" menes altså, at den tid det tager, maksimalt vokser polynomisk med størrelsen af problemet, og det er de problemer, vi kan håndtere med nuværende computere. De er indeholdt i kompleksitetsklassen P. Blandt disse findes nogle, som alle de andre problemer i klassen kan reduceres til. De betegnes som P-komplette problemer, og en løsning til et af dem, vil automatisk også give en løsning til alle andre problemer i P. Men der findes et helt *polynomisk hierarki* af kompleksitetsklasser, og udfordringen er, at rigtig mange af de hårde problemer, vi meget gerne vil kunne løse, ligger uden for P og dermed uden for vores nuværende rækkevidde. Og det vil de blive ved med, så længe den stærke Church-Turing-tese står ved magt. En af de kompleksitetsklasser, der er meget fokus på, er NP. Det er de problemer, for hvilke man med en Turingmaskine effektivt kan verificere en løsning, når den er fundet. Der er derimod ingen kendte effektive algoritmer til at finde løsningerne. Det er fortsat et stort åbent problem at afklare om P er lig med NP. Den amerikanske datalog Scott Aaronson formulerer det således [3]:

> "Look, this P versus NP question, what can I say? People like to describe it as "probably the central unsolved problem in theoretical computer science." That's a comical understatement. P vs. NP is one of the deepest questions that human beings have ever asked."

I NP findes der ligeledes en hård kerne af NP-komplette problemer. De omfatter bl.a. kombinatoriske optimeringsproblemer som *travelling salesman problemet*, og det er nogle af de problemer, det virkelig vil gøre en kæmpe forskel for både forskning og industri at finde effektive løsninger på. Men det er nok for meget at bede om, og mindre kan også gøre det. Selvom det som sagt er et åbent uafklaret spørgsmål, så er der en generel overbevisning om, at der i NP findes problemer, som ikke er indeholdt i P og dermed at $\mathsf{P} \neq \mathsf{NP}$. Og derudover, så er der en formodning om, at nogle af de problemer ikke er NP-komplette, det gælder eksempelvis *faktoriseringsproblemet*, der omhandler primtalsfaktorisering af heltal. Som vi skal se om et øjeblik, så er netop det problem blevet helt centralt for kvantecomputere.

Kvantecomputeren

Der eksisterer ikke en entydig definition af, hvad en kvantecomputer er. Der findes nemlig flere fysisk mulige modeller for, hvordan kvanteberegninger kan realiseres. Den dominerende model er på nuværende tidspunkt kredsløbsmodellen, men derudover findes fx også topologiske, målingsbaserede og adiabatiske modeller. Arbejder man inden for kredsløbsmodellen, så giver DiVincenzos tjekliste [7] nogle retningslinjer for, hvilke elementer kvantecomputeren skal rumme: (1) en fysisk implementering af mange qubits, (2) som skal kunne initialiseres i bestemte starttilstande og (3) manipuleres hhv. et universelt sæt af en- og to-qubitgates. Derudover skal der være (4) en effektiv oversættelse af beregningsproblemet til et kvantekredsløb og (5) de enkelte qubits sluttilstande skal kunne udlæses og omformes til klassiske bitværdier.

Lad os derefter få slået fast, at der med kvantecomputeren *ikke* ændres på, hvilke beregningsproblemer, det helt fundamentalt er muligt at løse. Alt hvad kvantecomputeren kan, kan også simuleres af en sædvanlig computer udstyret med en tilfældighedsgenerator til at håndtere kvantefysikkens ikke-deterministiske aspekter, og derfor er den oprindelige Church-Turing-tese ikke truet. Men hvorfor så overhovedet bruge kræfter på at udvikle en kvantecomputer? Fordi den i nogle tilfælde kan rykke grænserne markant for, hvad der er *praktisk muligt* og *praktisk umuligt*. Eller med andre ord, hvilke problemer vi kan løse effektivt og hvilke vi ikke kan. Kvantecomputeren er nemlig ikke bare en ny kraftigere variant af den computer vi allerede kender. Den udnytter nogle ganske andre fysiske principper og opnår derfor sin *universalitet* – evnen til i princippet at kunne udføre en vilkårlig beregning – ad andre veje end den klassiske computer. Information er i sidste ende noget fysisk, og enhver processering af information vil derfor også nødvendigvis involvere fysiske vekselvirkninger i henhold til fysikkens love. Den klassiske computers halvlederteknologi er grundlæggende set kvantefysisk, og Moores lov ville uden tvivl være blevet overtrumfet af Murphys lov langt tidligere, uden den forståelse. Men til forskel fra den klassiske computer, så funktionaliserer kvantecomputerens hardware kvantemekanikkens fænomener og udnytter dem aktivt. Vi ved fra mange og utvetydige eksperimentelle tests af Bells ulighed [4, 5], at kvantefysikken rummer fænomener, der ikke kan reduceres til en klassisk fysisk beskrivelse, og ved at bringe dem i spil, gør kvantecomputeren det muligt at indkode og processere information på helt nye måder. Dermed opstår også nye grænser mellem let og svært i kompleksitetslandskabet af beregningsproblemer. I 1997 definerede Bernstein og Vazirani [6] således den kompleksitetsklasse af problemer, som kvantecomputeren kan løse effektivt. Den hedder BQP – Bounded-Error Quantum Polynomial-Time. Præcis hvordan BQP passer ind i det polynomiske hierarki, hvis overhovedet, er endnu et åbent spørgsmål. Der er derfor heller ikke en klar forståelse af, hvilke typer af problemer kvantecomputeren kan løse effektivt.

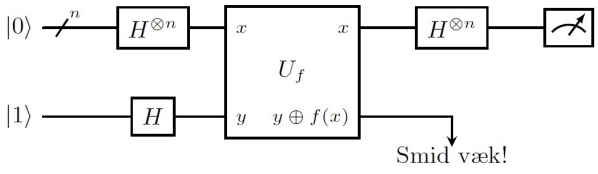
Ud i krogene af Hilbertrummet

For at vende tilbage til billedsprogets verden, så kan man med nogen ret sige, at vi med kvantecomputeren er ved at udvikle en meget specialiseret og kraftfuld form for slagværktøj, men vi har endnu langt fra det fulde billede af, hvilke søm den kan anvendes på. Men vi har trods alt en klar fornemmelse af, at der i BQP findes problemer som ikke er indeholdt i P, og det er i sig selv interessant, da effektive kvantealgoritmer til de problemer udfordrer den stærke Church-Turing-tese. Selvom vi endnu ikke entydigt kan sige, hvad der gør et problem til et godt kvante-problem – ét hvor en kvantealgoritme vil kunne levere en bedre løsning – så kan vi prøve at danne os et billede af de ønskede karakteristika.

Et generelt træk er, at algoritmen skal udnytte den eksponentielle skalering af det vektorrum, kaldet et Hilbert-rum, de kvantefysiske qubits er beskrevet i. Tilstandsvektoren for en qubit lever i et sådan vektorrum af dimension 2, udspændt af basisvektorerne $\{|0\rangle, |1\rangle\}$, og alle normaliserede linearkombinationer af basisvektorerne af formen $\alpha|0\rangle + \beta|1\rangle$, hvor $\alpha, \beta \in \mathbb{C}$ og $|\alpha|^2 + |\beta|^2 = 1$, er gyldige tilstande for qubitten. Til sammenligning er mulighederne for en sædvanlig bit givet ved det 1-dimensionale tilstandsrum $\mathcal{S} = \{0, 1\}$. Allerede her ser vi en forskel mellem de to grundlæggende informationsenheder, men det bliver for alvor tydeligt, når vi sammenligner systemer af flere bits og qubits. For n bits er det samlede tilstandsrum af dimension n og givet ved det *cartesiske* produkt af de individuelle udfaldsrum, og rummer i alt 2^n forskellige muligheder, $\mathcal{S}_n = \{\ (0,0,\ldots,0,0),\ (0,0,\ldots,0,1),\ (0,0,\ldots,1,0),\ \ldots,\ (1,1,\ldots,1,1)\ \}$. For et system af n qubits er den samlede kvantetilstand beskrevet i et Hilbertrum givet ved *tensorproduktet* af de individuelle rum, $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n$. Det er et vektorrum med dimension 2^n udspændt af basisvektorerne $\{|x\rangle\} = \{|00\cdots00\rangle, |00\cdots01\rangle, |00\cdots10\rangle, \ldots, |11\cdots11\rangle\}$, og den samlede tilstand for de n qubits er generelt givet ved $|\psi\rangle = \sum_{i=1}^{2^n} \alpha_i |x_i\rangle$. Tilstanden er en superposition af alle Hilbertrummets basisvektorer, og hvis $\alpha_i \neq 0$ for alle i, vil den rumme en komponent af hver af de klassisk mulige bit-tilstande på én gang. Men endnu vigtigere, så rummer en tilstand af den form *entanglement* og kan derfor ikke skrives som et produkt af uafhængige tilstande for hver af de n qubits. Ved at udnytte korrelationer på tværs af qubits, som entanglement repræsenterer, kan kvantealgoritmer nå ud i alle krogene af Hilbertrummet, og ikke kun de isolerede rum for hver enkelt qubit. En entanglet tilstand kan naturligvis indkodes i sædvanlige bits og i princippet simuleres på en klassisk computer, men en repræsentation af den generelle kvantetilstand ovenfor kræver specificering af $2(2^n - 1)$ reelle parametre (der fratrækkes én pga. normaliseringskravet), og den eksponentielle skalering gør, at det *ikke* kan gøres effektivt.

Deutsch-Jozsa-algoritmen

En af de første kvantealgoritmer der blev udviklet, og som udnytter kvante-parallelisme til at løse et problem hurtigere end den bedste klassiske løsning er *Deutsch-Jozsa algoritmen* fra 1992, udviklet af briterne David Deutsch og Richard Jozsa. Problemet algoritmen løser er følgende: Vi har en "black box", som udregner funktionen $f : \{0,1\}^n \to \{0,1\}$. f tager altså et heltalligt input x i intervallet $[0, 2^n - 1]$, repræsenteret ved n bits, og giver en funktionsværdi $f(x)$, som enten er 0 eller 1. Funktionen f har én af to mulige egenskaber; enten er den konstant og giver samme funktionsværdi for alle x, eller den er balanceret og giver 0 for halvdelen af x-værdierne og 1 for den anden halvdel. Opgaven er nu at afgøre om f er konstant eller balanceret ved brug af færrest mulige ressourcer. Figur 2 viser kredsløbet for den kvantealgoritme, som Deutsch og Jozsa udviklede til at klare opgaven.



Figur 2. En af de tidligste kvantealgoritmer, der demonstrerede kvantecomputerens potentiale til at udføre beregninger mere effektivt end klassiske computere er Deutsch-Jozsa algoritmen fra 1992.

Algoritmen bruger $n + 1$ qubits, som starter i tilstanden $|\psi_1\rangle = |00\cdots1\rangle$, og som hver bringes i en superpositionstilstand vha. n Hadamardgates, H. Hadamardgaten har den egenskab, at $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ og $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, og derfor vil de første n qubits blive bragt i en superposition af alle 2^n mulige værdier for x. Alle mulige input fødes dermed samtidigt ind i den unitære operator U_f, som implementerer funktionen f gennem transformationen $|x, y\rangle \to |x, y \oplus f(x)\rangle$, og pga. lineariteten af kvantemekanikken udregnes alle funktionsværdierne $f(x)$ parallelt. Resultatet er, at funktionsværdierne bliver indkodet i amplituderne for superpositionstilstanden af de første n qubits:

$$|\psi_2\rangle = \sum_{x=0}^{2^n-1} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Selvom det ikke altid er tilfældet – det afhænger bla. af værdien af n – så vil der generelt være en grad af entanglement i tilstanden $|\psi_2\rangle$, og algoritmen bevæger sig derved ud i den interessante del af Hilbertrummet.

I næste trin af algoritmen anvendes igen Hadamardgates på hver af de n første qubits, og derved kommer endnu en central komponent af stort set alle kvantealgoritmer i spil. I første omgang brugte vi Hadamardgates for at skabe superpositionstilstande, og efter at have indkodet funktionsværdierne i superpositionernes komplekse amplituder, så *interfereres* amplituderne nu ved at anvende Hadamardgates endnu en gang. For at forstå effekten af dette, starter vi med at se på en enkelt af de 2^n bitstrenge, der indgår i $|\psi_2\rangle$ og hvordan den transformeres, dvs. $H^{\otimes n}|x\rangle$. For hver Hadamard vi anvender, får vi en faktor $1/\sqrt{2}$ og i hvert tilfælde får vi en superposition af $|0\rangle$ og $|1\rangle$, dvs. i alt en

sum over de 2^n mulige bitstrenge $|x\rangle$:

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (?) \cdot |j\rangle$$

Vi mangler altså bare at bestemme amplituden for hvert led i superpositionen. Hvis der er et 1-tal på den k'te plads i $|x\rangle$ fører Hadamard transformationen til et bidrag til den resulterende superpositionstilstand med $-|1\rangle$ på den k'te plads – husk, at $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Fortegnet på det j'te led i superpositionen afhænger af antallet af 1-taller på positioner der overlapper med 1-taller i $|x\rangle$ og er derfor givet ved $(-1)^{x\cdot j}$, hvor $x \cdot j = x_0 j_0 \oplus x_1 j_1 \oplus \cdots \oplus x_{n-1} j_{n-1}$. Vi får derfor, at

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{x\cdot j}|j\rangle,$$

og kombineres dette med udtrykket for $|\psi_2\rangle$ ser vi, at den samlede tilstand for de $n+1$ qubits efter anden blok af Hadamardgates er,

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{j=0}^{2^n-1} \left[\sum_{x=0}^{2^n-1} (-1)^{x\cdot j + f(x)}|j\rangle \right] \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right).$$

Det kan vi nu oversætte til sandsynligheder, $p_i = |\langle i|\psi_3\rangle|^2$, for udfald af de målinger, der i sidste ende foretages på de første n qubits. Specielt er vi interesseret i udfaldet $|00\cdots0\rangle$, hvor alle qubits altså måles i tilstanden $|0\rangle$. Amplituden for dette udfald er $\langle 00\cdots0|\psi_3\rangle = \sum_x (-1)^{f(x)}/2^n$. Hvis f er konstant (0 eller 1 for alle x), så antager amplituden værdien ±1, svarende til en sandsynlighed $p = 1$. Hvis f i stedet er balanceret, så vil de positive og negative bidrag til amplituden for resultatet $|00\cdots0\rangle$ præcist udligne hinanden, og vi får $p = 0$. Det betyder, at i det tilfælde vil mindst én af de n målinger give et resultat forskellig fra 0. Konklusionen er altså, at vi med blot en enkelt måling af de n qubits kan løse opgaven og afklare, om funktionen f er konstant eller balanceret. Til sammenligning, så kan man i en klassisk løsning af opgaven kun evaluere f for én x-værdi ad gangen. Da der er 2^n forskellige x-værdier skal vi i værste tilfælde udregne f for $2^n/2+1$ af dem, for at afgøre om funktionen er konstant eller balanceret.

Deutsch-Jozsa-algoritmen er en af de absolut simpleste kvantealgoritmer, og det problem den løser har ingen praktisk relevans overhovedet. Ikke desto mindre, så illustrerer den nogle af de grundlæggende ingredienser, der gør det muligt at konstruere effektive kvantealgoritmer.

Fourier, fase og faktorisering

En række af langt mere kraftfulde – og tilsvarende mere komplicerede – algoritmer, relaterer sig til periodicitet og det at finde og udnytte periodiske egenskaber ved funktioner eller talfølger. En del af nøglen til løsningerne er netop at drage nytte af en underliggende struktur i problemerne, og som John Preskill beskriver det i sine forelæsningsnoter: "[W]hile finding needles in a haystack may be difficult, finding *periodically* spaced needles in a haystack can be far easier." [11].

En absolut hjørnesten for disse algoritmer er muligheden for effektivt at implementere den diskrete Fourier transformation (*Quantum Fourier Transform*) på en kvantecomputer. Kredsløbsimplementeringen af kvante-Fourier transformationen på n qubits forbruger et antal af gates, der skalerer som n^2. Til sammenligning skalerer antallet af gates der kræves for en klassisk Fast Fourier Transformation (FFT) som $n2^n$, og kvantealgoritmen giver derfor en eksponentiel speed-up i forhold til klassiske beregninger.

Som vi tidligere har været inde på, så er interferens af qubit-tilstandenes sandsynlighedsamplituder et af de elementer, der spiller en afgørende rolle for kvantealgoritmernes potentiale. Interferensen hænger direkte sammen med muligheden for at bringe qubits i kohærente superpositionstilstande og dermed eksistensen af en relative fase mellem sandsynlighedsamplituderne for superpositionens komponenter. Centralt for mange kvantealgoritmer er derfor også muligheden for effektivt at estimere sådanne relative faser for ukendte qubittilstande. Det lader sig gøre takket være *Phase Estimation* algoritmen, som igen henter sin styrke fra kvante-Fourier transformationen.

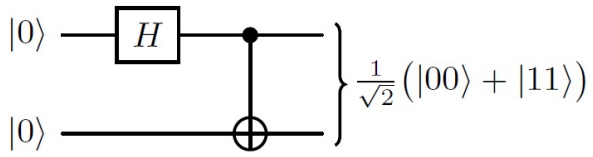
Sidst men ikke mindst, så spiller periodicitet en helt afgørende rolle for den ubetinget mest omtalte af alle kvantealgoritmer – *Shors algoritme*, der gør det muligt at bestemme primtalsfaktoriseringen for heltal (*heltalsfaktoriseringsproblemet*) med en eksponentiel speed-up. Også i dette tilfælde er det kvante-Fourier transformationen, der er den egentlige motor bag algoritmen. (Et par gode og pædagogiske fremstillinger af Shors algoritme findes i [9, 10]). Primtalsfaktorisering kan måske synes som et problem med beskeden relevans, men sikkerheden i såkaldt public key kryptografi, som anvendes i mere end 90% af alle digitale transaktioner (netbank, handel på nettet osv.) hænger på, at der ikke findes en effektiv løsning af netop det problem for klassiske computere. Shors algoritme, og muligheden for at kunne eksekvere den på kommende kvantecomputere, har derfor skabt meget stor global opmærksomhed omkring teknologien, og det er fortsat en af de helt store drivkræfter bag nationale så vel som private investeringer i udviklingen af kvantecomputere.

Ud over løsning af matematiske problemer, så har kvantecomputeren også et enormt potentiale i forhold til fysiske simuleringer af materialer, molekyler, kemiske processor osv. Kvantefysiske systemer bliver meget hurtigt praktisk umulige at simulere på klassiske computere pga. det eksponentielt store Hilbertrum, som tilstandene og processerne er beskrevet i. Med kvantecomputeren lader det sig imidlertid gøre at indkode processerne én-til-én og simulere naturens processer præcist som de foregår.

Hvad giver kvantealgoritmer speed-up?

Entanglement er som nævnt et entydigt kvantemekanisk fænomen, som ikke lader sig rumme i den klassiske fysik. Derfor anses entanglement også ofte som en essentiel ingrediens i kvantealgoritmer og for kvante-

teknologi generelt. Tilstedeværelsen af entanglement i et problem giver ofte en pejling om, at vi har at gøre med noget, der ikke lader sig beregne eller simulere effektivt på en sædvanlig computer. I 2002 viste Jozsa og Linden [12], at det er en *nødvendig* betingelse for enhver kvantealgoritme, at den involverer entanglement på tværs af ubegrænset mange af de involverede qubits, hvis den skal kunne levere en eksponentiel speed-up i forhold til klassiske beregninger.



Figur 3. En makismalt entanglet to-qubit tilstand kan generes ved brug af en Hadamardgate og to-qubitgaten "controlled-NOT", der flipper tilstanden af *target* qubitten (nederst), hvis *control* qubitten (øverst) er i tilstanden $|1\rangle$.

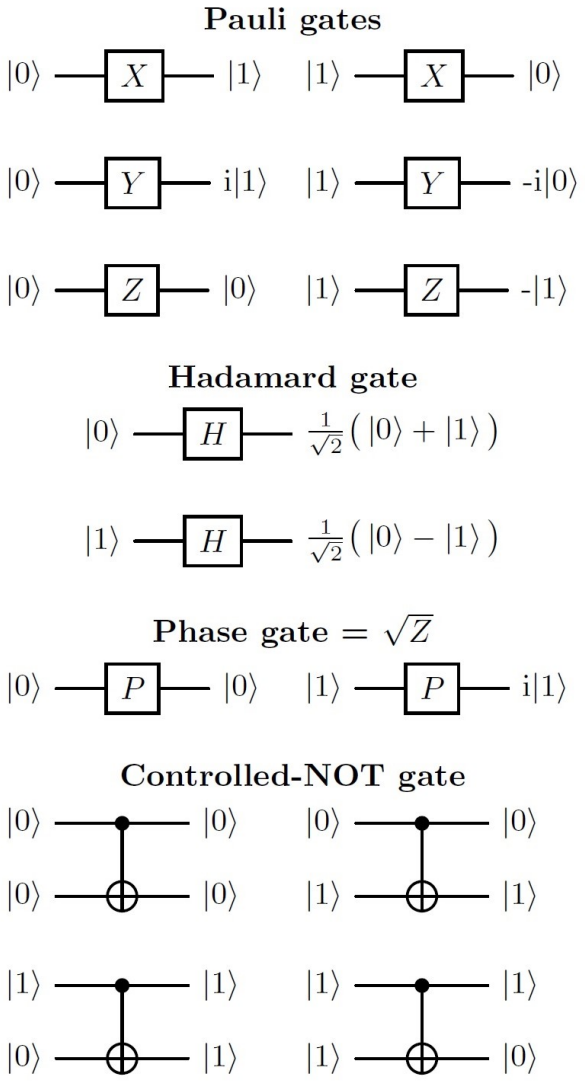
Men, som de også skrev i artiklen, så er det ikke den fulde fortælling i forhold til, hvad der gør et problem hårdt og dermed potentielt interessant at anvende en kvantecomputer på.

> "According to our main results above we can say that entanglement is necessary in a quantum algorithm (on pure states) if the algorithm is to offer an exponential speed-up over classical computation. Does this mean that entanglement can be identified as an essential resource for quantum computational power? [...] we will suggest (perhaps somewhat surprisingly) that this is *not* a good conclusion to draw from our results!" [12]

Et andet teoretisk resultat, som hjælper med at indkredse, hvornår en kvantealgoritme kan levere den eftertragtede eksponentielle speed-up er *Gottesman-Knill teoremet* [14] (her som det er gengivet af Nielsen & Chuang i [13]:

> "Suppose a quantum computation is performed which involves only the following elements: state preparations in the computational basis, Hadamard gates, phase gates, controlled-NOT gates, Pauli gates, and measurements of observables in the Pauli group, together with the possibility of classical control conditioned on the outcome of such measurements. Such a computation may be efficiently simulated on a classical computer."

De gates der refereres til i Gottesman-Knill teoremet kaldes også for Cliffordgates (se figur 4), og det er altså nødvendigt at anvende gates uden for det sæt, for at opnå en virkelig interessant kvantealgoritme. Eksempler på sådanne gates er bla. *T* gaten, der giver en faserotation på $\pi/4$ på en enkelt qubit, og *Toffoli*-gaten, der er en 3-qubit version af controlled-NOT-gaten (se figur 5). Tilføjelse af mindst én ikke-Cliffordgate er nødvendig for at opnå et universelt sæt af gates, således at alle operationer kan implementeres. Desværre – men måske nok forventeligt – så er disse gates også nogle af de allersværeste at realisere. Og det bliver ikke bedre af, at Aaronson og Gottesman i 2004 viste, at det ikke er nok bare med et par enkelte af disse gates, faktisk skal algoritmen benytte et antal, der skalerer polynomisk med antallet af qubits i algoritmen for ikke at kunne simuleres effektivt med en klassisk computer [15, 16].

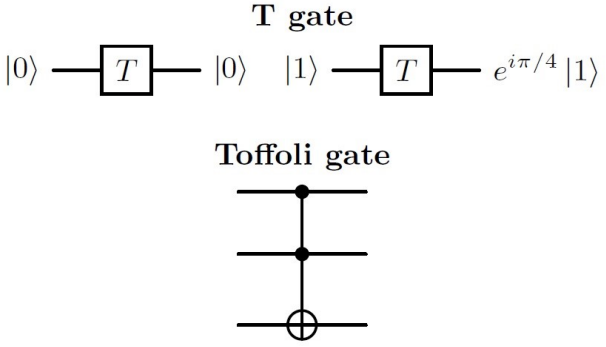


Figur 4. En kvantealgoritme, der udelukkende anvender gates fra Clifford sættet ovenfor, kan simuleres effektivt af en klassisk computer og vil derfor aldrig levere eksponentiel speed-up.

En måde at frembringe disse noget drilske gates er via såkaldte "magic states" (nogle gange er videnskabsfolk ikke så videnskabelige i deres navngivning) og bruge disse meget specifikke kvantetilstande som en ressource til at realise ikke-Cliffordgates. Magic states er selvfølgelig heller ikke lette at skabe, men der findes en metode, hvorved tilstande af højere og højere kvalitet kan destilleres ud fra en lille smule "magic" til at begynde med.

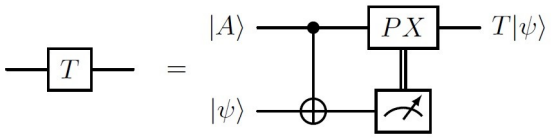
Et eksempel på en magic state er tilstanden

$$|A\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{i\pi/4}|1\rangle\big),$$



Figur 5. To eksempler på ikke-Cliffordgates. T-gaten virker på en enkelt qubit og introducerer en faserotation på $\pi/4$. Toffoligaten er en controlled-controlled-NOT-gate, der flipper tilstanden for target-qubitten, hvis og kun hvis begge control-qubits er i tilstanden $|1\rangle$.

og med den til rådighed kan man med kredsløbet i figur 6 implementere en T-gate [13]: Først "mikses" qubit 1 i magic-tilstanden med qubit 2 i input tilstanden $|\psi\rangle$ vha. en controlled-NOT-gate, og derefter udføres en måling på qubit 2. Hvis resultatet af målingen er 0 gøres ingenting, og hvis det er 1 udføres en kombineret PX-gate på den tilbageværende qubit. Resultatet er transformationen $|\psi\rangle \to T|\psi\rangle$.

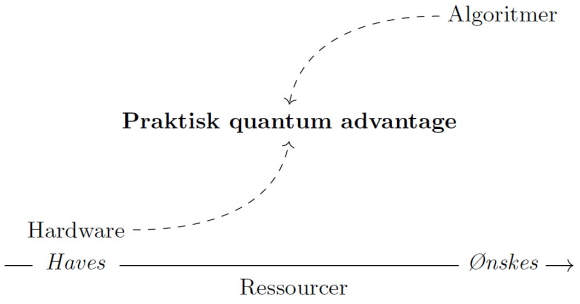


Figur 6. Med magic-tilstanden $|A\rangle$ som ressource kan en T-gate realiseres på en qubit ved hjælp af et ækvivalent kredsløb.

Quantum advantage

Lad os afslutningsvist vende blikket lidt udad. Som vi har set, så er den grundlæggende motivation for udviklingen af kvantecomputeren at tilvejebringe den hardware, der vil sætte os i stand til at afvikle kvantealgoritmer, som kan løse beregningsproblemer og simulationer, der er praktisk umulige at håndtere med klassiske computere. Det mål betegnes ofte som opnåelse af "quantum advantage". Allerede i 2019 annoncerede Google Quantum AI at have nået den milepæl [17], men dog for et problem uden real praktisk anvendelighed. Og selvom der siden er blevet opnået lignende quantum advantage resultater på andre hardware-platforme, så har vi stadig til gode at se praktisk quantum advantage demonstreret utvetydigt. Hvornår det kommer til at ske er svært at gisne om, men vi nærmer os stødt gennem kombinationen af to parallelle udviklingsprocesser.

Hardwareudviklingen er den del, der oftest er størst bevågenhed omkring, og der annonceres hyppigt nye landvindinger på tværs af forskellige qubit-modaliteter (superledende transmons, ioner, neutrale atomer, spins, fotonik m.fl), som hver har deres fordele og ulemper. Udviklingen i antallet af qubits er vokset støt og fideliteten af de gates der kan eksekveres er i nogle tilfælde så høj som 99,9955%. Mere interessant end blot at fylde flere qubits på kvanteprocessorerne er realiseringen af den første logiske qubit. Der skelnes mellem fysiske og logiske qubits, hvor førstnævnte er bruttoantallet af qubits, mens den logiske er en qubit, der er indkodet gennem et større antal fysiske qubits, og som via fejlkorrektion gøres robust over for dekohærens. Netop i den sammenhæng spiller ovennævnte *magic states* en helt afgørende rolle. Realiseringen af logiske qubits er en absolut forudsætning for at kunne køre komplekse kvantealgoritmer og opnå praktisk quantum advantage.



Figur 7. Den praktiske anvendelighed af kvantecomputeren er en kombination af modenheden af hardwaren og i hvilken grad algoritmerne formår at udnytte den tilgængelige hardware. Hardwaremodningen foregår relativt langsomt og vokser op fra bunden i takt med at nye teknologiske og mere skalerbare løsninger udvikles. På algoritmesiden findes løsninger med quantum advantage allerede, men de er hardwaremæssigt for ressourcekrævende til at have praktisk relevans. Men ressourcekravene reduceres hastigt i kraft af algoritmiske udviklinger, og det gør at de to udviklingsfronter af teknologien fra hver sin side nærmer sig det punkt, hvor praktisk quantum advantage bliver en realitet.

Den anden front af udviklingen er algoritmerne, og her sker der ligeledes hastige fremskridt. Der kommer hele tiden nye algoritmer til, og der har i de seneste år været et stort fokus på såkaldte variationelle kvantealgoritmer som *Variational Quantum Eigensolver* (VQE) [19] og *Quantum Approximate Optimization Algorithm* (QAOA) [18]. VQE er en hybridalgoritme, der kombinerer klassisk og kvantecomputing til at løse optimeringsproblemer, fx at bestemme grundtilstandsenergien for et molekyle. Givet en kvanteindkodning af cost-funktionen H og en parametriseret tilstandsfunktion $|\psi(p)\rangle$ bestemmes forventningsværdien $E = \langle\psi(p)|H|\psi(p)\rangle$ på kvantecomputeren og denne minimeres iterativt gennem klassisk optimering af parametrene p mellem hver evaluering af E. QAOA er ligeledes en hybrid iterativ algoritme, men er målrettet løsning af kombinatoriske optimeringsproblemer. Trods interessante og lovende anvendelser af de variationelle algoritmer, så er deres relevans nok af midlertidig

karakter og begrænset til den nuværende NISQ-æra, da de ikke kan levere den ønskede eksponentielle speed-up.

På sigt må man forvente, at fokus rykkes i retning af de kvantealgoritmer, der indtil videre er begrænset af manglen på logiske qubits, men har potentialet til at give en eksponentiel gevinst. Men også inden for den type algoritmer sker der nybrud, og det gør, at ressourcekravene til den kommende fejltolerante hardware hele tiden reduceres. Netop den optimering og modning af algoritmerne vil være helt afgørende for at fremrykke tidspunktet for opnåelsen af praktisk quantum advantage.

Ambitionen om at udvikle en kvantecomputer bliver ofte sammenlignet med Apolloprojektets "putting a man on the moon" målsætning, og der er da heller ingen tvivl om, at det er en virkelig krævende videnskabelig og teknologisk bedrift. Kvantealgoritmerne er helt essentielle for projektets succes, og som vi har forsøgt at illustrere, så er der også på den front store ubesvarede spørgsmål. Men så meget desto mere grund er der til at fokusere på de hårde matematiske problemer – på at identificere og analysere dem, optimere de klassiske løsninger og presse udviklingen forbi nuværende heuristikker og brute force-metoder. Den indsigt er helt afgørende for at forstå, hvilke problemer eller delproblemer, der på sigt skal uddelegeres til kvanteprocessorer, og hvordan vi på sigt optimerer samspillet mellem klassisk high-performance computing og kvantecomputere.

En lille fun fact perspektivering: Dåseåbneren (patenteret af Ezra J. Warner i 1858) blev først udviklet 50 år efter konservesdåsen. Indtil da var den foretrukne løsningsmodel hammer og mejsel.

Litteratur

[1] A. Lovelace, fra blogs.bodleian.ox.ac.uk/adalovelace/about-ada-lovelace/ "Ada Lovelace – celebrating 200 years of a computer visionary," Bodleian Libraries.

[2] J. Preskill (2022) "The Physics of Quantum Information", arxiv.org/abs/2208.08064v1.

[3] S. Aaronsen (2013) "Quantum Computing since Democritus", Cambridge University Press.

[4] J. S. Bell (1964) "On the Einstein Podolsky Rosen paradox", *Physics*, bind **1**, side 195.

[5] The Nobel Prize in Physics 2022, Popular science background, "How entanglement has become a powerful tool". www.nobelprize.org/uploads/2022/10/popular-physicsprize2022-3.pdf

[6] E. Bernstein og U. Vazirani (1997) "Quantum complexity theory", *SIAM J. Comput.*, bind **26**, side 1411–1473.

[7] D. P. DiVincenzo (2000) "The physical implementation of quantum computatiuon", *Fortschritte der Phyzik: Progress of Physics*, bind **48**, side 771.

[8] R. Jozsa (1997) "Entanglement and quantum computation", arxiv.org/abs/quant-ph/9707034.

[9] P. W. Shor (2001) "Introduction to quantum algorithms", arxiv.org/pdf/quant-ph/0005003.pdf.

[10] S. Aaronson (2007) "Shor, I'll do it", scottaaronson.blog/?p=208 (blogindlæg).

[11] J. Preskill, "Lecture Notes for Physics 229: Quantum Computation", theory.caltech.edu/∼preskill/ph229/notes/chap6.pdf.

[12] R. Jozsa og N. Linden (2002) "On the role of entanglement in quantum computational speed-up", arxiv.org/pdf/quant-ph/0201143.pdf.

[13] M. A. Nielsen og I. L. Chuang (2000) "Quantum Computation and Quantum Information", Cambridge University Press .

[14] D. Gottesman (1998) "The Heisenberg representation of quantum computers", arxiv.org/pdf/quant-ph/9807006.pdf.

[15] S. Aaronson og D. Gottesman (2004) "Improved simulation of stabilizer circuits", *Phys. Rev. Lett.*, bind **70**, side 052328. arxiv.org/abs/quant-ph/0406196.

[16] S. Aaronson, "Stabilizer Formalism", www.scottaaronson.com/qclec/28.pdf (forelæsningsnoter).

[17] F. Arute m.fl. (2019) "Quantum supremacy using a programmable superconducting processor", *Nature*, bind **574**, side 505.

[18] E. Farhi, J. Goldstone og S. Gutmann (2014) "A quantum approximate optimization algorithm´´, arxiv.org/abs/1411.4028.

[19] A. Peruzzo m.fl. (2014) "A variational eigenvalue solver on a photonic quantum processor", *Nature Commun.*, bind **5**, side 4213, arxiv.org/abs/1304.3061.

Ulrich Busk Hoff ph.d. i fysik og arbejder med udvikling og anvendelse af kvantealgoritmer som Quantum Engagement Specialist hos Kvantify. Han er desuden en ivrig videnskabsformidler.

En trykfejl i et af formeludtrykkene side 20 er rettet i denne onlineudgave. Se evt. "Errata" side 10 i Kvant nr. 1 fra 2024.