Průvodce implementace v ATOS QLM s úvodem do kvantového počítání

Best Practice Guide

Jiří Tomčala, Marek Lampart

15. listopadu 2022

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Průvodce implementace v ATOS QLM s úvodem do kvantového počítání

Jiří Tomčala & Marek Lampart

Obsah

Předmluva

Kvantové počítače jsou dnes jedním z nejžhavějších technologických odvětví budoucnosti. Tato technologie umožňuje řešit výpočetní problémy, které byly dříve považovány za neřešitelné, a významně ovlivnila kryptografii, chemii, optimalizaci, strojové učení a mnohé další.

Tyto kvantové počítače sice s největší pravděpodobností zcela nenahradí klasické počítače, ale kvantová technologie docela určitě významně změní způsob fungování světa.

Výzkum, který probíhá v mnoha významných technologických společnostech a start-upech, mezi nimiž jsou Google, IBM, Microsoft, Intel, Honeywell a další, vedl k řadě technologických průlomů v budování „bránových" kvantových počítačových systémů. Společnost D-Wave zaujala výrazně odlišný přístup k této nové technologii a věnuje se tzv. „kvantovému žíhání". Mnoho subjektů se zaměřuje na vývoj hardwaru pro kvantové výpočty, protože právě jeho nedokonalost je dnes hlavní překážkou k praktickému využití kvantové technologie. Opět mezi ně patří jak výše uvedení technologičtí giganti, tak relativně nové startupy jako jsou IQM, AQT, QCWare, Rigetti, IonQ, Quantum Circuits a další.

Cílem tohoto textu je seznámit čtenáře se základními principy kvantového počítání v praxi. Naleznete zde základní pojmy a vlastnosti nutné ke konstrukci důležitých algoritmů s návodem implementace v prostředí QLM od společnosti ATOS.

V tomto textu nenajdete úvody do teorie kvantové mechaniky, hluboké matematické struktury, rozsáhlé technické kapitoly o korekcích kvantových chyb, technických problémech implemetace, transpilerech a dalších. Tato a mnohá další témata lze najít například v [1]. Vhodnou alternativou tohoto textu může být [2], který nám byl častou inspirací.

Jako vhodnou prerekvizitu tento text požaduje základní znalosti lineární a tenzorové algebry, základní znalosti programovacího jazyka Python a hlavně chuť a zvědavost skloubenou s touhou proniknout do tajů kvantového počítání či informatiky.

v Ostravě 15. listopadu 2022

autoři
Ing. Jiří Tomčala, Ph.D.
a
prof. RNDr. Marek Lampart, Ph.D.

1 Úvod

Ideu kvantového počítače představil v roce 1982 laureát Nobelovy ceny Richard Feynman [3], který poukázal na to, že přesně a efektivně simulovat kvantové mechanické systémy na klasickém počítači není možné, ale že je k tomu zapotřebí nový druh stroje. Počítač, který je sám „postavený z kvantově mechanických prvků, které se řídí zákony kvantové mechaniky". Takovýto systém není možné ze své podstaty simulovat klasickým počítačem. Kvantové počítače, oproti klasickým, využívají jedinečné vlastnosti kvantových systémů (provázání, dekoherence, superpozice, kvantový paralelismus atd.), což jim umožňuje zpracovávat exponenciálně velké množství informací v polynomiálním čase.

V roce 1994 jako první sestavil Peter Shor průlomový kvantový algoritmus [4], který dal jednoznačnou odpověď na palčivou otázku, zda kvantové počítání přináší, ve srovnání s klasickým, podstatné výpočetní benefity. Tento kvantový algoritmus umožňuje faktorizaci čísla v polynomiálním čase, což poskytuje exponenciální zrychlení oproti nejrychlejším klasickým algoritmům. V kryptografii je běžně používaný algoritmus RSA, jehož bezpečnost závisí na skutečnosti, že neexistuje žádný známý účinný klasický algoritmus, který by rozložil dostatečně velké číslo (tzv. veřejný klíč) na prvočísla v reálném čase [5]. Bylo spočítáno, že pokud bychom měli k dispozici kvantový počítač s 4099 perfektně stabilními kvantovými bity, pak by šel tímto kvantovým algoritmem prolomit standardní 2048 bitový RSA klíč za 10 sekund [6]. Připomeňme, že takovýto klíč by šlo prolomit na současném běžném domácím počítači zhruba za 300 biliónů let [6] a na jednom z největších superpočítačů současnosti (Summit, IBM) teoreticky za 90 miliónů let.

Shorův kvantový algoritmus patří mezi první, není však jediný. V současné době je známo a dokonce kategorizováno velké množství kvantových algoritmů [7]. V tomto průvodci osvědčenými postupy se budeme detailně zabývat základními kvantovými algoritmy autorů E. Bernsteina, U. Vaziraniho, L. Grovera a P. Shora. Dříve, než nastíníme jejich principy a ukážeme návod implementace ve volně dostupném vývojovém prostředí Qiskit od IBM [8], formulujme nejprve základní aparát nutný ke konstrukci a pochopení zmíněných algoritmů.

2 Qubit

V klasické počítačové vědě se používá základní jednotka informace *bit*. Jedná se o abstraktní pojem. Hodnota *klasického bitu* je 0 nebo 1.

Na rozdíl od počítače klasického, počítač kvantový pracuje se stavy, které vychází ze světa kvantové fyziky [9, 10], což je důvodem, proč je kvantové počítání diametrálně odlišné od toho klasického. V případě kvantového počítání se objevují dvě neobvyklé vlastnosti, které se v klasickém případě nevyskytují, a to superpozice a provázanost. První vlastnost *superpozice* vychází z toho, že se kvantová částice ve stejné chvíli může nacházet ve stavech, které se navzájem vylučují, například nachází se „tady" i „tam" současně. Jednotky kvantové informace - kvantové bity - tedy nejsou jen 0 či 1, ale taky jejich libovolná kombinace, superpozice těchto stavů. Druhá vlastnost je příčinou extrémně vysoké efektivity kvantového počítání. *Provázanost* (entanglement) je nemožnost matematického oddělení stavů dvou různých fyzikálních objektů, například elementů kvantového počítače [11].

Kvantový bit (zkráceně *qubit* [12]) je dvoudimenzionální kvantově mechanický systém, který je ve stavu

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

kde α i β jsou komplexní čísla, což jsou *amplitudy* kvantových stavů $|0\rangle$ resp. $|1\rangle$. Druhé mocniny absolutních hodnot těchto amplitud představují pravděpodobnosti, že daný kvantový

stav bude pozorován v okamžiku měření. Pro amplitudy proto platí $|\alpha|^2 + |\beta|^2 = 1$.

V této definici se užívá standardní *bra-ketová*[1] (Diracova[2]) notace, tady označuje

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ a } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{3}$$

nejpoužívanější bázi v kvantovém počítání, takzvanou *výpočetní bázi*. Mimo výše uvedenou bázi je možno použít libovolnou ortonormální bázi, například

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ a } |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{4}$$

Geometrická interpretace kvantového stavu qubitu je možná pomocí takzvané *Blochovy sféry* zobrazené na obr. 1. Tedy qubit je možno reprezentovat pomocí dvou úhlů ϕ a θ. Přesněji, všechny lineární kombinace $\alpha\,|0\rangle + \beta\,|1\rangle$ v $\mathbb{C}^2$ odpovídají bodům (ϕ, θ) na jednotkové (Blochově) sféře. V tomto případě $\alpha = \cos(\theta/2)$ a $\beta = e^{i\phi}\sin(\theta/2)$:

$$\begin{aligned} |\psi\rangle &= \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle & (5)\\ &= \cos(\theta/2)\,|0\rangle + (\cos(\phi) + i\sin(\phi))\sin(\theta/2)\,|1\rangle\,, & (6) \end{aligned}$$

kde $0 \leq \theta \leq \pi$ a $0 \leq \phi < 2\pi$.

3 Kvantové registry

Kvantové výpočty, stejně jako ty klasické, neprobíhají na jediném qubitu, ale na *kvantovém registru*, který zahrnuje více qubitů najednou. Kvantový registr je tedy analogie klasického procesorového registru a kvantové počítače provádějí výpočty pomocí manipulací qubitů v rámci daného registru. Každý qubit v registru je superpozicí prvků výpočetní báze $|0\rangle$ a $|1\rangle$. Tedy registr s n qubity je superpozicí všech možných 2^n bitových řetězců, které mohu být reprezentovány n bity. Stavový prostor n kvantového registru je lineární kombinací n bázových vektorů délky 2^n

$$|\psi_n\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle \tag{7}$$

kde i je celé číslo v desítkové soustavě reprezentující číslo délky n ve dvojkové soustavě.

Například tří qubitový registr má tvar

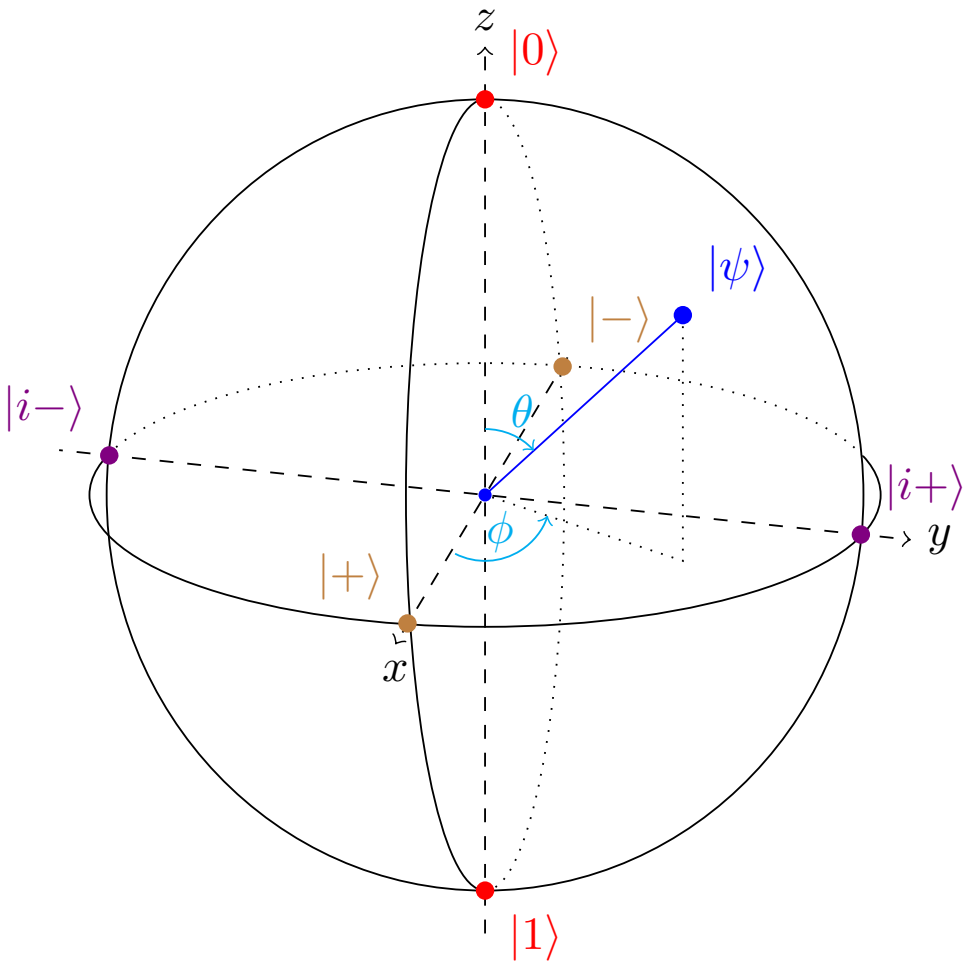
$$|\psi_3\rangle = \alpha_0\,|000\rangle + \alpha_1\,|001\rangle + \alpha_2\,|010\rangle + \alpha_3\,|011\rangle + \alpha_4\,|100\rangle + \alpha_5\,|101\rangle + \alpha_6\,|110\rangle + \alpha_7\,|111\rangle \tag{8}$$

[1] V této notaci $|u\rangle$ označuje sloupcový vektor

$$|u\rangle = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \tag{2}$$

nazývaný *ket-u*. Duální vektor $\langle u| = \overline{u^T} = (\overline{u}_1\,\overline{u}_2\,\ldots\,\overline{u}_n)$ je *bra-u*, zde $\overline{u}$ je komplexně sdružený vektor vektoru u.

[2] Jedná se o způsob zápisu vektorů, který je běžně používaný v kvantové mechanice a kvantové teorii pole. Jde o zápis vektorů v Hilbertově prostoru, který zavedl P.A.M. Dirac.



Obrázek 1: Blochova sféra

Každá bitová konfigurace v kvantové superpozici je označena jako tenzorový součin jednotlivých qubitů[3]. Například $|010\rangle$, která reprezentuje v bitovém řetězci číslo 2, má tvar:

$$|010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = (0\,0\,1\,0\,0\,0\,0\,0)^{\mathrm{T}}. \tag{10}$$

Pro (7) platí normalizační podmínka pro jednotlivé amplitudy pravděpodobností:

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \tag{11}$$

Logicky, součet pravděpodobností všech možných stavů musí být 1, protože žádný jiný stav nastat nemůže a zároveň se jednotlivé stavy navzájem vylučují. Celkově tedy, kvantové registry jsou přímým rozšířením qubitů.

4 Kvantové logické brány

Klasické logické brány jsou matematicky popsány pomocí Booleovské algebry. Kvantové logické brány fungují na podobném principu. Kvantové logické brány aplikované na kvantové registry zobrazují kvantovou superpozici na jinou a společně umožňují vývoj systému do nějakého požadovaného konečného stavu, správné odpovědi.

Kvantové logické brány jsou matematicky reprezentovány pomocí transformací matic (lineárních operací) aplikovaných na kvantový registr tenzorováním transformačních matic maticí reprezentující daný registr. Všechny matice odpovídající kvantové logické bráně jsou *unitární*[4]. Unitární transformace prováděné na jednom qubitu mohou být efektivně vizualizovány na Blochově sféře.

Tak jako v případě klasických logických bran je i v případě kvantových logických bran standardní množinou používaných bran množina zavedená v [13].

[3] Připomeňme, že $\otimes$ značí operaci *tenzorového součinu* definovaného pro dva vektory $|u\rangle$ a $|v\rangle$ dimenze m resp. n následujícím způsobem

$$|u\rangle\,|v\rangle = |uv\rangle = |u\rangle \otimes |v\rangle = \begin{pmatrix}u_1\\u_2\\\vdots\\u_m\end{pmatrix} \otimes \begin{pmatrix}v_1\\v_2\\\vdots\\v_n\end{pmatrix} = \begin{pmatrix}u_1v_1\\u_1v_2\\\vdots\\u_1v_n\\u_2v_1\\u_2v_2\\\vdots\\u_2v_n\\\vdots\\u_mv_1\\u_mv_2\\\vdots\\u_mv_n\end{pmatrix}. \tag{9}$$

Výsledný vektor $|uv\rangle$ je tedy dimenze mn. Tenzorový součin je distributivní i asociativní, ale obecně není komutativní.

[4] Komplexní matice U je *unitární* právě tehdy, když $U^{-1} = U^\dagger$, kde $U^\dagger$ je matice konjugovaná k matici U ($U^\dagger = \overline{U}^T$). Navíc platí $UU^\dagger = U^\dagger U = I$.

Nejprve uveďme *Hadamardův operátor* H

$$-\boxed{H}- = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}\langle 0| + \frac{|0\rangle - |1\rangle}{\sqrt{2}}\langle 1|$$

Aplikováním tohoto operátoru na $|0\rangle$ nebo $|1\rangle$ dostaneme superpozici stavů $|0\rangle$ a $|1\rangle$ se stejnou pravděpodobností jejich výskytu. Geometricky, zobrazeno na Blochově sféře, Hadamardův operátor provede nejprve rotaci $\pi/2$ kolem osy y a pak rotaci π kolem osy x.

Dalším příkladem jsou *Pauliho brány* X, Y a Z, které odpovídají rotaci o úhel π kolem osy x, y či z respektive. K zavedení těchto bran je zapotřebí *vnitřního* a *vnějšího součinu.* [5]

Pauliho brána X přehodí amplitudy $|0\rangle$ a $|1\rangle$

$$-\boxed{X}- = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1|$$

Pauliho brána Y přehodí amplitudy $|0\rangle$ a $|1\rangle$, násobí je komplexní jednotkou i a neguje amplitudu $|1\rangle$

$$-\boxed{Y}- = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} = i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1|$$

Pauliho brána Z neguje amplitudu $|1\rangle$ a ponechává amplitudu $|0\rangle$ beze změny

$$-\boxed{Z}- = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Brána fázového posunu R_φ nemění amplitudu $|0\rangle$, ale mění fázi $|1\rangle$ o faktor $e^{i\varphi}$ pro každou hodnotu φ

$$-\boxed{R_\varphi}- = \begin{bmatrix}1 & 0\\ 0 & e^{i\varphi}\end{bmatrix} = |0\rangle\langle 0| + e^{i\varphi}\,|1\rangle\langle 1|$$

Poznamenejme, že Z je speciální případ R_φ pro $\varphi = \pi$. Dalším speciálním případem R_φ pro $\varphi = \pi/2$ je *fázová* brána S, která mění fázi $|1\rangle$ faktorem i

$$-\boxed{S}- = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix} = |0\rangle\langle 0| + i\,|1\rangle\langle 1|$$

Jako poslední případ uveďme bránu T, je to brána R_φ pro $\varphi = \pi/4$

$$-\boxed{T}- = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix} = |0\rangle\langle 0| + e^{i\pi/4}\,|1\rangle\langle 1|$$

[5] Zde definujeme *vnitřní součin* $\langle u|v\rangle$ vektorů u a v téhož prostoru jako součin vektorů $\overline{u}^T$ a v:

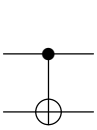
$$\langle u|v\rangle = \overline{u}^T v = (\overline{u}_1\,\overline{u}_2\,\ldots\,\overline{u}_n)\begin{pmatrix}v_1\\ v_2\\ \vdots\\ v_n\end{pmatrix} = \overline{u}_1 v_1 + \overline{u}_2 v_2 + \cdots + \overline{u}_n v_n,$$

vnější součin $|v\rangle\langle u|$ vektorů u a v prostorů dimenze m reps. n:

$$|v\rangle\langle u| = v\overline{u}^T = \begin{pmatrix}v_1\\ v_2\\ \vdots\\ v_n\end{pmatrix}(\overline{u}_1\,\overline{u}_2\,\ldots\,\overline{u}_m) = \begin{bmatrix}v_1\overline{u}_1 & v_1\overline{u}_2 & \ldots & v_1\overline{u}_m\\ v_2\overline{u}_1 & v_2\overline{u}_2 & \ldots & v_2\overline{u}_m\\ \vdots & \vdots & \ddots & \vdots\\ v_n\overline{u}_1 & v_n\overline{u}_2 & \ldots & v_n\overline{u}_m\end{bmatrix}$$

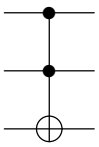
Výše uvedené brány jsou unární, což znamená, že se vždy působí jedním qubitem na danou bránu. S takovými operacemi si ale nevystačíme, potřebujeme další, binární i ternární. Mezi unární patří NOT, mezi binární AND, OR, CNOT, XOR, SWAP a ternární Toffoli, CCNOT a mnohé další viz [14].

Nejdůležitější a taky nejpoužívanější binární kvantovou branou je CNOT



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

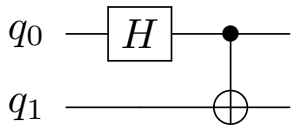
Pro úplnost ještě zaveďme nejpoužívanější ternární kvantovou Toffoliho bránu



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

5 Provázanost kvantových stavů

Jak již bylo řečeno, *provázanost* (entanglement) je nemožnost matematického oddělení dvou kvantových stavů. Co konkrétně je tím myšleno a jak se to projevuje lze nejlépe vysvětlit na příkladě jednoduchého kvantového obvodu, zobrazeného na obr. 2.



Obrázek 2: Kvantový obvod pro vytvoření provázanosti mezi qubity q_0 a q_1.

```
from qat.lang.AQASM import Program, H, CNOT
from qat.core.console import display

ko = Program()
q = ko.qalloc(2) # Alokace 2 qubitu

H(q[0])
CNOT(q[0], q[1])

# Vytvoreni spustitelne ulohy s 1000-nasobnym spustenim
# (cim vice spusteni, tim presnejsi urceni vyslednych pravdepodobnosti)
job = ko.to_circ().to_job(nbshots=1000)

from qat.qpus import get_default_qpu

qpu = get_default_qpu()

# Spusteni vytvorene ulohy
vysledky = qpu.submit(job)

for vysledek in vysledky:
    print("Stav: %s  Pravdepodobnost: %s" % (vysledek.state,vysledek.probability))
```

```
23
24 # Zobrazeni kvantoveho obvodu v "ASCII art" stylu
25 display(ko.to_circ())
26
27 # Pro ulozeni zobrazeni kvantoveho obvodu do souboru v prostredi "jupyter notebook", jako
28 # je na obrazku, je zapotrebi misto predchoziho prikazu pouzit tyto dva nasledujici:
29 # circuit = ko.to_circ()
30 # %qatdisplay circuit --file ko-Provazani.pdf
```

Zdrojový kód 1: Implementace kvantového obvodu na simulátoru ATOS QLM.

Výstup spuštění zdrojového kódu 1 na hardwarovém simulátoru QLM30 (*kvasi.csc.fi*). Možno použít i volně dostupný softwarový simulátor myQLM:

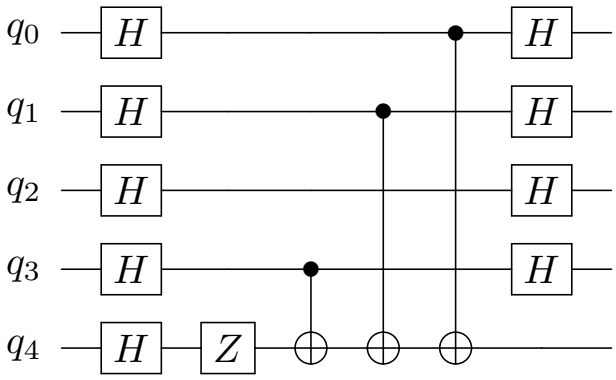
```
Stav: |00>  Pravdepodobnost: 0.498
Stav: |11>  Pravdepodobnost: 0.502
```

6 Bernstein-Vaziraniho algoritmus

Tento algoritmus je v podstatě variantou Deutsch-Jozsa algoritmu. Jeho implementací je dokonce stejný kvantový obvod.

Zatímco Deutsch-Jozsa algoritmus je určen pro rozhodnutí, zda je nějaká neznámá funkce konstantní nebo balancovaná, cílem Bernstein-Vaziraniho algoritmu je odhalit tajný kód v této neznámé funkci nastavený.

Tato neznámá funkce je implementovaná v tzv. černé skříňce (black box), jejíž vnitřní zapojení není známé. Taková černá skříňka se v odborné literatuře označuje taky někdy jako *orákulum (oracle)*. Ve schématech kvantových obvodů se ovšem toto orákulum označuje jako U_f a v kvantovém obvodě na obr. 3 je implementováno pomocí tří bran CNOT.



Obrázek 3: Kvantový obvod implementující Bernstein-Vaziraniho algoritmus. Tajný kód je 1101.

```
1  from qat.lang.AQASM import Program, H, X, Z, CNOT
2  from qat.core.console import display
3
4  n = 4 # pocet qubitu tajneho kodu
5  tajnyKod = "1101"
6
7  ko = Program()
8  q = ko.qalloc(n+1)
9
10 # Nastaveni pomocneho qubitu do stavu |->
11 H(q[n])
12 Z(q[n])
13
14 # Nastaveni vsech vstupu Orakula do stavu |+>
15 for i in range (n):
```



```
    H(q[i])

# Vytvoreni Orakula obsahujici tajny kod
for i in range (n):
    if tajnyKod[i] == "1":
        CNOT(q[i],q[n])

# Pouziti Hadamardovych bran na vystup z Orakula, cimz dojde k odhaleni
# tajneho kodu
for i in range(n):
    H(q[i])


job = ko.to_circ().to_job(nbshots=1000,qubits=list(range(n)))

from qat.qpus import get_default_qpu

qpu = get_default_qpu()

vysledky = qpu.submit(job)

for vysledek in vysledky:
    print("Stav: %s  Pravdepodobnost: %s" % (vysledek.state,vysledek.probability))

# Zobrazeni kvantoveho obvodu v "ASCII art" stylu
display(ko.to_circ())

# Pro ulozeni zobrazeni kvantoveho obvodu do souboru v prostredi "jupyter notebook", jako
# je na obrazku, je zapotrebi misto predchoziho prikazu pouzit tyto dva nasledujici:
# circuit = ko.to_circ()
# %qatdisplay circuit --file ko-Bernstein-Vazirani.pdf
```

Zdrojový kód 2: Implementace kvantového obvodu na simulátoru ATOS QLM.

Výstup spuštění zdrojového kódu 2 na hardwarovém simulátoru QLM30 (*kvasi.csc.fi*). Možno použít i volně dostupný softwarový simulátor myQLM:

```
Stav: |1101>  Pravdepodobnost: 1.0
```

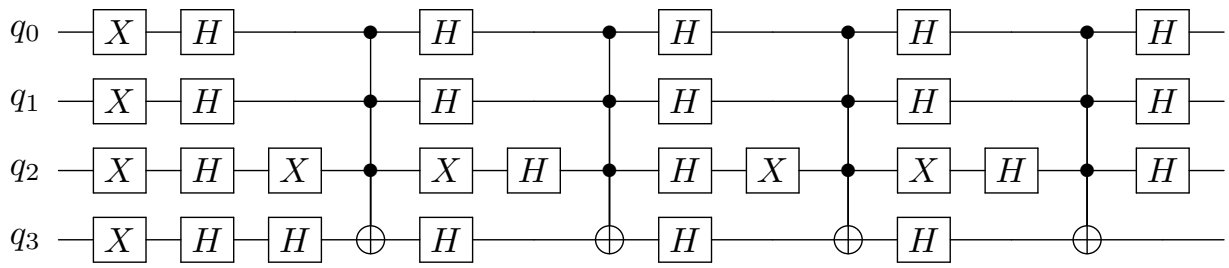
7 Groverův prohledávací algoritmus

Tento algoritmus je vhodný pro vyhledávání v tzv. nestrukturovaných datech. Při takovém vyhledávání může Groverův algoritmus dosáhnout až kvadratického zrychlení. Je to díky tzv. triku se zesílením amplitudy.

O co se jedná? Velmi stručně řečeno je zde opět zapotřebí sestavit orákulum, které pro hledanou vstupní kombinaci otočí fázi výstupního kvantového stavu. Za ním pak následuje tzv. *difuzér*, který zesílí amplitudu kvantového stavu s obrácenou fází na úkor amplitud ostatních kvantových stavů a zároveň tomuto zesílenému kvantovému stavu s obrácenou fází opět tuto fázi otočí. Na výstupu difuzéru je tak u hledané kombinace vyšší pravděpodobnost než u ostatních možných vstupních kombinací.

Tento postup (orákulum a difuzér) lze opět zopakovat, čímž dojde k ještě většímu zvýšení pravděpodobnosti výskytu hledané kombinace na výstupu a zároveň snížení pravděpodobností ostatních možných kombinací. Obecně je zapotřebí provést přibližně $\sqrt{N}$ průchodů těmito dvěma obvody (orákulum a difuzér), aby byla dosažena maximální možná pravděpodobnost hledané kombinace, kde N je počet všech možných vstupních kombinací.

Při klasickém hledání v nestrukturovaných datech je zapotřebí až $N - 1$ pokusů, aby byla hledaná vstupní kombinace nalezena, zatímco Groverův algoritmus k tomu samému potřebuje maximálně $\sqrt{N}$ pokusů. Proto tento algoritmus může dosáhnout kvadratického zrychlení a v některých případech dokonce i vyššího, protože již po několika průchodech orákulem a difuzérem může být jasné, u které kombinace se pravděpodobnost postupně zvyšuje.



Obrázek 4: Kvantový obvod implementující Groverův algoritmus. Hledaná vstupní kombinace je 1101.

```
from qat.lang.AQASM import Program, H, X
from qat.core.console import display

n = 4 # pocet qubitu tajneho kodu
hledanaVstupniKombinace = "1101"

ko = Program()
q = ko.qalloc(n)

# Nastaveni vsech vstupu Orakula do stavu |->
for i in range(n):
    X(q[i])
    H(q[i])

for j in range(2):  # Pro nejlepsi vysledek je zapotrebi dvakrat provest
                    # pruchod orakulem a difuzerem
    # Orakulum
    for i in range(n):
        if hledanaVstupniKombinace[i] == '0':
            X(q[i])
    H(q[n-1])
    X.ctrl().ctrl().ctrl()(q[0],q[1],q[2],q[3])
    H(q[n-1])
    for i in range(n):
        if hledanaVstupniKombinace[i] == '0':
            X(q[i])

    # Difuzer
    for i in range(n-1):
        H(q[i])
    X.ctrl().ctrl().ctrl()(q[0],q[1],q[2],q[3])
    for i in range(n-1):
        H(q[i])


job = ko.to_circ().to_job(nbshots=1000)

from qat.qpus import get_default_qpu

qpu = get_default_qpu()

vysledky = qpu.submit(job)

for vysledek in vysledky:
    print("Stav: %s  Pravdepodobnost: %s " % (vysledek.state,vysledek.probability))

# Zobrazeni kvantoveho obvodu v "ASCII art" stylu
display(ko.to_circ())

# Pro ulozeni zobrazeni kvantoveho obvodu do souboru v prostredi "jupyter notebook", jako
# je na obrazku, je zapotrebi misto predchoziho prikazu pouzit tyto dva nasledujici:
# circuit = ko.to_circ()
# %qatdisplay circuit --file ko-Grover.pdf
```

Zdrojový kód 3: Implementace kvantového obvodu na simulátoru ATOS QLM.

Výstup spuštění zdrojového kódu 3 na hardwarovém simulátoru QLM30 (*kvasi.csc.fi*). Možno

použít i volně dostupný softwarový simulátor myQLM:

```
Stav: |1101>  Pravdepodobnost: 0.921
Stav: |1100>  Pravdepodobnost: 0.006
Stav: |0010>  Pravdepodobnost: 0.005
Stav: |1011>  Pravdepodobnost: 0.005
Stav: |0110>  Pravdepodobnost: 0.006
Stav: |0101>  Pravdepodobnost: 0.004
Stav: |0001>  Pravdepodobnost: 0.007
Stav: |0100>  Pravdepodobnost: 0.004
Stav: |1000>  Pravdepodobnost: 0.008
Stav: |0000>  Pravdepodobnost: 0.009
Stav: |1010>  Pravdepodobnost: 0.008
Stav: |0011>  Pravdepodobnost: 0.003
Stav: |1111>  Pravdepodobnost: 0.004
Stav: |0111>  Pravdepodobnost: 0.003
Stav: |1001>  Pravdepodobnost: 0.004
Stav: |1110>  Pravdepodobnost: 0.003
```

8 Shorův faktorizační algoritmus

Shorův algoritmus [4] lze použít pro faktorizaci celých čísel v polynomiálním čase. Vzhledem k tomu, že nejlepší klasické algoritmy vyžadují pro tuto faktorizaci superpolynomiální čas, nejrozšířenější kryptovací systém (RSA) je založen na tom, že faktorizace dostatečně velkých celých čísel je v podstatě nemožná. V současnosti je doporučovaným standardem použití RSA s šířkou šifrovacího klíče 2048 bitů, jehož faktorizace by na dnešním běžném PC trvalo přibližně 3×10^{14} let. Na jednom z největších superpočítačů současnosti (Summit, IBM) by pak taková faktorizace trvala teoreticky 90 miliónů let.

Podrobné vysvětlení principu Shorova algoritmu je nad rámec této publikace, nicméně pro představu je vhodné zmínit alespoň zhruba jeho základní postup. Hlavním úkolem kvantového obvodu je najít periodu r funkce $f(x) = a^x \bmod N$, kde a je vhodně zvolené celé číslo a N je faktorizané číslo. Nalezená perioda r musí být sudá. Pokud tomu tak není, je zapotřebí zvolit jinou hodnotu a a opět najít periodu funkce $f(x)$. Jakmile je nalezena sudá perioda r, pak lze faktory jednoduše vypočítat jako největší společný dělitel čísel $(a^{r/2} + 1)$ a N a čísel $(a^{r/2} - 1)$ a N. Podrobnosti lze nalézt v [15].

Kvantový obvod je tedy zapotřebí pouze pro nalezení periody funkce $f(x)$. Přípravu vhodné hodnoty a a vypočtení hledaných faktorů z periody r pak může zajistit obsluhující klasický počítač.

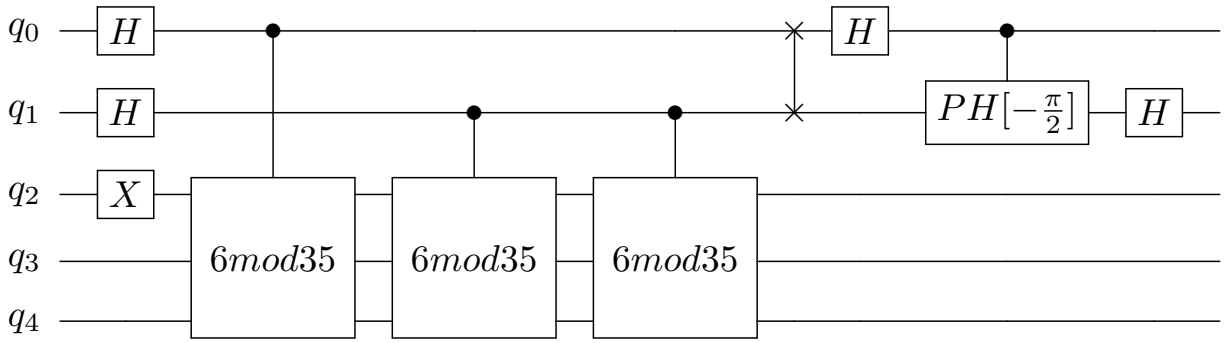
Jednoduchý příklad kvantového obvodu pro nalezení periody funkce $f(x)$ pro faktorizaci čísla $N = 35$ je zobrazen na obr. 5. Jako vhodné číslo bylo zvoleno $a = 6$, takže funkce $f(x) = 6^x \bmod 35$.

```python
from qat.lang.AQASM import Program, AbstractGate, QRoutine, H, X, SWAP, PH
from qat.core.console import display
import numpy as np

n = 2 # pocet qubitu, na ktere bude aplikovana funkce 6^x mod 35
m = 3 # pocet qubitu pro funkci 6^x mod 35

# Nasobeni (vstup x 6) mod 35
def ko_fx():
    qr = QRoutine()
    w = qr.new_wires(3)
    SWAP(w[0],w[2])
    X(w[1])
    return qr

ko_fx_gate = AbstractGate("x6mod35",[],arity=3,circuit_generator=ko_fx)
```



Obrázek 5: Kvantový obvod pro nalezení periody funkce $f(x) = 6^x \bmod 35$.

```
17
18 ko = Program()
19 q = ko.qalloc(n+m)
20
21 # Nastaveni qo a q1 do stavu |+ >
22 for i in range (n):
23     H(q[i])
24
25 # Nastaveni vstupu funkce 6^x mod 35 na |001>
26 X(q[n])
27
28 # Aplikace funkce 6^x mod 35 na superponovane q0 a q1
29 for i in range(n):
30     for j in range(2**i):
31         ko_fx_gate().ctrl()(q[i],q[n:(n+m)])
32
33 # Inverzni kvantova Fourierova transformace
34 for i in range(n//2):
35     SWAP(q[i], q[n-i-1])
36 for j in range(n):
37     for m in range(j):
38         PH(-np.pi/float(2**(j-m))).ctrl()(q[m],q[j])
39     H(q[j])
40
41
42 job = ko.to_circ().to_job(nbshots=1000,qubits=list(reversed(range(n))))
43
44 from qat.qpus import get_default_qpu
45
46 qpu = get_default_qpu()
47
48 vysledky = qpu.submit(job)
49
50 for vysledek in vysledky:
51     print("Stav: %s  Pravdepodobnost: %s" % (vysledek.state,vysledek.probability))
52
53 # Zobrazeni kvantoveho obvodu v "ASCII art" stylu
54 display(ko.to_circ())
55
56 # Pro ulozeni zobrazeni kvantoveho obvodu do souboru v prostredi "jupyter notebook", jako
57 # je na obrazku, je zapotrebi misto predchoziho prikazu pouzit tyto dva nasledujici:
58 # circuit = ko.to_circ()
59 # %qatdisplay circuit --file ko-Shor.pdf
```

Zdrojový kód 4: Implementace kvantového obvodu na simulátoru ATOS QLM.

Výstup spuštění zdrojového kódu 4 na hardwarovém simulátoru QLM30 (*kvasi.csc.fi*). Možno použít i volně dostupný softwarový simulátor myQLM:

```
Stav: |00>  Pravdepodobnost: 0.486
Stav: |10>  Pravdepodobnost: 0.514
```

Reference

[1] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Quantum computation and quantum information. 2000.

[2] Emma Strubell. An introduction to quantum algorithms. 2011.

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

[4] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994.

[5] R. L. Rivest, A. Shamir, and L. M. Adleman. *A method for obtaining digital signatures and public key cryptosystems*, pages 217–239. Secure Communications and Asymmetric Cryptosystems. 2019.

[6] Andreas Baumhof. Breaking rsa encryption – an update on the state-of-the-art, 2019. Last access: April 19, 2022.

[7] Stephen Jordan. Quantum algorithm zoo, 2021. Last updated: February 1, 2021.

[8] MD SAJID ANIS et all. Qiskit: An open-source framework for quantum computing, 2021.

[9] John Polkinghorne; přeložil Pavel Cejnar. *Kvantová teorie - průvodce pro každého*. Praha: Dokořán. 2007.

[10] George Johnson; přeložili Jiří Podolský a Pavel Cejnar. *Zkratka napříč časem - cesta ke kvantovému počítači*. Praha: Argo. 2004.

[11] Pavel Cejnar. https://ipnp.cz/cejnar/publikace/qcomp.htm. Last access: April 19, 2022.

[12] Benjamin Schumacher. Quantum coding. *Phys. Rev. A*, 51:2738–2747, Apr 1995.

[13] A. Barenco, C. H. Bennett, R. Cleve, D. P. Divincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995.

[14] Alexei Y. Kitaev, Alexander Shen, and Mikhail N. Vyalyi. Classical and quantum computation. In *Graduate studies in mathematics*, 2002.

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

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, the United Kingdom, France, the Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, the Republic of North Macedonia, Iceland, and Montenegro. This project has received funding from the Ministry of Education, Youth and Sports of the Czech Republic (ID:MC2101).

