Instytut Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk
44-100 Gliwice, ul. Bałtycka 5, http://www.iitis.gliwice.pl

Pakiet `quantum-octave` do symulacji obliczeń kwantowych

Jarosław Miszczak
Zespół Kwantowych Systemów Informatyki

27 września 2004 roku

IITiS

Plan prezentacji

- Dlaczego warto symulować maszyny kwantowe?
- Cel stworzenia pakietu `quantum-octave`
- Omówienie pakietu
- Przykład zastosowania
- Podsumowanie i wnioski

1. Potrzeba symulacji komputerów kwantowych

Jedną z własności pomiaru w mechanice kwantowej jest jego destruktywność. W związku z tym nie jest możliwe obserwowanie stanu układu kwantowego podczas wykonywania algorytmu bądź protokołu kwantowego. Symulacja układów kwantowych daje jedyną możliwość takiej obserwacji. Wykonując symulację możemy przeanalizować krok po kroku przebieg zmian następujących w układzie kwantowym.

IITiS

2. Cel stworzenia pakietu `quantum-octave`

Pakiet `quantum-octave` daje możliwość wykorzystania języka GNU Octave do symulacji układów kwantowych. Pozwala w łatwy sposób zaimplementować algorytmy i protokoły kwantowe. Cecha, która wyróżnia pakiet spośród dostępnych symulatorów, to możliwość operowania na stanach mieszanych. Zastosowanie modelu stanów mieszanych pozwala na analizę algorytmów i protokołów kwantowych pod względem odporności na błędy oraz badanie takich zjawisk jak splątanie i dekoherencja.

3. Możliwości pakietu `quantum-octave`

3.1. Modelowanie i symulacja

- konstruowanie bramek złożonych z bramek elementarnych
- przygotowanie stanów czystych i mieszanek stanów
- ewolucja unitarna i nieunitarna oraz pomiar

3.2. Analiza stanów

- operacje śladu częściowego i transpozycji częściowej, miary splątania, wierności oraz odległości między rozkładami prawdopodobieństw
- możliwość graficznej obserwacji amplitud stanów czystych oraz rozkładów prawdopodobieństwa stanów mieszanych

4. Przykład wykorzystania – symulacja algorytmu Grovera

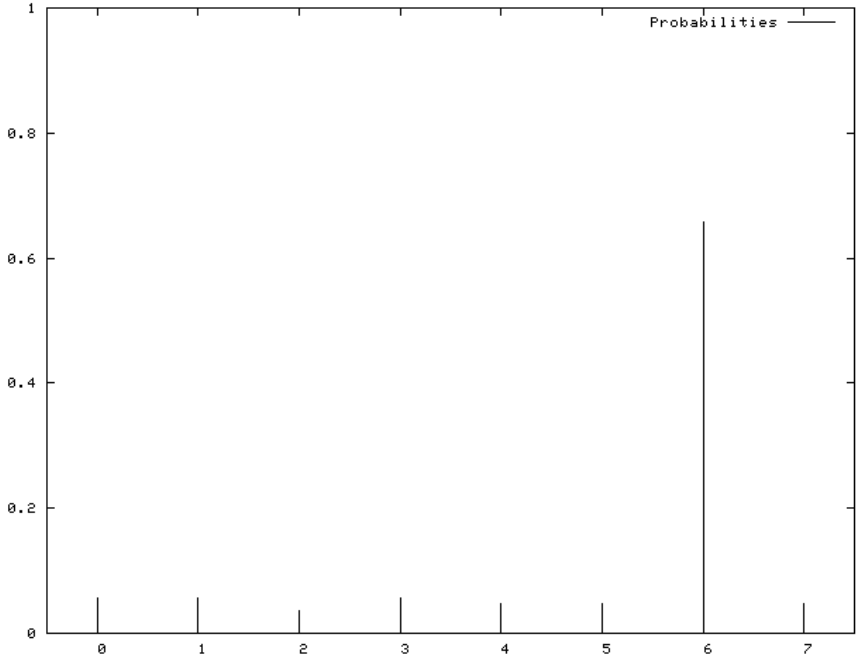
```
function ret = grover( num, state )
# initial data
qubits = log2(size(state)(1));
tvec = [1:qubits];
k = ceil((pi/8)*sqrt(size(state)(1)));
# global Hadamard transformation
bigh = ProductGate(qubits,H,tvec);
# Grover iterations
ret = Evolve(bigh,state);
for i = 1:k
ret = Evolve(oracle(num, qubits),ret);
ret = Evolve(diffuse(qubits),ret);
endfor
endfunction
```

```
function ret = oracle (num, qubits)
# check if num<2^qubits
ret =  eye (2^qubits);
ret(num+1,num+1) = -1;
endfunction

function ret = diffuse(qubits)
tvec = [1:qubits];
tmp = zeros(1,qubits);
ret = ProductGate(qubits,H,tvec);
ret = ret*(2*Projection(tmp) - Id(qubits));
ret = ret*ProductGate(qubits,H,tvec);
endfunction
```

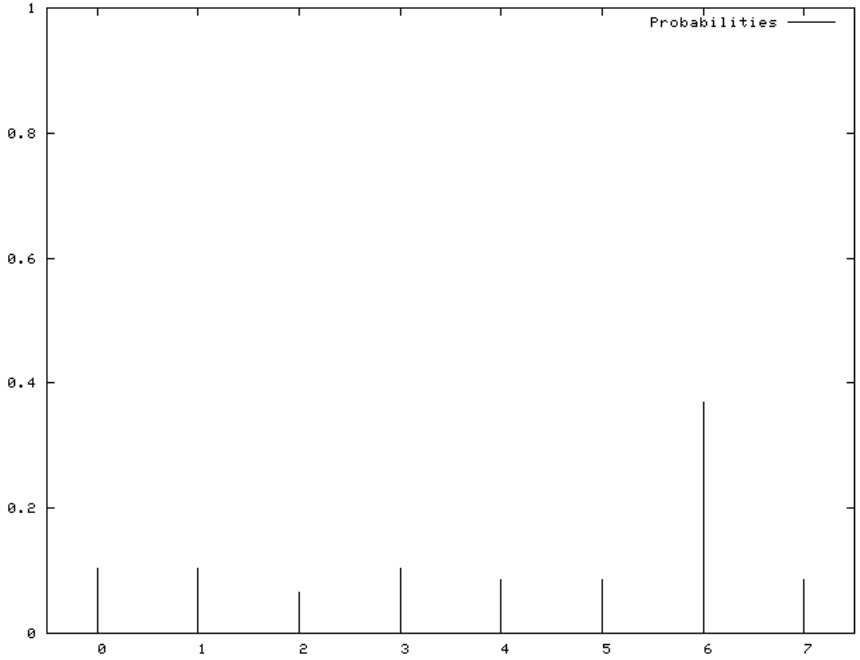
4.1. Rozkład prawdopodobieństwa w przypadku „prawie” czystego stanu

```
PlotProbs(grover(6,sm1));
```



4.2. Rozkład prawdopodobieństwa w przypadku stanu silnie mieszanego

```
PlotProbs(grover(6,sm2));
```



5. Dalsze prace zespołu

- optymalizacja symulacji poprzez zastosowanie macierzy rzadkich
- analiza wpływu szumów na teleportację kwantową
- modelowanie gier kwantowych

IITiS

Dziękuję za uwagę