

Memóriafelhasználás optimalizálása

kvantumalgoritmusok szimulációja során

Nemkin Viktória

dr. Friedl Katalin
Számítástudományi és Információelméleti Tanszék



Motiváció: "Protein folding" megoldása kvantumalgoritmussal

- **Protein:**

- ▶ Aminosavakból alkotott lánc.
 - ★ **Piros = Hidrofób ("vízkerülő").**
 - ★ **Kék = Poláris ("vízszerető").**
- ▶ Hajtogatás: 3D kockarácson elhelyezés.
- ▶ Cél: Minimális energiájú elhelyezés megtalálása.

- **Kódolás:**

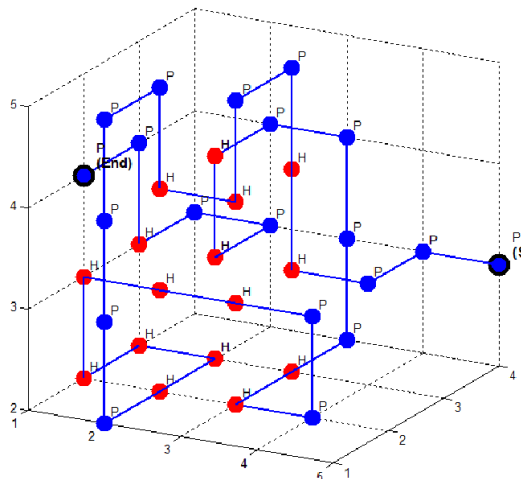
- ▶ Origóból, lépésenként 6 irány = 6 "qubit".

- **Orákulum:**

- ▶ Energiaviszonyok lepontozása.

- **Grover keresés:** $O(\sqrt{N})$

- ▶ "Kvantum párhuzamosság"-ot kihasználva.
- ▶ Energiaminimum megtalálása.



Egy összehajtogatott protein.

Cél

- Algoritmus kipróbálása és elemzése.
- Eszköztár: Regiszterek + Hadamard, Grover, Sum, MC-NOT
- Kvantumszámítógép: Publikusan nem elérhető (elég nagy).
- Klasszikus szimuláció: Túl sok memóriát fogyaszt.

Probléma: Memóriahasználat

Lánc	Qubitek	Regiszter	Operátor
n	$6(n - 1)$	$2^{6(n-1)} \cdot 16B$	$2^{12(n-1)} \cdot 16B$

Probléma: Memóriahasználat

Lánc	Qubitek	Regiszter	Operátor
n	$6(n-1)$	$2^{6(n-1)} \cdot 16B$	$2^{12(n-1)} \cdot 16B$
2	6	1 KB	64 KB
3	12	64 KB	256 MB
4	18	4 MB	1 TB
5	24	256 MB	4 PB
6	30	16 GB	16384 PB
7	36	1 TB	67108864 PB

Probléma: Memóriahasználat

Lánc	Qubitek	Regiszter	Operátor
n	$6(n-1)$	$2^{6(n-1)} \cdot 16B$	$2^{12(n-1)} \cdot 16B$
2	6	1 KB	64 KB
3	12	64 KB	256 MB
4	18	4 MB	1 TB
5	24	256 MB	4 PB
6	30	16 GB	16384 PB
7	36	1 TB	67108864 PB

- Optimalizációk (regiszter és operátor esetében is):

- ▶ Ritka mátrixos tárolás:
 - ★ IBM Qiskit, Google Cirq, stb.
 - ★ Nem mindig ritkák a mátrixok.
- ▶ Döntési fa alapú adatszerkezet:
 - ★ Újabb kutatási irány.
 - ★ Nem mindig segít.

Megoldás: "On-the-fly" és "Függvény-alapú" operátorok

Lánc	Qubitek	Regiszter	Operátor	"On-the-fly"	"Függvény-alapú"
n	$6(n-1)$	$2^{6(n-1)} \cdot 16B$	$2^{12(n-1)} \cdot 16B$	= Regiszter.	Nem kell tárolni.
2	6	1 KB	64 KB	1 KB	0 B
3	12	64 KB	256 MB	64 KB	0 B
4	18	4 MB	1 TB	4 MB	0 B
5	24	256 MB	4 PB	256 MB	0 B
6	30	16 GB	16384 PB	16 GB	0 B
7	36	1 TB	67108864 PB	1 TB	0 B

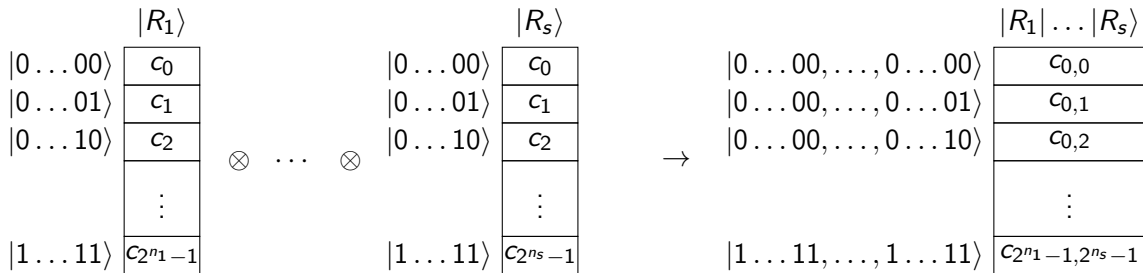
Megoldás: "On-the-fly" és "Függvény-alapú" operátorok

Lánc	Qubitek	Regiszter	Operátor	"On-the-fly"	"Függvény-alapú"
n	$6(n-1)$	$2^{6(n-1)} \cdot 16B$	$2^{12(n-1)} \cdot 16B$	= Regiszter.	Nem kell tárolni.
2	6	1 KB	64 KB	1 KB	0 B
3	12	64 KB	256 MB	64 KB	0 B
4	18	4 MB	1 TB	4 MB	0 B
5	24	256 MB	4 PB	256 MB	0 B
6	30	16 GB	16384 PB	16 GB	0 B
7	36	1 TB	67108864 PB	1 TB	0 B

- Qiskit, stb. nyílt forráskódúak...
- ...de szerves része a kódnak az operátor tárolása.
- → saját szimulátor implementáció.

Kvantumregiszterek állapotának tárolása

- Minden bitsorozathoz egy komplex valószínűségi amplitúdó.
- Szuperpozíció és összefonódás \rightarrow indexek Descartes-szorzata.
- TODO: $2^{n_1}, \dots, 2^{n_s} \rightarrow \prod_{i=1}^s 2^{n_i}$.



Operátor végrehajtás általánosan

TODO:

- Első két qubitra.
- Utolsó két qubitra.
- Köztes esetek: ebből van sok.

Mappelés

TODO: zavaros nyilas ábra.

QUBIT MAP:

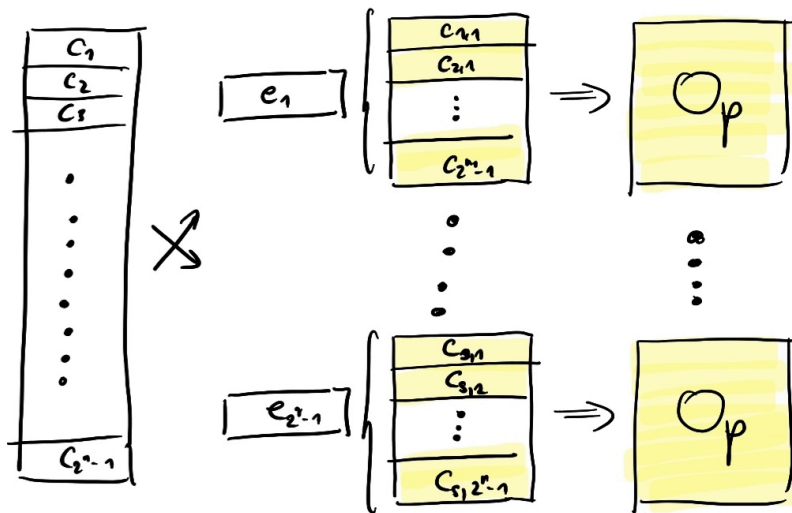
$$|R_1, R_2, R_3, R_4, \dots, R_{s-1}, R_s\rangle \Rightarrow |R_1, R_4, \dots, R_{s-1}, R_2, R_3, \dots, R_s\rangle$$

INDEX MAP:

$$\begin{array}{ccc} |0..0, 0..0, 0..0, 0..0, \dots, 0..0, 0..0\rangle & & |0..0, 0..0, \dots, 0..0, 0..0, 0..0, \dots, 0..0\rangle \\ \vdots & \times & \vdots \\ |1..1, 1..1, 1..1, 1..1, \dots, 1..1, 1..1\rangle & & |1..1, 1..1, \dots, 1..1, 1..1, 1..1, \dots, 1..1\rangle \end{array}$$

Végrehajtás (párhuzamosíthatóan)

Tipikus eset.



Operátorok megvalósítása

- Visitor minta (inverzió):
 - ▶ Operátornak odaadom a regisztert, végrehajtja magát rajta.
 - ▶ Belső működés eltakarva: mátrix reprezentáció nem szükséges.
- Strategy minta:
 - ▶ Egységes algoritmus interfész: több lehetséges implementáció.
- Operátor csak egy handle, példány nem foglal memóriát.

Megvalósítás:

- "On-the-fly" operátorok:
 - ▶ A mátrix egy sorát generálok.
 - ▶ Hadamard, Grover.
- "Függvény-alapú" operátorok:
 - ▶ $u : |0 \dots 0, in\rangle \rightarrow |out, in\rangle$
 - ▶ Sum: $\sum : |0 \dots 0, in\rangle \rightarrow |count(in), in\rangle$
 - ▶ MC-NOT: $mcnot : |0, in\rangle \rightarrow |any(in), in\rangle$

- Eredmények:
 - ▶ Ritka mátrixosan tárolt regiszterek.
 - ▶ Regiszterkezelés tetszőleges célregiszterekkel.
 - ▶ "On-the-fly" operátorok: Hadamard, Grover.
 - ▶ "Függvény-alapú" operátorok: Sum, Multi-Controlled NOT.
 - ▶ Könnyű bővíthetőség új operátorral.
- Jövőbeli terv:
 - ▶ Protein folding algoritmus implementálása.
 - ▶ Döntési fa alapú tárolás kipróbálása.

Bíráló kérdései - 1.

A dolgozat bevezetésében írja, hogy az egyik motivációt a munkához a bioinformatika adja, ezen belül is a fehérje feltekeredés (protein folding) vizsgálata.

A dolgozatban azonban - érthető okokból - egy egyszerűbb problémával, a Sudoku általános változatával dolgozik.

Mégis, meg tudná mondani, hogyan lehetne (milyen jellegű továbbfejlesztésekkel) a kidolgozott eljárást a bioinformatikában használni?

Bíráló kérdései - 2.

Mi a szerző várakozása a kifejlesztett keretrendszer működési korlátait illetően?

Például:

Hány qubites Grover-keresést fog tudni implementálni? Mekkora n esetén tudja implementálni a Sudoku verifiert?

A maximális méretű problémát implementáló kód mennyi idő alatt fog lefutni egy laptopon?

Bíráló kérdései - 3.

A keretrendszer struktúrájában vagy implementációjában meg kell-e különböztetni a CPU-n és a GPU-n való futtatás esetét?

Bíráló kérdései - 4.

Valódi kvantumszámítógépekben a kvantumbitek és a környezet kölcsönhatása dekoherenciához vezet.

A Grover-keresés működőképes marad-e, ha a dekoherenciát figyelembe vesszük a szimuláció során?

Lehetséges-e a dolgozatban kifejlesztett hatékony keretrendszert általánosítani dekoherencia jelenléte esetére, és ha igen, akkor meg lehet-e ezt tenni úgy hogy a hatékonyság is megmarad?