US 20060224547A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2006/0224547 A1**

Ulyanov et al. (43) **Pub. Date: Oct. 5, 2006**

(54) **EFFICIENT SIMULATION SYSTEM OF QUANTUM ALGORITHM GATES ON CLASSICAL COMPUTER BASED ON FAST ALGORITHM**

(76) Inventors: **Sergey V. Ulyanov**, Polo Didattico E Di Recerca Di Crema (DE); **Sergey A. Panfilov**, Polo Didattico E Di Recerca Di Crema (DE)

Correspondence Address:
KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614 (US)

(21) Appl. No.: **11/089,421**

(22) Filed: **Mar. 24, 2005**

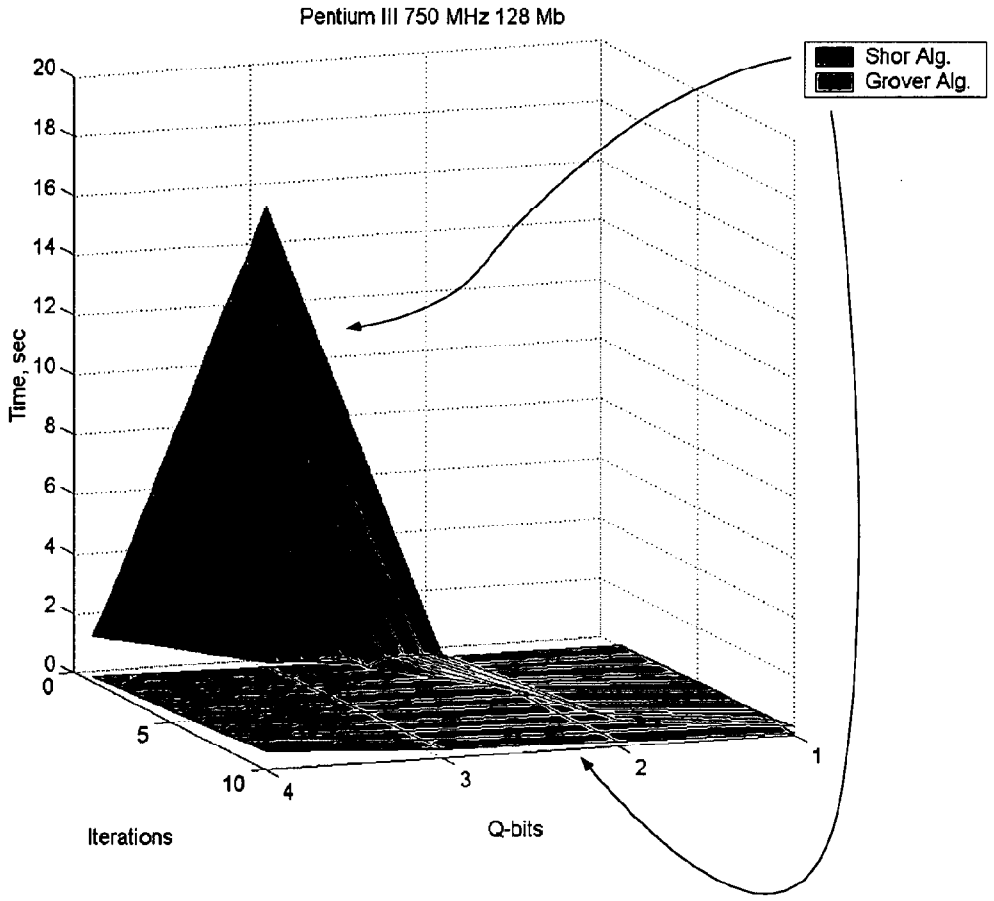
Publication Classification

(51) **Int. Cl.**
G06F 15/18 (2006.01)

(52) **U.S. Cl.** **706/62**

(57) **ABSTRACT**

An efficient simulation system of quantum algorithm gates for classical computers with a Von Neumann architecture is described. In one embodiment, a Quantum Algorithm is solved using an algorithmic-based approach, wherein matrix elements of the quantum gate are calculated on demand. In one embodiment, a problem-oriented approach to implementing Grover's algorithm is provided with a termination condition determined by observation of Shannon minimum entropy. In one embodiment, a Quantum Control Algorithm is solved by using a reduced number of quantum operations.



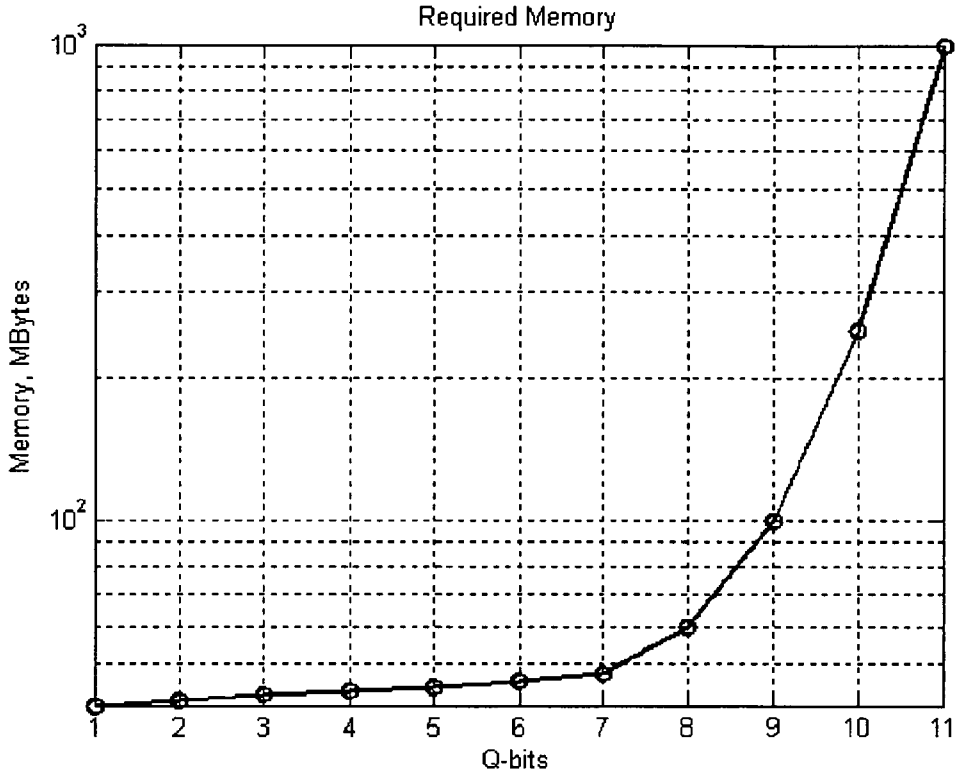

Required Memory
10^3
10^2
Memory, MBytes
1
2
3
4
5
6
7
8
9
10
11
Q-bits

Figure 1

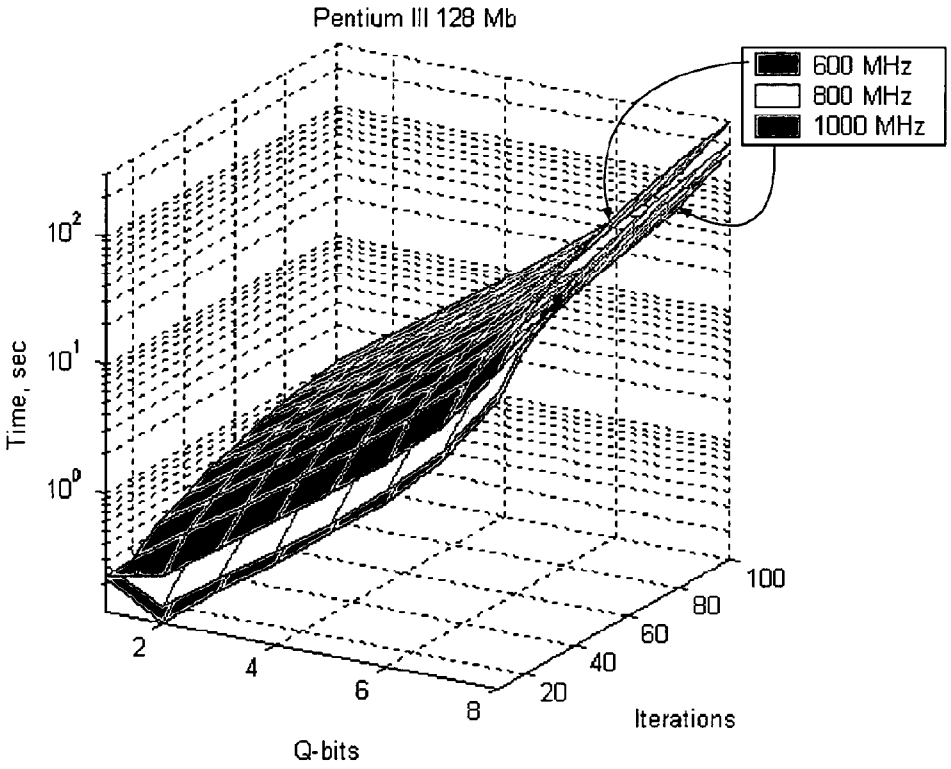

Pentium III 128 Mb
600 MHz
800 MHz
1000 MHz
Time, sec
10^2
10^1
10^0
2
4
6
8
Q-bits
20
40
60
80
100
Iterations

Figure 2

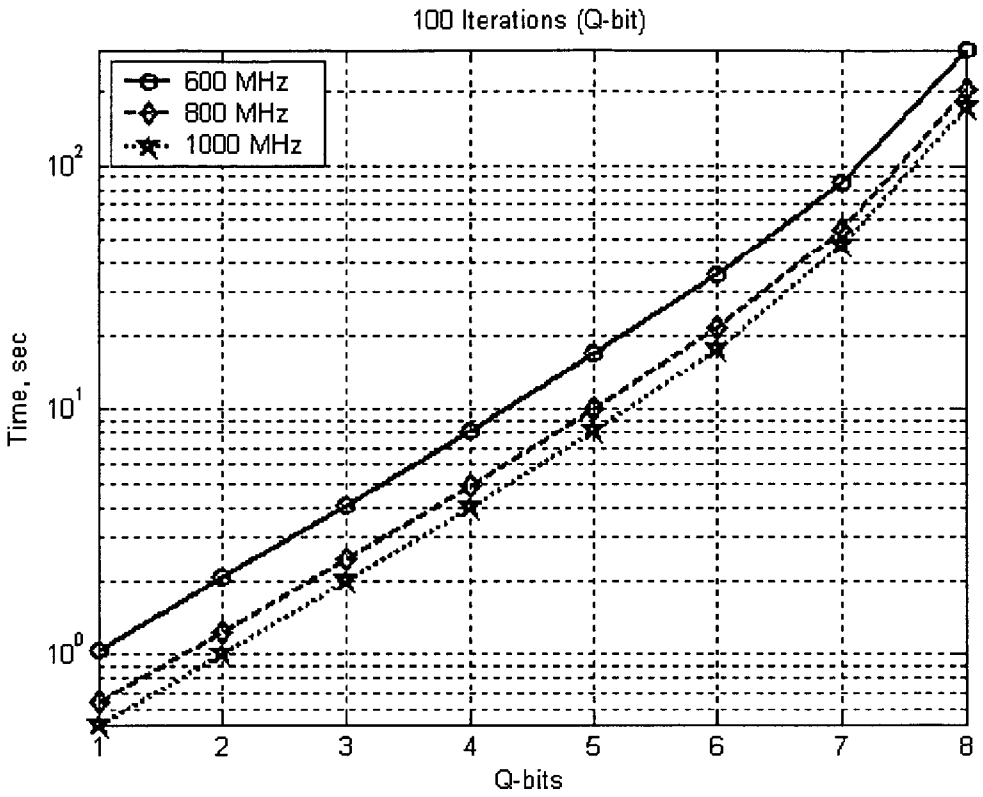

100 Iterations (Q-bit)
600 MHz
800 MHz
1000 MHz
Time, sec
10^2
10^1
10^0
1
2
3
4
5
6
7
8
Q-bits

Figure 3

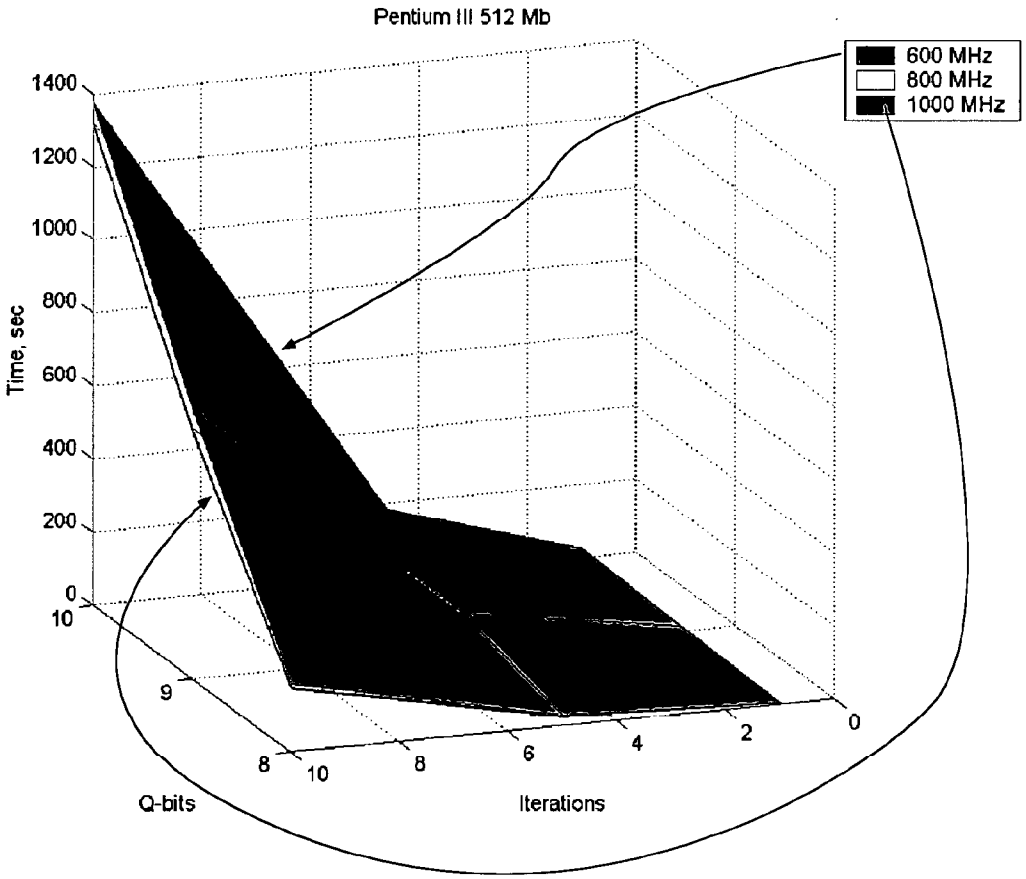

Pentium III 512 Mb
600 MHz
800 MHz
1000 MHz
Time, sec
1400
1200
1000
800
600
400
200
0
10
9
8
Q-bits
10
8
6
4
2
0
Iterations

Figure 4A

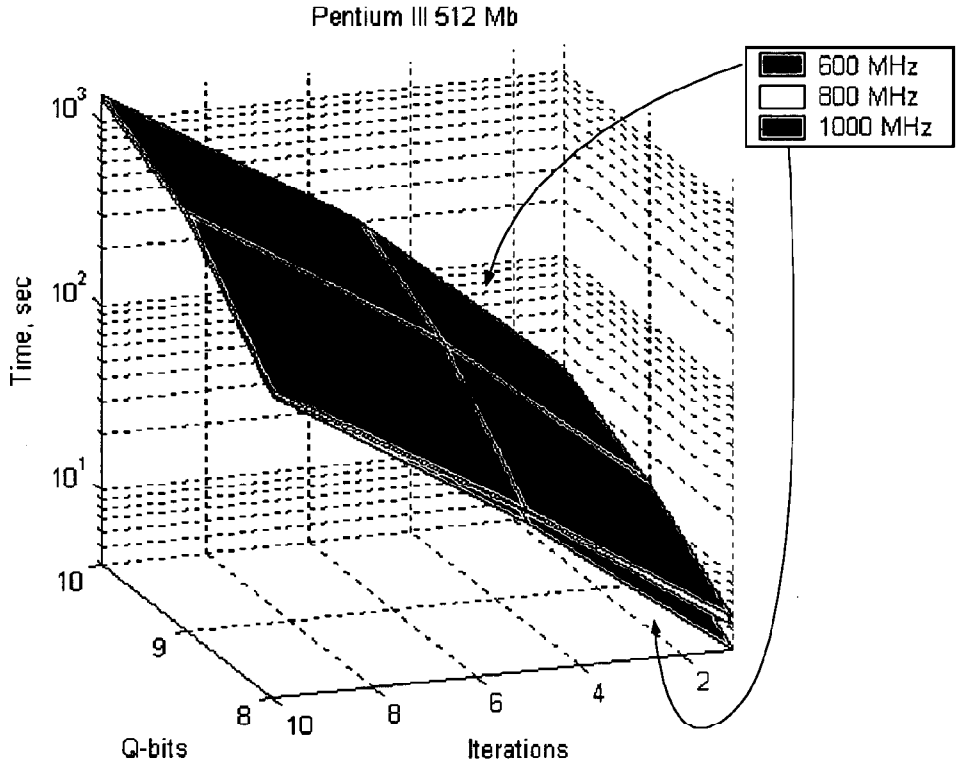

Pentium III 512 Mb
600 MHz
800 MHz
1000 MHz
Time, sec
10^3
10^2
10^1
10
Q-bits
9
8
10
8
6
4
2
Iterations

Figure 4B

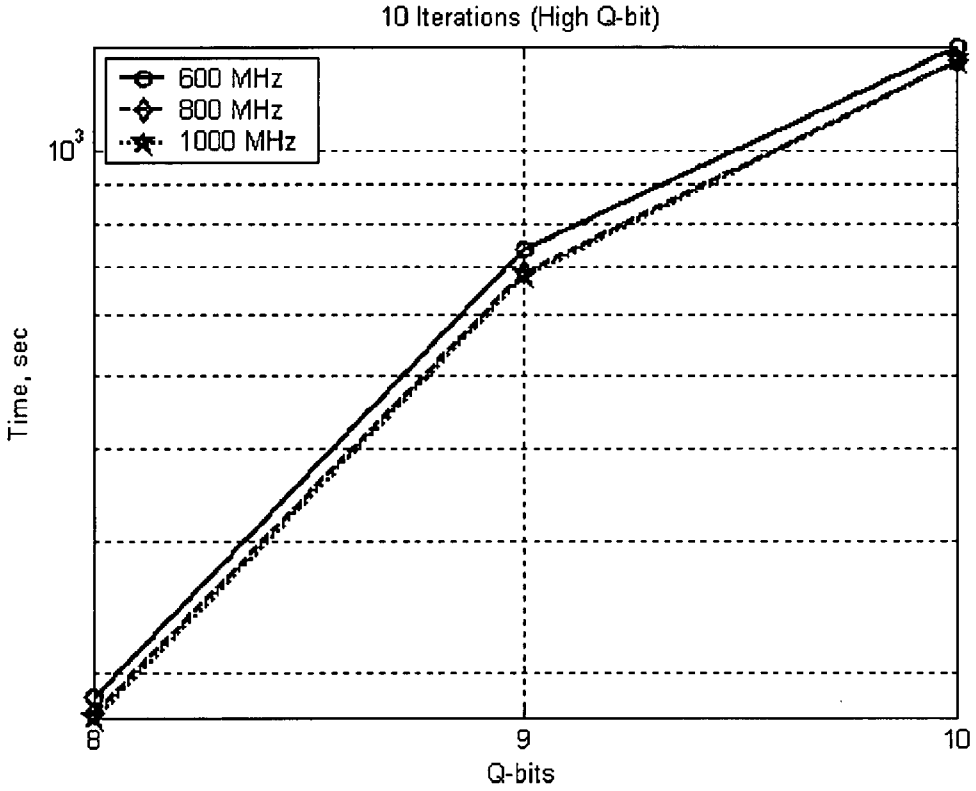

10 Iterations (High Q-bit)
600 MHz
800 MHz
1000 MHz
10^3
Time, sec
8
9
10
Q-bits

Figure 5

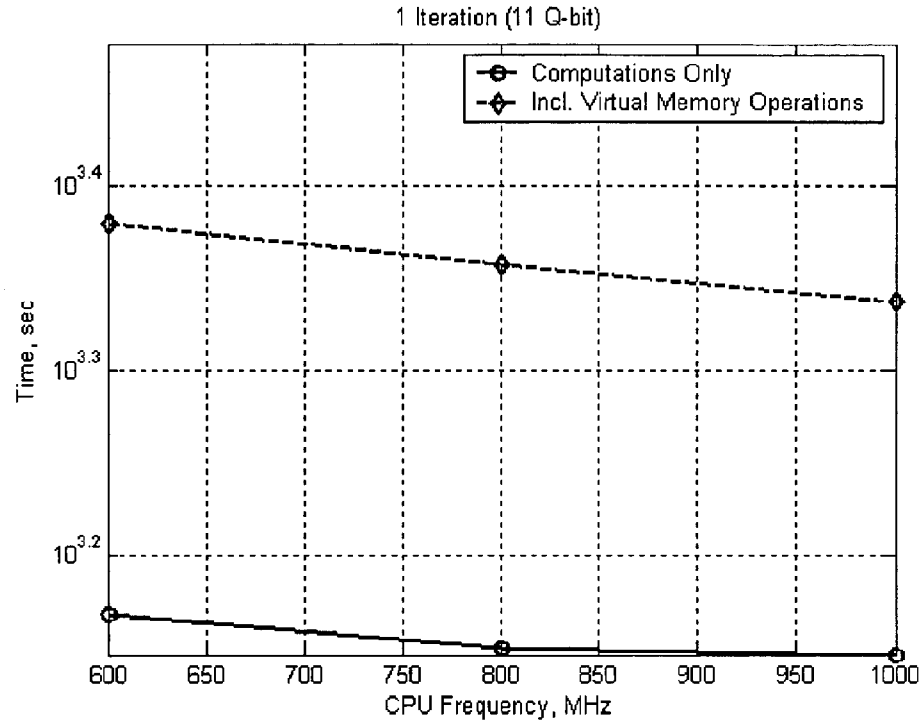

1 Iteration (11 Q-bit)
Computations Only
Incl. Virtual Memory Operations
$10^{3.4}$
$10^{3.3}$
$10^{3.2}$
Time, sec
600
650
700
750
800
850
900
950
1000
CPU Frequency, MHz

Figure 6

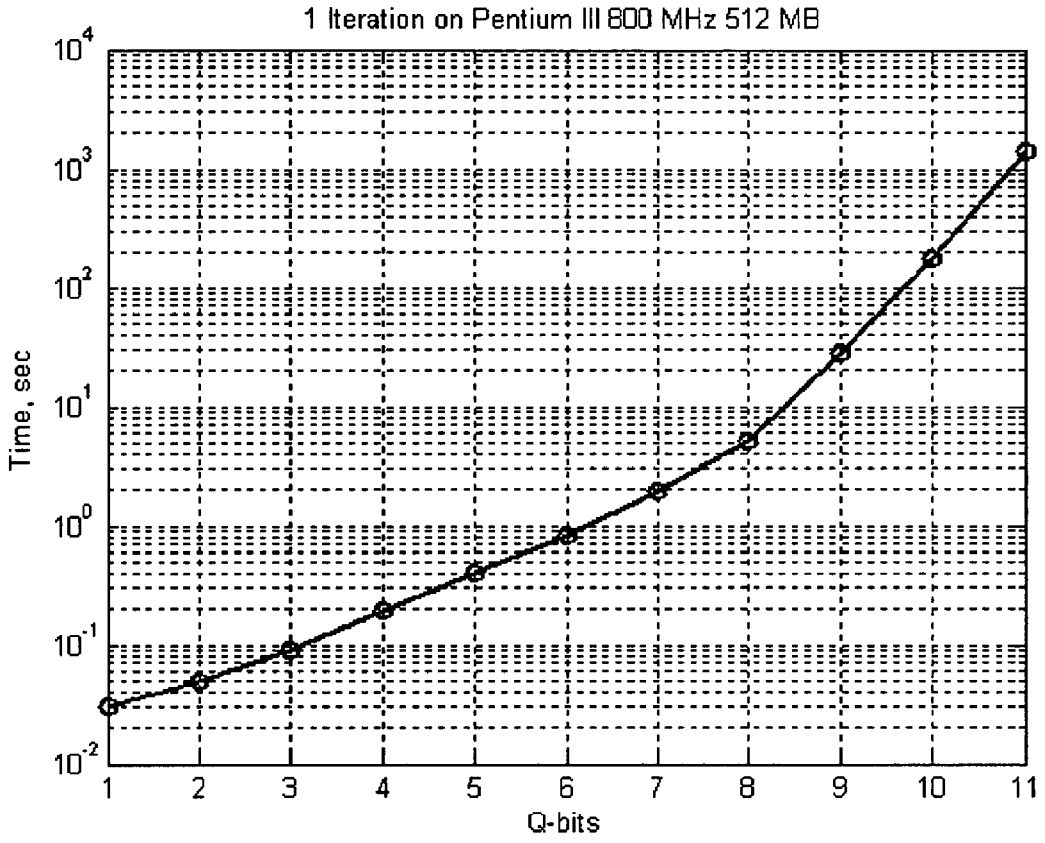

1 Iteration on Pentium III 800 MHz 512 MB
Time, sec
Q-bits
10^4
10^3
10^2
10^1
10^0
10^-1
10^-2
1
2
3
4
5
6
7
8
9
10
11

Figure 7

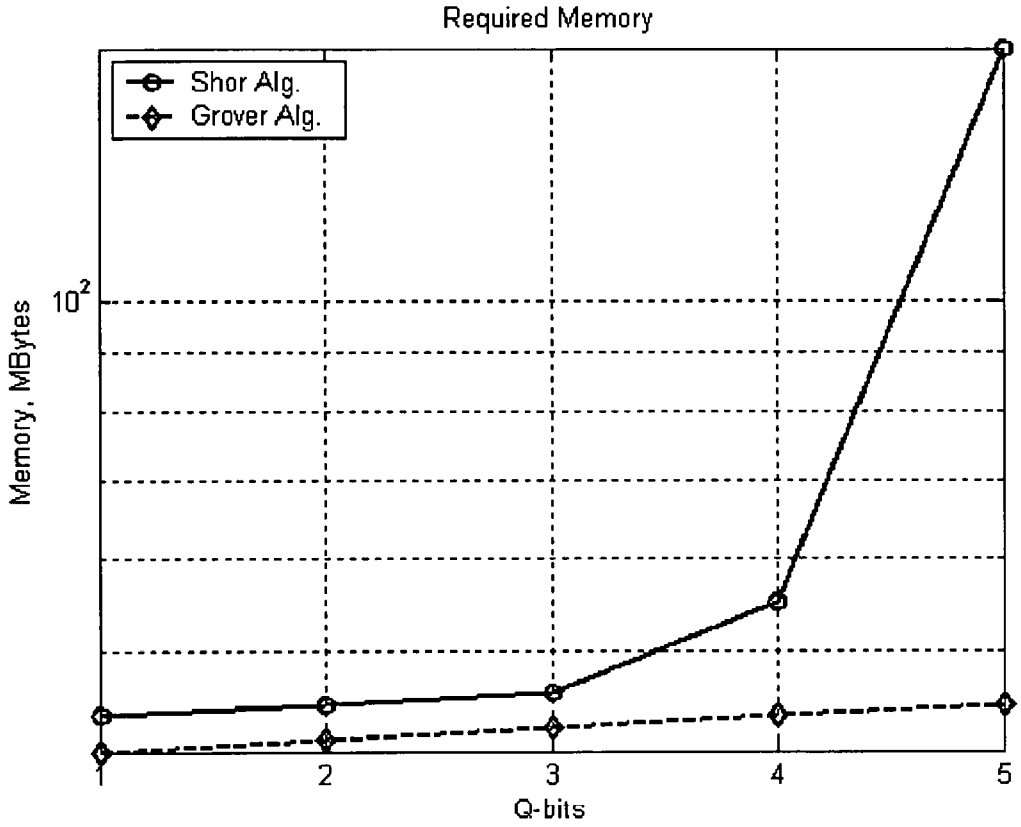

Required Memory
Shor Alg.
Grover Alg.
Memory, MBytes
10^2
1
2
3
4
5
Q-bits

Figure 8

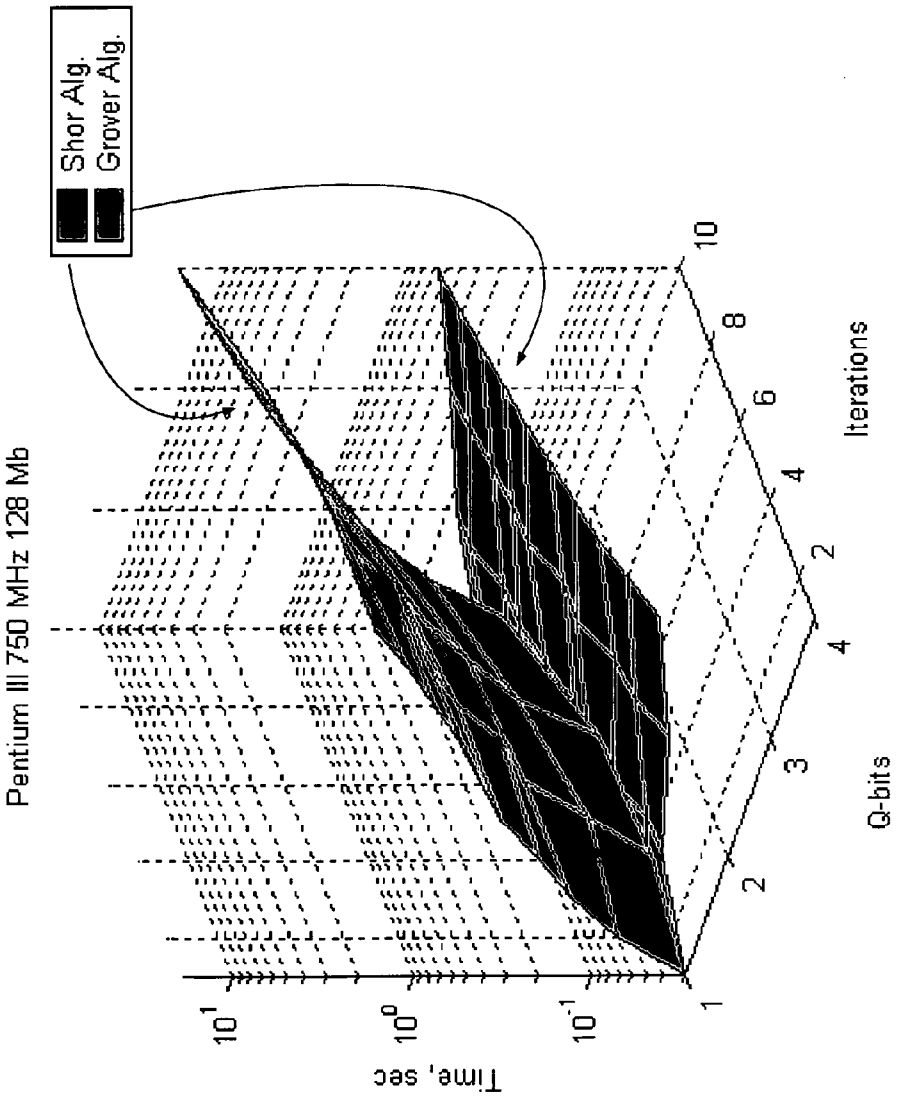

Shor Alg.
Grover Alg.
Pentium III 750 MHz 128 Mb
Time, sec
10^{1}
10^{0}
10^{-1}
Q-bits
1
2
3
4
Iterations
2
4
6
8
10

Figure 9

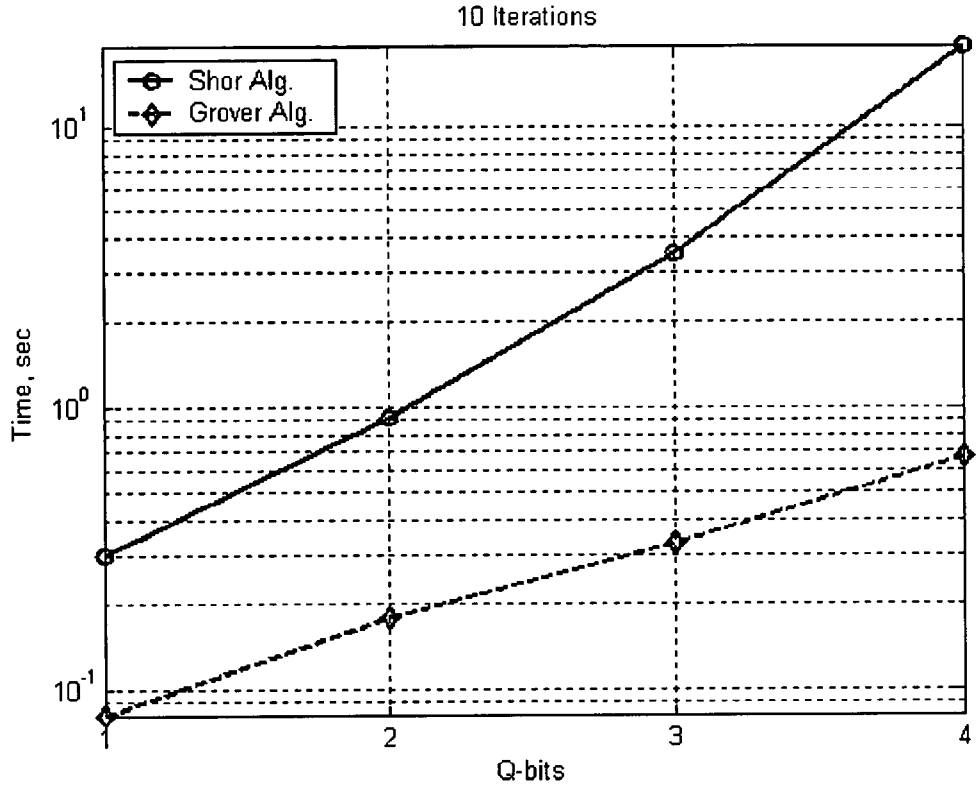

10 Iterations
Shor Alg.
Grover Alg.
Time, sec
10^1
10^0
10^{-1}
1
2
3
4
Q-bits

Figure 10

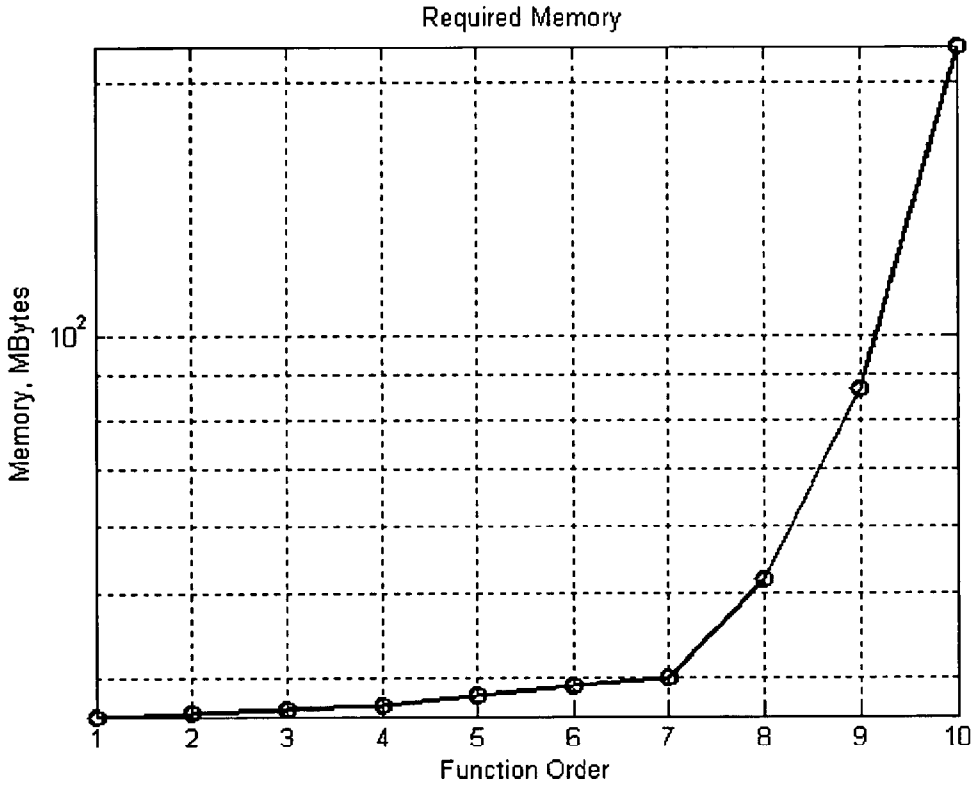

Required Memory
Memory, MBytes
10^2
1
2
3
4
5
6
7
8
9
10
Function Order

Figure 11

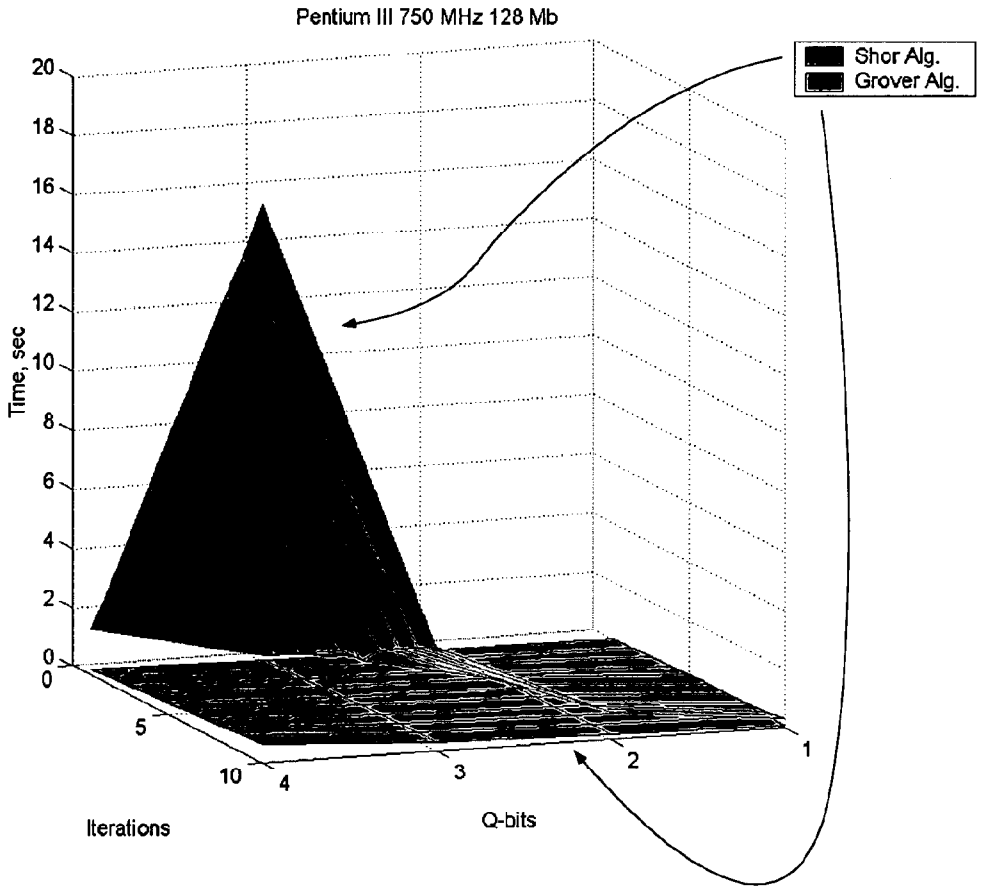

Pentium III 750 MHz 128 Mb
Shor Alg.
Grover Alg.
Time, sec
20
18
16
14
12
10
8
6
4
2
0
0
5
10
Iterations
4
3
2
1
Q-bits

Figure 12

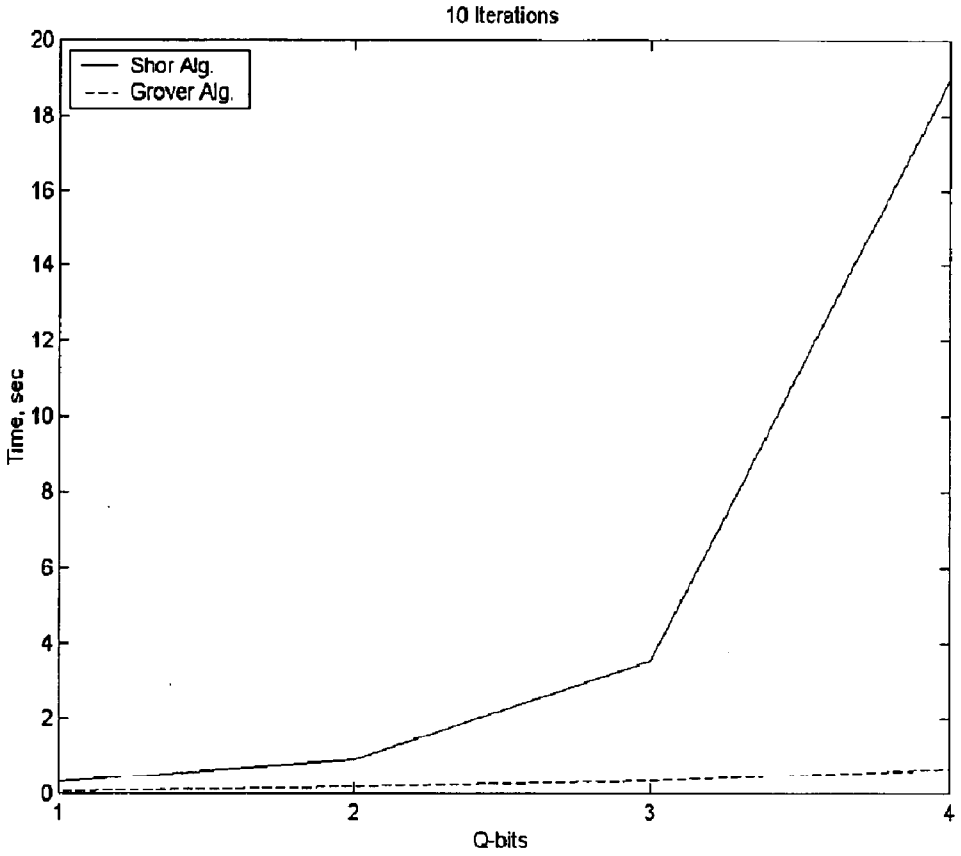

10 Iterations
Shor Alg.
Grover Alg.
Time, sec
Q-bits
0
2
4
6
8
10
12
14
16
18
20
1
2
3
4

Figure 13

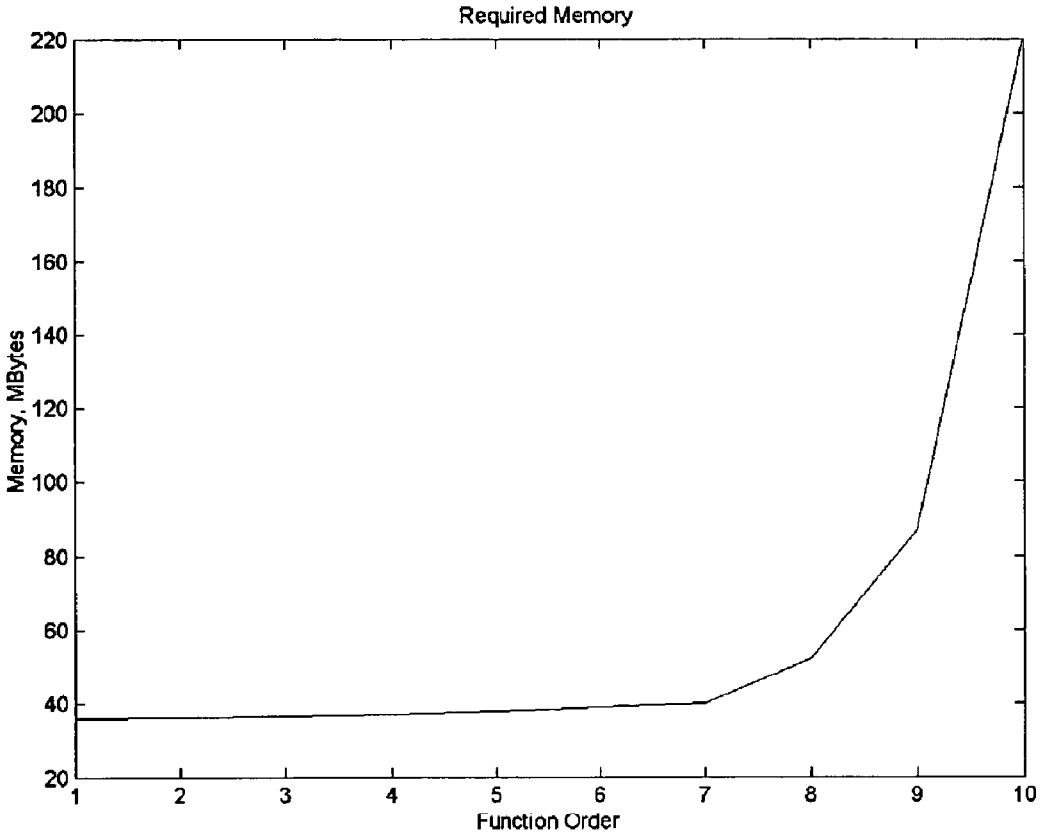

Required Memory
Memory, MBytes
220
200
180
160
140
120
100
80
60
40
20
1
2
3
4
5
6
7
8
9
10
Function Order

Figure 14

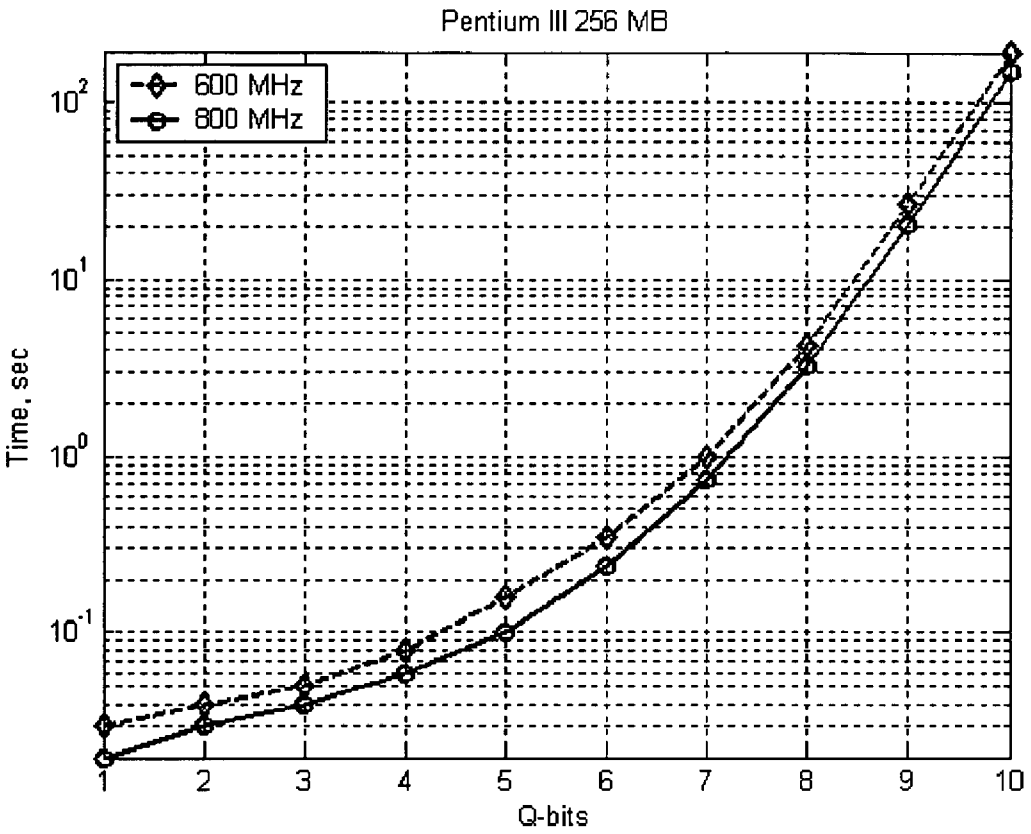

Pentium III 256 MB
600 MHz
800 MHz
Time, sec
10^2
10^1
10^0
10^{-1}
1
2
3
4
5
6
7
8
9
10
Q-bits

Figure 15

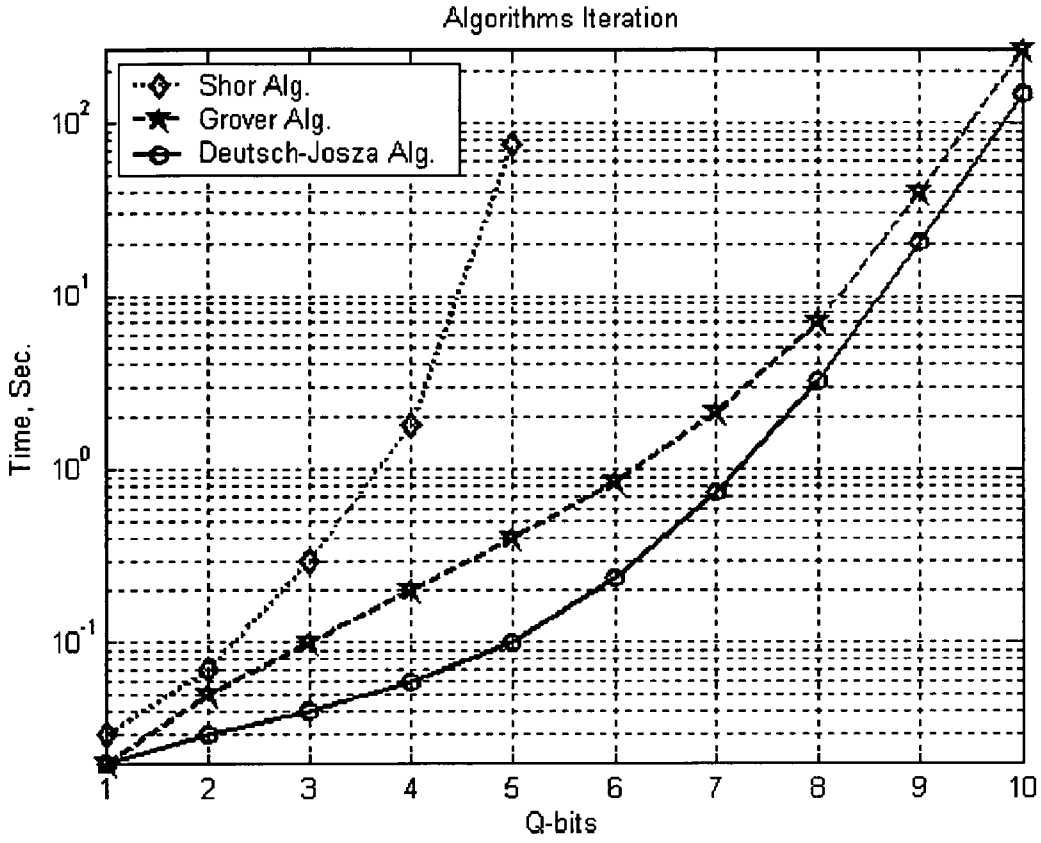

Algorithms Iteration
Shor Alg.
Grover Alg.
Deutsch-Josza Alg.
Time, Sec.
10^2
10^1
10^0
10^{-1}
1
2
3
4
5
6
7
8
9
10
Q-bits

Figure 16

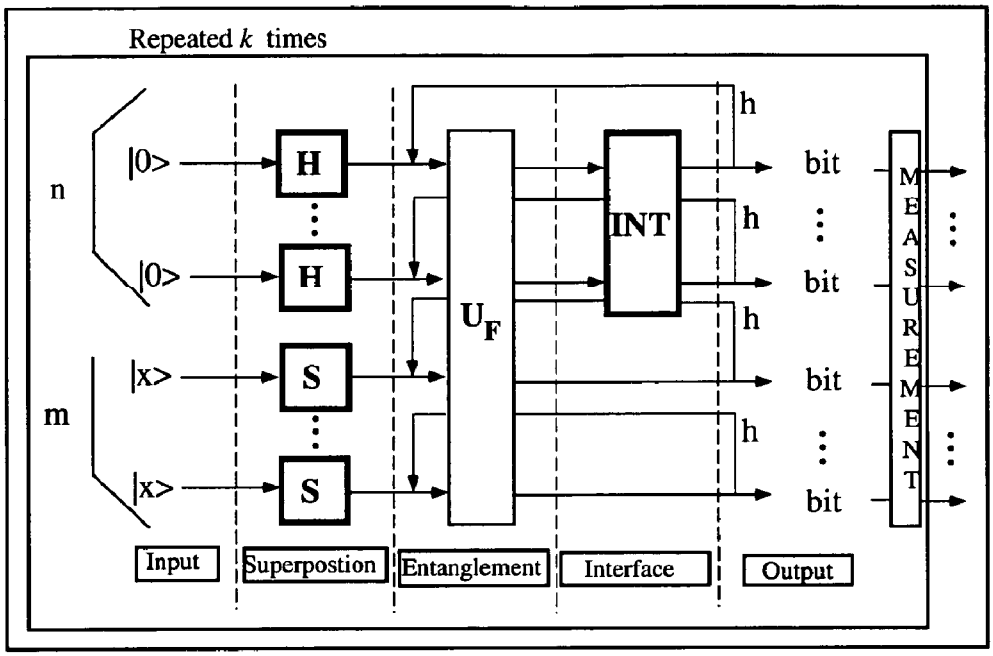


Figure 17a

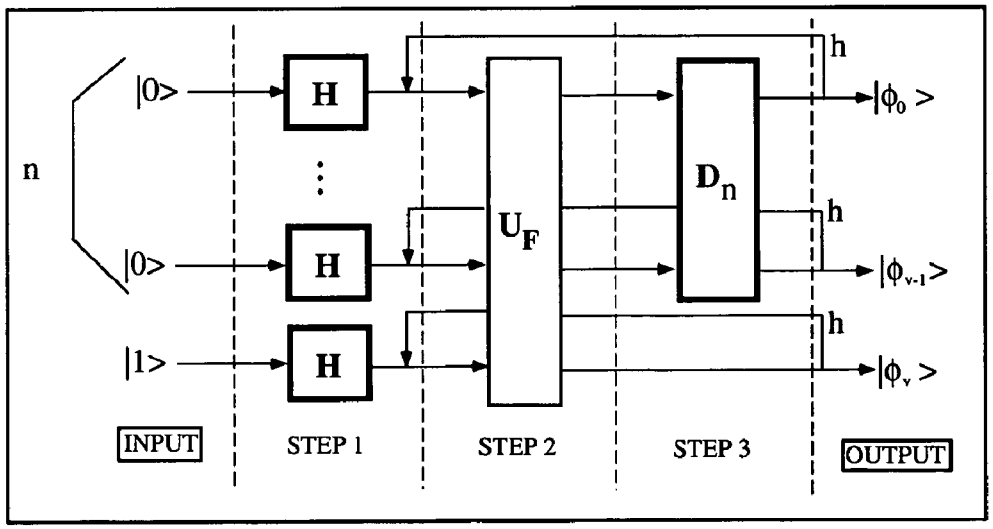


Figure 17b

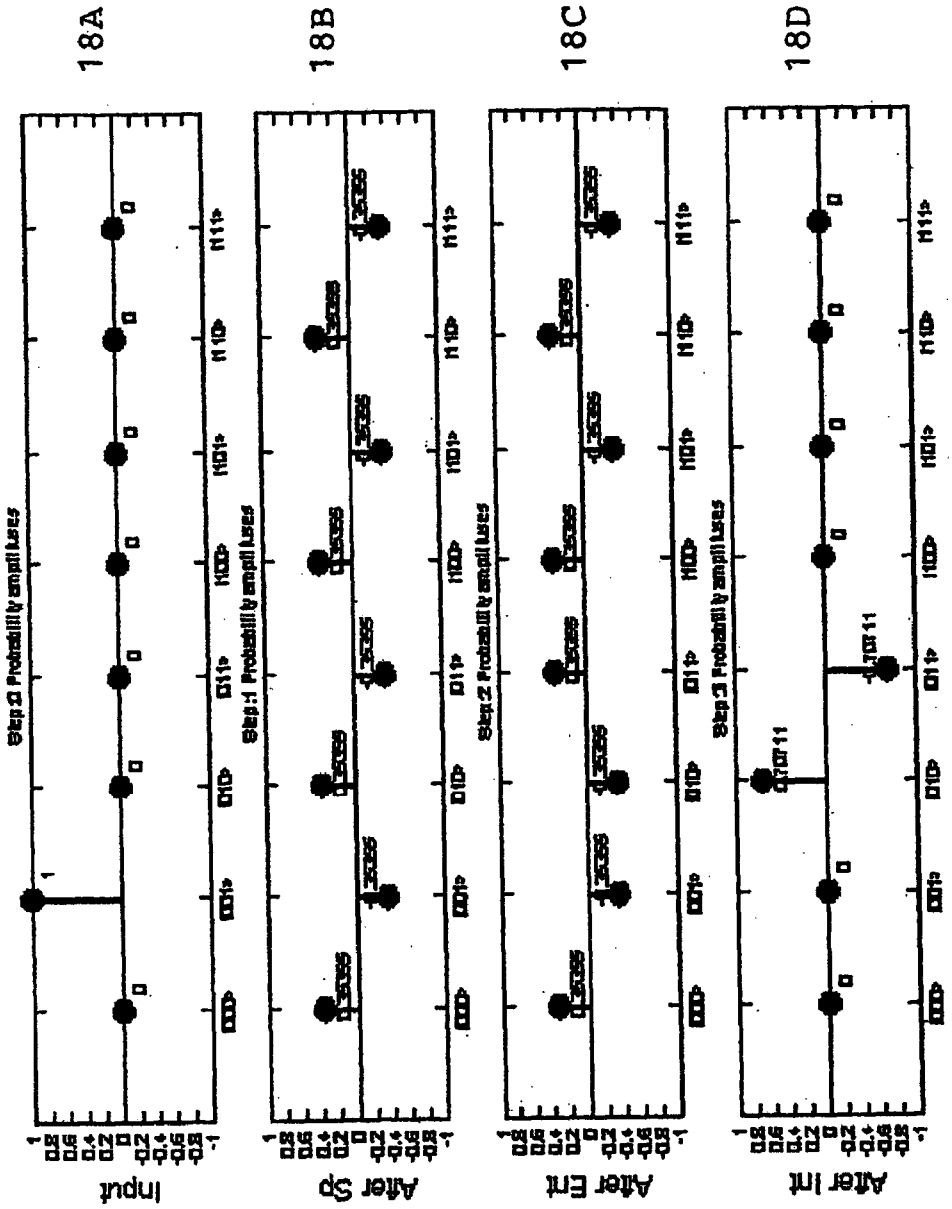


Figure 18

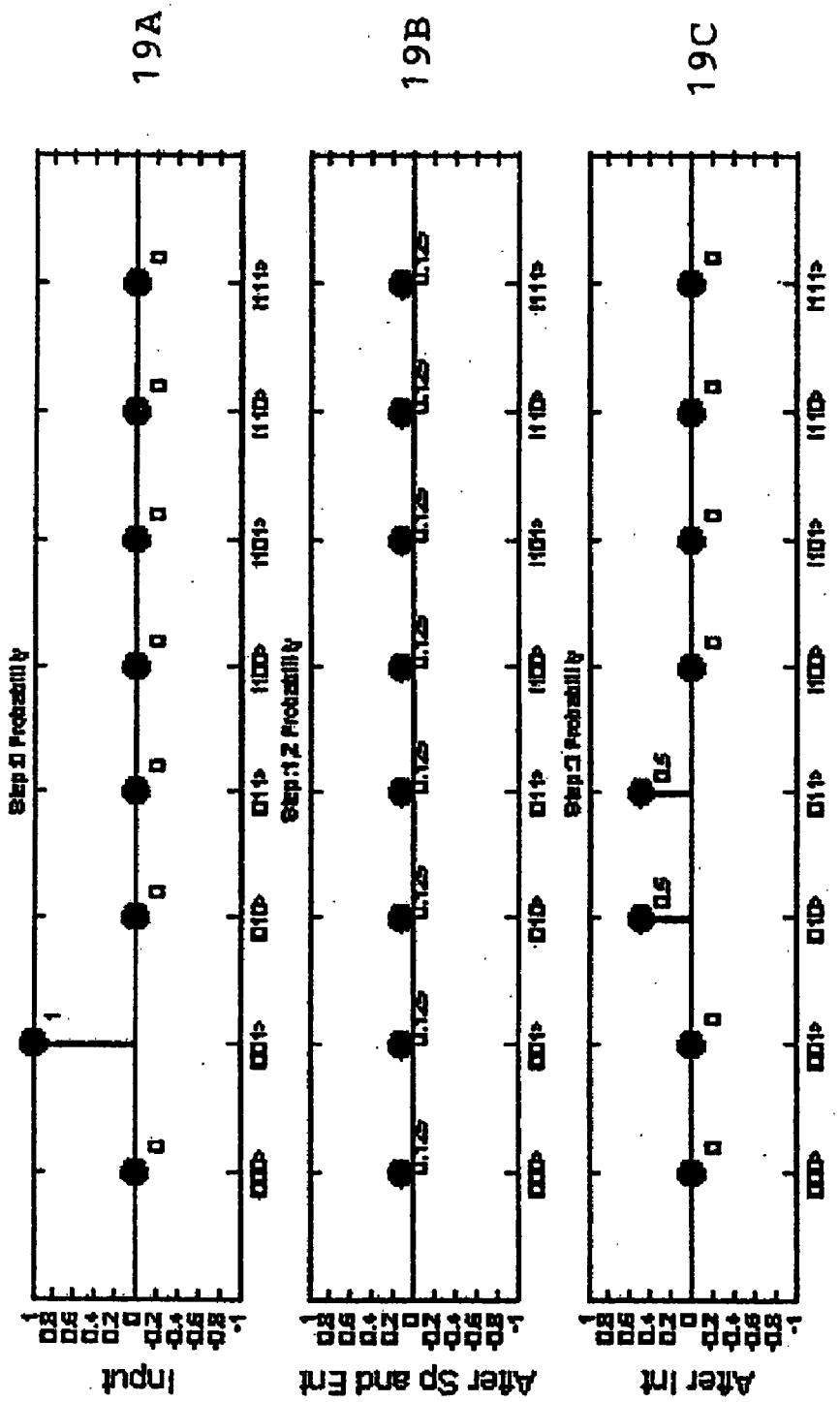


Figure 19

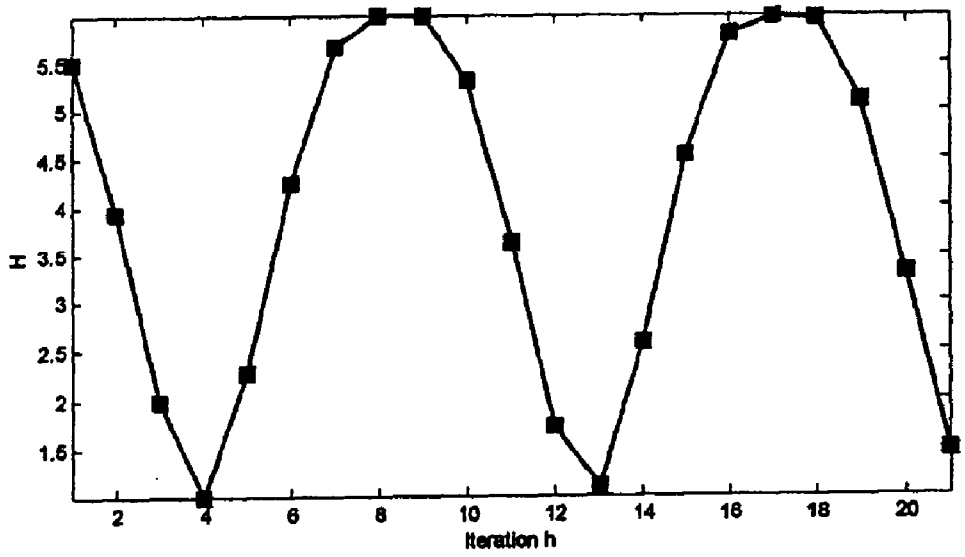


Figure 20

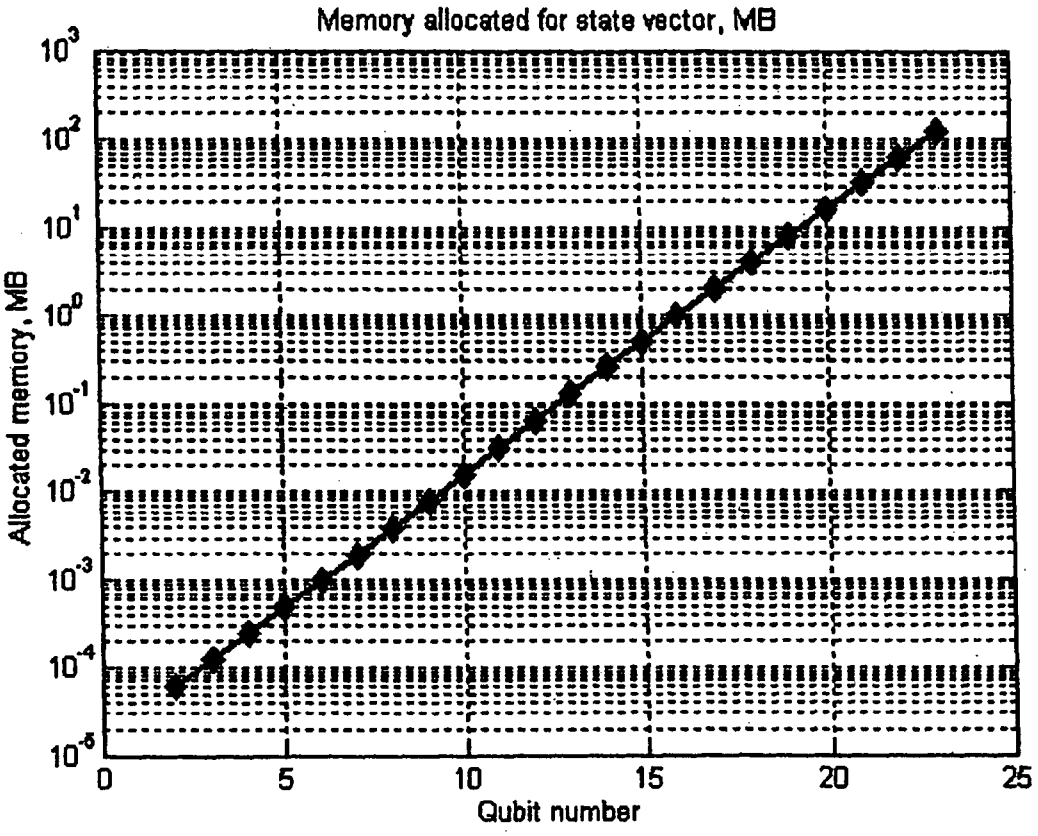

Memory allocated for state vector, MB
Allocated memory, MB
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5
10
15
20
25
Qubit number

Figure 21

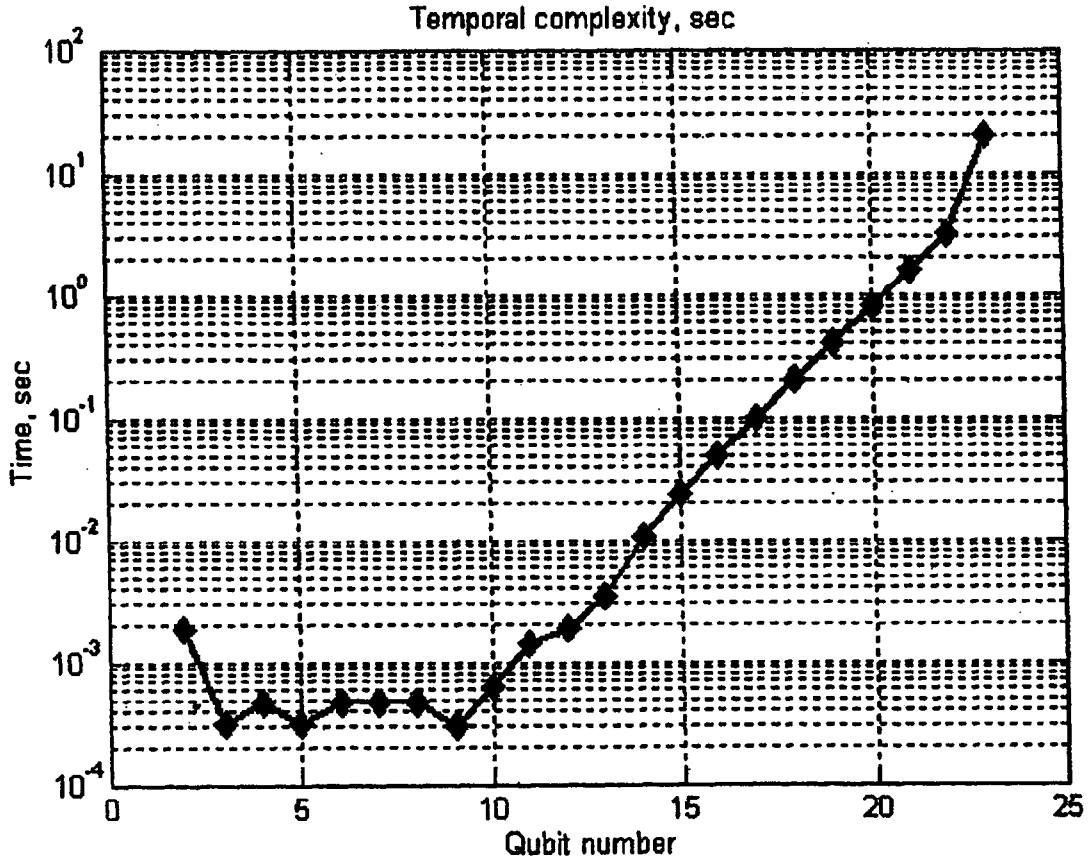

Temporal complexity, sec
Time, sec
Qubit number
10^2
10^1
10^0
10^-1
10^-2
10^-3
10^-4
0
5
10
15
20
25

Figure 22

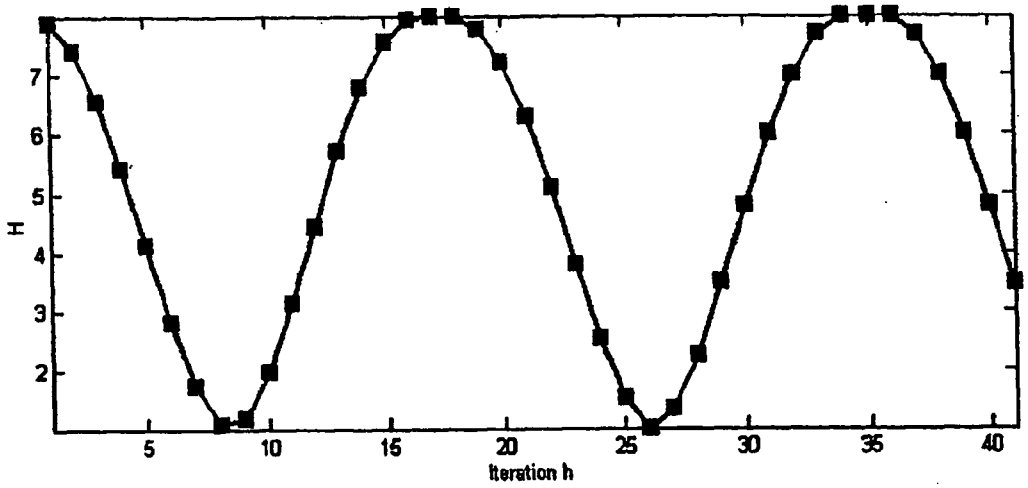


Figure 23

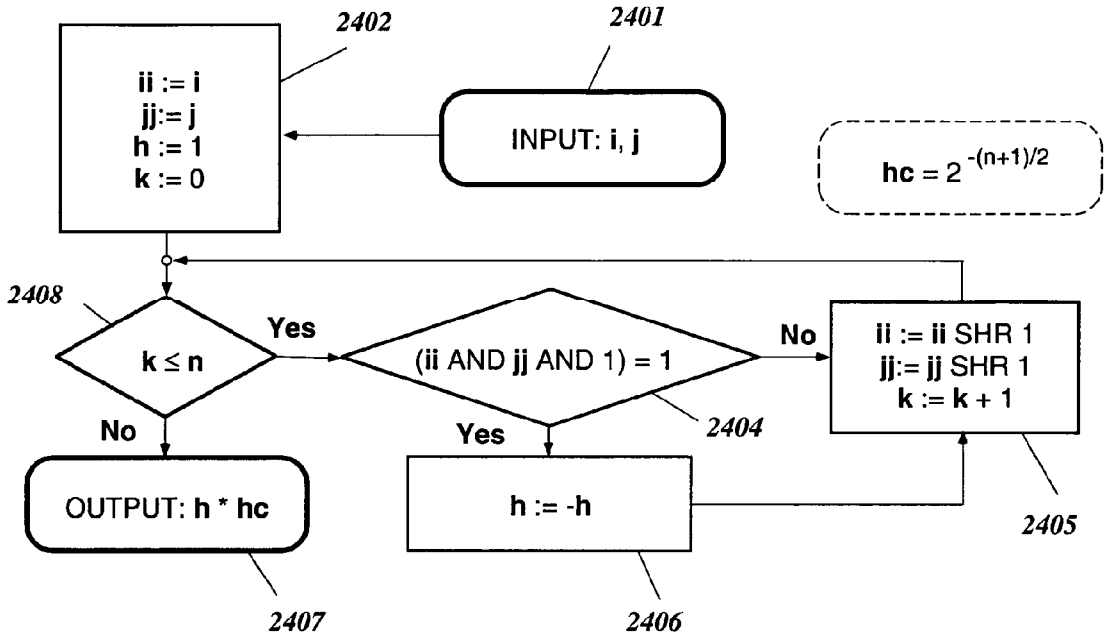

2402
2401
ii := i
jj:= j
h := 1
k := 0
INPUT: i, j
hc = 2 -(n+1)/2
2408
k ≤ n
Yes
(ii AND jj AND 1) = 1
No
ii := ii SHR 1
jj:= jj SHR 1
k := k + 1
2404
No
Yes
OUTPUT: h * hc
h := -h
2405
2407
2406

Figure 24a

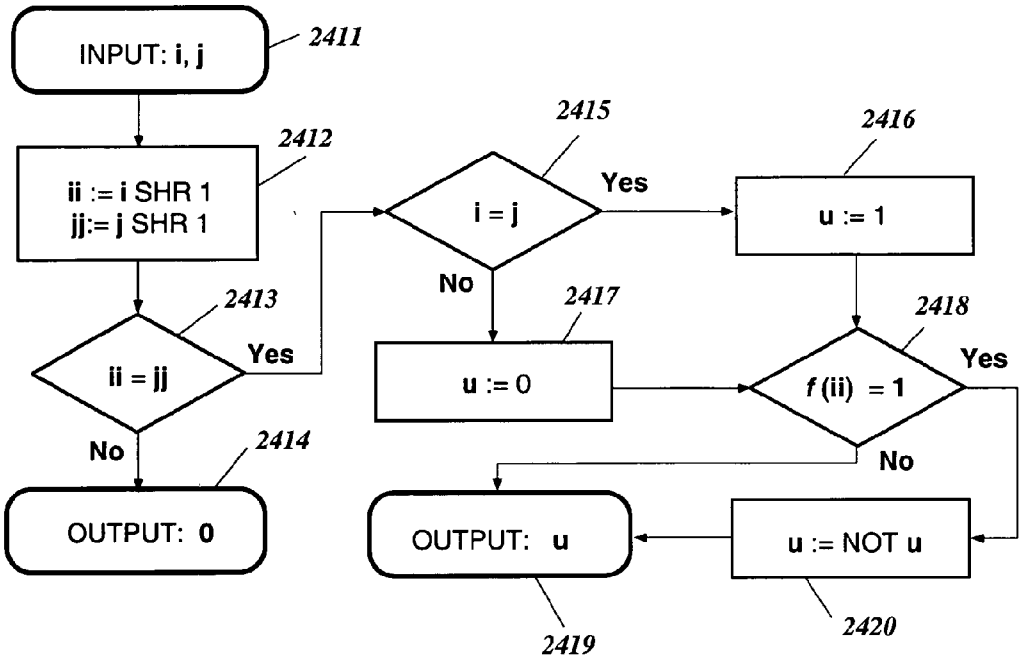

INPUT: i, j
2411
ii := i SHR 1
jj:= j SHR 1
2412
ii = jj
2413
Yes
No
2414
OUTPUT: 0
i = j
2415
Yes
No
2417
u := 0
u := 1
2416
f (ii) = 1
2418
Yes
No
OUTPUT: u
2419
u := NOT u
2420

Figure 24b

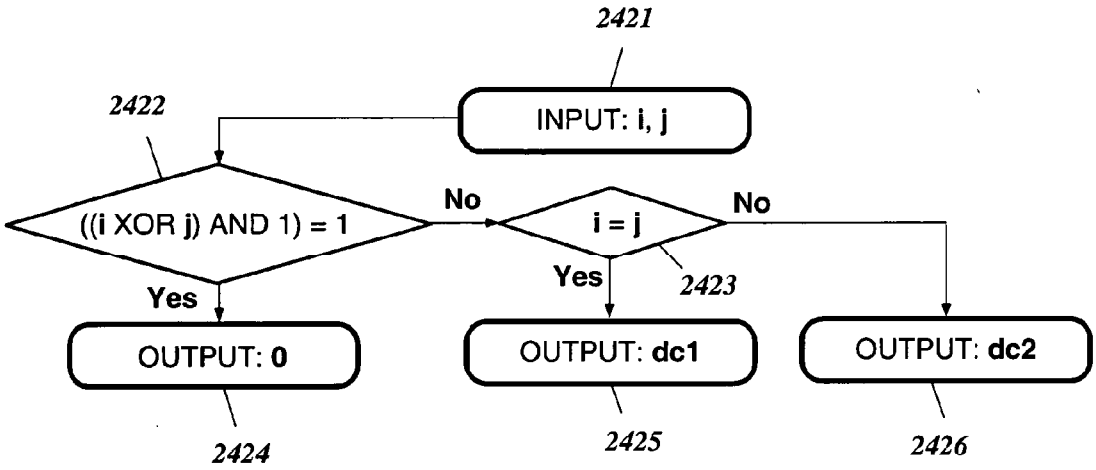

2421
INPUT: i, j
2422
((i XOR j) AND 1) = 1
No
i = j
No
Yes
2423
Yes
OUTPUT: 0
OUTPUT: dc1
OUTPUT: dc2
2424
2425
2426

Figure 24c

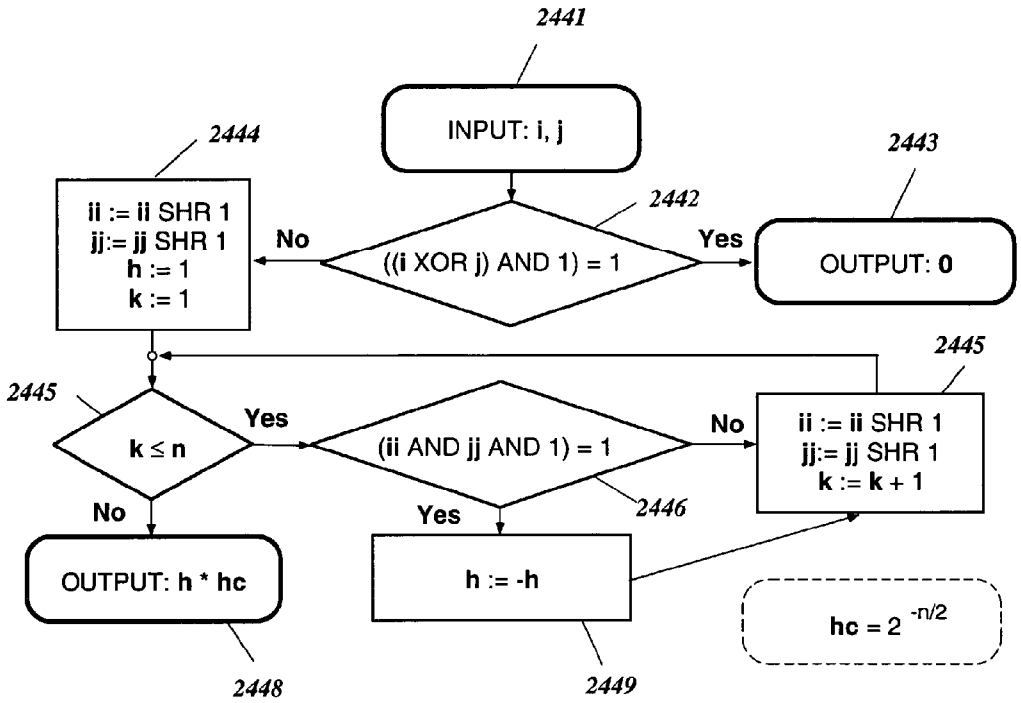

2441
INPUT: i, j
2444
ii := ii SHR 1
jj:= jj SHR 1
h := 1
k := 1
No
2442
((i XOR j) AND 1) = 1
Yes
2443
OUTPUT: 0
2445
2445
k ≤ n
Yes
(ii AND jj AND 1) = 1
No
2445
ii := ii SHR 1
jj:= jj SHR 1
k := k + 1
No
Yes
2446
OUTPUT: h * hc
h := -h
hc = 2 -n/2
2448
2449

Figure 24d

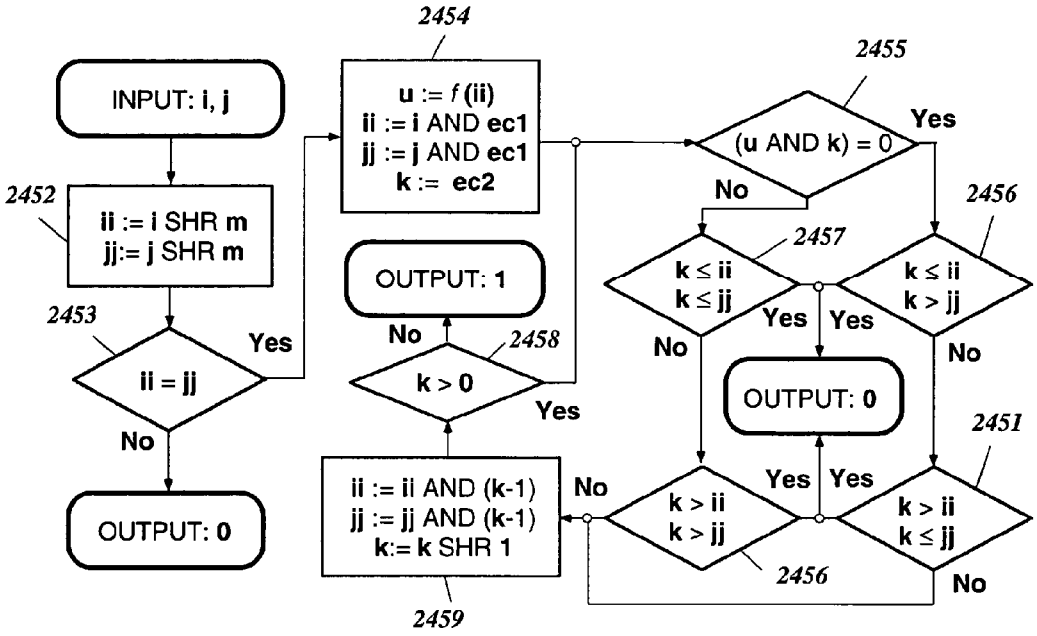

INPUT: i, j
2452
ii := i SHR m
jj:= j SHR m
2453
ii = jj
Yes
No
OUTPUT: 0
2454
u := f (ii)
ii := i AND ec1
jj := j AND ec1
k := ec2
OUTPUT: 1
No
2458
k > 0
Yes
ii := ii AND (k-1)
jj := jj AND (k-1)
k:= k SHR 1
2459
2455
(u AND k) = 0
Yes
No
2456
2457
k ≤ ii
k ≤ jj
Yes
No
k ≤ ii
k > jj
Yes
No
OUTPUT: 0
2451
k > ii
k > jj
Yes
Yes
No
2456
k > ii
k ≤ jj
No

Figure 24e

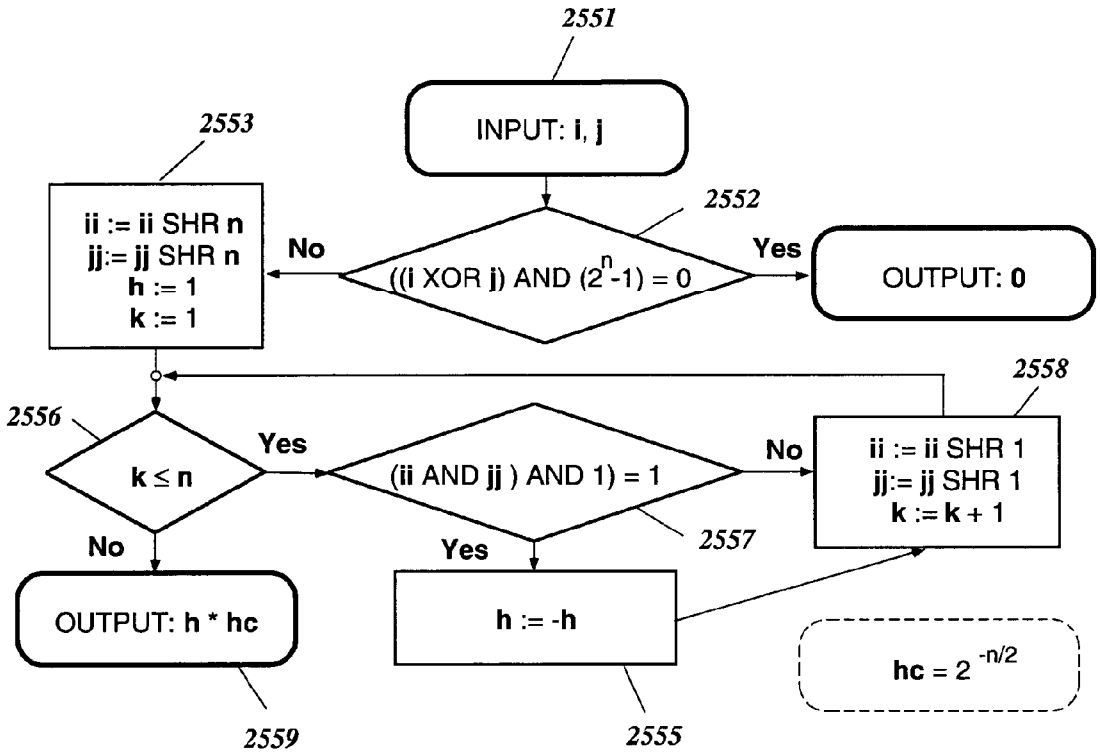

2551
INPUT: i, j
2553
ii := ii SHR n
jj:= jj SHR n
h := 1
k := 1
2552
No
((i XOR j) AND (2^n-1) = 0
Yes
OUTPUT: 0
2558
2556
k ≤ n
Yes
(ii AND jj) AND 1) = 1
No
ii := ii SHR 1
jj:= jj SHR 1
k := k + 1
2557
No
Yes
OUTPUT: h * hc
h := -h
hc = 2^-n/2
2559
2555

Figure 24f

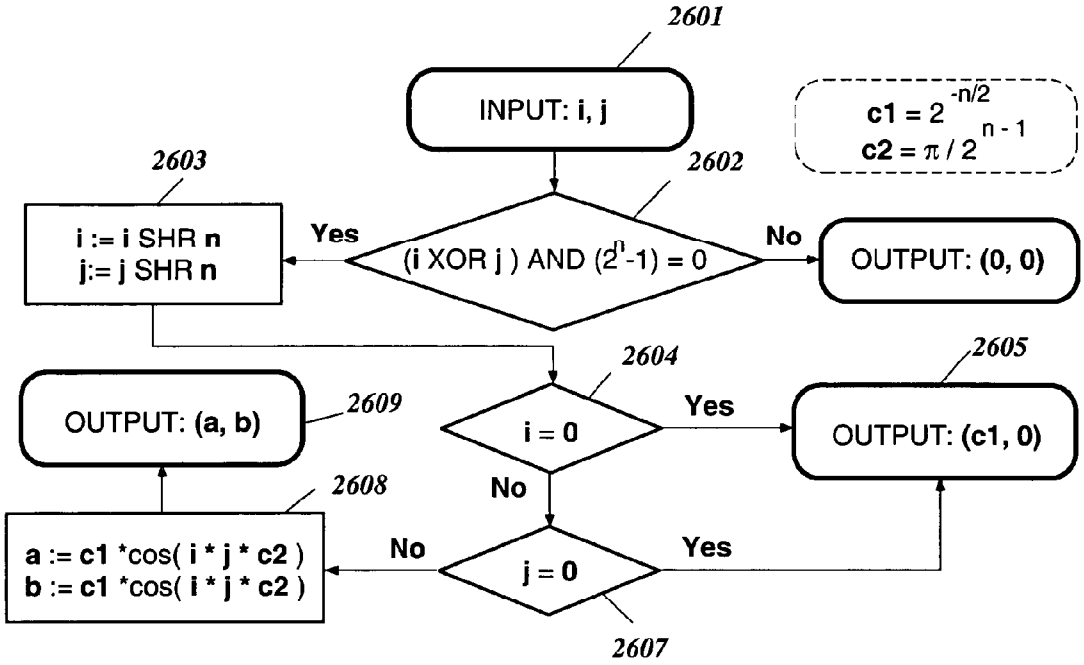

2601
INPUT: i, j
c1 = 2^-n/2
c2 = π / 2^n - 1
2603
2602
i := i SHR n
j:= j SHR n
Yes
(i XOR j) AND (2^n-1) = 0
No
OUTPUT: (0, 0)
2604
2605
OUTPUT: (a, b)
2609
i = 0
Yes
OUTPUT: (c1, 0)
No
2608
a := c1 *cos(i * j * c2)
b := c1 *cos(i * j * c2)
No
j = 0
Yes
2607

Figure 24g

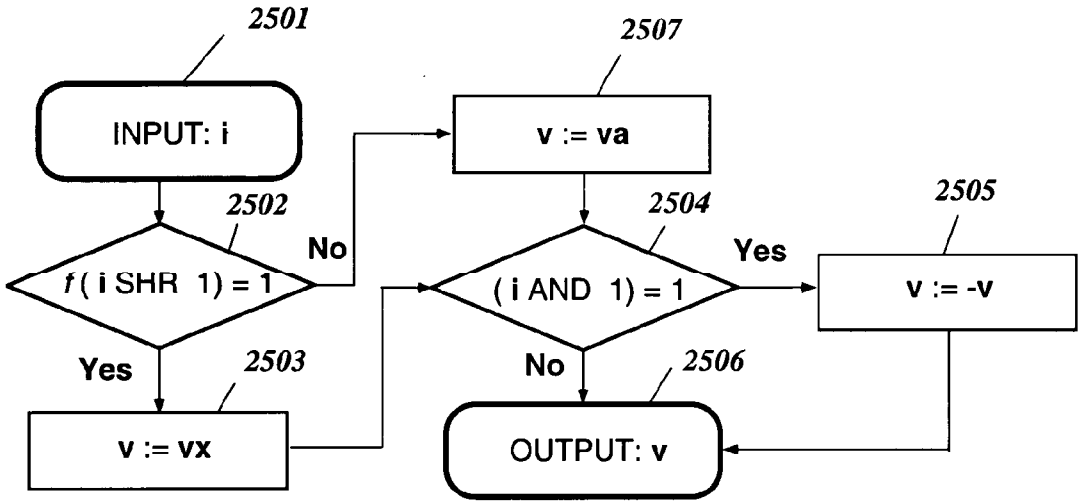

2501
INPUT: i
2502
f(i SHR 1) = 1
No
Yes
2503
v := vx
2507
v := va
2504
(i AND 1) = 1
Yes
2505
v := -v
No
2506
OUTPUT: v

Figure 25

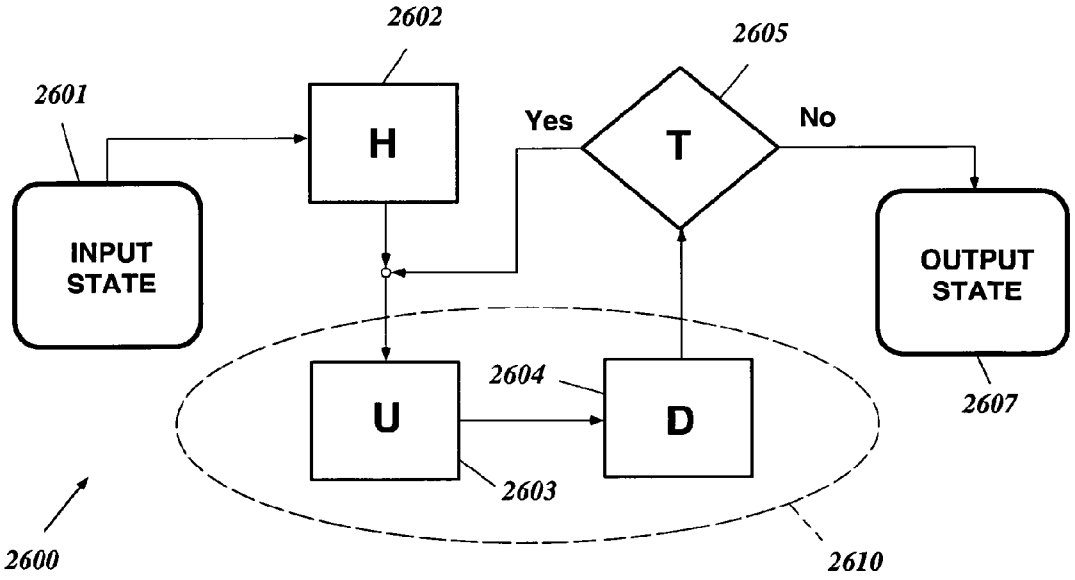

2602
2605
2601
H
Yes
T
No
INPUT STATE
OUTPUT STATE
2604
U
D
2607
2603
2600
2610

Figure 26

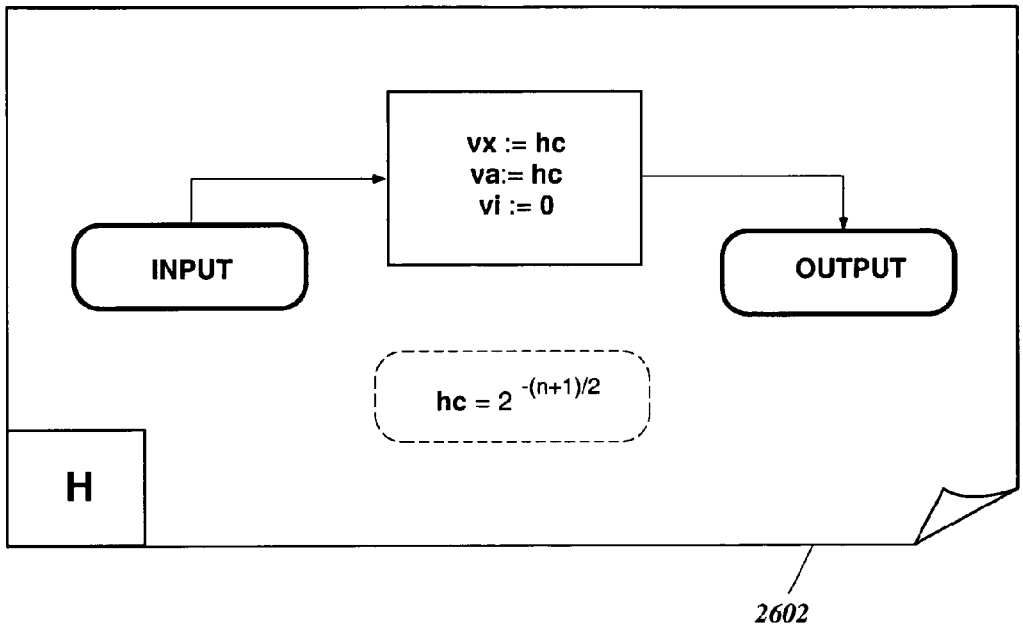


Figure 27

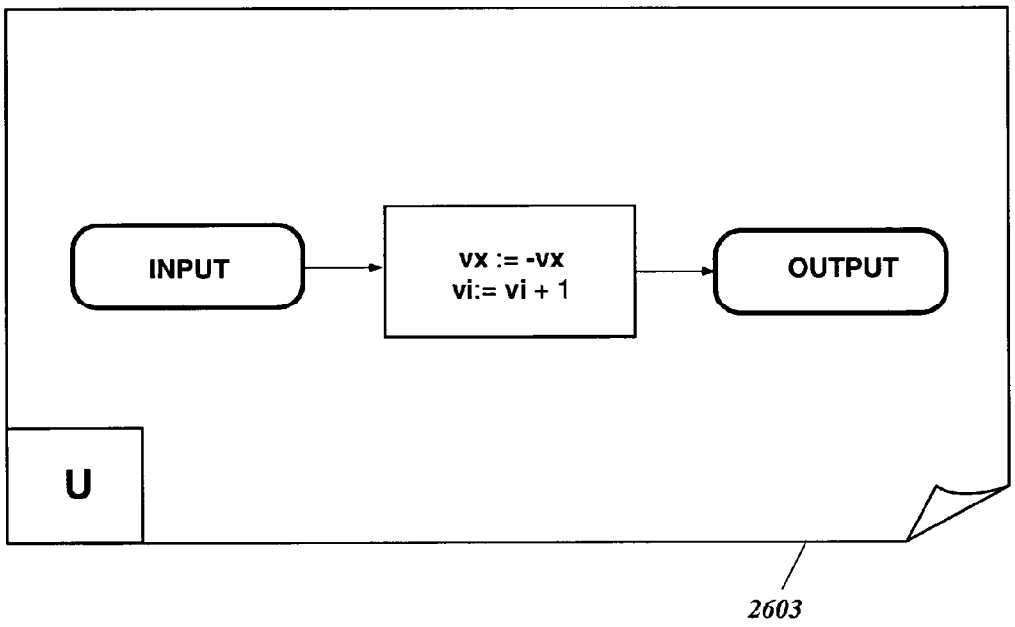


Figure 28a

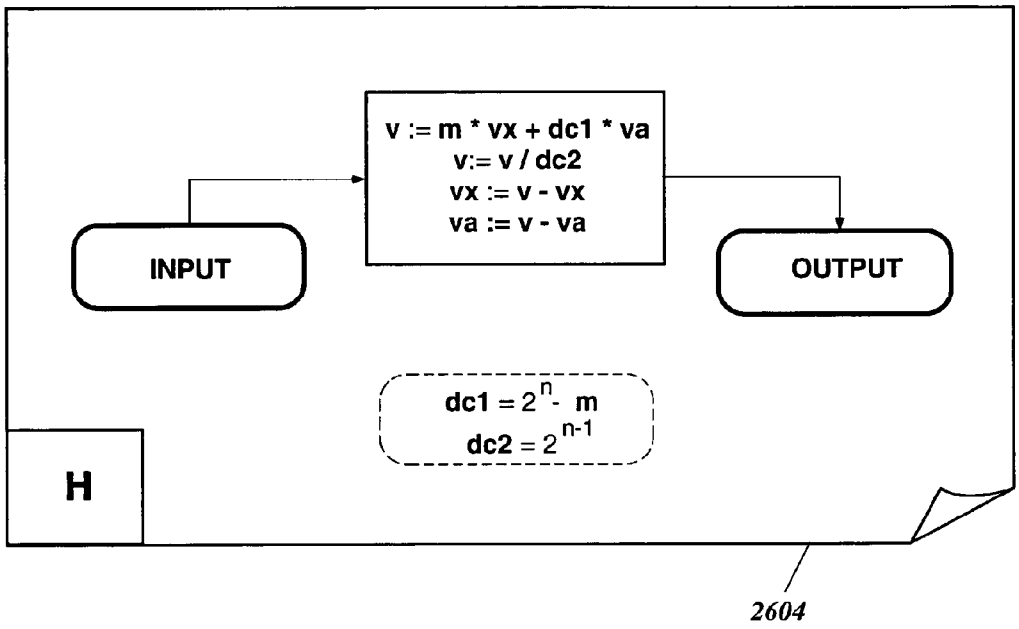

v := m * vx + dc1 * va
v:= v / dc2
vx := v - vx
va := v - va
INPUT
OUTPUT
dc1 = 2^n - m
dc2 = 2^n-1
H
2604

Figure 28b

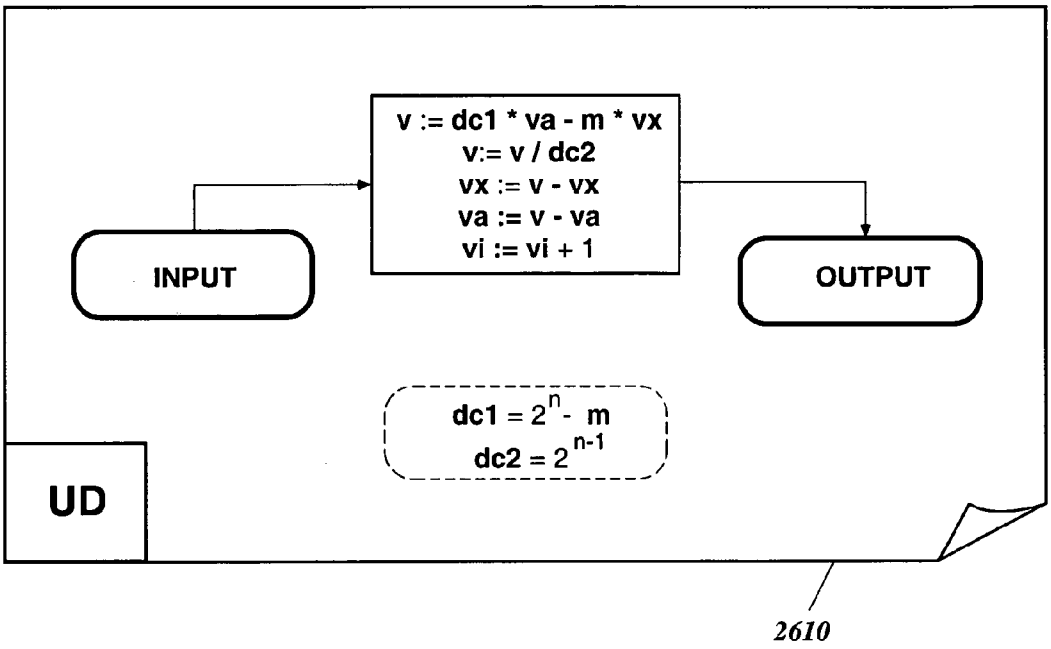

v := dc1 * va - m * vx
v:= v / dc2
vx := v - vx
va := v - va
vi := vi + 1
INPUT
OUTPUT
dc1 = 2^n - m
dc2 = 2^n-1
UD
2610

Figure 28c

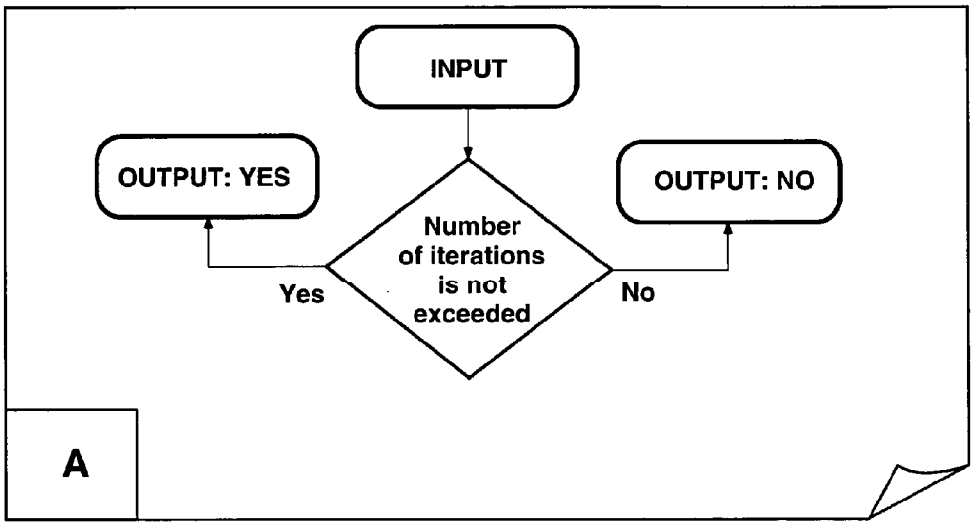

INPUT
OUTPUT: YES
OUTPUT: NO
Number of iterations is not exceeded
Yes
No
A

Figure 29

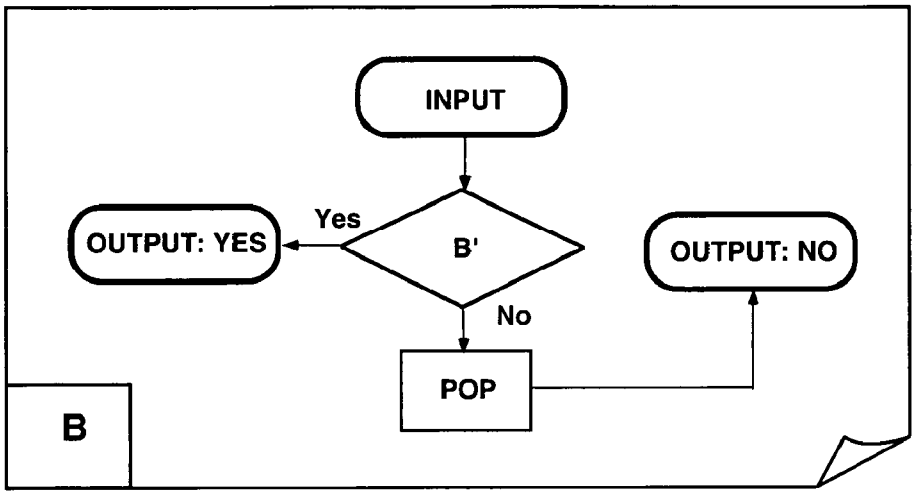


Figure 30

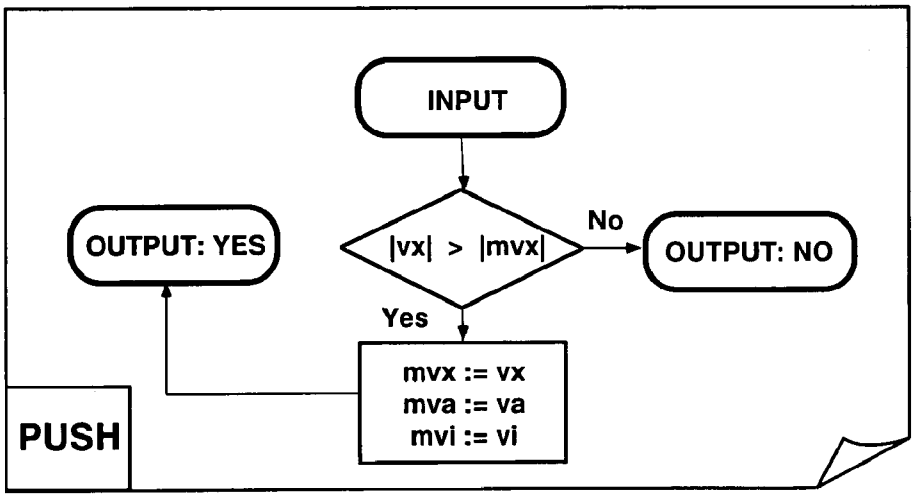


Figure 31A

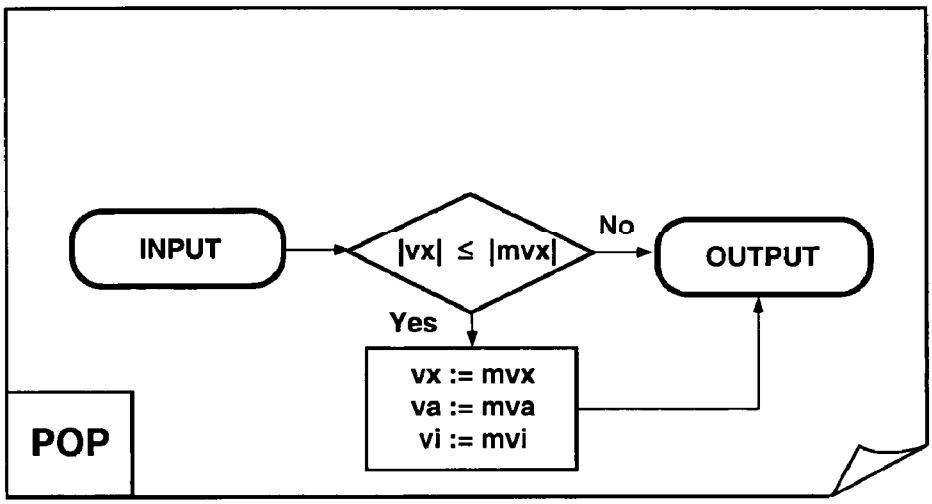


Figure 31B

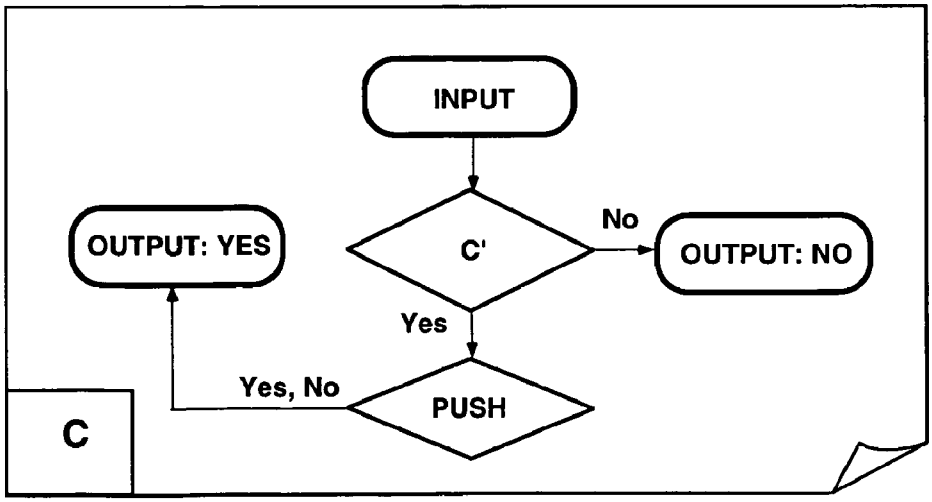


Figure 32

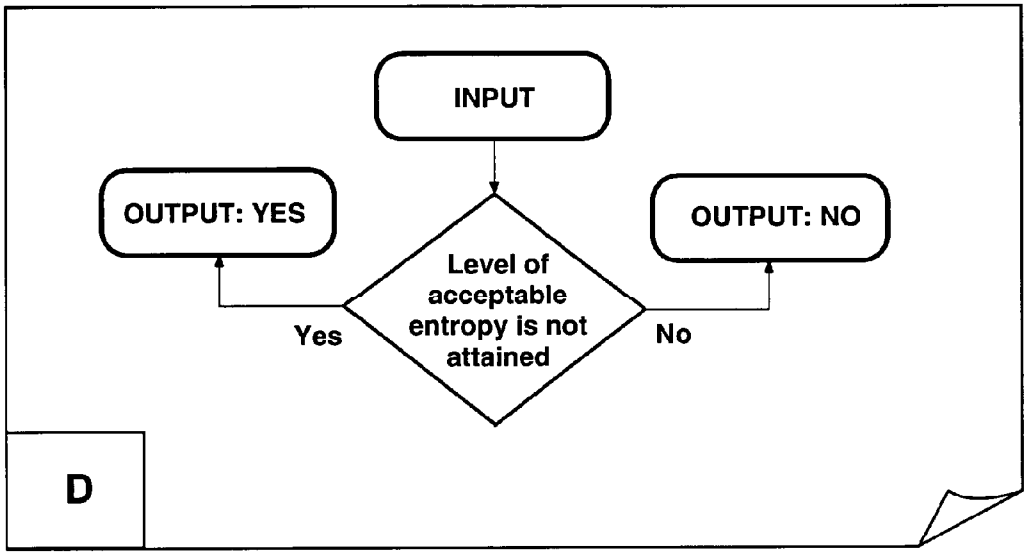

INPUT
OUTPUT: YES
OUTPUT: NO
Level of acceptable entropy is not attained
Yes
No
D

Figure 33

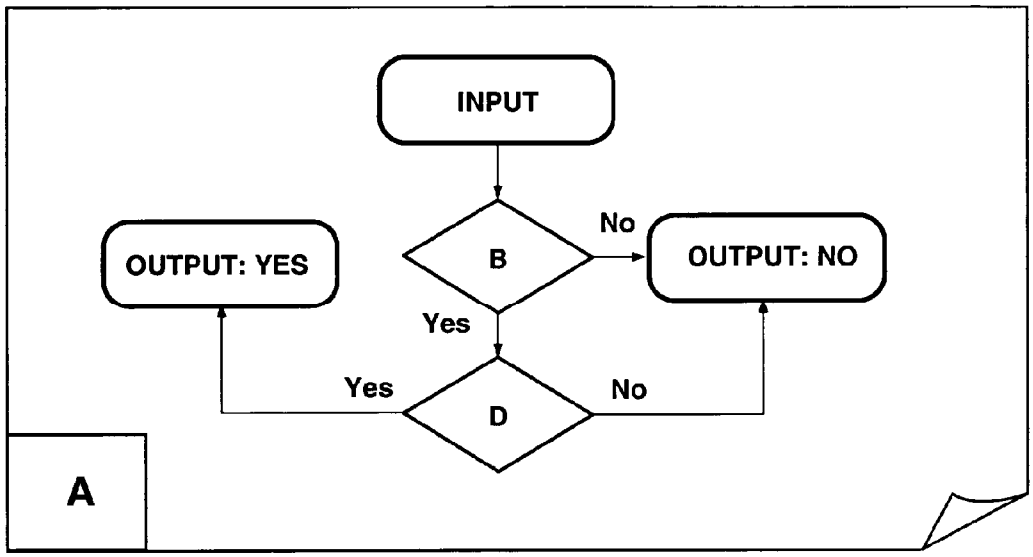


Figure 34

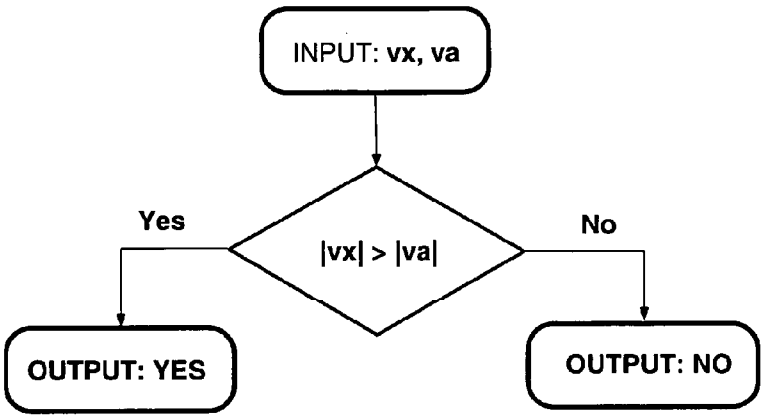


Figure 35

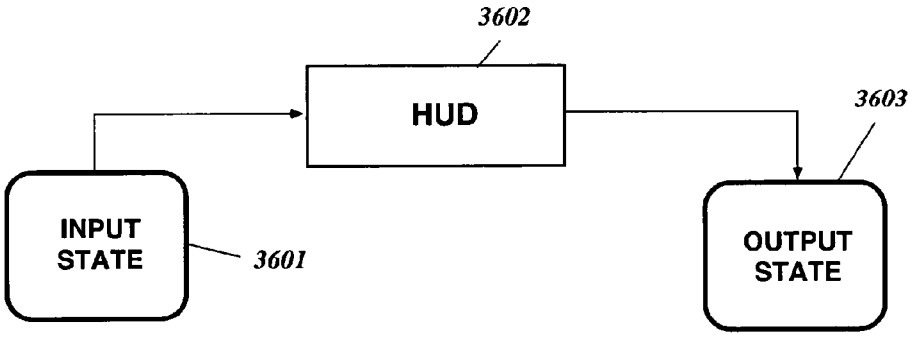


Figure 36

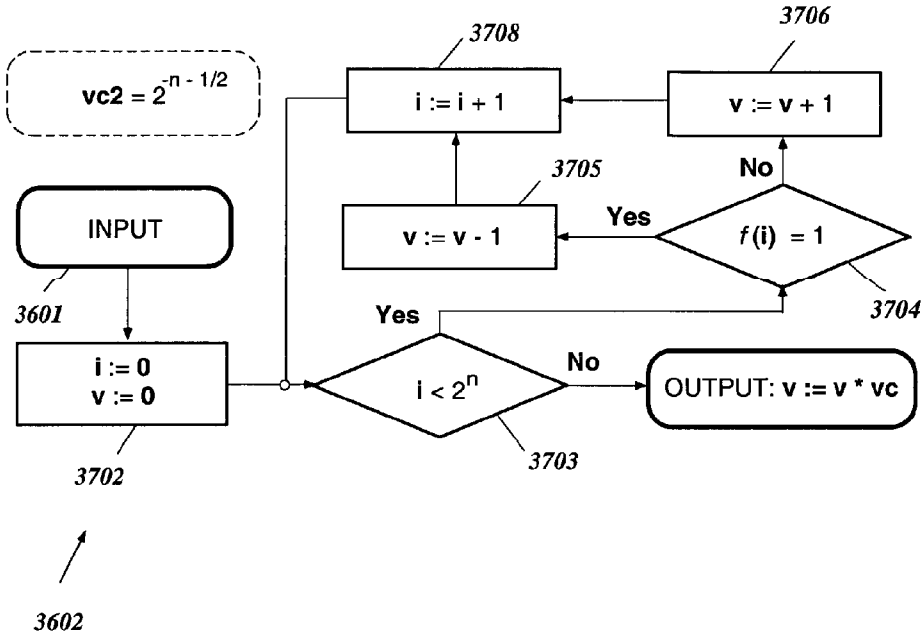

vc2 = 2^-n - 1/2
3708
i := i + 1
3706
v := v + 1
3705
No
INPUT
v := v - 1
Yes
f(i) = 1
3601
3704
Yes
i := 0
v := 0
i < 2^n
No
OUTPUT: v := v * vc
3702
3703
3602

Figure 37

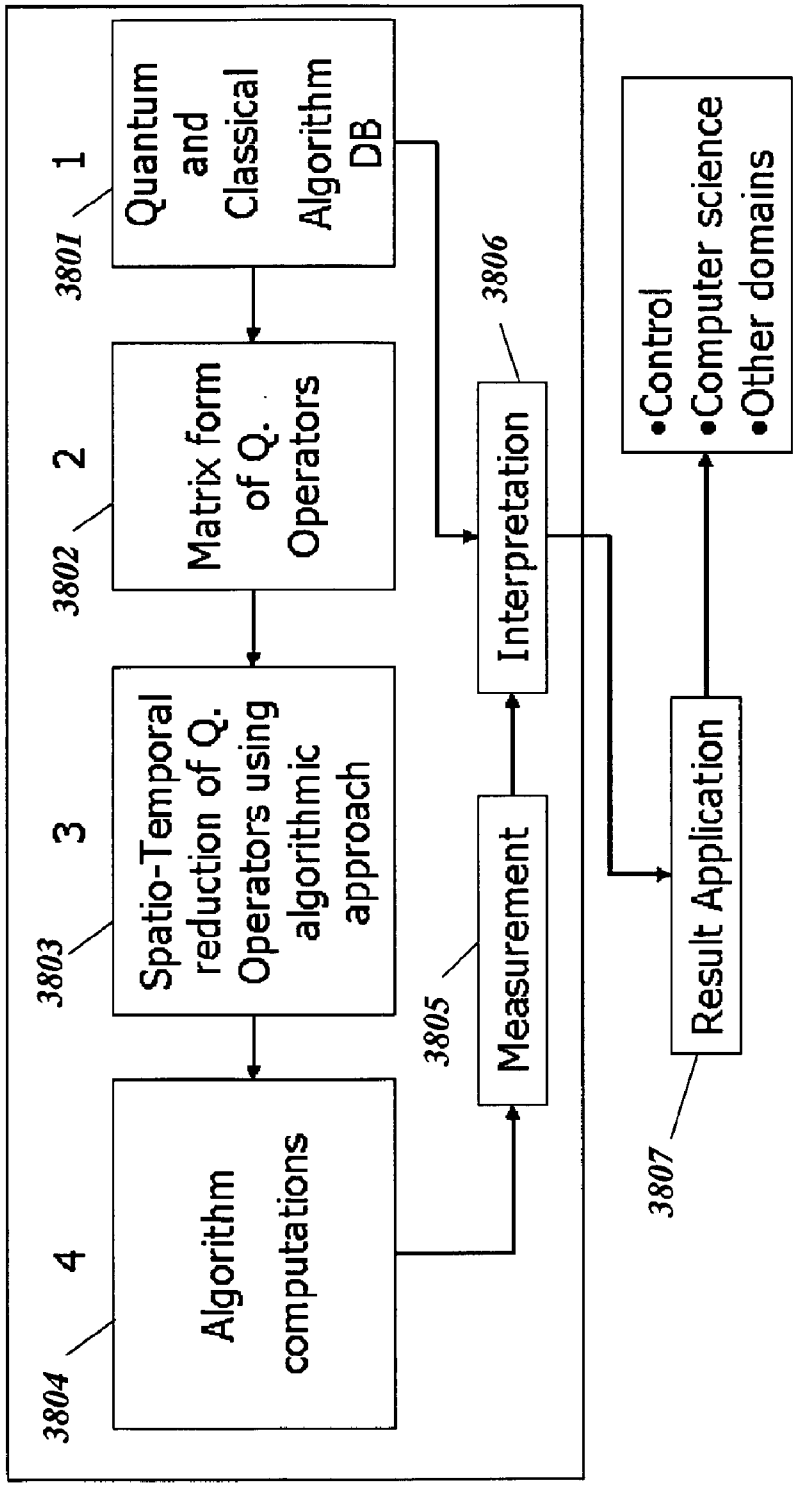

1
Quantum and Classical Algorithm DB
3801
2
Matrix form of Q. Operators
3802
3
Spatio-Temporal reduction of Q. Operators using algorithmic approach
3803
4
Algorithm computations
3804
3806
Interpretation
3805
Measurement
Result Application
3807
•Control
•Computer science
•Other domains

Figure 38

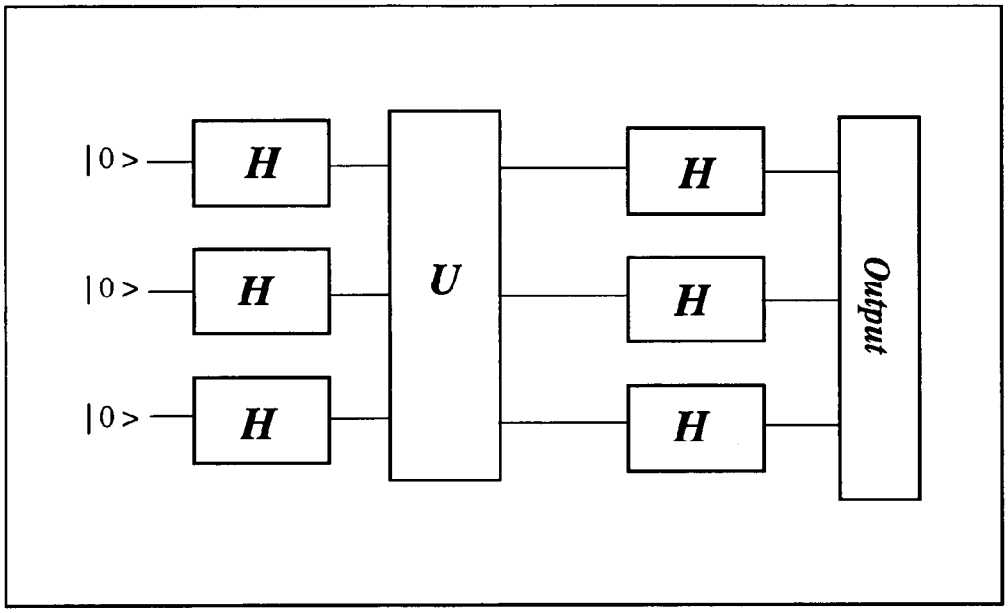


Figure 39

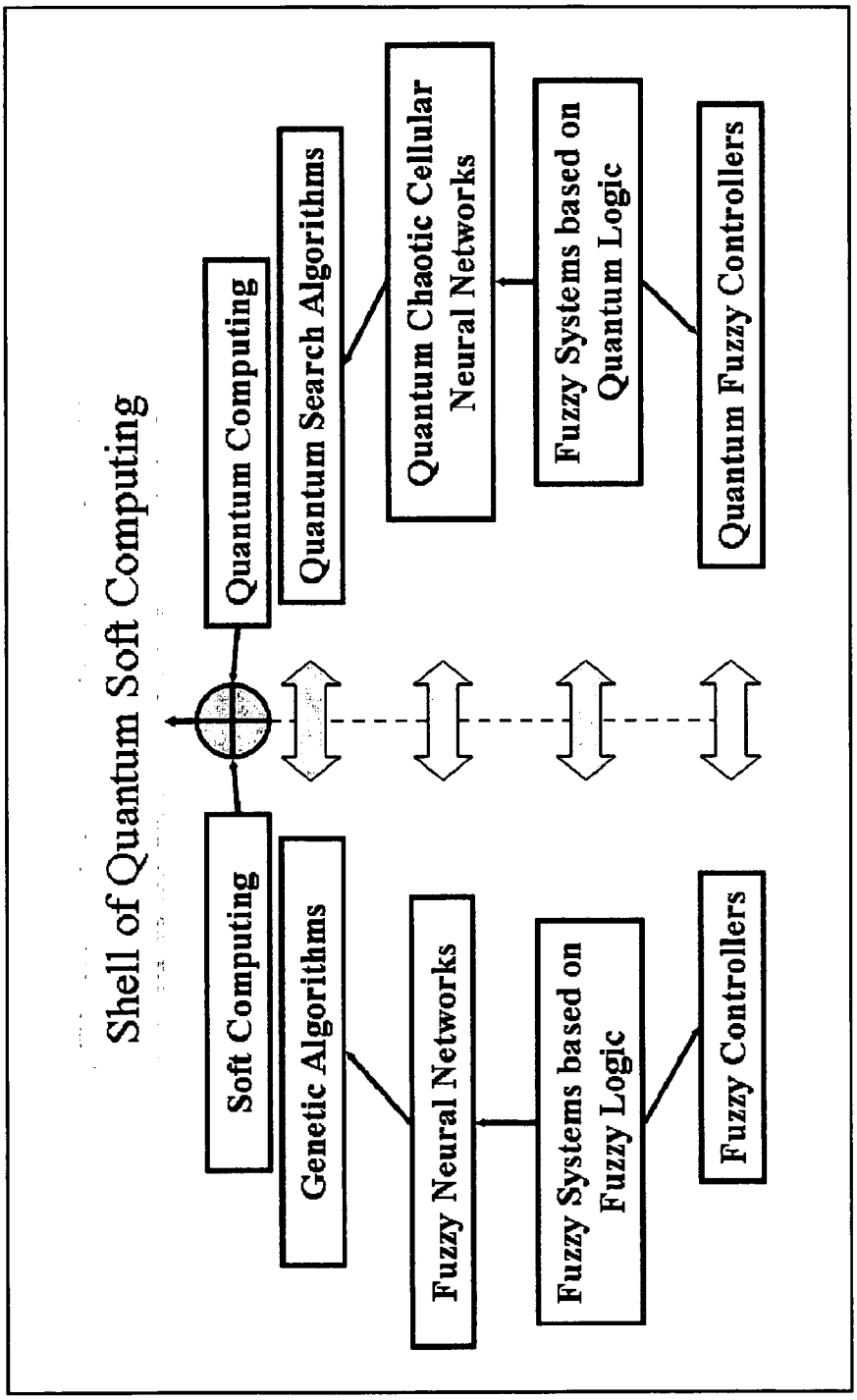

Shell of Quantum Soft Computing
Quantum Computing
Quantum Search Algorithms
Quantum Chaotic Cellular Neural Networks
Fuzzy Systems based on Quantum Logic
Quantum Fuzzy Controllers
Soft Computing
Genetic Algorithms
Fuzzy Neural Networks
Fuzzy Systems based on Fuzzy Logic
Fuzzy Controllers

Figure 40

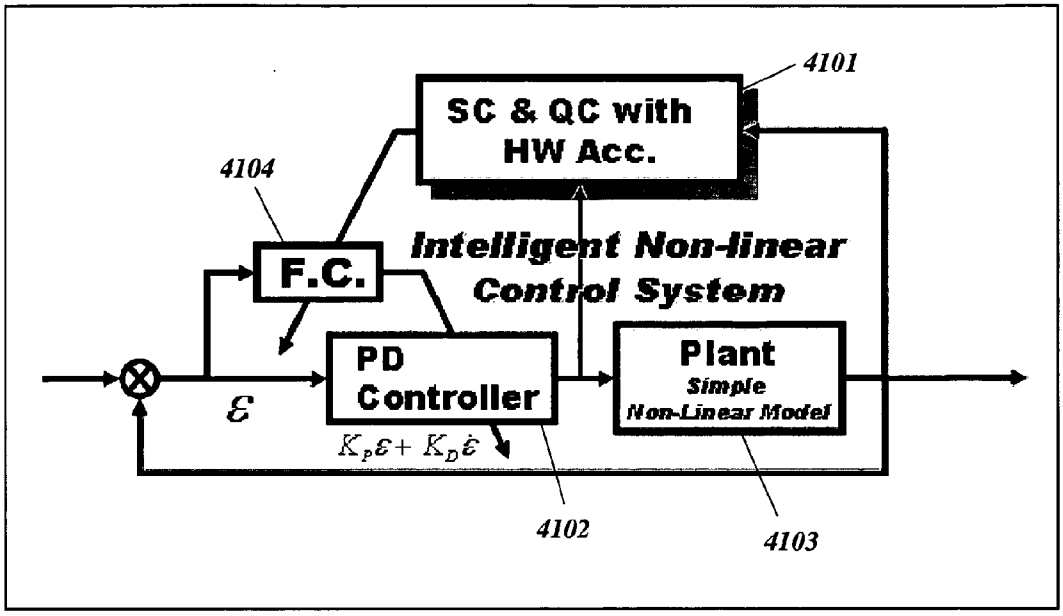

4101
SC & QC with HW Acc.
4104
F.C.
Intelligent Non-linear Control System
PD Controller
Plant
Simple Non-Linear Model
ε
$K_P\varepsilon + K_D\dot{\varepsilon}$
4102
4103

Figure 41a

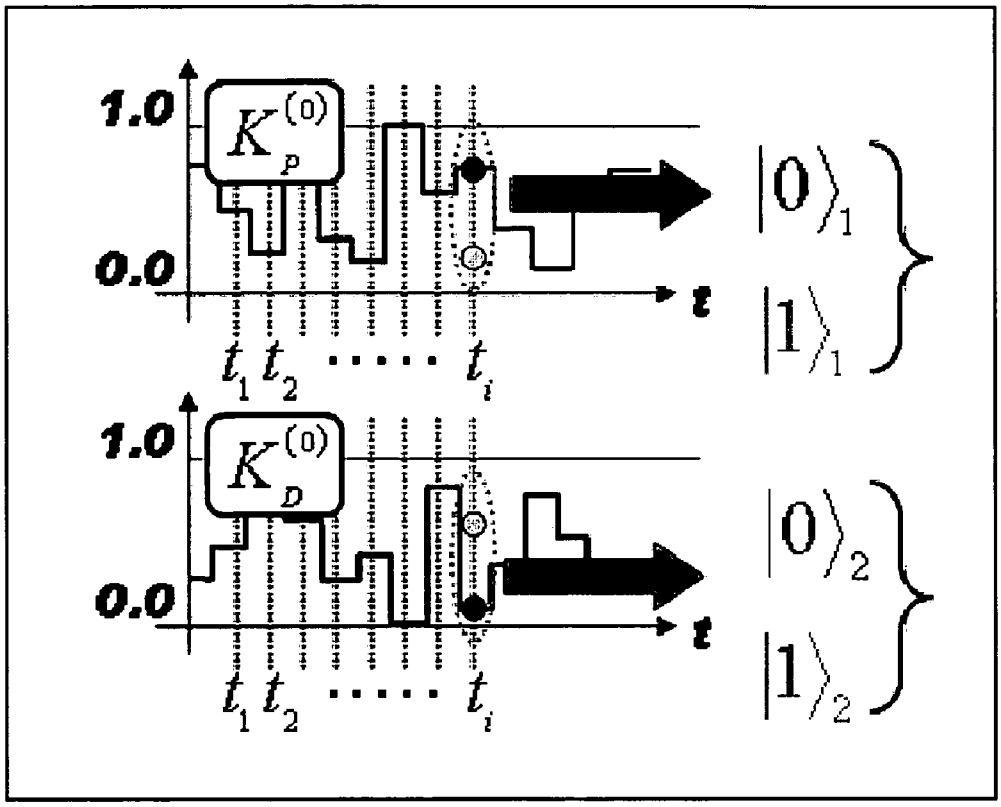

1.0
0.0
$K_P^{(0)}$
t
t_1 t_2 ..... t_i
$|0\rangle_1$
$|1\rangle_1$
1.0
0.0
$K_D^{(0)}$
t
t_1 t_2 ..... t_i
$|0\rangle_2$
$|1\rangle_2$

Figure 41b

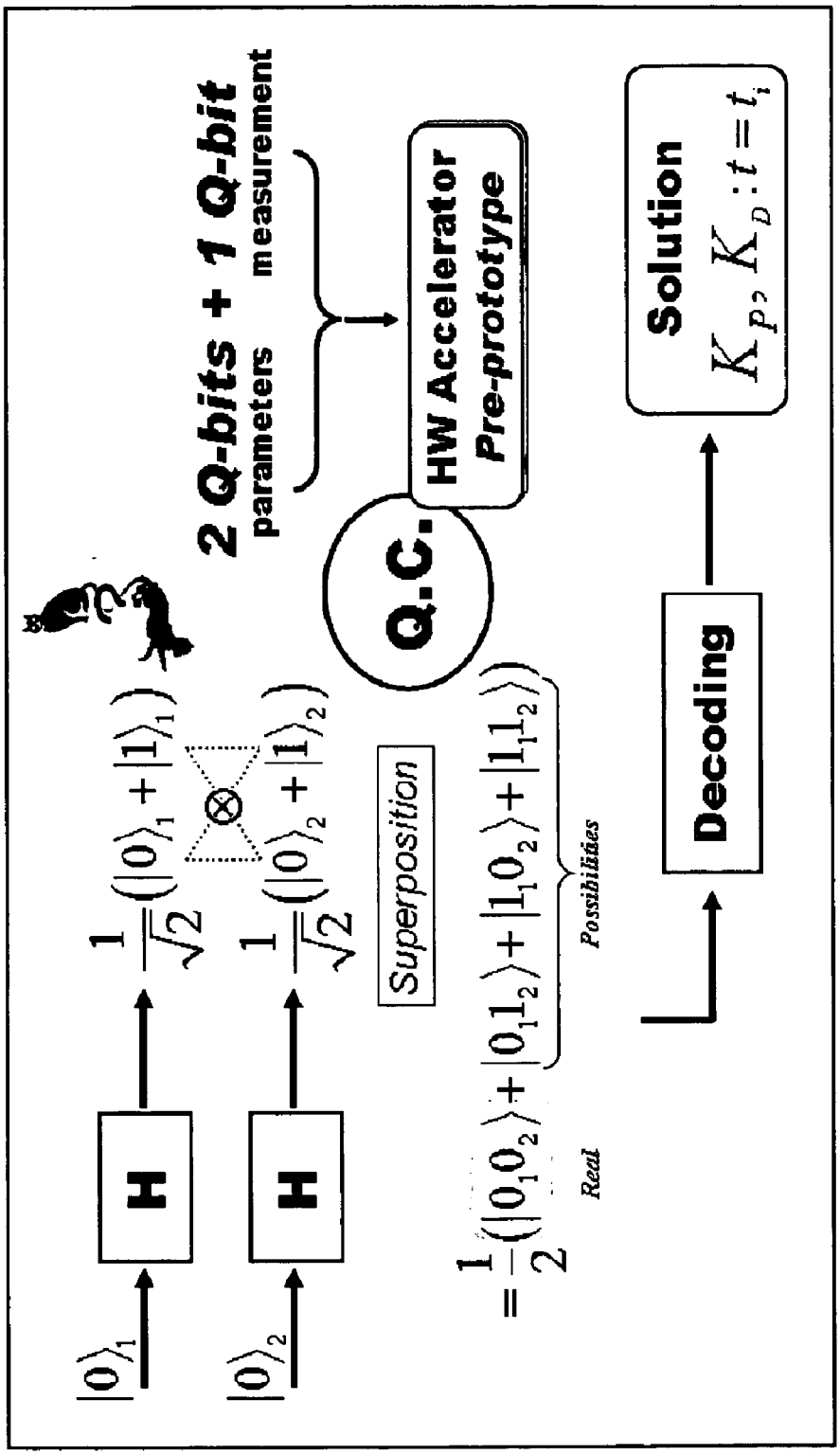

2 Q-bits + 1 Q-bit
parameters
measurement
Q.C.
HW Accelerator
Pre-prototype
Superposition
Real
Possibilities
Decoding
Solution

Figure 42

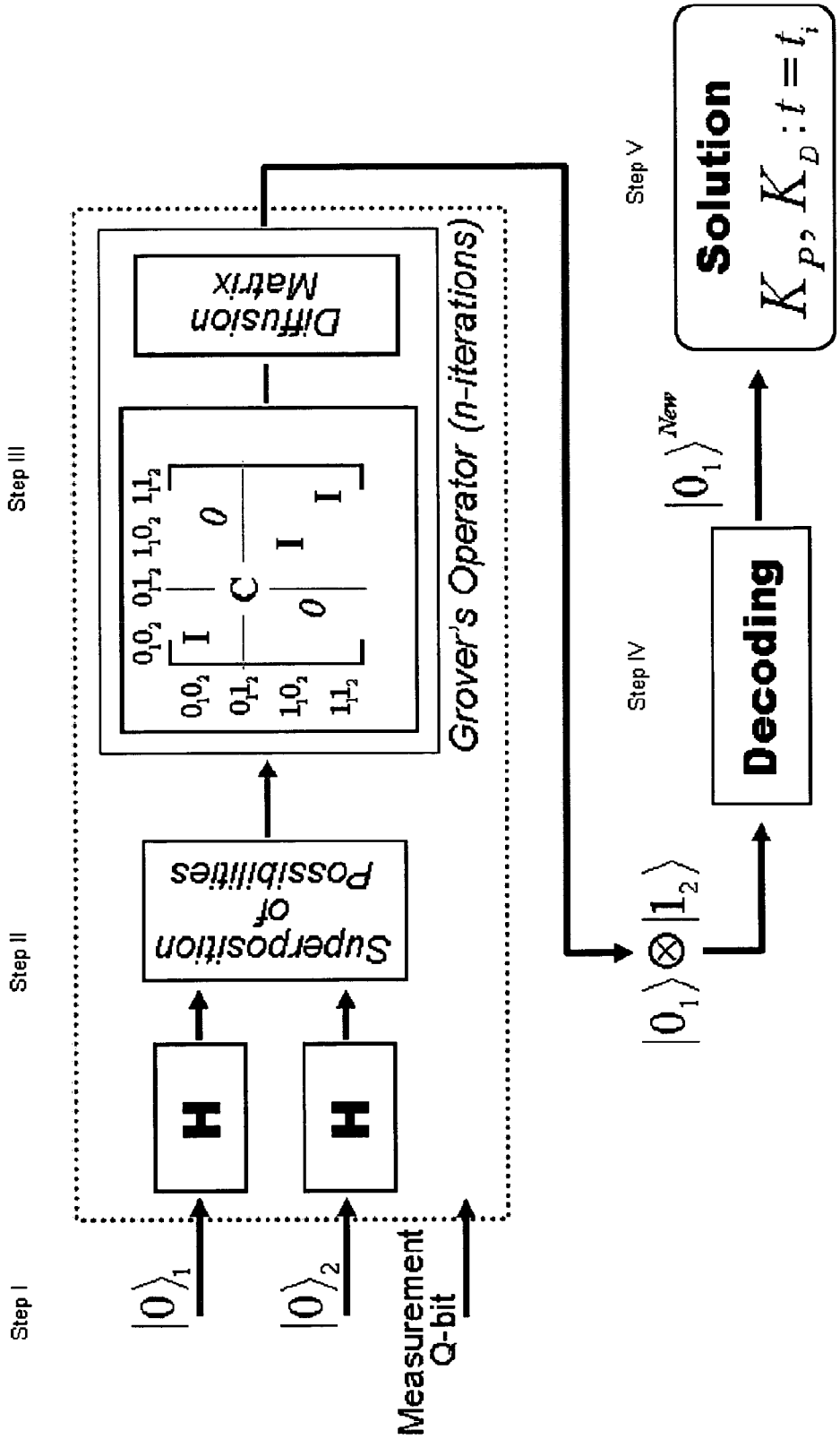

Step I
Step II
Step III
Step IV
Step V
Measurement Q-bit
H
H
Superposition of Possibilities
Diffusion Matrix
Grover's Operator (n-iterations)
Decoding
Solution
$K_p, K_D : t = t_i$
$|0_1\rangle^{New}$

Figure 43

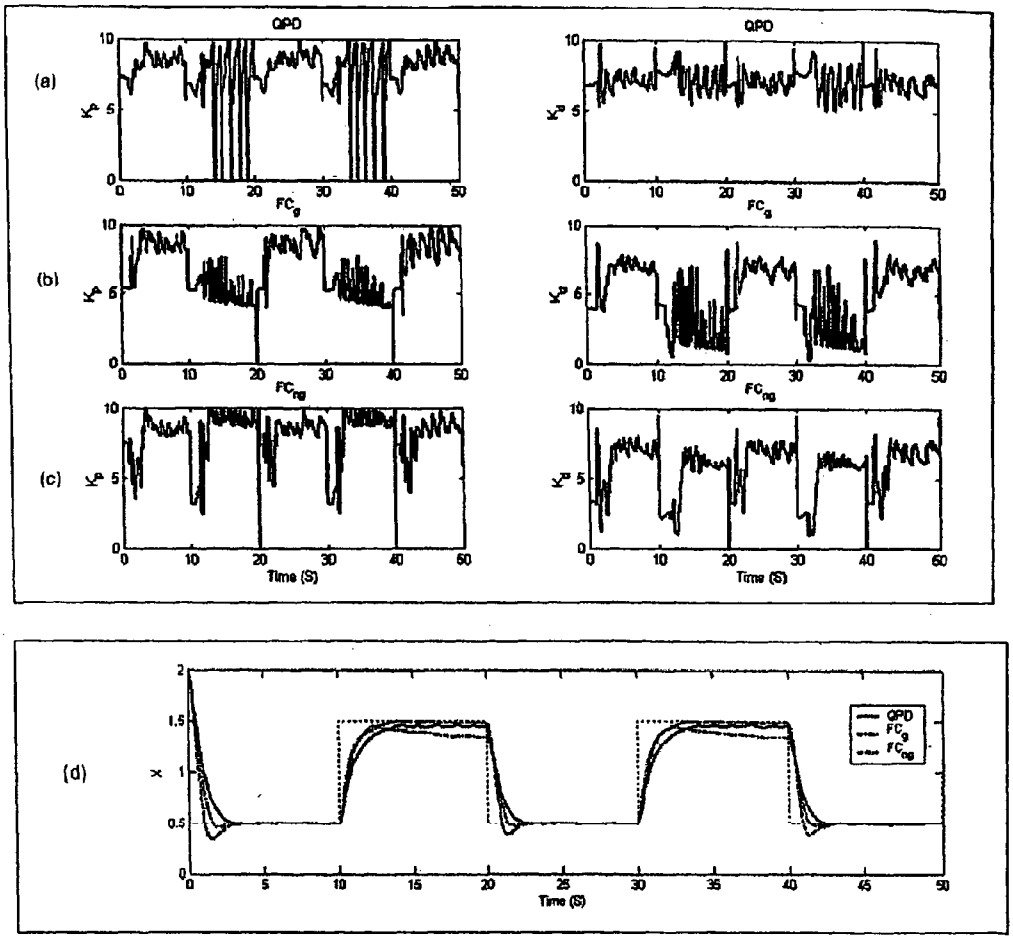


Figure 44a-d

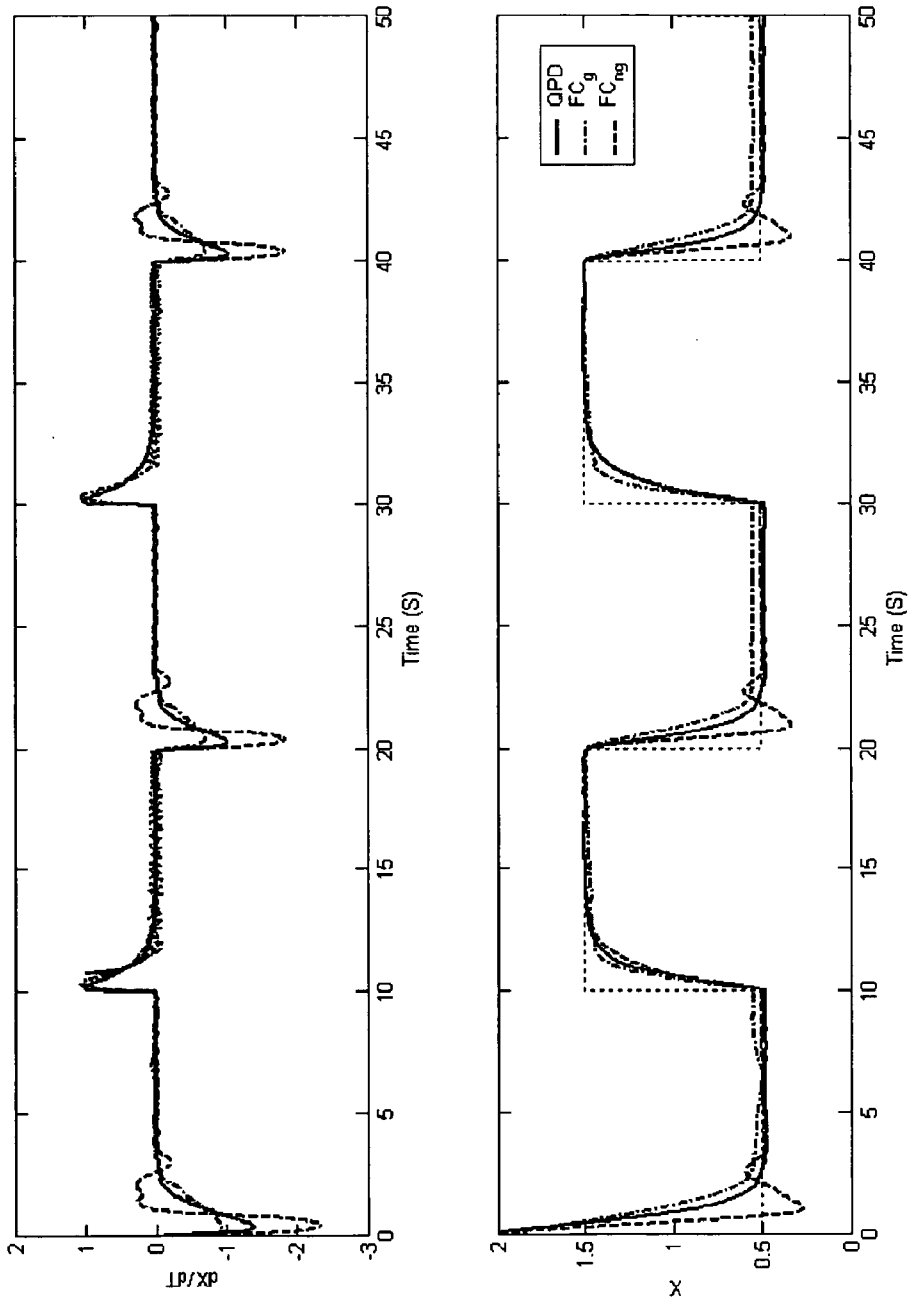


Figure 45

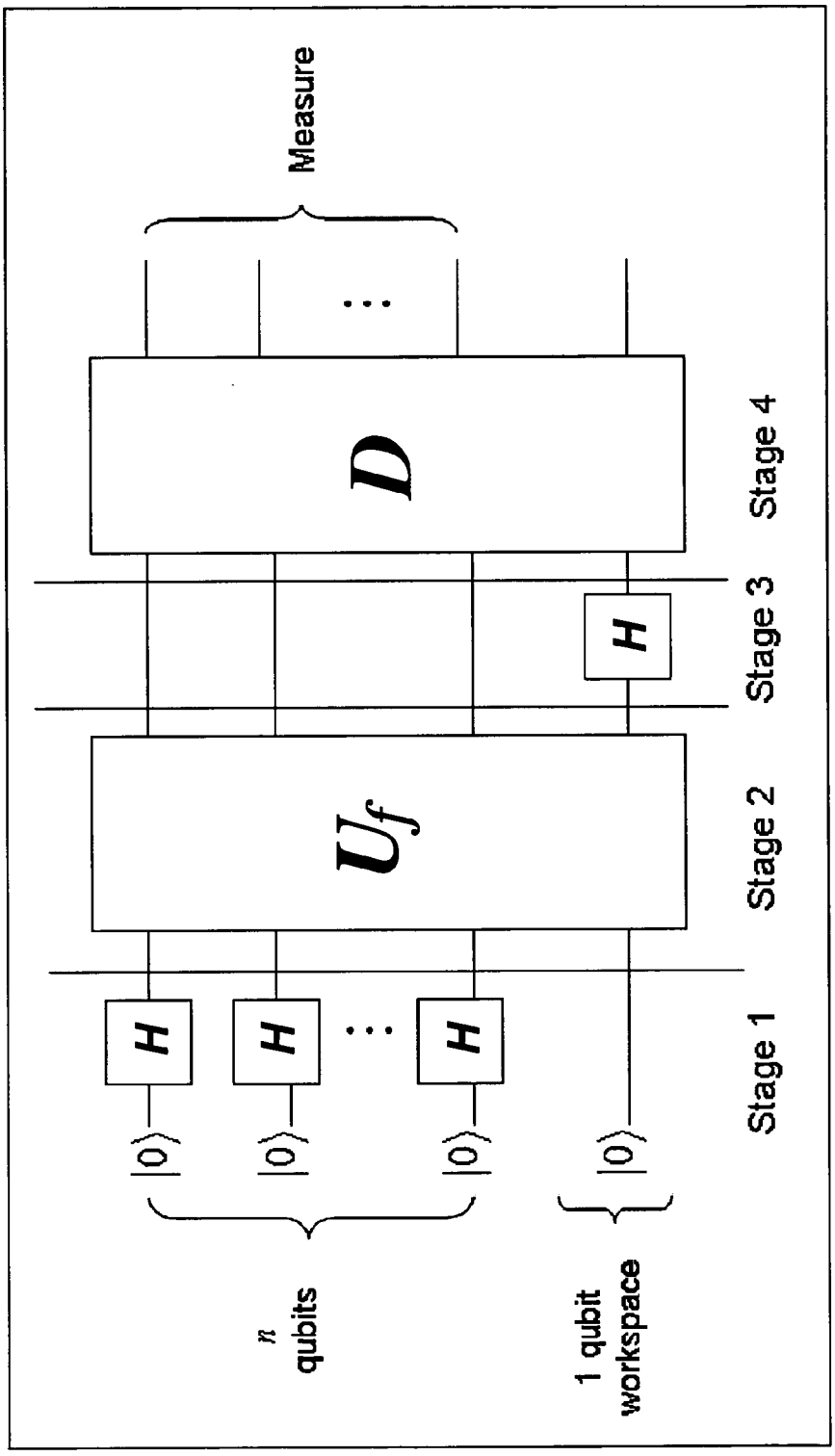

Measure
D
U_f
H
H
H
H
|0⟩
|0⟩
|0⟩
|0⟩
Stage 1
Stage 2
Stage 3
Stage 4
n qubits
1 qubit workspace

Figure 46

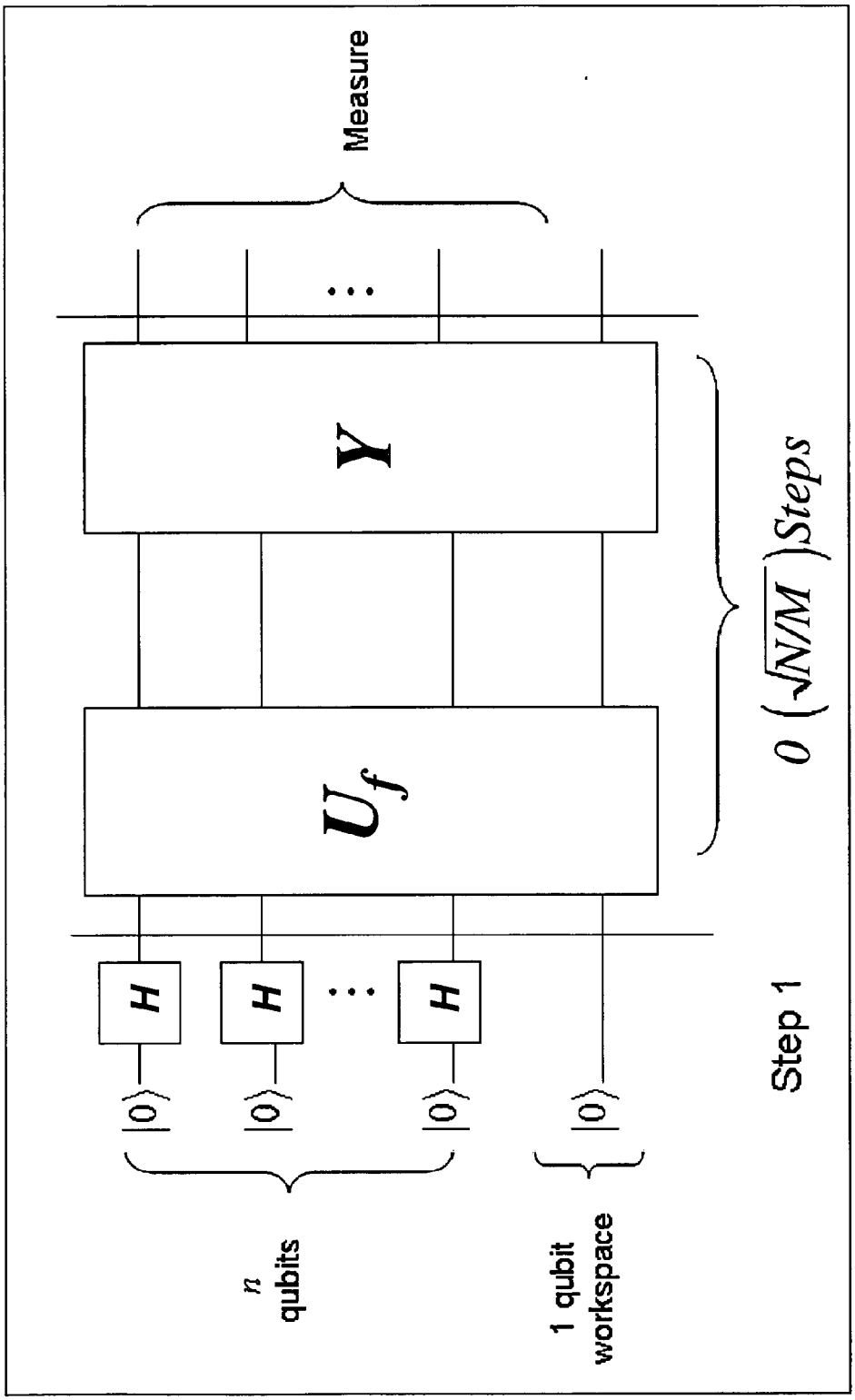

Measure
Y
U_f
$O\left(\sqrt{N/M}\right)$ Steps
Step 1
H
H
H
|0⟩
|0⟩
|0⟩
|0⟩
n qubits
1 qubit workspace

Figure 47

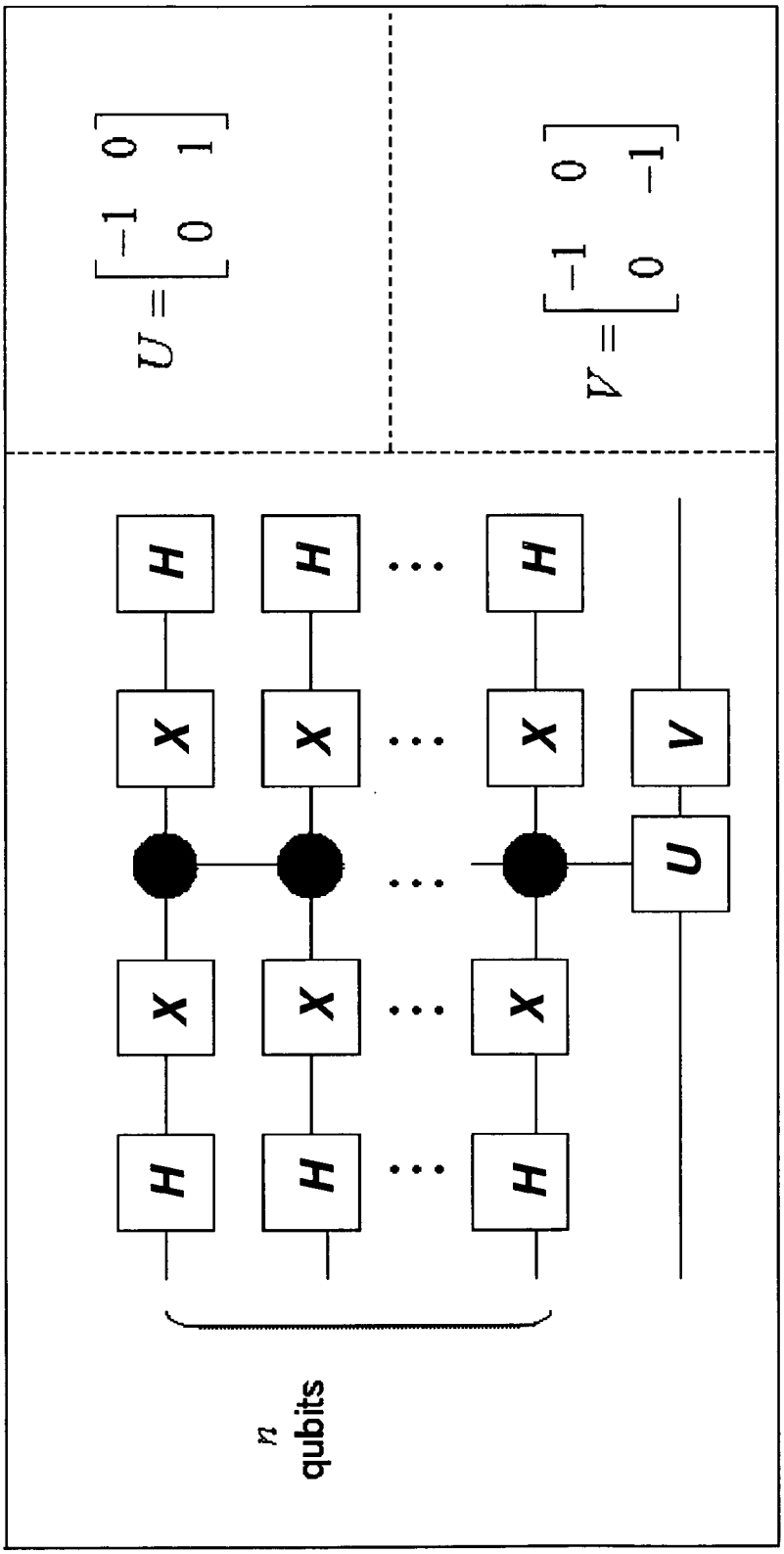

$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
$V = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
H
X
V
U
n qubits

Figure 48

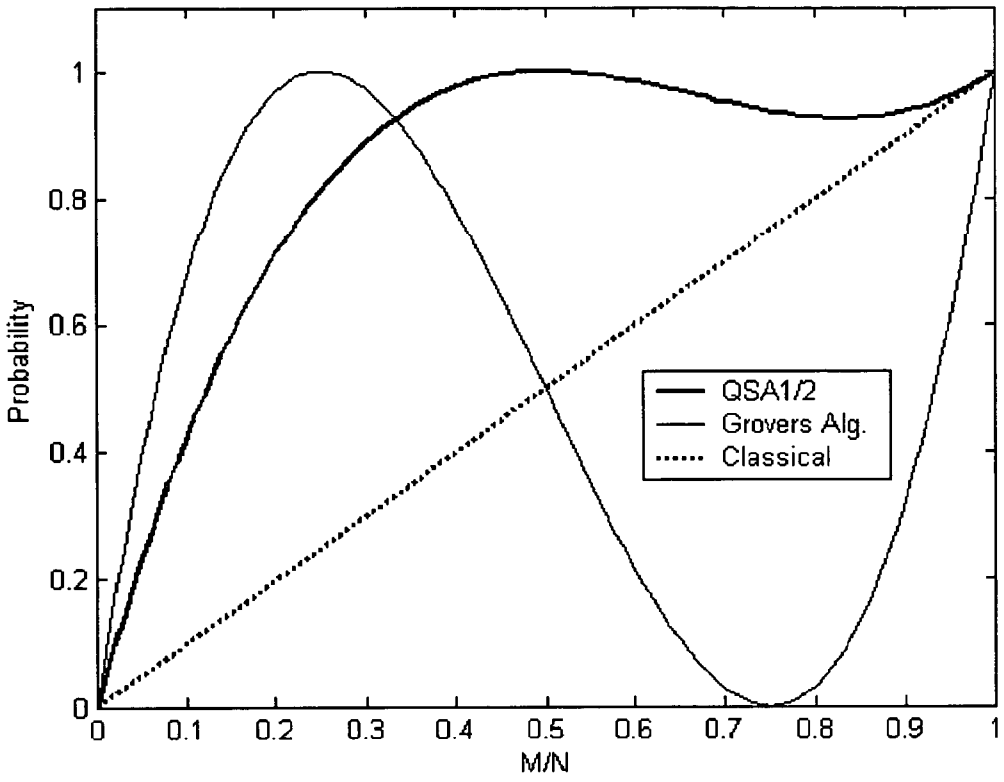

Probability
M/N
QSA1/2
Grovers Alg.
Classical
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9

Figure 49

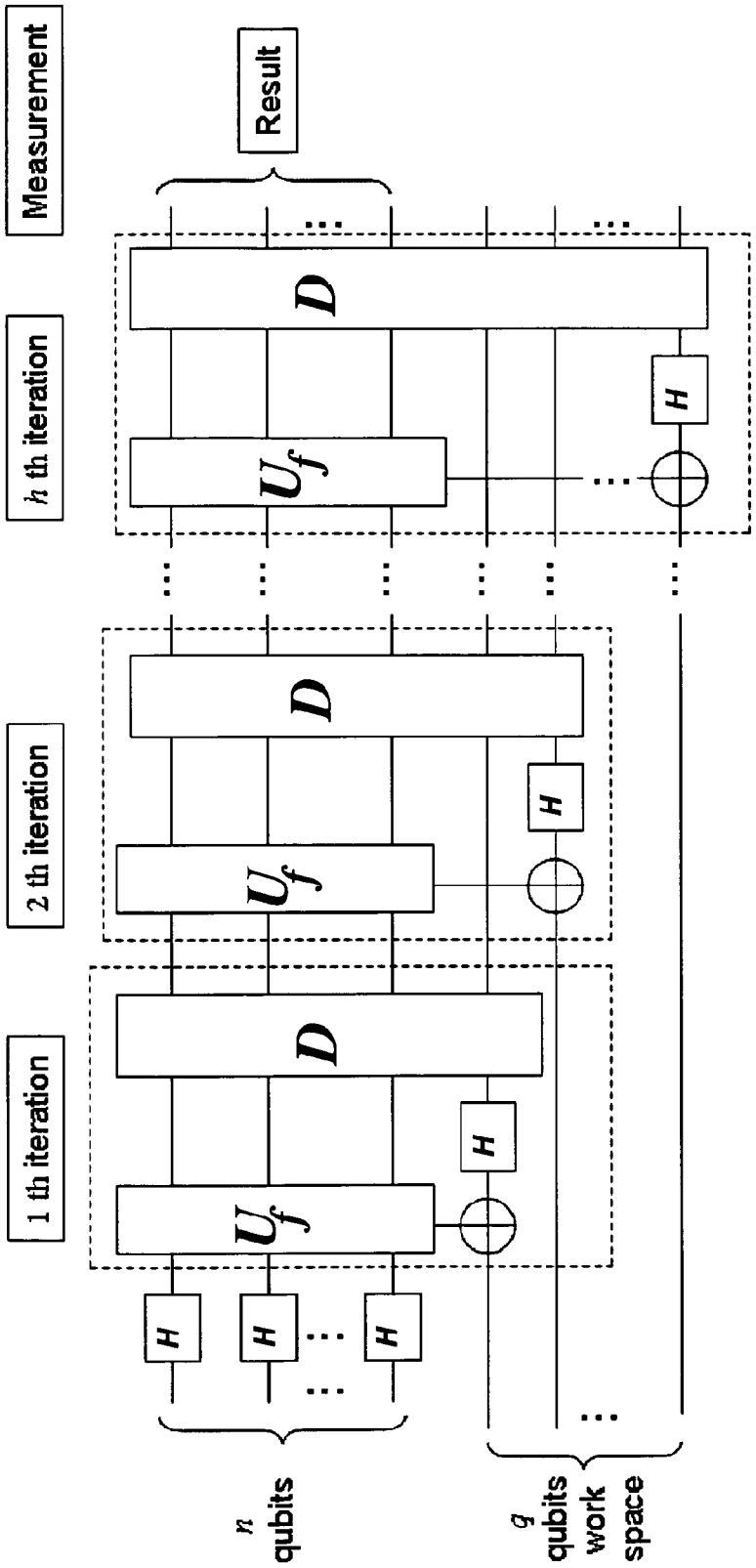

Measurement
Result
h th iteration
2 th iteration
1 th iteration
D
U_f
H
n qubits
q qubits work space

Figure 50

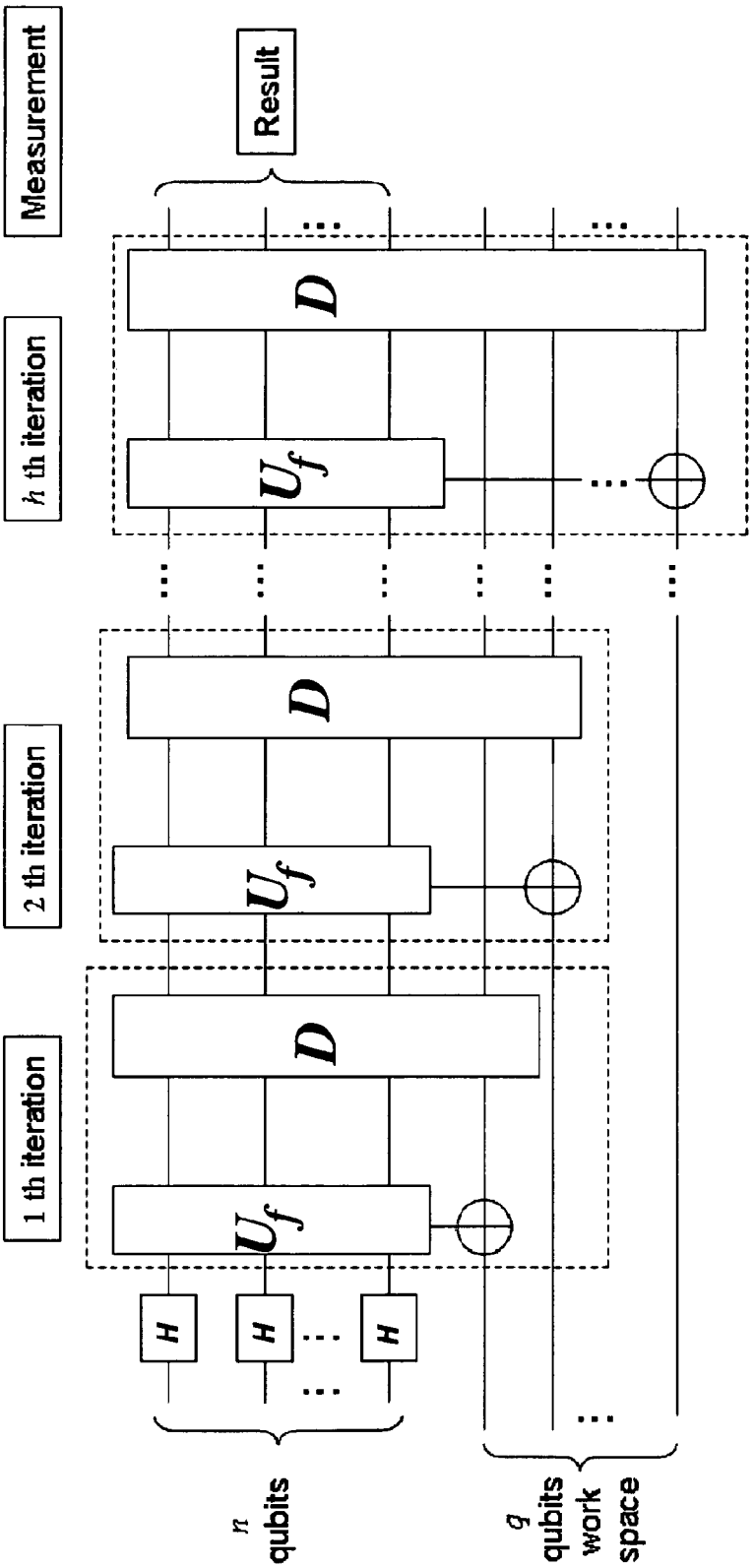

Measurement
Result
h th iteration
2 th iteration
1 th iteration
D
U_f
H
n qubits
q qubits work space

Figure 51

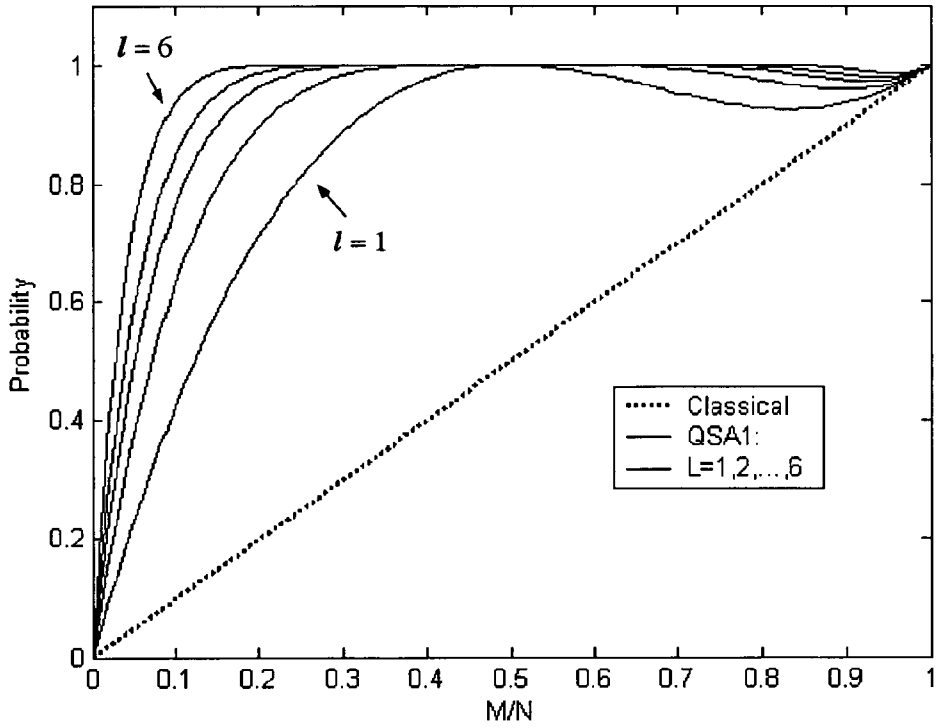

l = 6
l = 1
Probability
M/N
Classical
QSA1:
L=1,2,...,6
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9

Figure 52

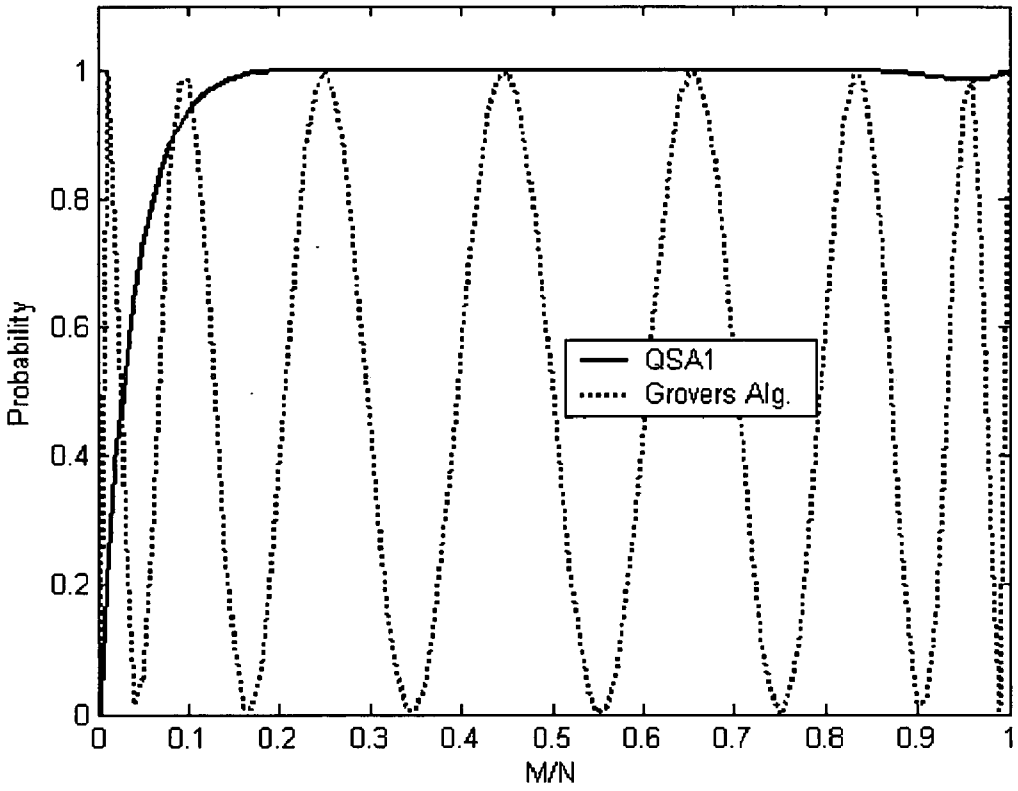


Figure 53

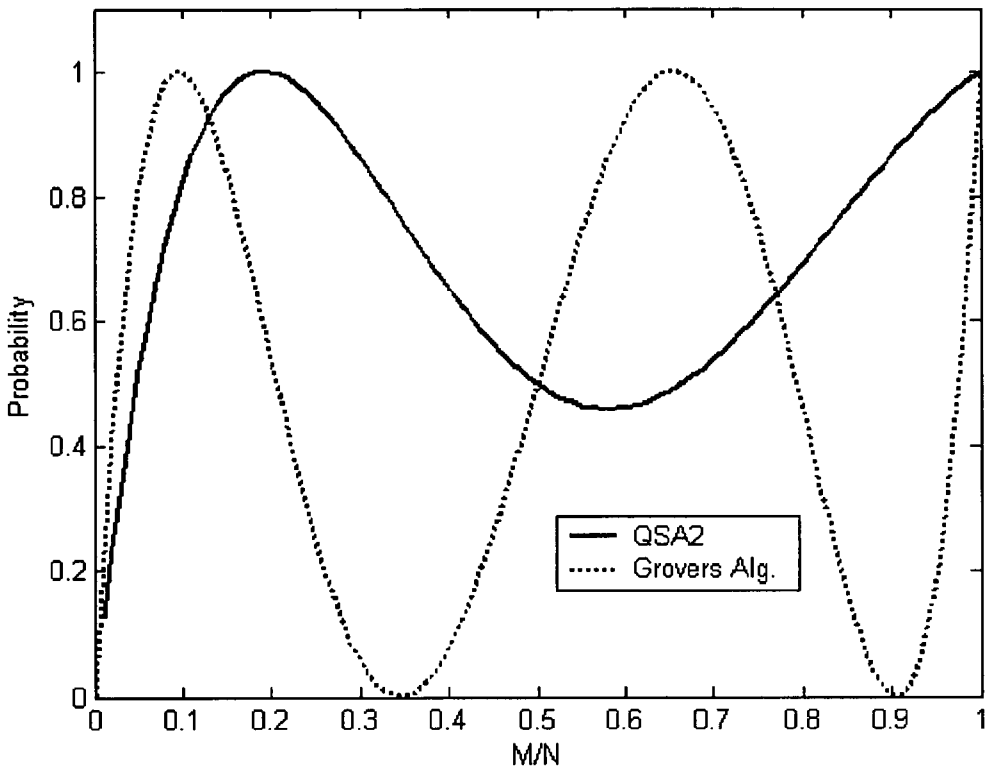


Figure 54

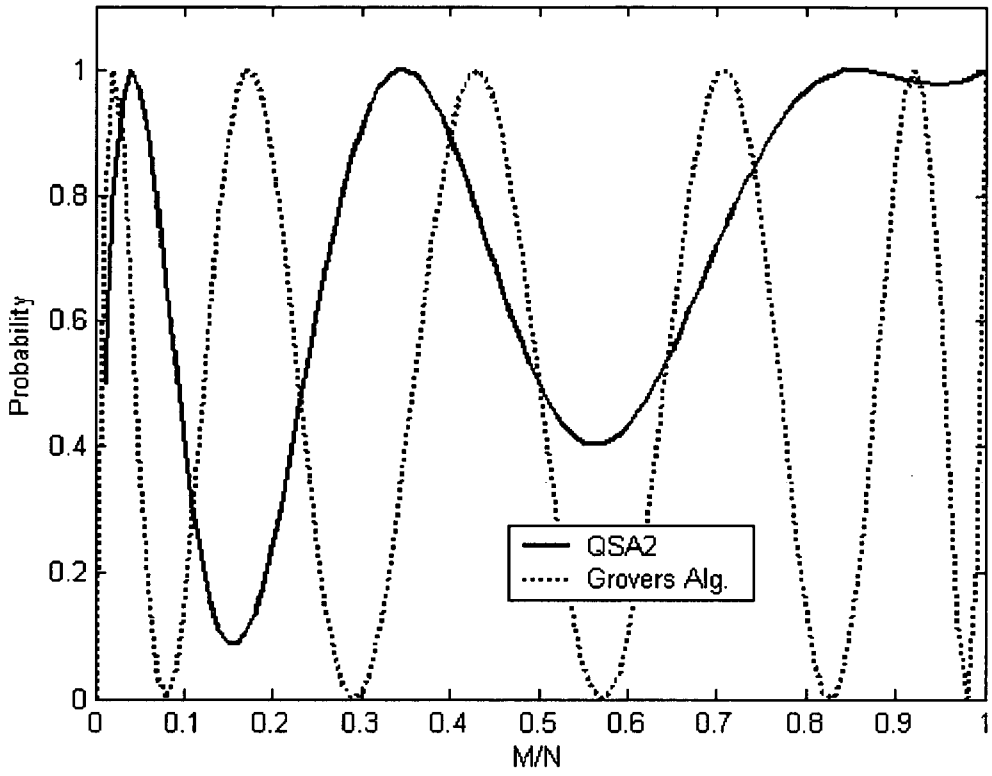


Figure 55

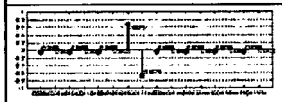
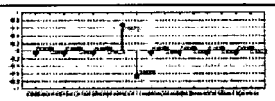
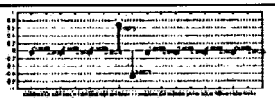
Matrix based approach	Algorithmic approach	Problem oriented approach (emulation, superposition is changed with replication, entanglement is realized as element perturbation, interference as a difference equation)	
		Entire state vector allocation	Sparse state vector allocation
1	2	3	4
Simulation results (4 qubits, state with minimum of Shannon entropy, x_0=011)			
			
Maximum order reached (number of qubits, simulation on PC with single CPU)			
11+1 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is 2^nx2^n)	19+1 (Limited by temporal complexity, we need $6x10^6$ seconds for one iteration with 20 qubits, and $12x10^9$ seconds for 21 qubits).	25 qubit (Limited by spatial complexity required for state vector allocation)	1023+1 without Shannon entropy calculation (Limited by floating point number representation see Figure 3.XX) 64+1 with Shannon entropy calculation (Limited by temporal complexity of calculations)
Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)			
10^3	$3x10^5$	10	~0 (q-bit number independent)
Remarks			
•Possible introduction of excitation; •Generalized for all QA; •High spatial and temporal complexity.	•Relatively high function order; •Required specific R&D for each algorithm; •Applicable for hardware realization; •No floating point operations (preparation and entropy calculation only); •Additional excitations may double temporal complexity.	•Practical application is impossible; •Can be used for state values estimation for high order functions; •Impossible introduction of excitations (excitations will cause exponential complexity and algorithm will vanish to 1st approach complexity) ; •Applicable only to few algorithms; •Simplest hardware realization.	

Figure 56A

Matrix based approach	Algorithmic approach	Problem oriented approach (emulation, superposition is changed with replication, entanglement is realized as element perturbation, interference as a difference equation)	
		Entire state vector allocation	Sparse state vector allocation
1	2	3	4
Maximum order reached (number of qubits, simulation on PC with single CPU)			
11+1 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is $2^n x 2^n$)	19+1 (Limited by temporal complexity)	*Requires more R&D*	> 1000 (Limited by floating point number representation)
Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)			
10^3	10^6	-	~0 (q-bit number independent)
Remarks			
•Possible introduction of excitation; •Generalized for all QA; •High spatial and temporal complexity.	•Relatively high function order; •Required specific R&D for each algorithm; •Applicable for hardware realization; •No floating point operations (preparation and entropy calculation only); •Additional excitations may double temporal complexity.	•Practical application is impossible; •Can be used for state values estimation for high order functions; •Impossible introduction of excitations (excitations will cause exponential complexity and algorithm will vanish to 1st approach complexity) ; •Applicable only to few algorithms; •Simplest hardware realization.	

Figure 56B

Matrix based approach	Algorithmic approach
1	2
Maximum order reached (number of qubits, simulation on PC with single CPU)	
5 + 5 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is 2^{n+m})	10+10 (Limited by temporal complexity)
Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)	
10^2	10^5
Remarks	
•Possible introduction of excitation; •Generalized for all QA; •High spatial and temporal complexity.	•Relatively high function order; •Required specific R&D for each algorithm; •Applicable for hardware realization; •No floating point operations (preparation and entropy calculation only); •Additional excitations may double temporal complexity.

Figure 56C

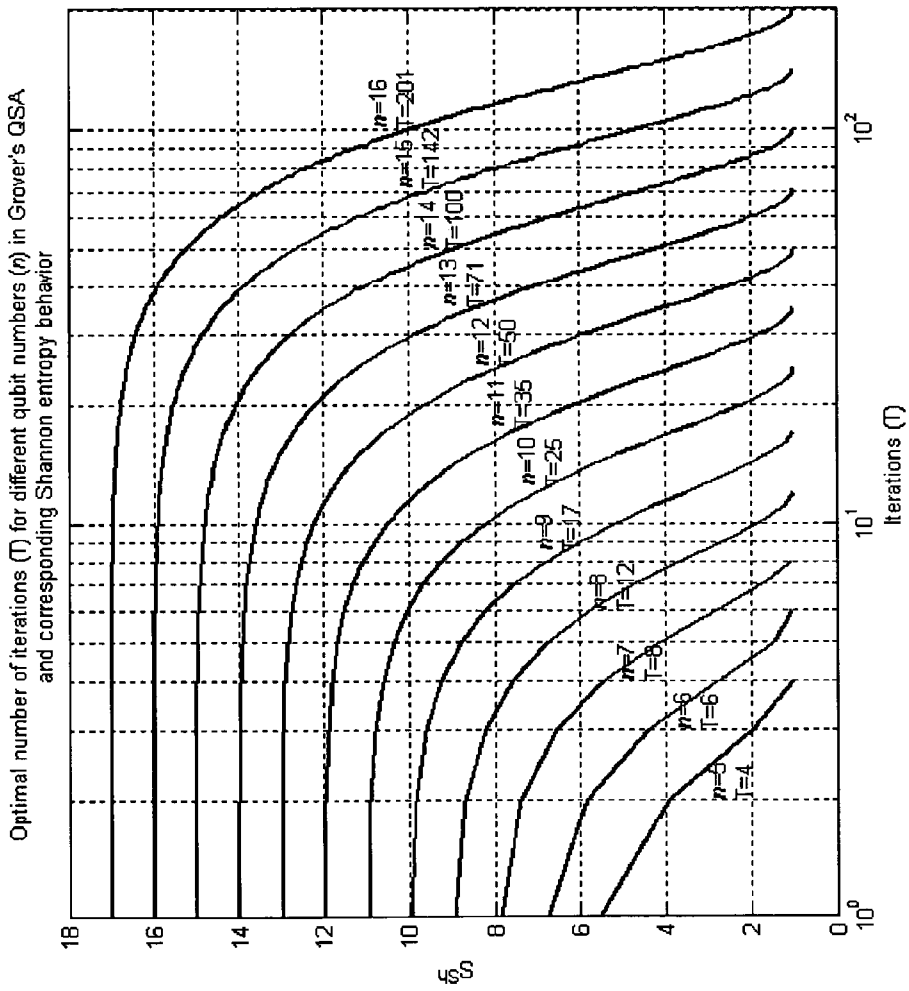

Optimal number of iterations (T) for different qubit numbers (n) in Grover's QSA
and corresponding Shannon entropy behavior
n=16 T=201
n=15 T=142
n=14 T=100
n=13 T=71
n=12 T=50
n=11 T=35
n=10 T=25
n=9 T=17
n=8 T=12
n=7 T=8
n=6 T=6
n=5 T=4
Iterations (T)
Sn

Figure 57A

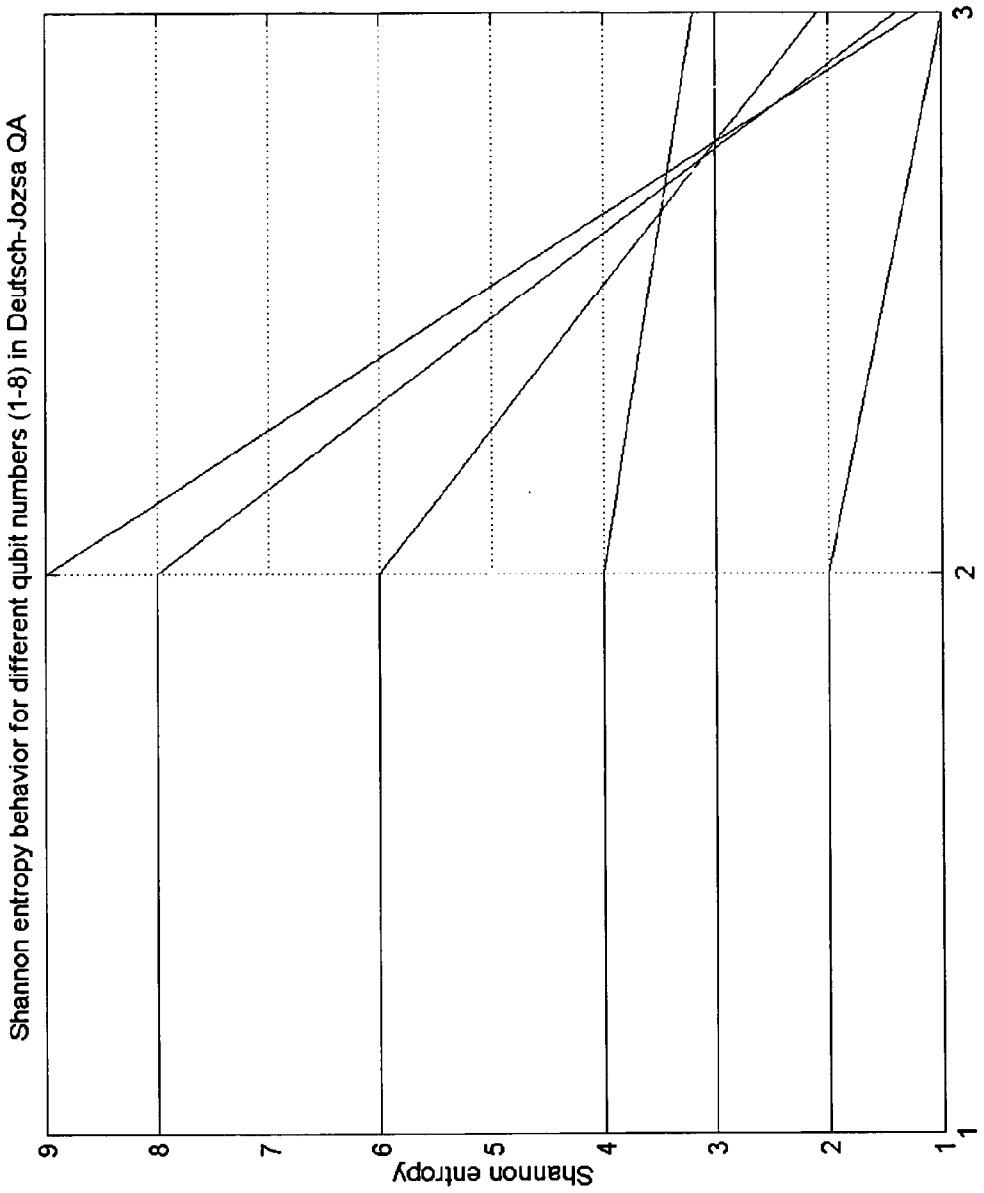

Shannon entropy behavior for different qubit numbers (1-8) in Deutsch-Jozsa QA
Shannon entropy
Quantum steps
9
8
7
6
5
4
3
2
1
1
2
3

Figure 57B

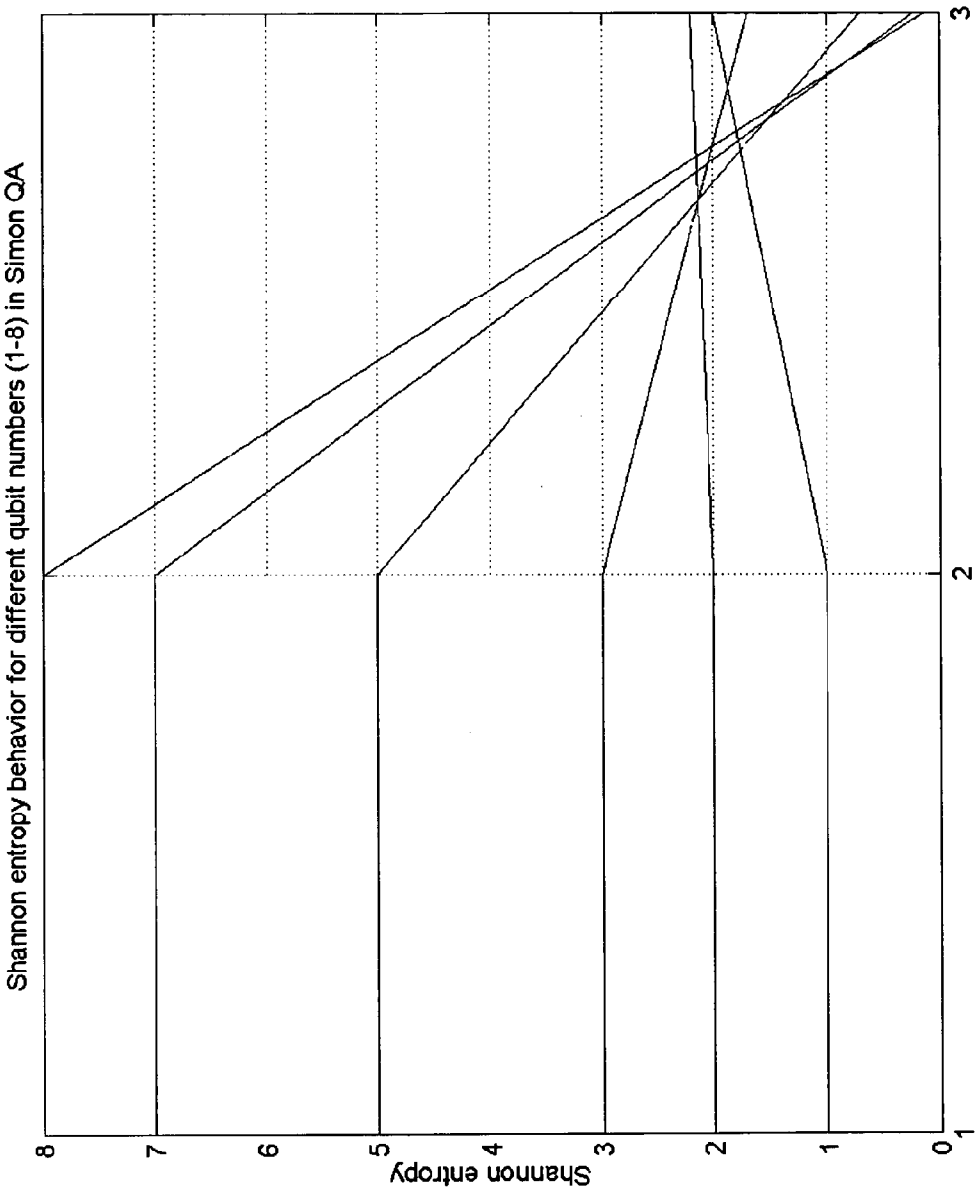

Shannon entropy behavior for different qubit numbers (1-8) in Simon QA
Shannon entropy
Quantum steps
0
1
2
3
4
5
6
7
8
1
2
3

Figure 57C

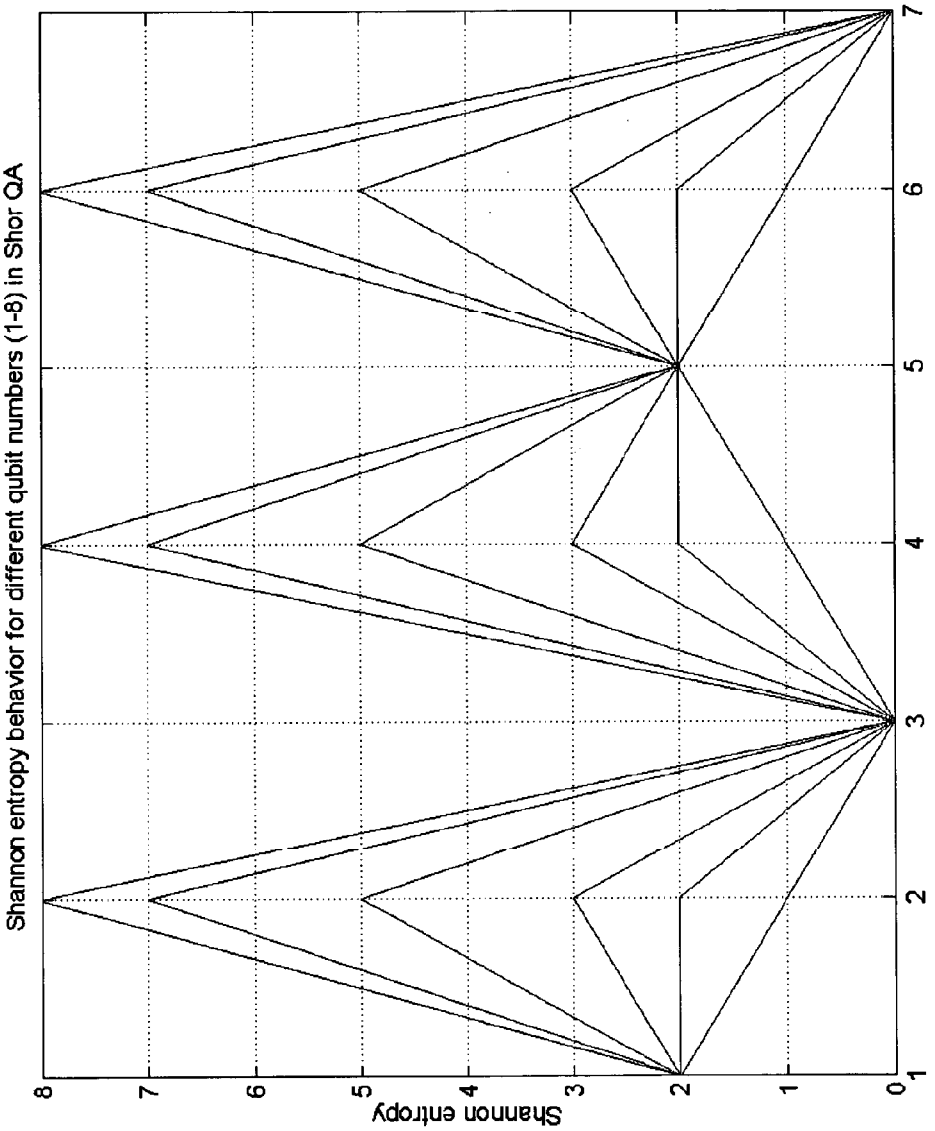

Shannon entropy behavior for different qubit numbers (1-8) in Shor QA
Shannon entropy
Iterations
0
1
2
3
4
5
6
7
8

Figure 57D

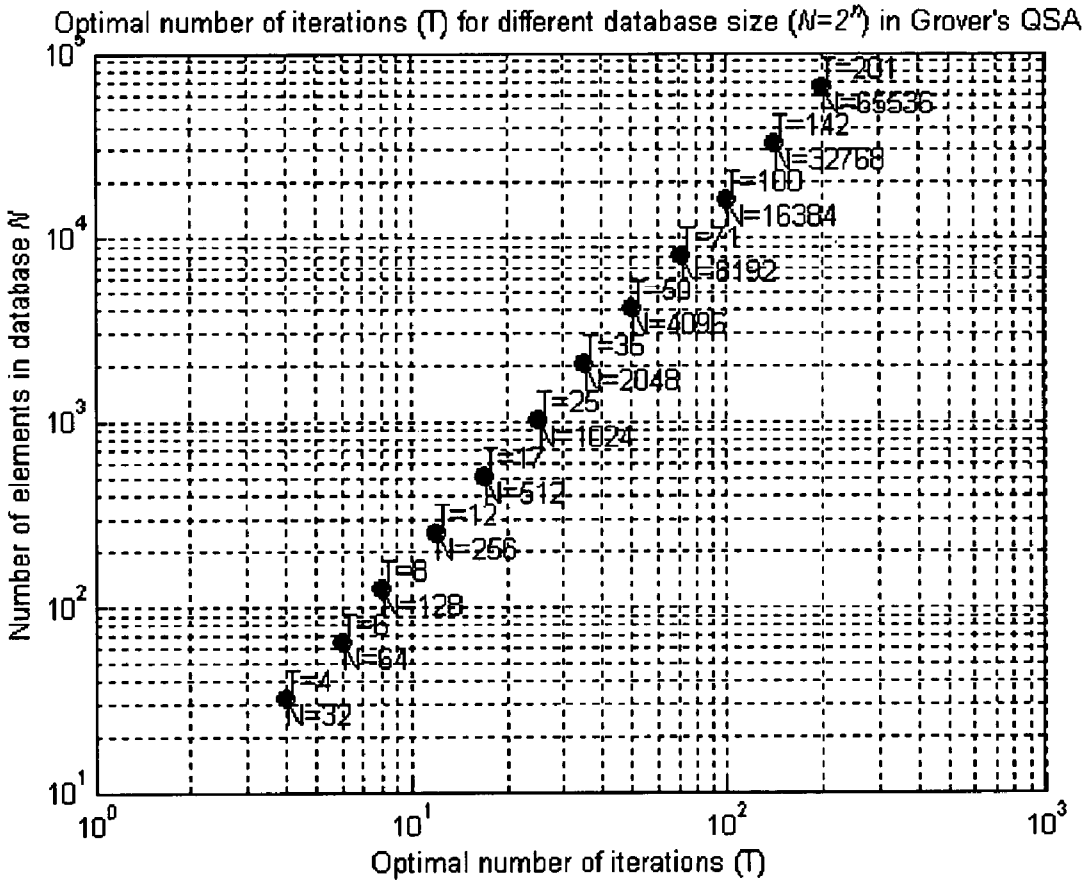

Optimal number of iterations (T) for different database size (N=2^n) in Grover's QSA
Number of elements in database N
Optimal number of iterations (T)
T=201
N=65536
T=142
N=32768
T=100
N=16384
T=71
N=8192
T=50
N=4096
T=35
N=2048
T=25
N=1024
T=17
N=512
T=12
N=256
T=8
N=128
T=6
N=64
T=4
N=32

Figure 58

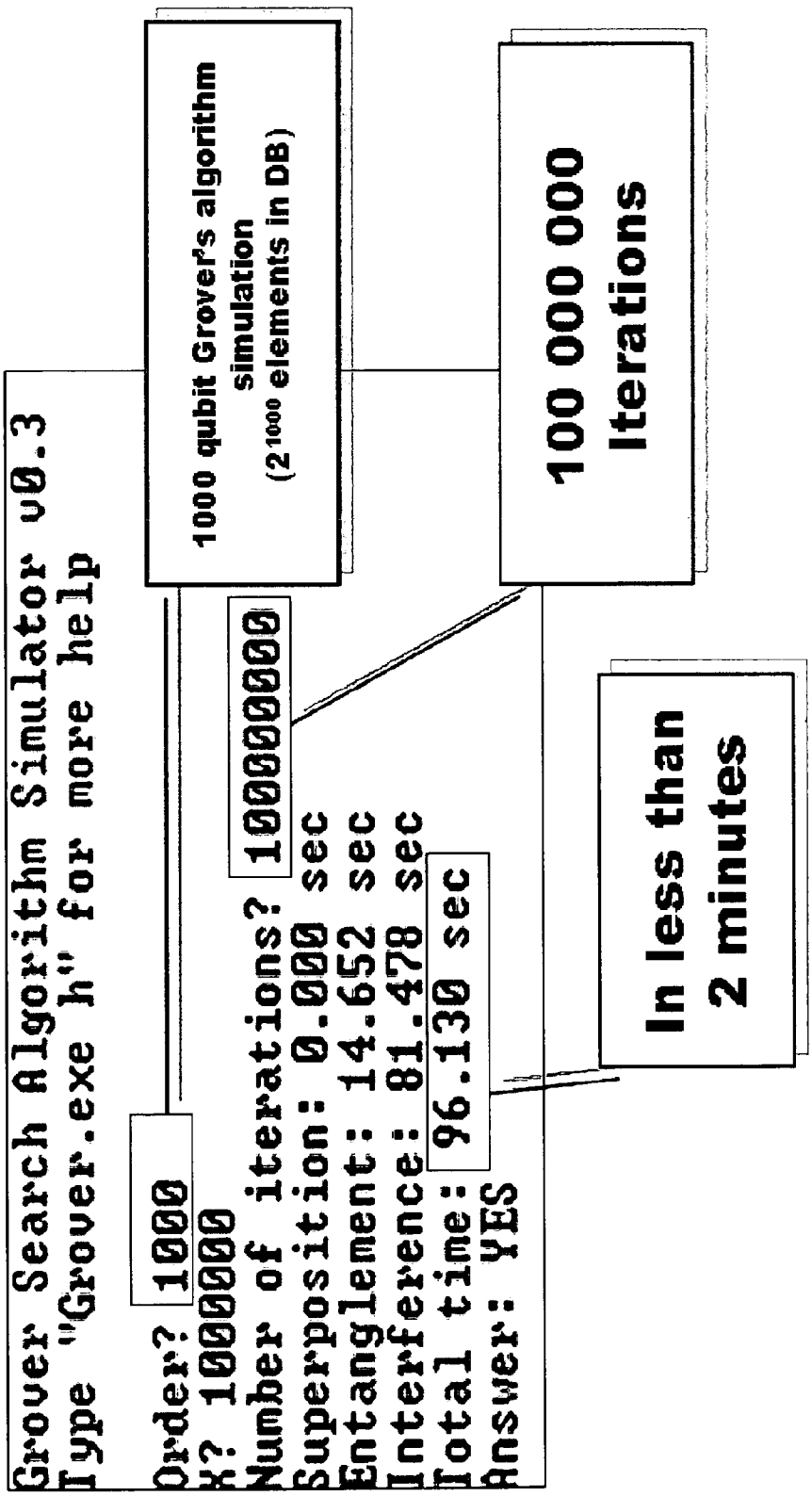

Grover Search Algorithm Simulator v0.3
Type "Grover.exe h" for more help
Order? 1000
k? 1000000
Number of iterations? 100000000
Superposition: 0.000 sec
Entanglement: 14.652 sec
Interference: 81.478 sec
Total time: 96.130 sec
Answer: YES
1000 qubit Grover's algorithm simulation
(2^1000 elements in DB)
100 000 000 Iterations
In less than 2 minutes

Figure 59

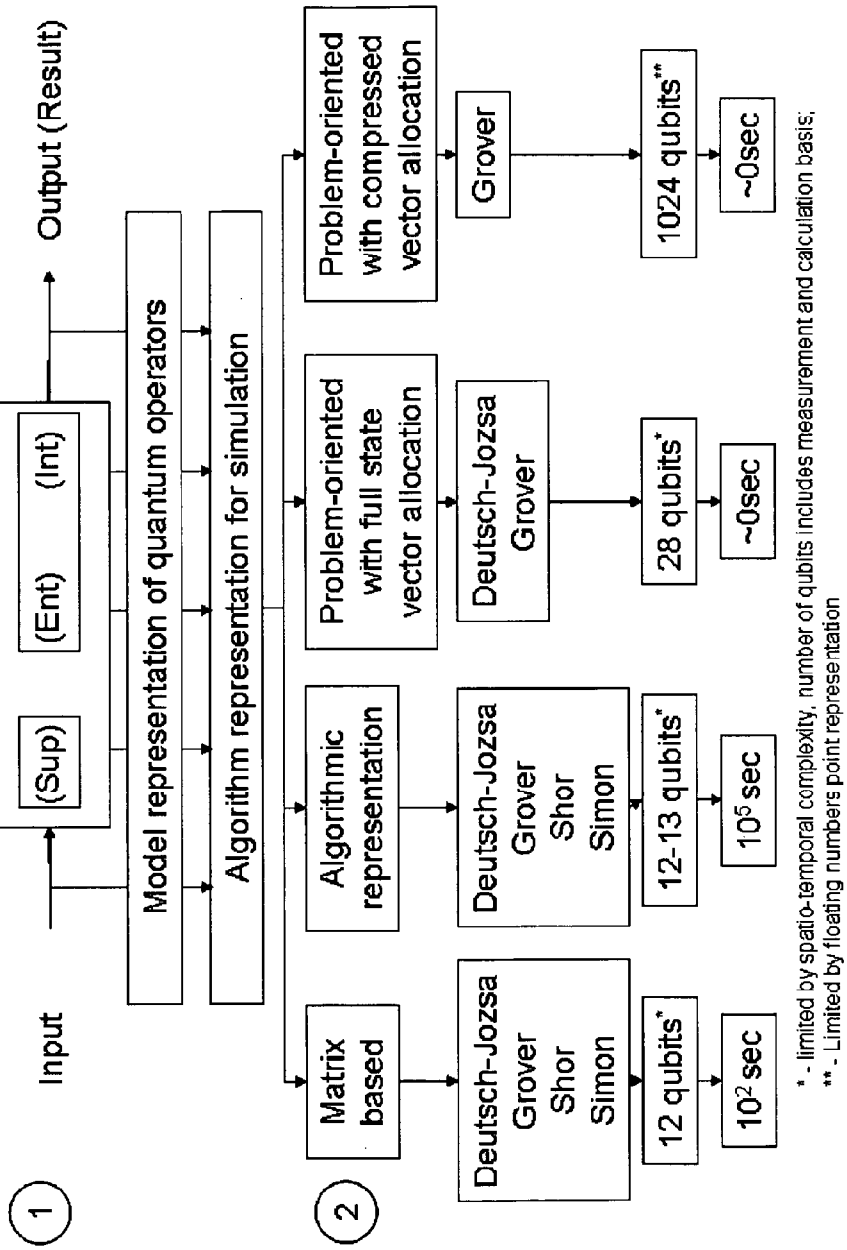

1
Input
QA-Gate
(Sup)
(Ent)
(Int)
Output (Result)
Model representation of quantum operators
Algorithm representation for simulation
2
Matrix based
Algorithmic representation
Problem-oriented with full state vector allocation
Problem-oriented with compressed vector allocation
Deutsch-Jozsa Grover Shor Simon
Deutsch-Jozsa Grover Shor Simon
Deutsch-Jozsa Grover
Grover
12 qubits*
12-13 qubits*
28 qubits*
1024 qubits**
10^2 sec
10^5 sec
~0sec
~0sec
* - limited by spatio-temporal complexity, number of qubits includes measurement and calculation basis;
** - Limited by floating numbers point representation

Figure 60

EFFICIENT SIMULATION SYSTEM OF QUANTUM ALGORITHM GATES ON CLASSICAL COMPUTER BASED ON FAST ALGORITHM

BACKGROUND

[0001] 1. Field of invention

[0002] The present invention relates to efficient simulation of quantum algorithms using classical computers with a Von Neumann architecture.

[0003] 2. Description of the Related Art

[0004] Quantum algorithms (QA) hold great promise for solving many heretofore intractable problems where classical algorithms are inefficient. For example, quantum algorithms are particularly suited to factorization and/or searching problems where the computational complexity increases exponentially when using classical algorithms. Use of quantum algorithms on true quantum computers is, however, rare because there is currently no practical physical hardware implementation of a quantum computer. All quantum computers to date have been too primitive for practical use.

[0005] The difference between a classical algorithm and a QA lies in the way that the QA is coded in the structure of the quantum operators. The initial input to the QA is a quantum register loaded with a superposition of initial states. The output of the QA is a function of the problem being solved. In some sense, the QA is given a problem to analyze and the QA returns its qualitative property in quantitative form as an answer. Formally, the problems solved by a QA can be stated as follows:

[0006] Input: A function f: $\{0,1\}^n \rightarrow \{0,1\}^m$

[0007] Problem: Find a certain property of f

[0008] Thus, the QA studies some qualitative properties of a function. The core of any QA is a set of unitary quantum operators or quantum gates. A quantum gate is a unitary matrix with a particular structure related to the algorithm needed to solve the given problem. The size of this matrix grows exponentially with the number of inputs, making it difficult to simulate a QA with more than 30-35 inputs on a classical computer with a Von Neumann architecture because of the memory required and the computational complexity of dealing with such a large matrix.

SUMMARY

[0009] The present invention solves these and other problems by providing an efficient simulation system of quantum algorithm gates and for classical Von Neumann computers. In one embodiment, a QA is solved using a matrix-based approach. In one embodiment, a QA is solved using an algorithmic-based approach wherein matrix elements of the quantum gate are calculated on demand. In one embodiment, a problem-oriented approach to implementing Grover's algorithm is provided with a termination condition determined by observation of Shannon entropy. In one embodiment, a QA is solved by using a reduced number of operators.

[0010] In one embodiment, at least some of the matrix elements of the QA gate are calculated as needed, thus avoiding the need to calculate and store the entire matrix. In this embodiment, the number of inputs that can be handled is affected by: (i) the exponential growth in the number of operations used to calculate the matrix elements; and (ii) the size of the state vector stored in the computer memory.

[0011] In one embodiment, the structure of the QA is used to provide an efficient algorithm. In Grover's QSA, the state vector always has one of the two different values: (i) one value corresponds to the probability amplitude of the answer; and (ii) the second value corresponds to the probability amplitude of the rest of the state vector. In one embodiment, two values are used to efficiently represent the floating-point numbers that simulate actual values of the probability amplitudes in the Grover's algorithm. For other QAs, more than two, but nevertheless a finite number of values will exist and such finiteness is used to provide an efficient algorithm.

[0012] In one embodiment, the QA is constructed or transformed such that entanglement and interference operators can by bypassed or simplified, and the result is computed based on superposition of the initial states (and deconstructive interference of final output patterns) representing the state of the designed schedule of control gains. In one embodiment, the Deutsch-Jozsa's algorithm, when entanglement is absent, is simulated by using pseudo-pure quantum states. In one embodiment, the Simon algorithm, when entanglement is absent, is simulated by using pseudo-pure quantum states. In one embodiment, an entanglement-free QA is used to optimize an intelligent control system.

BRIEF DESCRIPTION OF THE FIGURES

[0013] **FIG. 1** shows memory used versus the number of qubits in a MATLAB 6.0 simulation environment used for modeling quantum search algorithm.

[0014] **FIG. 2** shows the time required to make a fixed number of iterations as a function of processor clock frequency on a computer with a Pentium III processor.

[0015] **FIG. 3** shows a family of curves from **FIG. 2** for 100 iterations.

[0016] **FIGS. 4***a* and 4*b* show surface plots of the time required for a fixed number of iterations versus the number of qbits using processors of different internal frequency.

[0017] **FIG. 5** shows a family of curves from **FIG. 4** for 10 iterations.

[0018] **FIG. 6** shows the time for one iteration of 11 qubits, including curves for computations only and computation plus virtual memory operations.

[0019] **FIG. 7** shows the time for one iteration as a function of the number of qubits.

[0020] **FIG. 8** shows comparisons of the memory needed for the Shor and Grover algorithms.

[0021] **FIG. 9** shows the time required for a fixed number of iterations versus the number of qubits and versus the processor clock frequency.

[0022] **FIG. 10** shows the time required for 10 iterations with different clock frequencies.

[0023] **FIG. 11** shows the time required for one iteration as a function of the number of qubits.

[0024] **FIG. 12** shows the time versus number of iterations and versus the number of qbits for the Shor and Grover algorithms.

[0025] **FIG. 13** shows curves from **FIG. 12** for 10 iterations.

[0026] **FIG. 14** shows the spatial complexity of a quantum algorithm.

[0027] **FIG. 15** shows the difference between two quantum algorithms due to demands on the processor front side bus.

[0028] **FIG. 16** shows computational runtime differences between the Shor, Grover, and Deutch-Josza algorithms.

[0029] **FIG. 17***a* shows a generalized representation of a QA as a set of sequentially-applied smaller quantum gates.

[0030] **FIG. 17***b* shows an alternate representation of a QA.

[0031] **FIG. 18***a* shows a quantum state vector set up to an initial value.

[0032] **FIG. 18***b* shows the quantum state vector of **FIG. 18***a* after the superposition operator is applied.

[0033] **FIG. 18***c* shows the quantum state vector of **FIG. 18***b* after the entanglement operation in Grover's algorithm

[0034] **FIG. 18***d* shows the quantum state vector of **FIG. 18***c* after application of the interference operation.

[0035] **FIG. 19***a* shows the dynamics of Grover's QSA probabilities of the input state vector.

[0036] **FIG. 19***b* shows the dynamics of Grover's QSA probabilities of the state vector after superposition and entanglement.

[0037] **FIG. 19***c* shows the dynamics of Grover's QSA probabilities of the state vector after interference.

[0038] **FIG. 20** shows the Shannon information entropy calculation for the Grover's algorithm with 5 inputs.

[0039] **FIG. 21** shows spatial complexity of a Grover QA simulation.

[0040] **FIG. 22** shows temporal complexity of Grover's QSA.

[0041] **FIG. 23** shows Shannon entropy simulation of a QSA with 7-inputs.

[0042] **FIG. 24***a* shows the superposition operator representation algorithm for Grover's QSA.

[0043] **FIG. 24***b* shows an entanglement operator representation algorithm for Grover's QSA.

[0044] **FIG. 24***c* shows an interference operator representation algorithm for Grover's QSA.

[0045] **FIG. 24***d* shows an interference operator representation algorithm for Deutsch-Jozsa's QA.

[0046] **FIG. 24***e* shows an entanglement operator representation algorithm for Simon's and Shor's QA.

[0047] **FIG. 24***f* shows the superposition and interference operator representation algorithm for Simon's QA.

[0048] **FIG. 24***g* shows an interference operator representation algorithm for Shor's QA.

[0049] **FIG. 25** shows state vector representation algorithm for Grover's quantum search.

[0050] **FIG. 26** shows a generalized schema of simulation for Grover's QSA.

[0051] **FIG. 27** shows the superposition block for Grover's QSA.

[0052] **FIG. 28***a* shows emulation of the entanglement operator application of Grover's QSA.

[0053] **FIG. 28***b* shows emulation of interference operator application of Grover's QSA.

[0054] **FIG. 28***c* shows the quantum step block for Grover's quantum search.

[0055] **FIG. 29** shows the termination block for method 1.

[0056] **FIG. 30** shows component B for the termination block.

[0057] **FIG. 31***a* shows component PUSH for the termination block.

[0058] **FIG. 31***b* shows component POP for the termination block.

[0059] **FIG. 32** shows component C for the termination block.

[0060] **FIG. 33** shows component D for the termination block.

[0061] **FIG. 34** shows component E for the termination block.

[0062] **FIG. 35** shows final measurement emulation.

[0063] **FIG. 36** shows a generalized schema of simulation for Deutsch-Jozsa's QA.

[0064] **FIG. 37** shows a quantum block HUD for Deutsch-Jozsa's QA.

[0065] **FIG. 38** shows a generalized approach for QA simulation.

[0066] **FIG. 39** shows query processing.

[0067] **FIG. 40** shows a general structure of Quantum Soft Computing tools.

[0068] **FIG. 41***a* is a block diagram of an intelligent nonlinear control system.

[0069] **FIG. 41***b* shows a superposition of coefficient gains.

[0070] **FIG. 42** shows the structure of the design process.

[0071] **FIG. 43** shows robust KB design with a quantum algorithm.

[0072] **FIG. 44***a* shows coefficient gains of a Q-PD controller.

[0073] **FIG. 44***b* shows coefficient gains scheduled by a FC trained using Gaussian excitation.

[0074] **FIG. 44***c* shows coefficient gains scheduled by a FC trained using non-Gaussian excitation.

[0075] **FIG. 44***d* shows control object dynamics.

[0076] **FIG. 45** shows simulation result of the **FIG. 44***b*, under non-gaussian excitation.

[0077] **FIG. 46** shows the addition of a new Hadamard operator, as example, between the oracle (entanglement) and the diffusion operators in Grover's QSA.

[0078] **FIG. 47** shows the steps of QSA2.

[0079] **FIG. 48** shows one embodiment if a circuit implementation using elementary gates. The probability of finding a solution varies according to the number of matches M≠0 in the superposition.

[0080] **FIG. 49** shows the probability of success of the QSA1 and QSA2 algorithms after one iteration.

[0081] **FIG. 50** shows the iterating version of the algorithm QSA1.

[0082] **FIG. 51** shows the iterating version of the QSA2 algorithm.

[0083] **FIG. 52** shows the probability of success of the iterative version of the QSA1 algorithm.

[0084] **FIG. 53** shows the probability of success of the iterative version of the algorithm QSA1 after five iterations.

[0085] **FIG. 54** shows the probability of success of the iterative version of the QSA2 algorithm.

[0086] **FIG. 55** shows the probability of success of the iterative version of the QSA2 algorithm after five iterations.

[0087] **FIG. 56***a* shows results from different approaches for simulation of Grover's QSA.

[0088] **FIG. 56***b* shows results from different approaches for simulation of Deutsch-Jozsa's QA.

[0089] **FIG. 56***c* shows results from different approaches for simulation of Simon's and Shor's quantum algorithms.

[0090] **FIG. 57***a* shows the optimal number of iterations for different qubit numbers and corresponding Shannon entropy behavior of Grover's QSA simulation.

[0091] **FIG. 57***b* shows results of Shannon entropy behavior for different qubit numbers (1-8) in Deutsch-Jozsa's QA.

[0092] **FIG. 57***c* shows results of Shannon entropy behavior for different qubit numbers (1-8) in Simon's QA.

[0093] **FIG. 57***d* shows results of Shannon entropy behavior for different qubit numbers (1-8) in Shor's QA.

[0094] **FIG. 58** shows the optimal number of iterations for different database sizes.

[0095] **FIG. 59** shows simulation results of problem oriented Grover QSA according to approach **4** with 1000 qubits.

[0096] **FIG. 60** summarizes different approaches for QA simulation.

DETAILED DESCRIPTION

[0097] The simplest technique for simulating a Quantum Algorithm (QA) is based on the direct representation of the quantum operators. This approach is stable and precise, but it requires allocation of operator's matrices in the computer's memory. Since the size of the operators grows exponentially, this approach is useful for simulation of QAs with a relatively small number of qubits (e.g., approximately 11 qubits on a typical desktop computer). Using this approach it is relatively simple to simulate the operation of a QA and to perform fidelity analysis.

[0098] In one embodiment, a more efficient fast quantum algorithm simulation technique is based on computing all or part of the operator matrices on an as-needed basis. Using this technique, it is possible to avoid storing all or part of the operator matrices. In this case, the number of qubits that can be simulated (e.g., the number of input qubits, or the number of qubits in the system state register) is affected by: (i) the exponential growth in the number of operations required to calculate the result of the matrix products; and (ii) the size of the state vector that is allocated in computer memory. In one embodiment, using this approach it is reasonable to simulate up to 19 or more qubits on typical desktop computer, and even more on a system with vector architecture.

[0099] Due to particularities of the memory addressing and access processes in a typical desktop computer (such as, for example, a Pentium-based Personal Computer), when the number of qubits is relatively small, the compute-on-demand approach tends to be faster than the direct storage approach. The compute-on-demand approach benefits from a study of the quantum operators, and their structure so that the matrix elements can be computed more efficiently.

[0100] The study portion of the compute-on-demand approach can, for some QAs lead to a problem-oriented approach based on the QA structure and state vector behavior. For example, in Grover's Quantum Search Algorithm (QSA), the state vector always has one of the two different values: (i) one value corresponds to the probability amplitude of the answer; and (ii) the second value corresponds to the probability amplitude of the rest of the state vector. Using this assumption, it is possible to configure the algorithm using these two different values, and to efficiently simulate Grover's QSA. In this case, the primary limit is a representation of the floating-point numbers used to simulate the actual values of the probability amplitudes. After the superposition operation, these probability amplitudes are very small

$$\left(\frac{1}{2^{n/2}}\right).$$

Thus, it is possible to simulate Grover's QSA with this approach simulating 1024 qubits or more without termination condition calculation and up to 64 qubits or more with termination condition estimation based on Shannon entropy.

[0101] Other QAs do not necessarily reduce to just two values. For those algorithms that reduce to a finite number of values, the techniques used to simplify the Gover QSA can be used, but the maximum number of input qubits that can be simulated will tend to be smaller, because the probability amplitudes of other algorithms have relatively more complicated distributions. Introduction of an external excitation can decrease the possible number of qubits for some algorithms.

[0102] In some algorithms, the entanglement and interference operators can be bypassed (or simplified), and the output computed based only on a superposition of the initial

states (and deconstructive interference of the final output patterns) representing the state of the designed schedule of control gains. For example, a particular case of Deutsch-Jozsa's and Simon algorithms can be made entanglement free by using pseudo-pure quantum states.

[0103] The disclosure that follows begins with a comparative analysis of the temporal complexity of several representative QAs. That analysis is followed by an introduction of the generalized approach in QA simulation and algorithmic representation of quantum operators. Subsequent portions describe the structure representation of the QAs applicable to low level programming on classical computer (PC), generalizations of the approaches and introduction of the general QA simulation tool based on fast problem-oriented QAs. The simulation techniques are then applied to a quantum control algorithm.

1. Spatio-Temporal Complexity of QA Simulation Based on the Full Matrix Approach

I. Spatio-Temporal Complexity of Grover's Quantum Algorithm

1.1. Introduction

[0104] Practical realization of quantum search algorithms on classical computers is limited by the available hardware resources. Well-known algorithmic estimations for the number database transactions required by the Grover search algorithm cannot be considered directly on von Neumann computers. Classical versions of QAs depend on the effectiveness and efficiency of the mathematical models used to simulate the quantum-mechanical operations.

[0105] Thus, it is useful to analyze quantum algorithms to determine, or at least estimate, time expenses, influence of processor clock frequency, memory requirements, and Shannon entropy behavior of the QA. Evaluating time expenses of the Grover QSA includes evaluating the number of oracle queries (temporal complexity) for a fixed number of iterations of the Grover's QSA as a function of the number of qubits. Evaluating the effect of the central processor clock time includes estimating the influence of the central processor frequency on the time required for making a fixed number of iterations. Runtime does not necessarily scale linearly with processor clock speed due to effects of memory access, cache access, processor wait states, processor pipelines, processor branch estimation, etc. The required physical memory size (spatial complexity) depends on the algorithm and the number of qubits. The Shannon entropy behavior provides insight into the number of iterations required to arrive at a solution, and thus provides insight into the temporal complexity of the QA. The understanding gained from examining the spatio-temproral complexity helps in understanding the computing resources needed to simulate a desired QA with a desired number of qubits.

1.2. Computational Examples

[0106] **FIG. 1** shows the memory requirements versus number of qubits for a MATLAB 6.0 simulation environment used for modeling a QSA. **FIG. 1** shows that 128 MB of memory allows simulation of up to 8 qubits (corresponding to 2^8 elements in the database). **FIG. 2** shows the time required to simulate Grover's QSA versus the number of qubits and versus the number of iterations on a Pentium III computer with 128 MB of main memory and processor clock frequencies of 600, 800, and 1000 MHz. **FIG. 3** shows the influence of processor internal frequency on the time required for making 100 iterations (from **FIG. 2**). As shown in **FIG. 3**, the runtime does not scale linearly with processor speed.

[0107] A linear increase of the number of qubits results in an exponential increase in the amount of memory required. In one embodiment, a computer with 512 MB of memory running MATLAB 6.0 is able to simulate 10 qubits before memory limitations begin to dominate. **FIGS. 4 and 5** show runtime versus number of iterations and versus number of qubits (from 8 to 10) for the 512 MB hardware configuration.

[0108] Once the computer physical memory is full, a further increase in the number of qubits causes virtual memory paging and performance degrades rapidly, as shown in **FIG. 6**. **FIG. 6** shows time required for making one iteration of Grover's QSA for 11 qubits on a computer with 512 MB of physical memory—with and without virtual memory operations. As shown in the figure, the time required to perform virtual memory operations accounts for 50-70% of the time required to do calculations only.

[0109] **FIG. 7** shows the exponentially increasing time required for making one iteration versus the number of qubits (from 1 to 11) on a computer with 512 MB physical memory and an Intel Pentium III processor running at 800 MHz. Since the time required for making one iteration grows exponentially as the number of qubits increases, it is useful to determine the minimum number of iterations that guarantees a high probability of obtaining a correct answer.

[0110] The Shannon entropy can be considered as a criteria for solution of the QA-termination problem. Table 1.1 shows tabulated results of the number of qubits, Shannon entropy, and the number of iterations required.

TABLE 1.1

Number of qubit	Shannon entropy	Number of iterations
1	2.0	1
2	1.0	2
3	1.00351	7
4	1.0965	10
4	1.00721	16
5	1.01362	5
6	1.05330	7
6	1.02879	32
7	1.07123	9
7	1.00021	27
8	1.00002	13
9	1.00024	18
10	1.00024	26

[0111] The timing results presented above are provided by way of explanation and for trend analysis, and not by way of limitation. Different programming systems would likely yield different absolute values for the measured quantities, but the trends would nevertheless remain. Thus, several observations can be drawn from the data shown in **FIGS. 1-7**. According to contemporary standards of personal computer hardware, QSAs can be adopted for relatively small databases (up to 2^{11}-2^{12} elements). For a system with more than 2 qubits, the correct result calculation correlates with achieving a minimum value of Shannon entropy. Thus, the

minimum number of iterations needed to achieve a desired accuracy can be estimated from the number of qubits.

II. Temporal complexity of Grover's quantum algorithm in comparison with Shor's QA

2.1. Introduction

[0112] The results in **FIGS. 1-7** were obtained by simulating Grover's QSA. **FIG. 8** shows a comparison of the memory used by Shor's algorithm as compared to Grover's algorithm for 1 to 5 qubits. As shown in **FIG. 8**, Shor's algorithm requires considerably more memory. The qualitative properties of functions analyzed by Grover algorithm take Boolean values "true" and "false." By contrast, Shor's algorithm analyzes functions that can take various values as input parameters. This fact inevitably leads to a considerable increase in the amount of memory required for a given number of qubits. For Shor's algorithm, directly simulating a system with 5 qubits is practical, but a simulation with 6 qubits becomes impractical because the memory requirements are increasing exponentially. **FIG. 9** shows the time required to run Shor's algorithm and Grover's algorithm versus the number of qubits and the number of iterations. **FIG. 10** corresponds to **FIG. 9** where the number of iterations is fixed at 10. **FIG. 11** shows an exponential increase in the time required for making one iteration as the number of qubits increases from 1 to 5. **FIG. 12** and **FIG. 13** shows comparisons of computer hardware requirements of Shor's and Grover's quantum algorithms concerning time of execution.

[0113] The comparative analysis of Shor's and Grover's quantum algorithms afforded by **FIGS. 8-12** shows that maximum number of qubits that can be simulated in Shor's algorithm is relatively smaller than in Grover's algorithm (for direct simulation). Since realization of Shor's algorithm on classical computers is more demanding to hardware resources than realization of Grover's algorithm, appropriate hardware acceleration for practically significant applications is relatively more important for Shor's algorithm than for Grover's algorithm.

III. Comparative Temporal Complexity of Grover's QA, Shor's QA and Deutsch-Jozsa's QA

[0114] **FIG. 14** shows the runtime needed for 10 iterations of the Shor and Grover algorithms on a representative computer versus the number of qubits. The exponential increase shown by Shor's algorithm is much faster than the time increase shown by Grover's algorithm. **FIG. 15** shows how the frequency of the processor front side bus (FSB) on a Pentium III processor affects the time needed to make one iteration of a QA.

[0115] **FIG. 16** shows the runtime differences between the Shor, Grover, and Deutsch-Josza quantum algorithms as a function of the number of qubits. As shown in **FIG. 16**, Shor's algorithm runs considerably slower than either the Grover or the Deutsch-Josza algorithms. This result arises from the structure of Shor's algorithm. In Shor's quantum algorithm, the number of qubits used for measurement is equal to the number of input qubits. This means that running a Shor's algorithm simulation for 5 qubits is the same as running a Grover's algorithm simulation with 9 qubits. Moreover, Shor's algorithm requires twice as much memory in order to store with complex numbers. As shown in **FIG. 16**, for the tested hardware and software realization of Deutsch-Jozsa algorithm, simulation of systems with more than 11 qubits becomes increasingly impractical.

IV. Information Analysis of Quantum Complexity of QAs: Quantum Query Tree Complexity

[0116] The existing QAs described above can be naturally expressed using a black-box model. It is then useful to consider the spatio-temporal complexity of QAs from the quantum query complexity viewpoint. For example, in the case of Simon's problem, one is given a function $f: \{0,1\}^n \rightarrow \{0,1\}^n$ and a promise that there is an $s \in \{0,1\}^n$ such that $f(i)=f(j)$ iff $i=j \oplus s$. The goal is to determine whether $s=0$ or not. Simon's QA yields an exponential speed-up over a classical algorithm. Simon's QA requires an expected number of O (n) applications of f, whereas, every classical randomized algorithm for the same problem must make $\Omega(\sqrt{2^n})$ queries.

[0117] The function f can be viewed as a black-box $X=(x_0, \ldots, x_{N-1})$ of $N=2^n$ bits, and that an f-application can be simulated by n queries to X. Thus, Simon's problem fits squarely in the black-box setting, and exhibits an exponential quantum-classical separation for this promise-problem. The promise means that Simon's problem $f: \{0,1\}^n \rightarrow \{0,1\}^n$ is partial; i.e., it is not defined on all $X \in \{0,1\}^n$ but only on X that correspond to an X satisfying the promise.

[0118] Table 1.2 list the quantum complexity of various boolean functions such as OR, AND, PARITY, and MAJORITY

TABLE 1.2

	Some quantum complexities		
Function	Exact	Zero-error	Bounde-error
OR_N, AND_N	N	N	$\Theta(\sqrt{N})$
$PARITY_N$	$\frac{N}{2}$	$\frac{N}{2}$	$\frac{N}{2}$
$MAJORITY_N$	$\Theta(N)$	$\Theta(N)$	$\Theta(N)$

[0119] For example, consider the property $OR_N(X)=x_0 \vee \ldots \vee x_{N-1}$. The number of queries required to compute $OR_N(X)$ by any classical (deterministic or randomized) algorithm is $\Theta(N)$. The lower bound for OR implies a lower bound for the search problem where it is desired to find an i, such that $x_i=1$, if such an i exists. Thus, an exact or zero-error QSA requires N queries, in contrast to $\Theta(\sqrt{N})$ queries for the bounded-error case. On the other hand, the number of solutions is r and a solution can be found with probability 1 using

$$o\left(\sqrt{\frac{N}{k}}\right)$$

queries. Grover discovered a QSA that can be used to compute OR_N with small error probability using only $O(\sqrt{N})$ queries. In this case of OR_N, the function is total; however, the quantum speed-up is only quadratic instead of exponential.

[0120] A similar result holds for the order-finding problem, which is the core of Shor's efficient quantum factoring algorithm. In this case, the promise is the periodicity of a certain function derived from the number to be factored.

[0121] A boolean function is a function $f:\{0,1\}^n \rightarrow \{0,1\}$. Note that f is total, i.e., it is defined on all n-bit inputs. For an input $x \in \{0,1\}^n$, x_i to denotes its i th bit, so $x=\langle x_1 \ldots x_n \rangle$. The expression $|x|$ is used to denote the Hamming weight of x (its number of 1's). A more general form of a Boolean function can be defined as $f:\{0,1\}^n \supseteq A \rightarrow B = f(A) \subseteq \{0,1\}^m$, for some integers n, m>0. If S is a set of (indices of) variables, then x^s denotes the input obtained by flipping the S-variables in x. The function f is symmetric if $f(x)$ only depends on $|x|$. Some common symmetric functions are:

$$OR_n(x) = 1 \text{ iff } |x| \geq 1; \quad \text{(i)}$$

$$AND_n(x) = 1 \text{ iff } |x| = n; \quad \text{(ii)}$$

$$PARITY_n(x) = 1 \text{ iff } |x| \text{is odd}; \quad \text{(iii)}$$

$$MAJ_n(x) = 1 \text{ iff } |x| > \frac{n}{2}. \quad \text{(iv)}$$

[0122] The quantum oracle model is used to formalize a query to an input $x \in \{0,1\}^n$ as a unitary transformation O that maps $|i, b, z\rangle$ to $|i, b \oplus x_i, z\rangle$ is most some m-qubit basis state, where i takes $\lceil \log n \rceil$ bits, b is one bit. The value z denotes the $(m-\lceil \log n \rceil - 1)$-bit "workspace" of the quantum computer, which is not affected by the query. Applying the operator O_f twice is equivalent to applying the identity operator, and thus O_f is unitary (and reversible) as required. The mapping changes the content of the second register ($|b\rangle$) conditioned on the value of the first register $|i\rangle$.

[0123] The queries are implemented using unitary transformations O_j in the following standard way. The transformation O_j only affects the leftmost part of a basis state: it maps basis state $|i, b, z\rangle$ to $|i, b \oplus x_i, z\rangle$. Note that the O_j are all equal. This generalizes the classical setting where a query inputs an i into a black-box, which returns the bit x_i. Applying O to the basis state $|i,0,z\rangle$ yields $|i,x_i,z\rangle$, from which the i th bit of the input can be read. Because O has to be unitary, it is specified to map $|i,1,z\rangle$ to $|i,1-x_i,z\rangle$. Note that a quantum computer can make queries in superposition: applying O once to the state

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i, 0, z\rangle \text{ gives } \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i, x_i, z\rangle,$$

which in some sense contains all bits of the input.

[0124] A quantum decision tree has the following form: start with an m-qubit state $|\vec{0}\rangle$ where every bit is 0. Since it is desired to compute a function of X, which is given as a black-box, the initial state of the network is not very important and can be disregarded. Thus, the initial state is assumed to be $\vec{0}\rangle$ always. Next, apply a unitary transformation U_0 to the state, then apply a query O, then another transformation U_1, etc. A T-query quantum decision tree thus, corresponds to a unitary transformation $A=U_T O U_{T-1} \ldots O U_1 O U_0$. Here the U_i are fixed unitary transformations, independent of the input x. The final state $A|\vec{0}\rangle$ depends on the input x only via the T applications of O. The output obtained by measuring the final state and outputting the rightmost bit of the observed basis state. Without loss of generality, it can be assumed that there are no intermediate measurements.

[0125] A quantum decision tree is said to compute f exactly if the output equals $f(x)$ with probability 1, for all $x \in \{0,1\}^n$. The tree computes f with bounded-error if the output equals $f(x)$ with probability at least

$$\frac{2}{3},$$

for all $x \in \{0,1\}^n$.

[0126] The function $Q_E(f)$ denotes the number of queries of an optimal quantum decision tree that computes f exactly, $Q_2(f)$ is the number of queries of an optimal quantum decision tree that computes f with bounded-error. Note that the number of queries is counted, not the complexity of the U_i.

[0127] Unlike the classical deterministic or randomized decision trees, the QAs are not necessarily trees anymore (the names "quantum query algorithm" or "quantum black-box algorithm" can also be used). Nevertheless, the term "quantum decision tree" is useful, because such QAs generalize classical trees in the sense that they can simulate them as described below.

[0128] Consider a T-query deterministic decision tree. It first determines which variable it will query first; then it determines the next query depending upon its history, and so on for T queries. Eventually, it outputs an output-bit depending on its total history. The basis states of the corresponding QA have the form $|i, b, h, a\rangle$, where i, b is the query-part, h ranges over all possible histories of the classical computation (this history includes all previous queries and their answers), and a is the rightmost qubit, which will eventually contain the output. Let U,map the initial state $|\vec{0},0,\vec{0},0\rangle$ to $|i,0,\vec{0},0\rangle$, and x_i is the first variable that classical tree would query. Now, the QA applies O, which turns the state into $|i, x_i, \vec{0},0\rangle$. Then the algorithm applies a transformation U_1 that maps $|i, x_i, \vec{0},0\rangle$ to $|j,0,h,0)$, where h is the new history (which includes i and x_i) and x_j is the variable that the classical tree would query given the outcome of the previous query. Then when the quantum tree applies O for the second time, it applies a transformation U_2 that updates the workspace and determines the next query, etc. Finally, after T queries, the quantum tree sets the answer bit to 0 or 1 depending on its total history. All operations U_i performed here are injective mappings from basis states to basis states, hence they be extended to permutations of basis states, which are unitary transformations. Thus a T-query deterministic decision tree can be simulated by an exact a T-query quantum decision tree with the same error probability (basically because a superposition can "simulate" a probability distribution). Accordingly,

$$Q_2(f) \leq R_2(f) \leq D(f) \leq n \text{ and } Q_2(f) \leq Q_E(f) \leq D(f) \leq n \text{ for all } f.$$

[0129] If f is non-constant and symmetric, then

$$D(f)=(1-o(1))n; \quad \text{(i)}$$

$$R_2(f)=\Theta(n); \quad \text{(ii)}$$

$$Q_E(f)=\Theta(n); \quad \text{(iii)}$$

$$Q_2(f)=\Theta(\sqrt{n(n-\Gamma(f))}), \quad \text{(iv)}$$

where $\Gamma(f)=\min\{|2k-n+1|:f_k\neq f_{k+1}\}$ is quantity measure of length of the interval around hamming weight

$$\frac{n}{2}$$

where f_k is constant. The function f flips value if the hamming weight of the input changes from k to k+1 (this $\Gamma(f)$ is a number that is low if f flips for inputs with hamming weight close to

$$\frac{n}{2}).$$

This can be compared with the classical bounded-error query complexity of such functions, which is $\Theta(n)$. Thus, $\Gamma(f)$ characterizes the speed-up that QAs give for all total functions.

[0130] Unlike classical decision trees, a quantum decision tree algorithm can make queries in a quantum superposition, and therefore, may be intrinsically faster than any classical algorithm. The quantum decision tree model can also be referred to as the quantum black-box model.

[0131] Let $Q(f)$ be the quantum decision tree complexity of f with error-bounded probability by

$$\frac{1}{3}.$$

It is possible to derive a general lower bound for $Q(f)$ in terms of Shannon entropy $S^{Sh}(f)$ defined as follows. For any f, define the entropy of f, $S^{Sh}(f)$, to be the Shannon entropy of $f(X)$, where X is taken uniformly random from A:

$$S^{Sh}(f)=-\sum_{y\in B}p_y\log_2 p_y,$$

where $p_y=Pr_{x\in_R A}[f(x)=y]$. For any f,

$$Q(f)=\Omega\left(\frac{S^{Sh}(f)}{\log n}\right). \quad (1.1)$$

[0132] In this case, the computation process can be viewed as a process of communication. To make a query, the algorithm sends the oracle $\lceil \log n \rceil$ bits, which are then returned by the oracle. The first $\lceil \log n \rceil$ bits specify the location of the input bit being queried and the remaining one bit allows the oracle to write down the answer. The QA runs on

$$\frac{1}{\sqrt{|A|}}\sum_{x\in A}|x\rangle_X|y\rangle_Y,$$

where X(Y) denotes the qubits that hold the input (intermediate results of computing), respectively. It is useful to now consider the von Neumann entropy, $S^{vN(t)}(f)$, of the density matrix ρ_Y after t th query. If the QA computes f in T queries, at the end of computation, one expect to have a vector close to

$$\frac{1}{\sqrt{|A|}}\sum_{x\in A}\left|x\right\rangle_X\left|f(x)\right\rangle_Y.$$

For the initial (pure) state, $S^{vN(0)}(f)=0$. By using Holevo's theorem, one can show that $S^{vN(T)}(f)\approx S^{Sh}(f)$. Furthermore, by the sub-additivity of the von Neumann entropy

$$|S^{vN(t+1)}(f)-S^{vN(t)}(f)|=O(\log n) \text{ for any } t \text{ with } 0\leq t\leq T-1.$$

[0133] Therefore,

$$T=\Omega\left(\frac{S^{Sh}(f)}{\log n}\right).$$

This bound is tight.

[0134] This means one quantum query can get log n bits of information, while any classical query get no more than 1 bit of information. This power of getting $\omega(1)$ bits of information from a query is not useful in computing total functions, which are functions that are defined on every string in $\{0,1\}^n$, in the sense that each quantum query can only yield O(1) bits of information on average.

[0135] For this more general case, for any total function f,

$$Q(f)=\Omega(S^{Sh}(f)). \quad (1.2)$$

[0136] Thus, the minimum of Shannon entropy in the final solution output of the QA means its has minimal quantum query complexity. The interrelations in Eqs (1.1) and (1.2) between quantum query complexity and Shannon entropy are used in the solution of QA-termination problem (see below in Section 3). As mentioned above, the number of queries is counted, not the complexity of the U_i. The complexity of a quantum operator U_i and its interrelations with the temporal complexity of a QA is considered below.

[0137] The matrix-based approach can be efficiently realized for a small number of input qubits. The matrix approach is used above as a useful tool to illustrate complexity issues associated with QA simulation on classical computer.

2. Algorithmic Representation of the Quantum Operators and Quantum Algorithms

2.1. Structure of QA Gate System Design

[0138] As shown in **FIG. 17***a*, a QA simulation can be represented as a generalized representation of a QA as a set of sequentially-applied smaller quantum gates. From the structural point of view, each QA is based on a particular set of quantum gates, but generally speaking, each particular set can be divided into superposition operators, entanglement operators, and interference operators.

[0139] This division into superposition operators, entanglement operators, and interference operators permits a generalization of the design of a simulation and allows creation of a classical tool to simulate QAs. Moreover, local optimization of QA components according to specific hardware realization makes it possible to develop appropriate hardware accelerators for QA simulation using classical gates.

2.2. Generalized Approach in QA Simulation

[0140] In general, any QA can be represented as a circuit of smaller quantum gates as shown in **FIGS. 17***a-b*. The circuit shown in the **FIG. 17***a* is divided into five general layers: input, superposition, entanglement, interference, output.

[0141] Layer 1: Input. The quantum state vector is set up to an initial value for this concrete algorithm. For example, input for Grover's QSA is a quantum state $|\phi_0\rangle$ described as a tensor product

$$\begin{aligned}|\phi_0\rangle &= a_1|0\rangle\otimes\ldots\otimes|0\rangle\otimes|0\rangle + a_2|0\rangle\otimes\ldots\otimes|0\rangle\otimes|1\rangle + \\ &\quad a_3|0\rangle\otimes\ldots\otimes|1\rangle\otimes|0\rangle + \ldots + a_n|1\rangle\otimes\ldots\otimes|1\rangle\otimes|1\rangle \\ &= 1|0\rangle\otimes\ldots\otimes|0\rangle\otimes|1\rangle \\ &= |0\ldots01\rangle,\end{aligned} \tag{2.1}$$

where $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}; |1\rangle = \begin{pmatrix}0\\1\end{pmatrix};$

$\hat{\times}$ denotes Kronecker tensor product operation. Such a quantum state can be presented as shown on the **FIG. 18***a*.

[0142] The coefficients a_i in the Eq. (2.1) are called probability amplitudes. Probability amplitudes can take negative and/or complex values. However, the probability amplitudes must obey the following constraint:

$$\sum_i a_i^2 = 1 \tag{2.2}$$

[0143] The actual probability of the arbitrary quantum state $a_i|i\rangle$ to be measured is calculated as a square of its probability amplitude value $p_i=|a_i|^2$.

[0144] Layer 2: Superposition. The state of the quantum state vector is transformed by the Walsh-Hadamard operator so that probabilities are distributed uniformly among all basis states. The result of the superposition layer of Grover's QSA is shown in **FIG. 18***b* as a probability amplitude representation, and also in **FIG. 19***b* as a probability representation.

[0145] Layer 3: Entanglement. Probability amplitudes of the basis vector corresponding to the current problem are flipped while rest basis vectors left unchanged. Entanglement is typically provided by controlled-NOT (CNOT) operations. **FIGS. 18***c* and **19***c* show results of entanglement from the application of the operator to the state vector after superposition operation. An entanglement operation does not affect the probability of the state vector to be measured. Rather, entanglement prepares a state, which cannot be represented as a tensor product of simpler state vectors. For example, consider state ϕ_1 shown in the **FIG. 18***b* and state ϕ_2 presented on the **FIG. 18***c*:

$$\begin{aligned}\phi_1 &= 0.35355(|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + \\ &\quad |110\rangle - |111\rangle) \\ &= 0.35355(|00\rangle + |01\rangle + |10\rangle|11\rangle)(|0\rangle - |1\rangle)\end{aligned}$$

$$\begin{aligned}\phi_2 &= 0.35355(|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle + \\ &\quad |110\rangle - |111\rangle) \\ &= 0.35355(|00\rangle - |01\rangle + |10\rangle + |11\rangle)|0\rangle - 0.35355(|00\rangle + \\ &\quad |01\rangle + |10\rangle + |11\rangle)|1\rangle\end{aligned}$$

[0146] As shown above, the description of state ϕ_1 can be presented as a tensor product of simpler states, while state ϕ_2 (in the measurement basis $|0\rangle,|1\rangle$) cannot.

[0147] Layer 4: Interference. Probability amplitudes are inverted about the average value. As a result, the probability amplitude of states "marked" by entanglement operation will increase. **FIGS. 18***d* and **19***d* show the results of interference operator application. **FIG. 18***d* shows probability amplitudes and **FIG. 19***d* shows probabilities.

[0148] Layer 5: Output. The output layer provides the measurement operation (extraction of the state with maximum probability), followed by interpretation of the result. For example, in the case of Grover's QSA, the required index is coded in the first n bits of the measured basis vector.

[0149] Since the various layer of the QA are realized by unitary quantum operators, simulation of quantum operators depend on simulation of such unitary operators. Thus, in order to develop an efficient, simulation, it is useful to understand the nature of the QAs basic quantum operators.

2.3. Basic QA Operators

[0150] The superposition, entanglement and interference operators are now considered from the simulation viewpoint. In this case, the superposition operators and the interference operators have more complicated structure and differ from algorithm to algorithm. Thus, it is first useful to consider the entanglement operators, since they have a similar structure for all QAs, and differ only by the function being analyzed.

[0151] In general, the superposition operator is based on the combination of the tensor products Hadamard H operators

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

with identity operator I:

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

[0152] For most QAs the superposition operator can be expressed as

$$Sp = \left(\bigotimes_{i=1}^{n} H\right) \otimes \left(\bigotimes_{i=1}^{m} S\right), \tag{2.3}$$

[0153] where n and m are the numbers of inputs and of outputs respectively. The operator S depends on the algorithm and can be either the Hadamard operator H or the identity operator I. The numbers of outputs m as well as structures of the corresponding superposition and interference operators are presented in Table 2.1 for different QAs.

TABLE 2.1

Parameters of superposition and interference operators of main quantum algorithms

Algorithm	Superposition	m	Interference
Deutsch's	$H \otimes I$	1	$H \otimes H$
Deutsch-Jozsa's	${}^{n}H \otimes H$	1	${}^{n}H \otimes +0\ I$
Grover's	${}^{n}H \otimes H$	1	$D_n \otimes +0\ I$
Simon's	${}^{n}H \otimes {}^{n}I$	n	${}^{n}H \otimes {}^{n}I$
Shor's	${}^{n}H \otimes {}^{n}I$	n	$QFT_n \otimes +0\ {}^{n}I$

[0154] Superposition and interference operators are often constructed as tensor powers of the Hadamard operator, which is called the Walsh-Hadamard operator. Elements of the Walsh-Hadamard operator can be obtained as

$$[{}^{n}H]_{i,j} = \frac{(-1)^{i*j}}{2^{n/2}}[{}^{n-1}H] = \frac{1}{2^{n/2}}\begin{pmatrix} {}^{(n-1)}H & {}^{(n-1)}H \\ {}^{(n-1)}H & -{}^{(n-1)}H \end{pmatrix}. \tag{2.4}$$

where i=0,1, j=0,1, H denotes Hadamard matrix of ordder 2.

[0155] The rule in Eq. (2.4) provides way to speed up of the classical simulation of the Walsh-Hadamard operators, because the elements of the operator can be obtained by the simple replication described in Eq. (2.4) from the elements of the ${}^{n-1}H$ order operator. For example, consider the superposition operator of Deutsch's algorithm, n=1, m=1, S=I:

$$\begin{aligned}[Sp]_{i,j}^{Deutsch} &= \frac{(-1)^{i*j}}{2^{1/2}} \otimes I \\ &= \frac{1}{\sqrt{2}}\begin{pmatrix} (-1)^{0*0}I & (-1)^{0*1}I \\ (-1)^{1*0}I & (-1)^{1*1}I \end{pmatrix} \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}\end{aligned} \tag{2.5}$$

[0156] As a further example, consider the superposition operator of Deutsch-Jozsa's and of Grover's algorithm, for the case n=2, m=1, S=H:

$$\begin{aligned}[Sp]^{Deutsch\text{-}Jozsa's,Grover's} &= {}^{2}H \otimes H \\ &= \left(\frac{1}{\sqrt{8}}\right)^{3} H \\ &= \frac{1}{\sqrt{8}}\begin{pmatrix} {}^{2}H & {}^{2}H \\ {}^{2}H & -{}^{2}H \end{pmatrix} \\ &= \frac{1}{\sqrt{8}}\begin{pmatrix} H & H & H & H \\ H & -H & H & -H \\ H & H & -H & -H \\ H & -H & -H & H \end{pmatrix},\end{aligned} \tag{2.6}$$

where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

[0157] For yet another example, the superposition operator of Simon's and of Shor's algorithms, n=2, m=2, S=I can be expressed as:

$$\begin{aligned}[Sp]_{i,j}^{Simon,Shor} &= {}^{2}H \otimes {}^{2}I \\ &= \frac{1}{2}\begin{pmatrix} (-1)^{0*0}H & (-1)^{1*0}H \\ (-1)^{1*0}H & (-1)^{1*1}H \end{pmatrix} \otimes {}^{2}I \\ &= \frac{1}{2}\begin{pmatrix} H & H \\ H & -H \end{pmatrix} \otimes {}^{2}I \\ &= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \otimes {}^{2}I \\ &= \frac{1}{2}\begin{pmatrix} {}^{2}I & {}^{2}I & {}^{2}I & {}^{2}I \\ {}^{2}I & -{}^{2}I & {}^{2}I & -{}^{2}I \\ {}^{2}I & {}^{2}I & -{}^{2}I & -{}^{2}I \\ {}^{2}I & -{}^{2}I & -{}^{2}I & {}^{2}I \end{pmatrix}\end{aligned}$$

[0158] Interference operators are calculated for each algorithm according to the parameters listed in Table 2.1. The interference operator is based on the interference layer of the algorithm, which is different for various algorithms, and from the measurement layer, which is the same or similar for most algorithms and includes the m^{th} tensor power of the identity operator.

[0159] The interference operator of Deutsch's algorithm includes the tensor product of two Hadamard transformations, and can be calculated using Eq. (2.4) with n=2 as:

$$[Int^{Deutsch}]_{i,j} = {}^{2}H = \frac{(-1)^{i*j}}{2^{2/2}} \quad (2.7)$$

$$= \frac{1}{2}\begin{pmatrix} (-1)^{0*0}H & (-1)^{0*1}H \\ (-1)^{1*0}H & (-1)^{1*1}H \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

[0160] In Deutsch's algorithm, the Walsh-Hadamard transformation in the interference operator is used also for the measurement basis.

[0161] The interference operator of Deutsch-Jozsa's algorithm includes the tensor product of the n^{th} power of the Walsh-Hadamard operator with an identity operator. In general form, the block matrix of the interference operator of Deutsch-Jozsa's algorithm can be written as from the n−1 order matrix as:

$$[Int^{Deutsch\text{-}Jozsa's}] = {}^{n}H \otimes I \quad (2.8)$$

$$= \frac{1}{2^{n/2}}\begin{pmatrix} {}^{(n-1)}H & {}^{(n-1)}H \\ {}^{(n-1)}H & -{}^{(n-1)}H \end{pmatrix} \otimes I,$$

where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$,

[0162] Interference operator of Deutsch-Jozsa's algorithm, n=2, m=1:

$$[Int^{Deutsch\text{-}Jozsa's}] = {}^{2}H \otimes I$$

$$= \frac{1}{2}\begin{pmatrix} H & H \\ H & -H \end{pmatrix} \otimes I$$

$$= \frac{1}{2}\begin{pmatrix} I & I & I & I \\ I & -I & I & -I \\ I & I & -I & -I \\ I & -I & -I & I \end{pmatrix}.$$

[0163] The interference operator of Grover's algorithm can be written as a block matrix of the following form:

$$[Int^{Grover}]_{i,j} = D_n \otimes I \quad (2.9)$$

$$= \left(\frac{1}{2^{n/2}} - {}^{n}I\right) \otimes I$$

$$= \left(-1 + \frac{1}{2^{n/2}}\right) \otimes I\Big|_{i=j},$$

-continued

$$\left(\frac{1}{2^{n/2}}\right) \otimes I\Big|_{i\neq j} = \frac{1}{2^{n/2}}\begin{cases} -I, i = j \\ I, i \neq j \end{cases}$$

where i=0, . . . , 2^n−1, j=0, . . . , 2^n−1, D_n refers to diffusion operator

$$[D_n]_{i,j} = \frac{(-1)^{1 \ AND \ (i=j)}}{2^{n/2}}.$$

[0164] For example, the interference operator for Grover's QSA, when n=2, m=1 is:

$$[Int^{Grover}]_{i,j} = D_2 \otimes I \quad (2.10)$$

$$= \left(\frac{1}{2^{2/2}} - {}^{2}I\right) \otimes I$$

$$= \left(-1 + \frac{1}{2}\right) \otimes I\Big|_{i=j},$$

$$\frac{1}{2} \otimes I\Big|_{i\neq j} = \frac{1}{2}\begin{pmatrix} -I & I & I & I \\ I & -I & I & I \\ I & I & -I & I \\ I & I & I & -I \end{pmatrix}$$

[0165] As the number of qubits increases, the gain coefficient will become smaller. The dimension of the matrix increases according to 2^n, but each element can be extracted using Eq. (2.9), without allocation of the entire operator matrix.

[0166] The interference operator of Simon's algorithm is prepared in the same manner as the superposition (as well as superposition operators of Shor's algorithm) and can be described as follows from Eq. (2.3) and Eq. (2.6):

$$[Int^{Simon}]_{(i,j)} = {}^{n}H \otimes {}^{m}I = \frac{(-1)^{(i*j)}}{2^{n/2}}{}^{(n-1)}H \otimes {}^{m}I,$$

where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

[0167] In general, the interference operator of Simon's algorithm coincides with the interference operator of Deutsch-Jozsa's algorithm Eq. (2.8), but for each block of the operator matrix includes m tensor products of the identity operator.

[0168] The Interference operator of Shor's algorithm uses the Quantum Fourier Transformation operator (QFT), calculated as:

$$[QFT_n]_{i,j} = \frac{1}{2^{n/2}} e^{J(i*j)\frac{2\pi}{2^n}}, \quad (2.11)$$

where: J=$\sqrt{-1}$, i=0, . . . , 2^n−1 and, j=0, . . . , 2^n−1.

[0169] When n=1 then:

$$QFT_n|_{n=1} = \frac{1}{2^{\frac{1}{2}}}\begin{pmatrix} e^{J*(0*0)2\pi/2^1} & e^{J*(0*1)2\pi/2^1} \\ e^{J*(1*0)2\pi/2^1} & e^{J*(1*1)2\pi/2^1} \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H \quad (2.12)$$

[0170] Eq. (2.11) can also be presented in harmonic form using the Euler formula:

$$[QFT_n]_{i,j} = \frac{1}{2^{\frac{n}{2}}}\left(\cos\left((i*j)\frac{2\pi}{2^n}\right) + J\sin\left((i*j)\frac{2\pi}{2^n}\right)\right) \quad (2.13)$$

[0171] For some applications, the harmonic form of Eq (2.13) is preferable.

[0172] In general, entanglement operators are part of a QA when the information about the function being analyzed is coded as an input-output relation. Thus, it is useful to develop a general approach for coding binary functions into corresponding entanglement gates. Consider the arbitrary binary function: $f:\langle 0,1\rangle^n \rightarrow \langle 0,1\rangle^m$, such that:

$$f(x_0, \ldots, x_{n-1}) = (y_0, \ldots, y_{m-1})$$

[0173] In order to create unitary quantum operator, which performs the same transformation, first transform the irreversible function f into a reversible function F, as follows:

$$F:\langle 0,1\rangle^{m+n} \rightarrow \langle 0,1\rangle^{m+n},$$

such that: $F(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1}) == (x_0, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}) \oplus (y_0, \ldots, Y_{m-1}))$ where $\oplus$ denotes addition modulo 2.

[0174] For the reversible function F, it is possible to design an entanglement operator matrix using the following rule:

$$[U_F]_{i^B,j^B} = 1 \text{ iff } F(j^B) = i^B,\ i,\ j \in \left[\underbrace{0, \ldots, 0}_{n+m}; \underbrace{1, \ldots, 1}_{n+m};\right],$$

where B denotes binary coding. The resulting entanglement operator is a block diagonal matrix, of the form:

$$U_F = \begin{pmatrix} M_0 & & 0 \\ & \ddots & \\ 0 & & M_{2^n-1} \end{pmatrix} \quad (2.14)$$

[0175] Each block M_i,i=0, . . . , 2^n-1 includes m tensor products of I or of C operators, and can be obtained as follows:

$$M_i = \bigotimes_{k=0}^{m-1} \begin{cases} I, & \text{iff } F(i,k) = 0 \\ C, & \text{iff } F(i,k) = 1 \end{cases}, \quad (2.15)$$

where C represents the NOT operator, defined as:

$$C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The entanglement operator is a sparse matrix. Using sparse matrix operations it is possible to accelerate the simulation of the entanglement. Each row or column of the entanglement operation has only one position with non-zero value. This is a result of the reversibility of the function F.

[0176] For example, consider the entanglement operator for a binary function with two inputs and one output: $f:\langle 0,1\rangle^2 \rightarrow \langle 0,1\rangle^1$, such that: $f(x)=1|_{x=01}0|_{x\neq 01}$. The reversible function F in this case is:

[0177] $F:\langle 0,1\rangle^3 \rightarrow \langle 0,1\rangle^3$, such that:

(x, y)	$(x, f(x) \oplus y)$
00,0	$00,0 \oplus 0 = 0$
00,1	$00,0 \oplus 1 = 1$
01,0	$01,1 \oplus 0 = 1$
01,1	$01,1 \oplus 1 = 0$
10,0	$10,0 \oplus 0 = 0$
10,1	$10,1 \oplus 0 = 1$
11,0	$11,0 \oplus 0 = 0$
11,1	$11,1 \oplus 0 = 1$

[0178] The corresponding entanglement block matrix can be written as:

$$U_F = \begin{array}{c} \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{array}{c} \langle 00| \quad \langle 01| \quad \langle 10| \quad \langle 11| \\ \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} \end{array}$$

[0179] **FIG. 18***c* shows the result of the application of this operator in Grover's QSA. Entanglement operators of Deutsch and of Deutsch-Jozsa's algorithms have the general form shown in the above equation.

[0180] As a further example, consider the entanglement operator for a binary function with two inputs and two outputs: $f:\langle 0,1\rangle^2 \rightarrow \langle 0,1\rangle^2$, such that: $f(x)=10|_{x=01,11}00|_{x\neq 01,11}$ and

$$\langle 00| \quad \langle 01| \quad \langle 10| \quad \langle 11|$$

-continued

$$U_F = \begin{array}{c} \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{pmatrix} I\otimes I & 0 & 0 & 0 \\ 0 & \boxed{C\otimes I} & 0 & 0 \\ 0 & 0 & I\otimes I & 0 \\ 0 & 0 & 0 & \boxed{C\otimes I} \end{pmatrix}$$

[0181] The entanglement operators of Shor's and of Simon's algorithms have the general form shown in the above equation.

2.4. Results of Classical QA Gate Simulation

[0182] Analyzing the quantum operators described in Section 2.2 above leads to the following simplifications for increasing the performance of classical QA simulations:

[0183] a) All quantum operators are symmetrical around main diagonal matrices.

[0184] b) The state vector is a sparse matrix.

[0185] c) Elements of the quantum operators need not be stored, but rather can be calculated when necessary using Eqs. (2.6), (2.12), (2.14) and (2.15);

[0186] d) The termination condition can be based on the minimum of Shannon entropy of the quantum state, calculated as:

$$H = -\sum_{i=0}^{2^{m+n}} p_i \log p_i \qquad (2.16)$$

[0187] Calculation of the Shannon entropy is applied to the quantum state after the interference operation. The minimum of Shannon entropy in Eq. (2.16) corresponds to the state when there are few state vectors with high probability (states with minimum uncertainty are intelligent states).

[0188] Selection of an appropriate termination condition is important since QAs are periodical. **FIG. 20** shows results of the Shannon information entropy calculation for the Grover's algorithm with 5 inputs. **FIG. 20** shows that for five inputs of the Grover's QSA an optimal number of iterations, according to minimum of the Shannon entropy criteria for successful result, is exactly four. With more iterations, the probability of obtaining a correct answer will decrease and the algorithm may fail to produce a correct answer. The theoretical estimation for 5 inputs gives $\pi/4\sqrt{2^5}=4.44$ iterations. The Shannon entropy-based termination condition provides the number of iterations. More detailed description of the information-based termination condition is presented in Section 2.5.

[0189] Simulation results of a fast Grover QSA are summarized in Table 2.2. The number of iterations for the fast algorithm is estimated according to the termination condition based on minimum of Shannon entropy of the quantum intelligent state vector.

TABLE 2.2

Temporal complexity of Grover's QSA simulation on 1.2 GHz computer with two CPUs

		Temporal complexity, seconds	
n	Number of iterations h	Approach 1 (one iteration)	Approach 2 (h iterations)
10	25	0.28	~0
12	50	5.44	~0
14	100	99.42	~0
15	142	489.05	~0
16	201	2060.63	~0
20	804	—	~0
30	25.375	—	0.016
40	853.549	—	4.263
50	26.353.589	—	12.425

[0190] The following approaches were used in the simulations listed in Table 2.2. In Approach 1, the quantum operators are applied as matrices, elements of quantum operator matrices are calculated dynamically according to Eqs. (2.6), (2.12), (2.14) and (2.15). As shown in **FIG. 21**, the classical hardware limit of this approach to simulation on a desktop computer is around 20 or more qubits, caused by an exponential temporal complexity.

[0191] In Approach 2, the quantum operators are replaced with classical gates. Product operations are removed from the simulation as described above in Section 2.2. The state vector of probability amplitudes is stored in compressed form (only different probability amplitudes are allocated in memory). **FIG. 22** shows that with the second approach, it is possible to perform classical efficient simulation of Grover's QSA on a desktop computer with a relatively large number of inputs (50 qubits or more). **FIG. 22** shows that with allocation of the state vector in computer memory, this approach permits simulation 26 qubits on a conventional PC with 1 GB of RAM. By contrast, **FIG. 21** shows memory required for Grover's algorithm simulation when the entire state vector is stored in memory. Adding one qubit doubles the computer memory needed for simulation of Grover's QSA when state vector is allocated completely in memory.

2.5. Information Criteria for Solution of the QSA-Termination Problem

[0192] Quantum algorithms come in two general classes: algorithms that rely on a Fourier transform, and algorithms that rely on amplitude amplification. Typically, the algorithms includes a sequence of trials. After each trial, a measurement of the system produces a desired state with some probability determined by the amplitudes of the superposition created by the trial. Trials continue until the measurement gives a solution, so that the number of trials and hence, the running time are random.

[0193] The number of iterations needed, and the nature of the termination problem (i.e., determiming when to stop the iterations) depends in art on the information dynamics of the algorithm. An examination of the dynamics of Grover's QSA algorithm starts by preparing all m qubits of the quantum computer in the state $|s\rangle=|0\ldots0\rangle$. An elementary rotation in the direction of the sought state $|x_0\rangle$ with property $f(x_0)=1$ is achieved by the gate sequence:

$$Q = -[\underbrace{(I_s H^{\otimes 2m}) \cdot I_{x_0}}_{k\ times}] \cdot H^{\otimes 2m}, \tag{2.17}$$

where the phase inversion I_s with respect to the initial state $|s\rangle$ is defined by $I_s|S\rangle=-|S\rangle, 1_s|S\rangle=|S\rangle (x\neq s)$. The controlled phase inversion I_{x_0} with respect to the sought state $|x_0\rangle$ is defined in an analogous way. Because the state $|x_0\rangle$ is not known explicitly but only implicitly through the property $f(x_0)=1$, this transformation is performed with the help of the quantum oracle. This task can be achieved by preparing the ancillary of the quantum oracle in the state

$$|a_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

as the unitary and Hermitian transformation: $U_F:|x,a\rangle \rightarrow |x, f(x)\oplus a\rangle$. Thus, $|x\rangle$ is an arbitrary element of the computational basis and $|a\rangle$ is the state of an additional ancillary qubit. As a consequence, one obtains the required properties for the phase inversion I_{x_0}, namely:

$$|x, f(x)\oplus a_0\rangle \equiv |x, 0\oplus a_0\rangle = \frac{1}{\sqrt{2}}[|x,0\rangle - |x,1\rangle] = |x, a_0, \text{ for } x \neq x_0$$

$$|x, f(x)\oplus a_0\rangle \equiv |x, 1\oplus a_0\rangle = \frac{1}{\sqrt{2}}[|x,1\rangle - |x,0\rangle] = -|x, a_0, \text{ for } x \neq x_0$$

[0194] In order to rotate the initial state $|s\rangle$ into the state $|x_0\rangle$ one can perform a sequence of n such rotations and a final Hadamard transformation at the end, i.e., $|s_{fin}\rangle = HQ^n|s_{in}\rangle$. The optimal number n of repetitions of the gate Q in Eq. (2.17) is approximately given by

$$n = \frac{\pi}{4\arcsin(2^{-\frac{m}{2}})} - \frac{1}{2} \approx \frac{\pi}{4}\sqrt{2^m}, (2^m \gg 1). \tag{2.18}$$

[0195] The matrix D_n, which is called the diffusion matrix of order n, is responsible for interference in this algorithm. It plays the same role as QFT_n (Quantum Fourier Transform) in Shor's algorithm and of ^nH in Deutsch-Jozsa's and Simon's algorithms. This matrix is defined as

$$[D_n]_{i,j} = \frac{(-1)^{1 AND (i-j)}}{2^{n/2}}, \tag{2.19}$$

where i=0, . . . , 2^n-1, j=0, . . . , 2^n-1 n is a number of inputs.

[0196] The gate equation of Grover's QSA circuit is the following:

$$G^{Grover}=[(D_n \hat{\times} I)\cdot U_F]^{h\cdot(n+1}H) \tag{2.20}$$

[0197] The diagonal matrix elements in Grover's QSA-operators (as shown, for example, in Eq. (2.21) below) are connected to a database state to itself and the off-diagonal matrix elements are connected to a database state and to its neighbors in the database. The diagonal elements of the diffusion matrix have the opposite sign from the off-diagonal elements.

[0198] The magnitudes of the off-diagonal elements are roughly equal, so it is possible to write the action of the matrix on the initial state (see Table2.3).

TABLE 2.3

Diffusion matrix definition

D_n	$\|0\ldots0\rangle$	$\|0\ldots1\rangle$	$\ldots\|i\rangle$	$\ldots\|1\ldots0\rangle$	$\|1\ldots1\rangle$
$\|0\ldots0\rangle$	$-1+1/2^{n-1}$	$1/2^{n-1}$	$\ldots 1/2^{n-1}$	$\ldots 1/2^{n-1}$	$1/2^{n-1}$
$\|0\ldots1\rangle$	$1/2^{n-1}$	$-1+1/2^{n-1}$	$\ldots 1/2^{n-1}$	$\ldots 1/2^{n-1}$	$1/2^{n-1}$
. . .	. . .	. . .	. . . . .	. . . . .	. . .
$\|i\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	$\ldots -1+1/2^{n-1}$	$\ldots 1/2^{n-1}$	$1/2^{n-1}$
. . .	. . .	. . .	. . . . .	. . . . .	. . .
$\|1\ldots0\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	$\ldots 1/2^{n-1}$	$\ldots -1+1/2^{n-1}$	$1/2^{n-1}$
$\|1\ldots1\rangle$	$1/2^{n-1}$	$1/2^{n-1}$	$\ldots 1/2^{n-1}$	$\ldots 1/2^{n-1}$	$-1+1/2^{n-1}$

[0199] For example:

$$\begin{pmatrix} -a & b & b & b & b & b \\ b & -a & b & b & b & b \\ b & b & -a & b & b & b \\ b & b & b & -a & b & b \\ b & b & b & b & -a & b \\ b & b & b & b & b & -a \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \frac{1}{\sqrt{N}} = \tag{2.21}$$

$$\begin{pmatrix} -a+(N-3)b \\ -a+(N-3)b \\ +a+(N-1)b \\ -a+(N-3)b \\ -a+(N-3)b \\ -a+(N-3)b \end{pmatrix} \frac{1}{\sqrt{N}}, \text{ where } a = 1-b,\ b = \frac{1}{2^{n-1}}.$$

If one of the states is marked, i.e., has its phase reversed with respect to that of the others, the multimode interference conditions are appropriate for constructive interference to the marked state, and destructive interference to the other states. That is, the population in the marked bit is amplified. The form of this matrix is identical to that obtained through the inversion about the average procedure in Grover's QSA. This operator produces a contrast in the probability density of the final states of the database of

$$\frac{1}{N}[a+(N-1)b]^2$$

for the marked bit versus

$$\frac{1}{N}[a \quad (N \quad 3)b]^2$$

for the unmarked bits; where N is the number of bits in the data register.

[0200] Grover's algorithm gate in Eq, (2.20) is optimal and it is, thus, an efficient search algorithm. Thus, software based on the Grover algorithm can be used for search routines in a large database.

[0201] Grover's QSA includes a number of trials that are repeated until a solution is found. Each trial has a predetermined number of iterations, which determines the probability of finding a solution. A quantitative measure of success in the database search problem is the reduction of the information entropy of the system following the search algorithm. Entropy $S^{Sh}(P_i)$ in this example of a single marked state is defined as

$$S^{Sh}(P_i) = -\sum_{i=1}^{N} P_i \log P_i, \tag{2.22}$$

where P_i is the probability that the marked bit resides in orbital i. In general, the Von Neumann entropy is not a good measure for the usefulness of Grover's algorithm. For practically every value of entropy, there exist states that are good initializers and states that are not. For example,

$$S(\rho_{(n-1)-mix}) = \log_2 N - 1 = S\left(\rho_{\left(\frac{1}{\log_2 N}\right)-pure}\right),$$

but when initialized in $\rho_{(n-1)-mix}$, the Grover algorithm is not good at guessing the market state. Another example may be given using pure states $H|0\rangle\langle 0|H$ and $H|1\rangle\langle 1|H$. With the first, Grover finds the marked state with quadratic speed-up. The second is practically unchanged by the algorithm.

[0202] The information intelligent measure $\Im_T(|\psi\rangle)$ of the state $|\psi\rangle$ with respect to the qubits in T and to the basis B=$\{|i_1\rangle \times \ldots \times |i_n\rangle\}$ is

$$\Im_T(|\psi\rangle) = 1 - \frac{S_T^{Sh}(|\psi\rangle) - S_T^{VN}(|\psi\rangle)}{|T|}. \tag{2.23}$$

[0203] The intelligence of the QA state is maximal if the gap between the Shannon and the Von Neumann entropy in Eq. 2.23 for the chosen resultant qubit is minimal. Information QA-intelligent measure $\Im_T(|\psi\rangle)$ and interrelations between information measures $S_T^{Sh}(|\psi\rangle) \cong S_T^{VN}(|\psi\rangle)$ are used together with entropic relations of the step-by-step natural majorization principle for solution of the QA-termination problem. From Eq. (2.17) it can be seen that for pure states

$$\max \Im_T(|\psi\rangle) \mapsto 1 - \min\left(\frac{S_T^{Sh}(|\psi\rangle) - S_T^{VN}(|\psi\rangle)}{|T|}\right) \mapsto \min S_T^{Sh}(|\psi\rangle), \tag{2.24}$$

$$S_T^{VN}(|\psi\rangle) = 0,$$

[0204] From Eq.(2.17) the principle of Shannon entropy minimum is described as follows.

[0205] According to Eq. (1.2), the Shannon entropy shows the lower bound of quantum complexity of the QA. It means that the criterion in Eq. (2.24) includes both metrics for design of an intelligent QSA: (i) minimal quantum query complexity; and (ii) optimal termination of the QSA with a successful search solution.

[0206] The Shannon information entropy is used for optimization of the termination problem of Grover's QSA. A physical interpretation of the information criterion begins with an information analysis of Grover's QSA based on using of Eq. (2.23). Eq (2.23) gives a lower bound on the amount of entanglement needed for a sucessful search and of the computational time. A QSA that uses the quantum oracle calls $\langle O_g \rangle$ as $I-2|s\rangle\langle s|$ calls the oracle at least

$$T \geq \left(\frac{1-P_e}{2\pi} + \frac{1}{\pi \log N}\right)\sqrt{N}$$

times to achieve a probability of error P_e. The information system includes the N-state data register. Physically, when the data register is loaded, the information is encoded as the phase of each orbital. The orbital amplitudes carry no information. While state-selective measurement gives as result only amplitudes, the information is hidden from view, and therefore, the entropy of the system is maximum: $S_{init}^{Sh}(P_i) = -\log(1/N) = \log N$. The rules of quantum measurement ensure that only one state will be detected each time.

[0207] If the algorithm works perfectly, the marked state orbital is revealed with unit efficiency, and the entropy drops to zero. Otherwise, unmarked orbitals may occasionally be detected by mistake. The entropy reduction can be calculated from the probability distribution, using Eq. (2.22). The minimum Shannon entropy criteria is used for successful termination of Grover's QSA and realized in this case in digital circuit implementation. P **FIG. 23** shows the results of entropy analysis for Grover's QSA according to Eq. (2.16), for the case where n=7, $f(x_0)=1$. **FIG. 23** shows that minimum Shannon entropy is achieved on the 8^{th} iteration (the minimum value of the Shannon entropy is 1). A theoretical estimation for this case is

$$\frac{\pi}{4}\sqrt{2^7} \approx 9$$

iterations. On the ninth iteration, the probability of the correct answer already becomes smaller, and as a result, measurement of the wrong basis vector may happen.

[0208] Application of the Shannon entropy termination condition is presented below in Section 6 (see **FIGS. 48 and 49**) for different input qubit numbers of Grover's QSA. The role of majorization and its relationship to Shannon entropy is discussed below.

[0209] Majorization describes what it means to say that one probability distribution is more disordered than another. In the quantum mechanical context, majorization provides an elegant way to compare two probability distributions or two density matrices. The step-by-step majorization is found in the known instance of efficient QA's, namely in the QFT, in Grover's QSA, in Shor's QA, in the hidden affine function problem, in searching by quantum adiabatic evolution and in deterministic quantum walks algorithm in continuous time solving a classical hard problem. Moreover, majorization has found many applications in classical computer science like stochastic scheduling, optimal Huffman coding, greedy algorithms, etc. Majorization is a natural ordering on probability distributions. One probability distribution is more uneven than another one when the former majorizes the later. Majorization implies an entropy decrease, thus the ordering concept introduced by majorization is more restrictive and powerful than that associated with the Shannon entropy.

[0210] The notion of ordering from majorization is more severe than the one quantified by the standard Shannon entropy. If one probability distribution majorizes another, a set of inequalities must hold to constrain the former probabilities with respect to the latter. These inequalities lead to entropy ordering, but the converse is not necessarily true. In quantum mechanics, majorization is at the heart of the solution of a large number of quantum information problems. In QA analysis, the problem distribution associated with the quantum state in the computational basis is step-by-step majorized until it is maximally ordered. Then a measurement provides the solution with high probability. The way such a detailed majorization emerges in both algorithmic families (as Grover's and Shor's QA's, and phase-estimation QA) is intrinsically different. The analyzed instance of QA's support a step-by-step Majorization Principle.

[0211] Grover's algorithm is an instance of the principle where majorization works step by step until the optimal target state is found. Extensions of this situation are also found in algorithms based in quantum adiabatic evolution and the family of quantum phase-estimation algorithms, including Shor's algorithm. In a QA, the time arrow is a majorization arrow.

[0212] Majorization is often defined as a binary relation noted by $\rangle$ on vectors in ϵ^d. Notations are fixed by introducing the following basic definitions:

[0213] For $x,y \in \epsilon^d$,

$$x < y \text{ iff} \begin{cases} \sum_{i=1}^{k} x_{[i]} \le \sum_{i=1}^{k} y_{[i]}, & k = 1, \ldots, d-1 \\ \sum_{i=1}^{d} x_{[i]} \le \sum_{i=1}^{d} y_{[i]} \end{cases}$$

where $[z_{[1]} \ldots z_{[d]}] := \text{sort}_{\downarrow}(z)$ denotes the descendingly sorted (non-increasing) ordering of $z \epsilon^d$. If it exists, the least element x_l (greatest element x_g) of a partial order like majorization is defined by the condition $x_l \rangle x$, $\forall x \epsilon^d (x \rangle x_g, \forall x \epsilon^d)$

[0214] For example, consider two vectors x, y ϵR^d such that

$$\sum_{i=1}^{d} x_i = \sum_{i=1}^{d} y_i = 1,$$

[0215] whose components represent two different probabilistic distributions. Three definitions of majorization are given in the table below:

Definition 1	$x = \sum_j p_j P_j y$
Definition 2	$\sum_{i=1}^{k} x_i \le \sum_{i=1}^{k} y_i, \quad k = 1, \ldots, d$
Definition 3	$x = Dy$

[0216] Definition 1 says that distribution y majorizes distribution x, written $x \rangle y$, if and only if, there exists a set of permutation matrices P_j and probabilities p_j such that

$$x = \sum_j p_j P_j y.$$

[0217] Because the probability distribution x can be obtained from y by means of a probabilistic sum, the definition given above provides the intuitive notion that the x distribution is more disordered than y.

[0218] An alternative and usually more practical definition of majorization can be stated in terms of a set of inequalities to be held between two distributions as described in Definition 2 above. Consider the components of the two vectors sorted in decreasing order, written as $(z_1, \ldots z_d) \equiv z^{\downarrow}$. Then, $y^{\downarrow}$ majorizes $x^{\downarrow}$ if and only if the following relations are satisfied:

$$\sum_{i=1}^{k} x_i \leq \sum_{i=1}^{k} y_i, \quad k = 1, \ldots, d.$$

[0219] Probability sums, such as the ones appearing in the previous expression are referred to as "cumulants".

[0220] According to Definition 3 above, a real d×d matrix $D=(D_{ij})$ is said to be double stochastic if it has non-negative entries, and each row and column of D sums to 1. Then y majorizes x if and only if, there is a double stochastic matrix D such that x=Dy. Complementarily, the probability distribution x minorizes distribution y if and only if, y majorizes x.

[0221] A powerful relation involving majorization and common Shannon entropy

$$S^{Sh}(x) = -\sum_{i=1}^{d} x_i \log x_i$$

of probability distribution x is that: If $x \prec y$, then $-S^{Sh}(y) \geqq -S^{Sh}(x)$. This is a particular case of a more general result, stated in the following weak form:

$$x \prec y \Rightarrow F(x) < F(y), \quad \text{where } F(x) \equiv \sum_i f(x_i),$$

for any convex function $f:R \rightarrow R$ This result can be extended to the domain of operator functionals.

$$\rho \prec \sigma \Rightarrow F(\rho) < F(\sigma), \text{ where } F(\rho) \equiv \sum_i f(\lambda_i),$$

and λ_i are the eigenvalues of ρ, for any convex function $f:R \rightarrow R$

[0222] In particular, it follows that the von Neumann entropy $S^{vN}(\rho)=S^{Sh}(\lambda(\rho))$ also obeys $\rho \prec \sigma \Rightarrow -S^{vN}(\rho) \geqq -S^{vN}(\sigma)$.

[0223] Thus, if one probability distribution or one density operator is more disordered than another in the sense of majorization, then it is also more disordered according to the Shannon or the von Neumann entropies, respectively.

[0224] As the two previous theorems show, there are many other functions that also preserve the majorization relation. Any such function, called Schur-convex, can in a sense be used as a measure of order. The majorization relation is a stronger notion of disorder, giving more information than any Schur-convex function. The Shannon and the von Neumann entropies quantify the order in some limiting conditions, namely when many copies of a system are considered.

[0225] There is a majorization principle underlying the way QA's work. Denote by $|\Psi_m\rangle$ the pure state representing the state of the register in a quantum computer at an operating stage labeled by m=0,1, . . . , M−1, where M is the total number of steps of algorithm, and let N be the dimension of the Hilbert space. Also, denote as $\{|i\rangle\}_{i=1}^{N}$ the basis in which the final measurement is performed in the algorithm, one can naturally associate a set of sorted probabilities $[p^m_{[x]}]$, x=0,1, . . . ,2^n−1 to this quantum state of n qubits in the following way: decompose the register state in the computational basis i.e.,

$$|\Psi_m\rangle := \Sigma_{x=0}^{2^n-1} c^m_x |x\rangle$$

with

$$\{|x\rangle := |x_0 s_1 \ldots x_{n-1}\rangle\}_{x=0}^{2^n-1}$$

denoting basis states in digital or binary notation, respectively and

$$x := \Sigma_{j=0}^{n-1} x_j 2^j.$$

[0226] The sorted vectors to which majorization theory applies are precisely

$$[p^m_{[x]}] := [|c^m_{[x]}|^2] = [|\langle x|\psi_m\rangle|^2],$$

where x=1, . . . , N, which corresponds to the probabilities of all the possible outcomes if the computation is stopped at stage m and a measurement is performed.

[0227] Thus, in a QA, one deals with probability densities defined in $\{_+^d$, with $d=2^n$. With these ingredients, the main result can be stated as follows: in the QAs known so far, the set of sorted probabilities $[p_{[x]}]^m$ associated with the quantum register at each step m are majorized by the corresponding probabilities of the next step

$$[p^m_{[x]}] \prec [p^{m+1}_{[x]}], \begin{cases} \forall\, m = 0, 1, \ldots, M-2 \\ x = 0, 1, \ldots, 2^n - 1 \end{cases}, \text{ or}$$

$$p^{(m)} \prec p^{(m+1)}, \; p^{(m)} = [p^m_{[x]}].$$

[0228] Majorization works locally in a QA, i.e., step by step, and not just globally (for the initial and final states). The situation given in the above equation is a step-by-step verification, as there is a net flow of probability directed to the values of highest weight, in such a way that the probability distribution will be steeper as time flows.

[0229] In physical terms, this can be stated as a very particular constructive interference behavior, namely, a constructive interference that has to satisfy the constraints given above step-by-step. The QA builds up the solution at each time step by means of this very precise reordering of probability distribution.

[0230] The majorization is checked on a particular basis. Step-by-step majorization is a basis-dependent concept. The preferred basis is the basis defined by the physical implementation of the quantum computation or computational basis. The principle is rooted in the physical possibility to arbitrarily stop the computation at any time and perform a measurement. The probability distribution associated with this physically meaningful action obeys majorization and the QA-stopping problem can be solved by the principle of minimum of Shannon entropy.

[0231] Working with probability amplitudes in the basis $\{|i\rangle\}_{i=1}^{N}$, the action of a particular unitary gate at step m makes the amplitudes evolve to step m+1 in the following way:

$$c_i^{m+1} = \sum_{j=1}^{N} U_{ij} c_j^m,$$

where U_{ij} are the matrix elements in the chosen basis of the unitary evolution operator (namely, the propagator from step m to step m+1). Inverting the evolution gives

$$c_i^m = \sum_{j=1}^{N} A_{ij} c_j^{m+1},$$

where A_{ij} are the matrix elements of the inverse unitary evolution (which is unitary as well).Taking the square modulus

$$|c_i^m|^2 = \sum_j |A_{ij}|^2 |c_i^{m+1}|^2 + \text{interference terms.}$$

[0232] Should the interference terms disappear, majorization would be verified in a "natural" way between steps m and m+1 because the initial probability distribution could be obtained from the final one only by the action of a doubly stochastic matrix with entries $A_{ij}|^2$. This is so-called "natural majorization": majorization, which naturally emerges from the unitary evolution due to the lack of interference terms when making the square modulus of the probability amplitudes. There will be "natural minorization" between steps m and m+1 if and only if there is "natural majorization" between time steps m+1 and m.

[0233] Grover's QSA follows a step-by-step majorization. More concretely, each time Grover's operator is applied, the probability distribution obtained from the computational basis obeys the above constraints until the searched state is found. Furthermore, because of the possibility of understanding Grover's quantum evolution as a rotation in a two-dimensional Hilbert space the QA follows a step-by-step minorization when evolving far away from the marked state, until the initial superposition of all possible computational states is obtained again. The QA behaves such that majorization is present when approaching the solution, while minorization appears when escaping from it. A cycle of majorization and minorization emerges in the process proceeds through enough evolutions, due to the rotational nature of Grover's operator.

[0234] Grover's algorithm is an instance of the principle where majorization works step-by-step until the optimal target state is found. Extensions of this situation are also found in algorithms based in quantum adiabatic evolution and the family of quantum phase-estimation algorithms, including Shor's algorithm.

[0235] Grover's algorithm can conveniently be used as a starting point for majorization analysis of various quantum algorithms. This QA efficiently solves the problem of finding a target item in a large database. The algorithm is based on a kernel that acts symmetrically on the subspace orthogonal to the solution. This is clear from its construction

$K:=U_s U_{y0}$

$U_s:=2|s\rangle\langle s|-1$, $U_{y0}:=1-2|y_0\rangle\langle y_0|$

where $|s\rangle:=1/\sqrt{N}\Sigma_x|x\rangle$ and $|y_0\rangle$ is a searched item. The set of probabilities to obtain any of the N possible states in a database is majorized step-by-step along with the evolution of Grover's algorithm when starting from a symmetric state until the maximum probability of success is reached.

[0236] Shor's QA is analyzed inside of the broad family of quantum phase-estimation algorithms. A step-by-step majorization appears under the action of the last QFT when considered in the usual Coppersmith decomposition. The result relies on the fact that those quantum states that can be mixed by a Hadamard operator coming from the decomposition of the QFT only differ by a phase all along the computation. Such a property entails as well the appearance of natural majorization, in the way presented above. Natural majorization is relevant for the case of Shor's QFT. This particular algorithm manages step-by-step majorization in the most efficient way. No interference terms spoil the majorization introduced by the natural diagonal terms in the unitary evolution.

[0237] For efficient termination of QAs that give the highest probability of successful result, the Shannon entropy is minimal for the step m+1. This is the principle of minimum Shannon entropy for termination of a QA with the successful result. This result also follows from the principle of QA maximum intelligent state. For this case:

$$\max J_T(|\psi\rangle) = 1 - \min \frac{II_T^{SH}(|\psi\rangle)}{|I|}.$$

$S_T{}^{vN}(|\psi\rangle)=0$ (for pure quantum state). Thus, the principle of maximal intelligence of QAs include as particular case the principle of minimum Shannon entropy for QA-termination problem solution.

3. The Structure and Acceleration Method of Quantum Algorithm Simulation

[0238] The analysis of the quantum operator matrices that was carried out in the previous sections forms the basis for specifying the structural patterns giving the background for the algorithmic approach to QA modeling on classical computers. The allocation in the computer memory of only a fixed set of tabulated (pre-defined) constant values instead of allocation of huge matrices (even in sparse form) provides computational efficiency. Various elements of the quantum operator matrix can be obtained by application of an appropriate algorithm based on the structural patterns and particular properties of the equations that define the matrix elements. Each representation algorithm uses a set of table values for calculating the matrix elements. The calculation of the tables of the predefined values can be done as part of the algorithm's initialization.

3.1. Algorithmic Representation of the Grover's QA

[0239] **FIGS. 24***a-c* are flowcharts showing realization of such an approach for simulation of superposition (**FIG. 24***a*), entanglement (**FIG. 24***b*) and interference (**FIG. 24***c*) operators in Grover's QSA. Here n is a number of qubit, i and j are the indexes of a requested element, $hc=2^{-(n+1)/2}$, $dc1=2^{1-n}-1$ and $dc2=2^{1-n}$ are the table values.

[0240] In **FIG. 24***a*, in a block **2401**, the i,j values are specified and provided to an initialization block **2402** where loops control variables ii :=i, jj:=0, and k:=0 are initialized, and calculation variable h:=1 is initialized. The process then proceeds to a decision block **2403**. In the block **2403**, if k is less than or equal to n, then the process advances to a decision block **2404**; otherwise, the process advances to an output block **2407** where the output h*hc is computed (where $hc=2^{-(n+1)/2}$). In the decision block **2404**, if (ii and jj and 1)=1, then the process advances to a block **2406**; otherwise, the process advances to a block **2405**. In the block **2406**, the process sets h:=−h and advances to the block **2405**. In the block **2405**, the process sets ii:=ii SHR 1, jj:=jj SHR 1, and k:=k+1 (where SHR is a shift right operation), and then the process returns to the decision block **2403**.

[0241] In **FIG. 24***b*, the inputs i, j in an input block **2411** are provided to an initialization block **2412** which sets ii:=i SHR 1, and jj:=SHR 1 and then advances to a decision block **2413**. In the decision block **2413**, if ii==jj, then the process advances to a decision block **2415**, otherwise, the process advances to an output block **2414** which outputs 0. In the decision block **2415**, if i=j, then the process advances to a block **2416**; otherwise, the process advances to a block **2417**. In the block **2416**, the process sets u:=1 and then advances to a decision block **2418**. In the block **2417**, the process sets u:=0 and advances to the decision block **2418**. In the decision block **2418**, if f(ii)=1, then the process advances to a block **2420**; otherwise, the process advances to an output block that outputs u. The block **2420** sets u:=NOT u and advances to the output block **2419**.

[0242] In **FIG. 24***c*, if ((i XOR j) AND 1)=1 then the process outputs 0; otherwise, the process advances to a decision block **2423**. In the decision block **2423**, if i=j then the process outputs dc1, otherwise the process outputs dc2, where $dc1=2^{1-n}-1$ and $dc2=2^{1-n}$.

[0243] As described above, the superposition and entanglement operators for Deutsch-Jozsa's QA are the same with superposition and entanglement operators for Grover's QSA (**FIG. 24***a*, **FIG. 24***b*, respectively). The interference operator representation algorithm for Deutsch-Jozsa's QA is shown in **FIG. 24***d*, where $hc=2^{-n/2}$.

[0244] The entanglement operator for the Simon QA is shown in **FIG. 24***e*. Here m is an output dimension, $ec1=2^m-1$ and $ec2=2^{m-1}$ are the table values. In **FIG. 24***e*, the inputs i,j are provided to an initialization block **2452** that sets ii:=i SHR m and jj :=SHR m. The process then advances to a decision block **2453**. In the decision block **2453**, if ii=jj then the process advances to a block **2454**; otherwise, the process outputs 0. In the block **2454**, the process sets u:=f(ii), ii:=i AND ec1, jj:=j AND ec1, and k:=ec2; after which the process advances to a decision block **2455**. In the decision block **2455**, if (u AND k)=0, then the process advances to a decision block **2456**; otherwise, the process advances to a decision block **2457**. In the decision block **2456**, if k<=ii, and k>jj, then the process outputs 0; otherwise, the process advances to a decision block **2451**. In the decision block **2457**, if k<=ii AND k<=jj, then the process outputs 0; otherwise, the process advances to a decision block **2456**. In the decision block **2451**, if k>ii AND k<=jj, then the process outputs 0; otherwise, the process advances to a block **2459**. In the decision block **2456**, if k>ii AND k>jj then the process outputs 0; otherwise, the process advances to the block **2459**. In the block **2459**, the process sets ii:=jj AND (k−1), jj:=jj AND (k=1), and k:=K SHR 1, after which, the process advances to a decision block **2458**. In the decision block **2458**, if k>0, then the process loops back to the block **2455**; otherwise, the process outputs 1.

[0245] Superposition and interference operators for the Simon QA are identical (see Table 2.1) and are shown by flowchart in **FIG. 24***f*. In **FIG. 24***f*, the inputs i,j are provided to a decision block **2552**. In the decision block **2552**, if ((i XOR j) AND $(2^{n-1})=0$) then the process advances to a block **2553**; otherwise, the process outputs 0. In the block **2553**, the process sets ii:=i SHR n, jj :=j SHR n, h:=1, and k:=1, and then advances to a decision block **2556**. In the decision block **2556**, if k<=n, then the process advances to a decision block **2557**; otherwise, the process outputs h*hc. In the decision block **2557**, if (((ii AND jj) AND 1)=1) then the process sets J:=−h and advances to a block **2558**; otherwise, the process advances directly to the block **2558**. In the block **2558**, the process sets ii:=SHR 1, jj :=jj SHR 1, k:=k+1 and then loops back to the decision block **2556**.

[0246] **FIG. 24***g* is a flowchart showing calculation of the interference operator from the Shor QA. The Shor interference operator is relatively more complex, as explained above. Superposition and entanglement operators for the Shor algorithm are the same as the Simon's QA operators shown in **FIG. 24***f* and **FIG. 24***e*. The Shor interference operator is based on the Quantum Fourier Transformation (QFT) with table values $c1=2^{-n/2}$ and $c2=\pi/2^{n-1}$.

[0247] In **FIG. 24***g*, the inputs i,j are provided to a decision block **2602**. In the decision block **2602**, if ((i XOR j) AND $(2^n-1))=0$ then the process advances to a block **2603**; otherwise, the process outputs the complex number (0,0). In the block **2603**, the process sets i:=i SHR n, and j :=j SHR n, and then advances to a decision block **2604**. In the decision block **2604**, if i=0, then the process outputs the complex number (c1,0); otherwise, the process advances to a decision block **2607**. In the decision block **2607**, if j=0, then the process outputs the complex number (c1,0); otherwise, the process advances to a block **2608**, In the block **2608**, the process sets a:=c1*cos(i*j*c2), and b:=c1*sin(i*j*c2), and the outputs (a,b).

[0248] The time required for calculating the elements of an operator's matrix during a process of applying a quantum operator is generally small in comparison to the total time of performing a quantum step. Thus, the time burden created by exponentially-increasing memory usage tends to be less, or at least similar to, the time burden created by computing matrix elements as needed. Moreover, since the algorithms used to compute the matrix elements tend to be based on fast bit-wise logic operations, the algorithms are amenable to hardware acceleration.

[0249] Table 3.1 shows comparisons of the traditional and as-needed matrix calculation (when the memory used for the as-needed algorithm (Memory*) denotes memory used for storing the quantum system state vector.

TABLE 3.1

Different approaches comparison: Standard (matrix based) and algorithmic based approach

	Standard		Calculated Matrices	
Qubits	Memory, MB	Time, s	Memory*	Time, s
1	1	0.03	≈0	≈0
8	18	5.4	0.008	0.0325
11	1048	1411	0.064	2.3
16	—	—	2	4573
24	—	—	512	$3 * 10^8$
64	—	—	—	—

[0250] The results shown in Table 3.1 is based on the results of testing the software realization of Grover QSA simulator on a personal computer with Intel Pentium III 1 GHz processor and 512 Mbytes of memory. One iteration of the Grover QSA was performed.

[0251] Table 3.1 shows that significant speed-up is achieved by using the algorithmic approach as compared with the prior art direct matrix approach. The use of algorithms for providing the matrix elements allows considerable optimization of the software, including the ability to optimize at the machine instructions level. However, as the number of qubits increases, there is an exponential increase in temporal complexity, which manifests itself as an increase in time required for matrix product calculations.

[0252] Use of the structural patterns in the quantum system state vector and use of a problem-oriented approach for each particular algorithm can be used to offset this increase in temporal complexity. By way of explanation, and not by way of limitation, the Grover algorithm is used below to explain the problem-oriented approach to simulating a QA on a classical computer.

3.2. Problem-Oriented Approach Based on Structural Pattern of QA State Vector.

[0253] Let n be the input number of qubits. In the Grover algorithm, half of all 2^{n-1} elements of a vector making up its even components always take values symmetrical to appropriate odd components and, therefore, need not be computed. Odd 2^n elements can be classified into two categories:

[0254] The set of m elements corresponding to truth points of input function (or oracle); and

[0255] The remaining 2^n−m elements.

[0256] The values of elements of the same category are always equal.

[0257] As discussed above, the Grover QA only requires two variables for storing values of the elements. Its limitation in this sense depends only on a computer representation of the floating-point numbers used for the state vector probability amplitudes. For a double-precision software realization of the state vector representation algorithm, the upper reachable limit of q-bit number is approximately 1024. **FIG. 25** shows a state vector representation algorithm for the Grover QA. In **FIG. 25**, i is an element index, *f* is an input function, vx and va corresponds to the elements' category, and v is a temporal variable. The input i is provided to a decision block **2502**. In the decision block **2502**, if *f*(i SHR 1)=1, then the process proceeds to a block **2503**; otherwise, the process proceeds to a block **2507**. In the block **2503**, the process sets v:=vx and then advances to a decision block **2504**. In the block **2507**, the process sets v:=va and then advances to the decision block **2504**. In the decision block **2504**, if (i AND 1)=1), then the process outputs −v; otherwise, the process outputs v. Thus, the number of variables used for representing the state variable is constant.

[0258] A constant number of variables for state vector representation allows reconsideration of the traditional schema of quantum search simulation. Classical gates are used not for the simulation of appropriate quantum operators with strict one-to-one correspondence but for the simulation of a quantum step that changes the system state. Matrix product operations are replaced by arithmetic operations with a fixed number of parameters irrespective of qubit number.

[0259] **FIG. 26** shows a generalized schema for efficient simulation of the Grover QA built upon three blocks, a superposition block H **2602**, a quantum step block UD **2610** and a termination block T **2605**. **FIG. 26** also shows an input block **2601** and an output block **2607**. The UD block **2610** includes a U block **2603** and a D block **2604**. The input state from the input block **2601** is provided to the superposition block **2602**. A superposition of states from the superposition block **2602** is provided to the U block **2603**. An output from the U block **2603** is provided to the D block **2604**. An output from the D block **2604** is provided to the termination block **2605**. If the termination block terminates the iterations, then the state is passed to the output block **2607**; otherwise, the state vector is returned to the U block **2603** for another iteration.

[0260] As shown in **FIG. 27**, the superposition block H **2602** for Grover QSA simulation changes the system state to the state obtained traditionally by using n+1 times the tensor product of Walsh-Hadamard transformations. In the process shown in **FIG. 27**, vx:=hc, va:=hc, and vi:=0., where $hc=2^{-(n+1)/2}$ is a table value.

[0261] The quantum step block UD **2610** that emulates the entanglement and interference operators is shown on **FIGS. 28***a-c*. The UD block **2610** reduces of the temporal complexity of the quantum algorithm simulation to linear dependence on the number of executed iterations. The UD block **2610** uses ore-calculated table values $dc1=2^n-m$ and $dc2=2^{n-1}$. In the U block **2603** shown in **FIG. 28***a*, vx:=−vx and vi:=vi+1. In the D block **2604** shown in **FIG. 28***b*, v:=m*vx+dc1*va, v:=v/dc2, vx:=v=vx, and va:=v−va in the UD block shown in **FIG. 28***c*, v:=dc1*va=m*vx, v:=v/dc2, vx:=v+vx, va:=v−va, and vi:=vi+1.

[0262] The termination block T **2605** is general for all quantum algorithms, independently of the operator matrix realization. Block T **2605** provides intelligent termination condition for the search process. Thus, the block T **2605** controls the number of iterations through the block UD **2610** by providing enough iterations to achieve a high probability of arriving at a correct answer to the search problem. The block T **2605** uses a rule based on observing the changing of the vector element values according to two classification categories. The T block **2605** during a number of iterations, watches for values of elements of the same category monotonically increase or decrease while values of elements of another category changed monotonically in reverse direc-

tion. If after some number of iteration the direction is changed, it means that an extremum point corresponding to a state with maximum or minimum uncertainty is passed. The process can proceed here using direct values of amplitudes instead of considering Shannon entropy value, thus, significantly reducing the required number of calculations for determining the minimum uncertainty state that guarantees the high probability of a correct answer. The Termination algorithm realized in the block T **2605** can use one or more of five different termination models:

[**0263**] Model 1: Stop after a predefined number of iterations;

[**0264**] Model 2: Stop on the first local entropy minimum;

[**0265**] Model 3: Stop on the lowest entropy within a predefined number of iterations;

[**0266**] Model 4: Stop on a predefined level of acceptable entropy; and/or

[**0267**] Model 5: Stop on the acceptable level or lowest reachable entropy within the predefined number of iterations.

[**0268**] Note that models 1-3 do not require the calculation of an entropy value. **FIGS. 29-31** show the structure of the termination condition blocks T **2605**.

[**0269**] Since time efficiency is one of the major demands on such termination condition algorithm, each part of the termination algorithm is represented by a separate module, and before the termination algorithm starts, links are built between the modules in correspondence to the selected termination model by initializing the appropriate functions' calls.

[**0270**] Table 3.2 shows components for the termination condition block T **2605** for the various models. Flow charts of the termination condition building blocks are provided in **FIGS. 29-34**

TABLE 3.2

	Termination block construction		
Model	T	B'	C'
1	A	—	—
2	B	PUSH	—
3	C	A	B
4	D	—	—
5	C	A	E

[**0271**] The entries A, B, PUSH, C, D, E, and PUSH in Table 5 correspond to the flowcharts in **FIGS. 29, 30, 31, 32, 33, 34** respectively.

[**0272**] In model 1, only one test after each application of quantum step block UD is needed. This test is performed by block A. So, the initialization includes assuming A to be T, i.e., function calls to T are addressed to block A. Block A is shown in **FIG. 29**. As shown in **FIG. 29**, the A block checks to see if the maximum number of iterations has been reached, if so, then the simulation is terminated, otherwise, the simulation continues.

[**0273**] In model 2, the simulation is stopped when the direction of modification of categories' values are changed. Model 2 uses comparison of the current value of vx category with value mvx that represents this category value obtained in previous iteration:

[**0274**] (i) If vx is greater than mvx, its value is stored in mvx, the vi value is stored in mvi, and the termination block proceeding to the next quantum step.

[**0275**] (ii) If vx is less than mvx, it means that the vx maximum is passed and the process needs to set the current (final) value of vx :=o mvx, vi :=mvi, and stop the iteration process. So, the process stores the maximum of vx in mvx and the appropriate iteration number vi in mvi. Here block B, shown in **FIG. 30** is used as the main block of the termination process. The block PUSH, shown in the **FIG. 31***a* is used for performing the comparison and for storing the vx value in mvx (case a). A POP block, shown in **FIG. 31***b* is used for restoring the mvx value (case b). In the PUSH block of **FIG. 31***a*, if |vx|>|mvx|, then mvx:=vx, mva:=va, mvi:=vi, and the block returns true; otherwise, the block returns false. In the POP block of **FIG. 31***b*, if |vx|<=|mvx|, then vx:=mvx, va:=mva, and vi:=mvi.

[**0276**] The model 3 termination block checks to see that a predefined number of iterations is not exceeded (using block A in **FIG. 29**):

[**0277**] (i) If the check is successful, then the termination block compares the current value of vx with mvx. If mvx is less than, it sets the value of mvx equal to vx and the value of mvi equal to vi. If mvx is less using the PUSH block, then perform the next quantum step.

[**0278**] (ii) If the check operation fails, then (if needed) the final value of vx equal to mvx, vi equal to mvi (using the POP block) and the iterations are stopped.

[**0279**] The model 4 termination block uses a single component block D, shown in **FIG. 33**. The D block compares the current Shannon entropy value with a predefined acceptable level. If the current Shannon entropy is less than the acceptable level, then the iteration process is stopped; otherwise, the iterations continue.

[**0280**] The model 5 termination block uses the A block to check that a predefined number of iterations is not exceeded. If the maximum number is exceeded, then the iterations are stopped. Otherwise, the D block is then used to compare the current value of the Shannon entropy with the predefined acceptable level. If acceptable level is not attained, then the PUSH block is called and the iterations continue. If the last iteration was performed, the POP block is called to restore the vx category maximum and appropriate vi number and the iterations are ended.

[**0281**] **FIG. 35** shows measurement of the final amplitudes in the output state to determine the success or failure of the search. If |vx|>|va|, then the search was successful; otherwise, the search was not successful.

[**0282**] Table 3.3 lists results of testing the optimized version of Grover QSA simulator on personal computer with Pentium 4 processor at 2 GHz.

TABLE 3.3

High probability answers for Grover QSA

Qbits	Iterations	Time
32	51471	0.007
36	205887	0.018
40	823549	0.077
44	3294198	0.367
48	13176794	1.385
52	52707178	5.267
56	210828712	20.308
60	843314834	81.529
64	3373259064	328.274

[0283] The theoretical boundary of this approach is not the number of qubits, but the representation of the floating-point numbers. The practical bound is limited by the front side bus frequency of the personal computer.

[0284] Using the above algorithm, a simulation of a 1000 qubit Grover QSA requires only 96 seconds for 10^8 iterations.

[0285] The above approach can be used for simulation of the Deutsch-Jozsa's QA. The general schema of Deutsch-Jozsa's QA simulation is shown on **FIG. 36**, where an input state **3601** is provided to a quantum HUD block **3602** which generates an output state **3603**.

[0286] The structure of the HUD block **3602** is shown in **FIG. 37**, where the input **3601** is provided to an initialization block **3702**. The initialization block **3702** sets i:=0 and v:=0, and then the process advances to a decision block **3703**. In the decision block **3703**, if $i<2^n$, then the process advances to a decision block **3704**; otherwise, the process advances to an output block which outputs v:=v*vc, where $vc=2^{-n-1/2}$.

[0287] The quantum block HUD **2610** is applied only once to obtaining of the final state. Here v represents the vector |0..00> amplitude, f is an input function of order n, $vc=2^{-n-1/2}$ is a table value. After applying the block HUD, the value of v is considered in correspondence with Table 3.4.

TABLE 3.4

Possible answers for Deutsch-Jozsa's problem

Value of v	Answer
0	f is balanced
$\frac{1}{\sqrt{2}}$	f is constant 0
$-\frac{1}{\sqrt{2}}$	f is constant 1
Otherwise	f is something else

4. General Software and Hardware Approach in QC Based on Fast Algorithm Simulation

[0288] The structure of the generalized approach in QA simulation is shown in **FIG. 39**. From the available database of the QAs, its matrix representation is extracted. Then matrix operators are replaced with developed algorithmic or problem-oriented corresponding approaches, thus spatio-temporal characteristics of the algorithm will improve.

[0289] The simulation is then performed, and after obtaining final state vector, the measurement takes place in order to extract the result. Final results can be obtained by having the information about the algorithm and results of the measurement. After interpretation, results can be applied in the selected field of applications.

5. Simulation of Quantum Algorithms with Reduced Number of Quantum Operators: Application of Entanglement-Free Quantum Control Algorithm for Robust KB Design of FC

[0290] The simulation techniques described above for simulating quantum algorithms on classical computers permit design of new QAs, such as, for example, entanglement-free quantum control algorithms. The simulation of a QA can be made more efficient by arranging the QA to be entanglement-free. In one embodiment, the entanglement-free algorithm is used in the context of soft computing optimization for the design process of a robust Knowledge Base (KB) for a Fuzzy Controller (FC).

5.1. Models of Entanglement-Free Algorithms and Classical Efficient Simulation of Quantum Strategies without Entanglement.

[0291] Entanglement-free quantum speed-up algorithms are useful for many applications, including, but not limited to, simulation results in the robust KB-FC design process. The explanation of the entanglement-free quantum efficient algorithm begins with a statement of the following problem: Given an integer N function f: $x \rightarrow mx+b$, where x, m,b ϵZ_N, find m. The classical analysis reveals that no information about m can be obtained with only one evolution of the function f. Conversely, given the unitary operator U_f acting in a reversible way in the Hilbert space $Hil_N \times Hil_N$ such that

$$U_f|x>|y>=|x>|y+f(x)>, \quad (5.1)$$

(where the sum is to be interpreted as modulus N). A QA can be used to solve this problem with only one query to U_f.

[0292] A QA structure for solving the above problem is described as follows. Take $N=2^n$, being n the number of qubits. The QA for efficiently solving the above problem includes the following operations:

[0293] 1. Prepare two registers of n qubits in the state $|0 \ldots >|\psi_1>\epsilon H_N \times H_N$, where $|\psi_1>=QFT(N)^{-1}|1>$, and $QFT(N)^{-1}$ denotes the inverse quantum Fourier transform in a Hilbert space of dimension N.

[0294] 2. Apply QFT (N) over the first register.

[0295] 3. Apply U_f over the whole quantum state.

[0296] 4. Apply $QFT(N)^{-1}$ over the first register.

[0297] 5. Measure the first register and output the measured value.

[0298] This QA leads to the solution of the problem. The analysis raises two observations concerning the way both entanglement and majorization behave in the computational process. In the first step of the algorithm, the quantum state is separable, noting that the QFT (and its inverse) are applied on a well-defined state in the computational basis leads to a perfectly separable state. Actually, this separability holds

also step-by-step when the decomposition for the QFT is considered, such as the Coppersmith's decomposition. That is, the quantum state $|0 \ldots 0\rangle|\psi_1\rangle$ is un-entangled.

[0299] The second step of the algorithm corresponds to a QFT in the first register. This action leads to a step-by-step minorization of the probability distribution of the possible outcomes while it does not create any entanglement. Moreover, natural minorization is at work due to the absence of interference terms.

[0300] It can be verified that the quantum state

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi \frac{i}{N} j} |j\rangle \tag{5.2}$$

is an eigenstate of the operator $|y\rangle \rightarrow |y+f(x)\rangle$ with eigenvalue $e^{2\pi i f(x)/N}$.

[0301] After the third step, the quantum state reads

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i \frac{f(x)}{N}} |\psi_1\rangle = \frac{e^{2\pi i \frac{b}{N}}}{\sqrt{N}} \underbrace{\left(\sum_{x=0}^{N-1} e^{2\pi i \frac{mx}{N}} \right)}_{\text{First Register}} |\psi_1\rangle \tag{5.3}$$

[0302] The probability distribution of possible outcomes has not been modified, thus not affecting majorization. Furthermore, the pure quantum state of the first register in Eq.(5.3) can be written as QFT (N) m) (up to a phase factor), so this step has not created any entanglement among the qubits of the system.

[0303] In the fourth step of the algorithm, the action of the operator $QFT(N)^{-1}$ over the first register leads to the state $e^{2\pi i b/N}|m\rangle|\psi_1\rangle$.

[0304] A subsequent measurement in the computational basis over the first register provides the desired solution.

[0305] The inverse QFT naturally majorizes step-by-step the probability distribution attached to the different outputs. However, the separability of the quantum state still holds step-by-step.

[0306] The QA is more efficient than any of its possible classical counterparts, as it only needs a single query to the unitary operator U_f to obtain the solution. One can summarize this analysis of majorization for the present QA as follows: The entanglement-free efficient QA for finding a hidden affine function shows a majorization cycle based on the action of QFT(N) and $QFT(N)^{-1}$.

[0307] It follows that there can exist a quantum computational speed-up without the use of entanglement. In this case, no resource increases exponentially. Yet, a majorization cycle is present in the process, which is rooted in the structure of both the QFT and the quantum state.

[0308] Quantum mechanics affects game theory, and game theory can be used to show classical-quantum strategy without entanglement. For certain games, a suitable quantum strategy is able to beat any classical strategy. It is possible to demonstrate design of quantum strategies without entanglement using two simple examples of entanglement-free games: the PQ-game and the card game.

[0309] Consider, for example, the penny flipping game PQ PEANY FLIP game. The game is penny flipping, where player P places a penny head up in a box, after which player Q, then player P, and finally player Q again, can choose to flip the coin or not, but without being able to see it. If the coin ends up being head up, player Q wins, otherwise player P wins. The winning (or cheating, depending upon one's perspective) quantum strategy of Q now involves putting the penny into a superposition of head up and down. Since player P is allowed to interchange only up and down he is not able to change that superposition, so Q wins the game by rotating the penny back to its initial state.

[0310] Q produces a penny and asks P to place it in a small box, head up. Then Q, followed by P, followed by Q, reaches into box, without looking at the penny, and either flips it over or leaves it as it is. After Q's second turn they open the box and Q wins if the penny is head up.

[0311] Q wins every time they play, using the following quantum game gate:

$$|\psi_{fin}\rangle = \underbrace{\underbrace{H}_{Q\ strategy} \cdot \underbrace{\sigma_x(I_2)}_{P\ strategy} \cdot \underbrace{H}_{Q\ strategy}}_{} |0\rangle_{Initial\ state}$$

[0312] and the following quantum strategy:

Initial state and strategy	Player strategy	Result of operation
$\|0\rangle$	$\xrightarrow[H]{Q}$	$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
Classical strategy	$\xrightarrow[\sigma_x(\text{or } I_2)]{P}$	$\frac{1}{\sqrt{2}}(\|1\rangle + \|0\rangle)$ or $\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$
Quantum strategy	$\xrightarrow[H]{Q}$	$\|0\rangle$

[0313] Here 0 denotes "head" and 1 denotes "tail", and

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \equiv NOT$$

implements P's possible action of flipping the penny over. Q's quantum strategy of putting the penny into the equal superposition of "head" and "tail" on his first turn means that whether P flips the penny over or not, it remains in an equal superposition which Q rotates back to "head" by applying the Hadamard transformation H again, since

$$H = H^{-1} \text{ and } \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

After measurement, Q receives the state |0>. The second application of the Hadamard transformation plays the role of constructive interference. So when they open the box, Q always wins without using entanglement.

[0314] If Q were restricted to playing classically, i.e., to implementing only σ_x or I_2 on his turns, an optimal strategy for both players would be to flip the penny over or not with equal probability on each turn. In this case, Q would win only half the time, so he does substantially better by playing quantum mechanically.

[0315] Now, consider the interesting case of a classical-quantum card game without entanglement. In the classical game, one player A can always win with the probability

$$\frac{2}{3}.$$

But if the other player B performs quantum strategy, he can increase his winning probability from

$$\frac{1}{3}$$

to

$$\frac{1}{2}.$$

In this case, B is allowed to apply quantum strategy and the original unfair game turns into a fair and zero-sum game, i.e., the unfair classical game becomes fair in the quantum world. In addition, this strategy does not use entanglement.

[0316] The classical model of the card game is explained as follows. A has three cards. The first card has one circle on both sides, the second has one dot on both sides, and the third card has one circle on one side and one dot on the other. In the first step, A puts the three cards into a black box. The cards are randomly placed in the box after A shakes it. Both players cannot see what happens in the box. In the second step, B takes one card from the box without flipping it. Both players can only see the upper side of the card. A wins one coin if the pattern of the down side is the same as that of the upper side and loses one coin when the patterns are different. It follows that A has a

$$\frac{2}{3}$$

probability of winning and B only has a

$$\frac{1}{3}$$

chance of winning. B is in a disadvantageous situation and the game is unfair to him. Any rational player will not play the game with A because the game is unfair. In order to attract B to play with him, before the original second step, A allows B to have one chance to operate on the cards. That is, B has one step query on the box. In the classical world, B can only attain one card information after the query. Because the card is in the box, so what B knows is only one upper side pattern of the three cards. Except for this, he knows nothing about the three cards in the black box. So in the classical field, even having this one step query, B still will be in a disadvantaged state and the game is still unfair.

[0317] Now consider the quantized approach to the card game. In the quantum field, the whole game is changed. The game turns into a fair zero-sum game and both players are in equal situation. Consider first the case when A uses the classical strategy and B uses the quantum strategy. In the first step, A puts the cards in the box and shakes the box, that is, he prepares the initial state randomly. The card state is |0> if the pattern in the upper side is circle and |1> if it is dot. So the upper sides of the three cards in the box can be described as $|r\rangle=|r_0\rangle|r_1\rangle|r_2\rangle$, where $r_0, r_1, r_2 \in (0,1)$, which means $|r_0\rangle$, $|r_1\rangle$, $r_2\rangle$ are all eigenstate superpositions of |0> and |1>.

[0318] After the first step of the game, A gives the black box to B. Because A thinks in classical way, in his mind B cannot get information about all upper side patterns of the three cards in the box. So A can still win with higher probability. But what B uses is quantum strategy: He replaces the classical one step query with one step quantum query. The following shows how B queries the box.

[0319] Assume that B has a quantum machine that applies an unitary operator U on its three input qubits and gives three output qubits. This machine depends on the state |r> in the box that A gives B. The explicit expression of U and its relation with |r> is as following $U=U_0\hat{\times}U_1\hat{\times}U_2$ where

$$U_k = \begin{cases} I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ if } r_k = 0 \\ \sigma_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ if } r_k = 1 \end{cases} = \begin{pmatrix} 1 & 0 \\ 0 & \exp\{i\pi r_k\} \end{pmatrix}.$$

[0320] The processing of the query is shown in **FIG. 40**. After the process, the output state is

$$|\psi_{fin}\rangle=(H\hat{\times}H\hat{\times}II)U(II\hat{\times}II\hat{\times}II)|000\rangle=(IIU_0II)|0\rangle(IIU_1II)|0\rangle(IIU_2II)|0\rangle.$$

[0321] Because

$$HU_kH = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi r_k} \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1+e^{i\pi r_k} & 1-e^{i\pi r_k} \\ 1-e^{i\pi r_k} & 1+e^{i\pi r_k} \end{pmatrix}.$$

So

$$HU_kH|0\rangle = \frac{1+e^{i\pi r_k}}{2}|0\rangle + \frac{1-e^{i\pi r_k}}{2}|1\rangle = \begin{cases} |0\rangle \text{ if } r_k = 0 \\ |1\rangle \text{ if } r_k = 1 \end{cases} = |r_k\rangle$$

[0322] From the above equation, it follows that B can obtain the complete information about the upper patterns of

all the three cards through one query. There are only two possible kinds of output states in the black box, which is |0>|0>|1> or |1>|1>|0>, that is two circles and one dot on the upper side or two dots and one circle. Assume that the state of the cards after the first step is two circles and one dot, i.e., |0>|0>|1>. After the one-step query, B knows the complete information about the upper patterns, but has no individual information about which upper pattern corresponds to which card. Then he takes one card out of the box to see what pattern is on the upper side. If B finds out that he is in a disadvantage situation, the upper pattern of the card is dot (|1>), he refuses to play with A in this turn because he knows the down side is dot definitely. Otherwise if the upper side pattern is circle (|0>), then he knows that the down side pattern is circle 0> or dot |1>. So he continues his turn because the probability of winning is

$$\frac{1}{2}.$$

B will continue the game because he has probability

$$\frac{1}{2}$$

to win. Hence, the game becomes fair and is also zero-sum.

[0323] One of the reasons why the quantum strategies in games are better than classical strategies is that the initial state is maximally entangled. The quantum strategy in the card game applied by B includes no entanglement and is still better than the classical strategy.

[0324] The initial state input to the quantum machine is |0>|0>|0>, which is separable. After the Hadamard transformation, the state is

$$\frac{1}{\sqrt{2^3}}(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle).$$

[0325] Performed by U, the state becomes

$$\frac{1}{\sqrt{2^3}}(|0\rangle+e^{i\pi r_0}|1\rangle)\otimes(|0\rangle+e^{i\pi r_1}|1\rangle)\otimes(|0\rangle+e^{i\pi r_2}|1\rangle).$$

And the states, after the second Hadamard transformation, are in the output state $|r_0>r_1>r_2>$. The state is described by the tensor products of the states of the individual qubits, so it is unentangled. And because the operators (H and U) are also tensor products of the individual local operators on these qubits, in this quantum game there is no entanglement applied.

[0326] Entanglement is important for static games (such as the Prisoner's Dilemma) but may not be necessary in dynamic games (such as the PQ-game and the card game). In static games, each player can only control his qubit and his operation is local. So in the classical world, the operation of one player cannot have influence on others in the operational process. But in the quantum field, through entanglement, the strategy used by one player can influence not only himself, but also his opponents. In dynamic games, players can control all qubits at any step. So, as in QAs, in dynamic games, players can use quantum strategies without entanglement to solve problems, even entangled quantum strategies can be re-described with other quantum strategies without entanglement.

[0327] Thus, if B is given a quantum strategy (e.g., a quantum query) against his classical opponent A, the classical opponent cannot always win with high probability. Both players are on equal footing and the game is a fair zero-sum game. The quantum game includes no entanglement and quantum-over-classical strategy is achieved using only interference. Thus, quantum strategy can still be powerful without entanglement.

[0328] In general, the PQ game can be described as follows:

Definition	Main operations
(i)	A Hilbert space H (the possible states of the game) with N = dim H
(ii)	An initial state $\psi_0 \in H$
(iii)	Subset $Q_i \subset U(N)$, $i \in \{1, \ldots, k+1\}$ - the elements of Q_i are the moves Q chooses among on turn i
(iv)	Subset $P_i \subset S_N$, $i \in \{1, \ldots, k\}$, where S_N is the permutation group on N elements - the elements of P_i are the moves P chooses among on turn i
(v)	A projection operator Π on H (the subspace W_Q fixed by Π consists of the winning states for Q)

[0329] Since only P and Q play, these are two-player games; they are zero-sum since when Q wins, P loses, and vice versa. A pure quantum strategy for Q is a sequence $u_i \in Q_i$. A pure (classical) strategy for P is a sequence $s_i \in P_i$, while a mixed (classical) strategy for P is a sequence of probability distributions $f_i: P_i \rightarrow [0,1]$. If both Q and P play pure strategies, the corresponding evolution of the PQ-game is described by quantum game gate:

$$|\psi_{fin}\rangle = \prod_k u_{k+1} s_k u_k |\psi_{in}\rangle.$$

[0330] After Q's last move, the state of the game is measured with Π. According to the rules of quantum mechanics, the players observe the eigenvalue 1 with probability $\mathrm{Tr}(\psi^{\dagger}\Pi\psi)$; this is lo the probability that the state is projected into W_Q and Q wins. More generally, if P plays a mixed strategy, the corresponding evolution of the PQ-game is described by

$$\rho_f = u_{k+1}\left(\sum_{s_k \in P_k} f_k(s_k) s_k u_k \ \ldots\ u_2\left(\sum_{s_1 \in P_1} f_1(s_1) s_1 u_1 \rho_0 u_1^{*} s_1^{\dagger}\right) u_2^{\dagger} \ \ldots\ u_k^{\dagger} s_k^{\dagger}\right) u_{k+1}^{\dagger},$$

where $\rho_0 = |\psi_0\rangle\hat{\times}\langle\psi_0^{\dagger}|$. Again, after Q's last move ρ_f is measured with Π; the probability that ρ_f is projected into

$W_Q \hat{\times} W_q^{\dagger}$ and Q wins is Tr ($\Pi\rho_f$). 1 5 An equilibrium state is a pair of strategies, one for P and one for Q, such that neither player can improve his probability of winning by changing his strategy while the other does not. In general, unlike the simple case of the PQ-game, $W_Q=W_Q(\langle s_i\rangle)$ or $W_Q=W_Q(\langle f_i\rangle)$, i.e., the conditions for Q's win can depend on P's strategy. There are mixed/quantum equilibria at which Q does better than he would at any mixed/mixed equilibrium; there are some QAs, which outperform classical ones.

5.2. Interrelations Between QAs and Quantum Games Structures.

[0331] A QA for an oracle problem can be understood as a quantum strategy for a player in a two-player zero-sum game in which the other player is constrained to play classically. This correspondence can be formalized and the following development gives examples of games (and hence, oracle problems) for which the quantum player can do better than that would be possible classically. In the general case, entanglement (or some replacement resource) is required. However, an efficient quantum search of a "sophisticated" database requires no entanglement at any time step. A quantum-over-classical reduction in the number of queries is achieved using only interference, not entanglement, within the usual model of quantum computation.

TABLE 5.1

Oracle functions

Number	Title of oracle	Type	Definition
1	The phase oracle	P_f	$\lvert x\rangle\lvert b\rangle \rightarrow \exp\left\{\frac{2\pi i f(x)\cdot b}{2^n}\right\}\lvert x\rangle\lvert b\rangle$
2	The standard oracle	S_f	$\lvert x\rangle\lvert b\rangle \rightarrow \lvert x\rangle\lvert b\oplus f(x)\rangle$
3	The minimal (an erasing) oracle	M_f	$\lvert x\rangle \rightarrow \lvert f(x)\rangle$

[0332] Returning to the quantum oracle evaluation of multi-valued Boolean functions discussed in section 3, consider a multi-valued function F that is one-to-one and where the size of its domain and range is the same. The problem can be formulated as follows: Given an oracle

$$f(a, x):\langle 0,1\rangle^n\times\langle 0,1\rangle^n\rightarrow\langle 0,1\rangle$$

and a fixed (but hidden) value a_0, obtain the value of a_0 by querying the oracle $f(a_0, x)$. The algorithm evaluates the multi-valued Boolean function F through oracle calls and the main goal is to minimize the number of such oracle calls (the query complexity) using a quantum mechanism.

[0333] Query complexity is one of the issues in quantum computation, especially in proving lower bounds of QAs with oracles. Generally speaking, there are two popular techniques to derive quantum lower bounds: (i) polynomials; and (ii) adversary methods. For the bounded error case, evaluations of AND and OR functions need $\Theta(\sqrt{N})$ number of queries, while parity and majority functions at least

$$\frac{N}{2}$$

and $\Theta(N)$, respectively. Alternatively, define

$$F(x_0, \ldots, x_{N-1}) = \begin{cases} a \text{ if } x_a = 1 \text{ and } x_j = 0 \text{ for all } j \neq a \\ \text{undefined otherwise} \end{cases}$$

then evaluating this function F is the same as Grover's QSA. Moreover, if one defines

$$F(x_0, \ldots, x_{N-1}) = \begin{cases} a \text{ if } x_a = a\cdot i(\text{mod}2) \text{ for all } 0 \leq i \leq N-1 \\ \text{undefined otherwise} \end{cases}$$

then this is the same as the so-called Bernstein-Varzirani problem. Some lower bounds are easier to obtain using the quantum adversary method than the polynomials one. The lower bound of a bounded-error quantum query complexity of read-once functions is $\Omega(\sqrt{N})$.

[0334] Quantum evaluation assumes that it is possible to obtain the value of variable x_i only through an oracle O (i). Since both functions are one-to-one, and their domain and range are of the same size, it is possible to formulate the problem as follows.

[0335] Let n be an integer $\geqq 1$ and $N=2^n$. Then, given an oracle defined as a function

$$f(a,x):\langle 0,1\rangle^n\times\langle 0,1\rangle^n\rightarrow\langle 0,1\rangle$$

such that $f(a,x)\neq f(a_2,x)$ for some x if $a_1\neq a_2$, and a fixed (and hidden) value a, it is desired to obtain the value a, using the oracle $f(a, x)$.

[0336] For the Grover QSA, the definition

$$f(x, a) = \begin{cases} 1 \text{ if } x = a \\ 0, \text{ otherwise} \end{cases},$$

completely specifies the problem. This oracle is sometimes called the exactly quantum (EQ) oracle and is denoted by $EQ_a(x)$. Table 5.2 shows the case $f(x, a)=EQ_a(x)$ for n=4.

[0337] As can be seen from Table 5.2, $f(a, x)$ is given by a truth-table of size N×N, where each row gives the function F of the previous definition. For example, F (1, 0, . . . , 0)=0000 from the first row of the Table 5.2. If the hidden value a is 0010 for example, the oracle returns value 1 only when it is queried with x=0010 .

[0338] For the Bernstein-Vazirani problem, the similar definition is given as

$$f(a, x)=a\cdot x(\text{mod } 2),$$

[0339] which is called the inner product (IP) oracle and denoted by $IP_a(x)$. Its truth-table for n=4 is given in Table 5.3.

TABLE 5.2

a	x			
	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1
	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1
0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 1	0 1 0 0 ≡ I	0 0 0 0	0 0 0 0	0 0 0 0
0 0 1 0	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0
0 0 1 1	0 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0
0 1 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0
0 1 0 1	0 0 0 0	0 1 0 0 ≡ I	0 0 0 0	0 0 0 0
0 1 1 0	0 0 0 0	0 0 1 0	0 0 0 0	0 0 0 0
0 1 1 1	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 0 0	1 0 0 0	0 0 0 0
1 0 0 1	0 0 0 0	0 0 0 0	0 1 0 0 ≡ I	0 0 0 0
1 0 1 0	0 0 0 0	0 0 0 0	0 0 1 0	0 0 0 0
1 0 1 1	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0
1 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0
1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0 ≡ I
1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0
1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1

[0340] The above assumed that the domain of the Boolean function has the same size as its range. More general cases, e.g., the size of the range is larger than the domain, will be mentioned briefly below.

[0341] The quantum query complexity is a function of the number of oracle calls needed to obtain the hidden value a. The query complexity for the EQ-oracle is Θ(√N), while only O(1) for the IP-oracle. A difference exist between the EQ- and IP-oracles. The difference can be shown by comparing their truth-tables given in Tables 5.21 and 5.32, where Table 5.3 shows a truth-table for

$$f(x, a) = IP_a = \left\{a \cdot x = \sum_i a_i \cdot x_i (\mathrm{mod}2)\right\}, n = 4.$$

[0342] One can immediately see

TABLE 5.3

a	x			
	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1
	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
0 0 0 1	0 ⌈1 0 1	0 ⌈1 0 1	0 ⌈1 0 1	0 ⌈1 0 1
0 0 1 0	0 0 1⌋ 1	0 0 1⌋ 1	0 0 1⌋ 1	0 0 1⌋ 1
0 0 1 1	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0
0 1 0 0	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1
0 1 0 1	0 ⌈1 0 1	1 0 1⌉ 0	0 ⌈1 0 1	1 0 1⌉ 0
0 1 1 0	0 0 1⌋ 1	1 ⌊1 0 0	0 0 1⌋ 1	1 ⌊1 0 0
0 1 1 1	0 1 1 0	1 0 0 1	0 1 1 0	1 0 0 1

TABLE 5.3-continued

1 0 0 0	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lceil 1 & 0 & 1 \\ 0 & 0 & 1\rfloor & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lceil 1 & 0 & 1 \\ 0 & 0 & 1\rfloor & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1\rceil & 0 \\ 1 & \lfloor 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1\rceil & 0 \\ 1 & \lfloor 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
1 0 0 1				
1 0 1 0				
1 0 1 1				
1 1 0 0	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lceil 1 & 0 & 1 \\ 0 & 0 & 1\rfloor & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1\rceil & 0 \\ 1 & \lfloor 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1\rceil & 0 \\ 1 & \lfloor 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lceil 1 & 0 & 1 \\ 0 & 0 & 1\rfloor & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$
1 1 0 1				
1 1 1 0				
1 1 1 1				

[0343] The table for IP_a is well-balanced in terms of the numbers of 0's and 1's, but quite unbalanced for EQ_a. The natural consequence is that there should be intermediate oracles between those extreme cases for which the query complexity is also intermediate between $\Theta(\sqrt{N})$ and O(1). Furthermore, these intermediate oracles can be characterized by some parameter in such a way that the query complexity depends upon this parameter value and both EQ_a and IP_a are obtained as special cases.

[0344] For these two oracles, the EQ-oracle (defined as f (a, x)=1 iff x=a) and the IP-oracle (defined as $f(a,x)=a\cdot x$ mod2), the query complexity is $\Theta(\sqrt{N})$ for the EQ-oracle while only O(1) for the IP-oracle. To investigate what causes this large difference, the parameter K can be introduced as the maximum number of 1's in a single column of T_f where T_f is the N×N truth-table of the oracle $f(a, x)$. The quantum complexity is strongly related to this parameter K.

[0345] To develop models and estimation of quantum lower/upper bounds, let T_f be the truth-table of an oracle $f(a,x)$ like the oracles given in Tables 5.2 and 5.3. Assume without loss of generality that the number of 1's is less than or equal to the number of 0's in each column of T_f. Let $\#_i(T_f)$ denote the number of 1's

$$\left(\leq \frac{N}{2}\right)$$

in the i-th column of T_f and $\#(T_f)=\max_1 \#_i(T_f)$. This single parameter $\#(T_f)$ plays a key role, namely: (i) Let $f(a, x)$ be any oracle and $K=\#(T_f)$. Then the query complexity of the search problem for $f(a,x)$ is

$$\Omega\left(\sqrt{\frac{N}{K}}\right);$$

This lower bound is tight in the sense that it is possible to construct an explicit oracle whose query complexity is

$$O\left(\sqrt{\frac{N}{K}}\right).$$

This oracle again includes both EQ and IP oracles as special cases; (iii) The tight complexity,

$$\Theta\left(\frac{N}{K}+\log K\right),$$

is also obtained for the classical case. Thus, the QA needs a quadratically fewer number of oracle calls when K is small and this merit become larger when K is large, e.g., log K versus a constant when K=cN.

[0346] The quantum oracle models and reduction of query number problems frame the context for the discussion for the database search problem, that is, to identify a specific record in a large database. Formally, records are labeled ⟨0,1, . . . , N−1⟩ where, for convenience when writing the numbers in binary, it is convenient to take $N=2^n$ where n is a positive integer. In one embodiment, a quantum database search involves a database in which, when queried about a specific number, the oracle responds only that the guess is correct or not. On a classical reversible computer, one can implement a query by a pair of register (x,b), where x is an n-bit string representing the guess, and b is a single bit which the database will use to respond to the query. If the guess is correct, the database responds by adding 1(mod2) to b ; if it is incorrect, it adds 0 to b. That is, the response of the database is the operation: $|x\rangle|b\rangle\rightarrow|x\rangle|b\oplus f_a(x)\rangle$, where $f_a(x)=1$ when x=a, 0 otherwise. Thus, if b changes, one knows that the guess is correct. Classically, it takes N−1 queries to solve this problem with probability 1.

[0347] The following oracles are defined in Table 5.4 for a general function $f:\langle 0,1\rangle^m\rightarrow\langle 0,1\rangle^n$. Here x and b are strings of m and n bits respectively, $|x\rangle$ and $|b\rangle$ the corresponding computational basis states, and $\oplus$ is addition modulo 2^n. The oracles P_f and S_f are equivalent in power: each can be constructed by a quantum circuit containing just one copy of the other. Assuming m=n and assuming f is a known permutation on the set $\langle 0,1\rangle^n$ then $M_{\ƒ}$ is a simple invertible quantum map associated to f. Intuitively, erasing oracles seem at least as strong as standard ones, though it is not clear how to simulate the latter with the former without also having access to an oracle that map $|x\rangle$ to $|f^{-1}(x)\rangle$. One-way functions provide a clue: if f is one-way, then (by assumption) $|x\rangle|f(x)\rangle$ can be computed efficiently, but if $|f(x)\rangle$ could be computed efficiently given $|x\rangle$ then so could $|x\rangle$ given $|f(x)\rangle$, and hence f could be inverted. For some

problems, an exponential gap between query complexity given a standard oracle and query complexity given an erasing oracle.

[0348] QAs work by supposing that they will be realized in a quantum system, which can be in a superposition of "classical" states. These states form a basis for the Hilbert space whose elements represent states of the quantum system. More generally, Grover's QSA works with quantum queries which are linear combinations $\Sigma c_{x,b}|x,b\rangle$, where $c_{x,b}$ are complex numbers satisfying $\Sigma|c_{x,b}|^2=1$. The operations in QAs are unitary transformations, the quantum mechanical generalization of reversible classical operations. Thus, the operation of the database that Grover considered is implemented on superpositions of queries by a unitary transformation, which takes $|x,b\rangle$ to $|x\rangle|b\oplus f_a(x)\rangle$. By using

$$\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$$

quantum queries, it identifies the answer with probability close to 1: The final vectors for the N possible answers a are nearly orthogonal.

[0349] Consider one of the guessing game type that uses Grover's QSA for guessing of any number between 0 and N−1 and to discuss the role of different quantum oracle models in the reduction of query number. Assume, in PQ-game, the player Q boasts that if P picks any number between 0 and N−1, inclusive, he can guess it. P knows the Grover's QSA and realizes that for $N=2^n$, the player Q can determine the number he picks with high probability by playing the following strategy:

TABLE 5.4

$\|0\ldots0,0\rangle$	$\xrightarrow[H^{\otimes n}\otimes H\sigma_x]{Q}$	$\frac{1}{\sqrt{N}}\sum_{x00}^{n-1}\left\|x\right\rangle\otimes\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\right)$	$\Rightarrow$ (u_1)
	$\xrightarrow[s(f_a)]{P}$	$\frac{1}{\sqrt{N}}\sum_{x=0}^{n-1}(-1)^{\delta_{xa}}$	$\Rightarrow$ (s_1)
		$\|x\rangle\otimes\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\right)$	
	$\xrightarrow[H^{\otimes n}\otimes I_2\circ s(f_0)\circ H^{\otimes n}\otimes I_2]{Q}$ $\ldots,$		$\Rightarrow$ (u_2)

using the following quantum game gate:

$$G=[H^{\hat{}n}\times\hat{I}_2\circ s(f_0)\circ H^{\hat{}n}\times\hat{I}_2]\circ s(f_a)\circ[H^{\hat{}n}\times\hat{I}H\sigma_x]$$

which can be efficiently simulated using classical computer. Where $a\in[0,N-1]$ is P's chosen number, moves (s_1) and (u_2) are repeated a total of

$$k=\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$$

times, i.e., $(s_k=\ldots=s_1)$ and $(u_k=\ldots=u_2)$. For $f:Z_2^n\to Z_2$, the oracle $s(f)$ is the permutation (and hence unitary transformation) defined by (see Table 5.4) $s(f)|x,b\rangle=|x,b\oplus f(x)\rangle$. Each P's moves s_i can be thought of as the response of an oracle, which computes $f_x(x):=\delta_{xa}$ to respond to the quantum query defined by the state after the action of quantum strategy (u_i). After $O(\sqrt{N})$ such queries, a measurement by $\Pi=|a\rangle\langle a|\times I_2$ returns a win for Q with probability bounded above

$$\frac{1}{2},$$

i.e., Grover's QSA determines a with high probability.

[0350] If Q were to play classically, he could query P about a specific number at each time, but on the average it would take

$$\frac{N}{2}$$

turns to guess a. A classical equilibrium is for P to choose a random, and for Q to choose a permutation of $N=2^n$ uniformly at random and guess numbers in the corresponding order. Even when P plays such a mixed strategy, Q's quantum strategy is optimal; together they define a mixed / quantum equilibrium.

[0351] Knowing all this, P responds that he will play, but that Q should only get one guess, not

$$k=\lfloor \frac{\pi}{4}\sqrt{N} \rfloor.$$

Q protests that this is hardly fair, but he will play, as long as P tells how close his guess is to the chosen number. P agrees, and they play. Q wins every step.

[0352] In this case, Q uses a slightly improved Berstein-Vazirani algorithm: Guess x and answer a are vectors in Z_2^n, so $x\cdot a$ depends on the cosine of the angle between these vectors. Thus, it seems reasonable to define the oracle "how close a guess is to the answer" to be the oracle response $f_a(x)\mapsto g_a(x):=x·a$. Then Q plays as follows:

$\|0\ldots0,0\rangle$	$\xrightarrow[H^{\otimes n}\otimes H\sigma_x]{Q}$	$\frac{1}{\sqrt{N}}\sum_{x00}^{n-1}\left\|x\right\rangle\otimes\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\right)$	$\Rightarrow$ (u_1)
	$\xrightarrow[s(g_a)]{P}$	$\frac{1}{\sqrt{N}}\sum_{x=0}^{n-1}(-1)^{x\cdot a}\left\|x\right\rangle\otimes\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\right)$	$\Rightarrow$ (s_1)
	$\xrightarrow[H^{\otimes n}\otimes I_2]{Q}$	$\|a\rangle\otimes\frac{1}{\sqrt{2}}(\|0\rangle-\|1\rangle)\right)$	$\Rightarrow$ (u_2)

using the following (more simple) quantum game gate: $G=[H^{e,crc\times}n\times I_2]\circ g_a(x)\circ[H^{\times n}\times H\sigma_x]$. For $\Pi=|a\rangle\langle a|\times I_2$ again, Q wins with probability 1, having queried P only once.

[0353] The oracle, which responds in the Berstein-Vazirani algorithm with x·a (mod2), is a "sophisticated database" by comparison with Grover's oracle in QSA, which only responds that a guess is correct or incorrect. And finally, entanglement is not required in the Berstein-Vazirani QA for quantum-over-classical improvement. The improved version of the Berstein-Vazirani algorithm does not create entanglement at any time step, but still solves this oracle problem with fewer queries than is possible classically.

[0354] Quantum computing manipulates quantum information by means of unitary transformations, such as superpositions. For instance, a single-qubit Walsh-Hadamard operation H transforms a qubit from |0> to |+> and from |1> to |−>. When H is applied to a superposition such as |+>, it follows by the linearity of quantum mechanics that the resulting state is ½(|0>+|1>)+(|0>−|1>)=0. This illustrates the phenomenon of destructive interference, by which component |1> of the state is erased. Consider now an n-qubit quantum register initialized to $|0^n\rangle$. Applying a Walsh-Hadamard transform to each of these qubits yields an equal superposition of all n-bit classical states:

$$|0^n\rangle \xrightarrow{H} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

[0355] Consider now a function $f:\langle 0,1\rangle^n \rightarrow \langle 0,1\rangle$, that maps n-bit strings to a single bit. On a quantum computer, because unitary transformations are reversible, it is natural to implement it as a unitary transformation U_f that maps |x>|b> to |x>|b⊕f(x)>, where x is an n-bit string, b is a single bit, and "⊕" denotes the Exclusive −OR (XOR). Schematically,

$$|x\rangle|b\rangle \xrightarrow{U_f} |x\rangle|b \oplus f(x)\rangle.$$

[0356] Quantum computers can solve some problems exponentially faster than any classical computer provided the input is given as an oracle, even if bounded errors are allowed. In this model, some function $f:\langle 0,1\rangle^n \rightarrow \langle 0,1\rangle$ is given as a black-box, which means that the only way to obtain knowledge about f is to query the black-box on chosen inputs. In the corresponding quantum oracle model, a function f is provided by a black-box that applies unitary transformation U_f to any chosen quantum state, as described by:

$$|x\rangle|b\rangle \xrightarrow{U_f} |x\rangle|b \oplus f(x)\rangle.$$

[0357] The goal of the algorithm is to learn some property of the function f.

[0358] The linearity of quantum mechanics gives rise to quantum parallelism and two important phenomena, the first of which is quantum parallelism. It is possible to compute f on arbitrarily many classical inputs by a single application of U_f to a suitable superposition:

$$\sum_x \alpha_x |x\rangle|b\rangle \xrightarrow{U_f} \sum_x \alpha_x |x\rangle|f(x) \oplus b\rangle.$$

[0359] When this is done, the additional output qubit may become entangled with the input register;

[0360] The second phenomena is phase kick-back: The outcome of f can be recorded in the phase of the input register rather than being XOR-ed to the additional output qubit:

$$|x\rangle|-\rangle \xrightarrow{U_f} (-1)^{f(x)}|x\rangle|-\rangle;$$

$$\sum_x \alpha_x |x\rangle|-\rangle \xrightarrow{U_f} \sum_x \alpha_x (-1)^{f(x)}|x\rangle|-\rangle.$$

[0361] The fundamental questions in quantum computing are following:

[0362] The common measure of efficiency for computer algorithms is the amount of time required to obtain the solution as function of the input size. In the oracle context, this usually means the number of queries needed to gain a predefined amount of information about the solution. In contrast, one can fix a maximum number of oracle calls and to try to obtain as much Shannon information as possible about the correct answer. In this model, when a single oracle query is performed, the probability of obtaining the correct answer is better for the QA than for the optimal classical algorithm, and the information gained by that single query is higher. This is true even when no entanglement is ever present throughout the quantum computation and even when the state of the quantum computer is arbitrarily close to being totally mixed. QAs can be better than classical algorithms even when the state of the computer is almost totally mixed, which means that it contains an arbitrary small amount of information. It means that QAs can be better than classical algorithms even when no entanglement is present.

[0363] It is often believed that entanglement is essential for quantum computing. However, in many cases, quantum computing without entanglement is better than anything classically achievable, in terms the reliability of the outcome after a fixed number of oracle calls. It means that: (i) entanglement is not essential for all QAs; and (ii) some advantage of QAs over classical algorithms persists even when the quantum state contains an arbitrary small amount of information—that is, even when the state is arbitrarily close to being totally mixed.

[0364] A special quantum state known as a pseudo-pure state (PPS) can be used to describe entanglement-free quantum computation. PPS occurs naturally in the framework of Nuclear Magnetic Resonance (NMR) quantum computing. Consider any pure state |ψ> on n-qubits and some real number $0 \leqq \epsilon \leqq 1$. PPS has the following form:

$$\rho_{PPS}{}^n \equiv \epsilon|\psi\rangle\langle\psi| + (1-\epsilon)I.$$

[0365] It is a mixture of a pure state $|\psi\rangle$ with the totally mixed state

$$I = \frac{1}{2^n} I_{2^n}$$

(where I_{2^n} denotes the identity matrix of order 2^n). For example, the Werner state is a special case of PPS.

[0366] To understand why these states are called pseudo-pure, consider what happens if a unitary operation U is performed on state $\rho=\rho_{PPS}^{n}$.

[0367] First, the purity parameter ϵ of the PPS is conserved under a unitary transformation, since

$$\rho \xrightarrow{U} U\rho U^{\dagger}$$

and $UI\,U^{\dagger}=I$, and

$$U\rho U^{\dagger}=\epsilon U|\psi\rangle\langle\psi|U^{\dagger}+(1-\epsilon)UI\,U^{\dagger}=\epsilon|\phi\rangle\langle\phi|+(1-\epsilon)I,$$

where $|\phi\rangle=U|\psi\rangle$. In other words, unitary operations affect only the pure part of these states, leaving the totally mixed part unchanged and leaving the pure proportion ϵ intact.

[0368] For a PPS there exists some bias ϵ below which these states are never entangled. Thus, for any number n of qubits, a state ρ_{PPS}^{n} is separable whenever

$$\varepsilon < \frac{1}{1+2^{2n-1}},$$

regardless of its pure part $|\psi\rangle$.

[0369] Consider the density matrix $\rho_{PPS}^{n}\equiv\epsilon|\psi\rangle\langle\psi|+(1-\epsilon)I$. Its candidate ensemble probability satisfies

$$w(\vec{n}_1,\ldots,\vec{n}_N)=\frac{1-\varepsilon}{(4\pi)^N}+\varepsilon w(\vec{n}_1,\ldots,\vec{n}_N)\geq\frac{1-\varepsilon(1+2^{2N-1})}{(4\pi)^N}.$$

[0370] Therefore, ρ_{ϵ} is separable if

$$\varepsilon \leq \frac{1}{1+2^{2N-1}} \underset{N\to\infty}{\approx} \frac{2}{4^N}.$$

[0371] Here again, the density matrices in the neighborhood of the maximally mixed matrices are separable, and one obtains a lower bound on the size of the separable neighborhood. For $N\geqq 4$ the bound is better than the bound

$$\varepsilon \leq \frac{1}{(1+2^{N-1})^{N-1}}.$$

[0372] One illustrative example is the Greenberger-Horne-Zeilinger (GHZ) state, a state of three qubits with density matrix

$$\rho_{GHZ}=\frac{1}{2}(|111\rangle+|222\rangle)(\langle 111|+\langle 222|)=$$
$$=\frac{1}{8}(1_2\otimes 1_2\otimes 1_2+1_2\otimes\sigma_3\otimes\sigma_3+\sigma_3\otimes 1_2\otimes\sigma_3+$$
$$\sigma_3\otimes\sigma_3\otimes 1_2+\sigma_1\otimes\sigma_1\otimes\sigma_1-\sigma_1\otimes\sigma_2\otimes\sigma_2-$$
$$\sigma_2\otimes\sigma_1\otimes\sigma_2-\sigma_2\otimes\sigma_2\otimes\sigma_1,$$

which gives a representation

$$w_{GHZ}(\vec{n}_1,\ldots,\vec{n}_N)=\frac{1}{(4\pi)^3}[1+9(c_1c_2+c_2c_3+c_1c_3)+$$
$$27s_1s_2s_3\cos(\varphi_1+\varphi_2+\varphi_3)]\geq-\frac{26}{(4\pi)^3}.$$

[0373] Here $c_j\equiv\cos\theta_j$ and $s_j\equiv\sin\theta_j$, and the minimum occurs at $\theta_1=\theta_2=\theta_3=\pi/2$ and $\phi_1+\phi_2+\phi_3=\pi$. Thus, the mixed state $\rho_c=(1-\epsilon)M_8+\epsilon\rho_{GHZ}$ is separable if $\epsilon\leqq 1/27$, in which case, no measurement can reveal evidence of quantum entanglement.

[0374] Up to this point it has been assumed that the number of qubits is being fixed, and the boundary between separability and non-separability has been described as the amount of noise, specified by ϵ, changes. Now, the discussion shifts to thinking of the qubits as particles with spin and asking what happens as the number of particles or their dimension changes, while ϵ is held fixed. In general, going to more particles or higher spins, allows the system to tolerate more mixing with the maximally mixed state and still have states that are not separable. In other words, for a given ϵ, one can find states of sufficiently large numbers of particles or sufficiently high spin for which ρ_{ϵ} is non-separable. This yields an upper bound on the size of separable neighborhood around the maximally mixed state.

[0375] Consider now two spin-(d-1)/2 particles, each living in a d-dimensional Hilbert space. Each of these particles is an aggregate of N/2 spin-1/2 particles (qubits), in which case $d=2^{N/2}$. Consider a specific joint density matrix of the two particles,

$$\rho_{\epsilon}=(1-\epsilon)M_{d^2}+\epsilon|\psi\rangle\langle\psi|,$$

where $|\psi\rangle$ is a maximally entangled state of the two particles,

$$|\psi\rangle=\frac{1}{\sqrt{d}}(|1\rangle|1\rangle+|2\rangle|2\rangle+\ldots+|d\rangle|d\rangle).$$

[0376] Now project each particle onto the subspace spanned by $|1\rangle$ and $|2\rangle$. The state after projection is

$$\tilde{\rho} = \frac{1}{A}\left(\frac{1-\varepsilon}{d^2}1_4 + \frac{\varepsilon}{d}(|1\rangle|1\rangle + |2\rangle|2\rangle)\right)(\langle 1|\langle 1| + \langle 2|\langle 2|)$$
$$= (1-\varepsilon')M_4 + \varepsilon'|\phi\rangle\langle\phi|,$$

where

$$A = \frac{4}{d^2}\left[1 + \varepsilon\left(\frac{d}{2} - 1\right)\right]$$

is the normalization factor,

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|1\rangle|1\rangle + |2\rangle|2\rangle)$$

is a maximally entangle state of two qubits, and

$$\varepsilon' = \frac{2\varepsilon/d}{A} = \frac{\varepsilon d/2}{1+\varepsilon(d/2-1)}.$$

[0377] The projected state $\tilde{\rho}$ is a Werner state, a mixture of the maximally mixed state for two qubits, M_4, and the maximally entangled state $|\phi\rangle$. The proportion ϵ' of maximally entangled state increases linearly with d. Thus, as d increases for fixed ϵ, there is a critical dimension beyond which p becomes entangled. Indeed, the Werner state is non-separable for $\epsilon'>1/3$ which is equivalent to $d>\epsilon^{-1}-1$. Moreover, since the local projections on the two particles cannot create entanglement from a separable state, one can conclude that the state (14) of N qubits is non-separable under the same conditions, i.e., if

$$\varepsilon > \frac{1}{1+d} = \frac{1}{1+2^{N/2}}.$$

[0378] This result establishes an upper bound, scaling as $2^{-N/2}$ on the size of the separable neighborhood around the maximally mixed state. The general effect of noise on the computation, then the relationship between separability and noise is disclosed below.

[0379] Consider a pure-state computational protocol in which the computer starts in the state $|\psi_0\rangle$ and ends in the state $|\psi_f = U|\psi^0\rangle$, where U is the unitary time evolution operator which describes the computation. The corresponding computation starting with pseudo-pure state

$$\rho=(1-\epsilon)M+\epsilon|\psi_0\rangle\langle\psi_0|$$

ends up in the state

$$\rho=(1-\epsilon)M+\epsilon|\psi_f\rangle\langle\psi_f|.$$

[0380] Upon reaching the final state, a measurement is carried out and the result of the computation is inferred from the result of the measurement.

[0381] In the most favorable case, that the pure-state protocol gives the correct answer with certainty with a single repetition of the protocol and that if the result of computation is found, one can check it with polynomial overhead. The Pseudo Pure State (PPS) protocol uses the order of $1/\epsilon$ repetitions. Thus, if ϵ becomes exponentially small with N. the number governing the scaling of the classical problem (in other words, the noise becomes exponentially large with N), the protocol requires an exponential number of repetitions to get the correct answer. So, for this amount of noise, the quantum protocol with a PPS cannot transform an exponential problem into a polynomial one: even in the best possible case that the pure-state protocol takes one computational step, the protocol with noise takes exponentially many steps. This conclusion applies quite generally to pseudo-state quantum computing and is independent of the discussion of separability, which follows later.

[0382] In the PPS there is a probability ϵ of finding the computer in the "correct" final state $|\psi_f\rangle$ arising from the term $\epsilon|\psi_\eta\rangle|\psi_f\rangle$. As stated above, assume here the most favorable case, that if the state is $|\psi_f$ then, from the outcome of the final measurement, one can infer the solution to the computational problem with certainty with one repetition. In general protocols, such as Shor's algorithm, for example, a single repetition of the protocol is not sufficient to find the correct answer.

[0383] There is also the probability $(1-\epsilon)$ of finding the computer in the maximally mixed state M. In this case, there is a possibility that the correct answer will be found, since the noise term contains all possible outcomes with some probability. However, the probability of finding the correct answer from the noise term must be at least exponentially small with N. Otherwise, there would be no need to prepare the computer at all: one could find the correct answer from the noise term simply by repeating the computation a polynomial number of times. In fact, if the probability of finding the correct answer from the noise term did not become exponentially small with N, one could dispense with the computer altogether. For using a classical probabilistic protocol, which selected from all the possibilities at random, one would get the correct answer with probability of the order of one with only a polynomial number of trials.

[0384] Thus, the probability of finding the correct answer from the pseudo-pure state is essentially ϵ and so the computation must be repeated $1/\epsilon$ times on average to find the correct answer with probability of order one.

[0385] Now consider whether reaching entangled states during the computation is a necessary condition for exponential speed-up. This is addressed by investigating what can be achieved with separable states. Specifically, impose the condition that the pseudo-pure state remains separable during the entire computation. For an important class of computational protocols, it is shown that this condition implies an exponential amount of noise.

[0386] The example protocols shown herein use $n=n_1+n_2$ qubits of which n_1 are considered to be the input registers, and the remaining n_2 are the output registers. Assume that n_1 and n_2 are polynomial in the number N which describes how the classical problem scales. As stated earlier, the problems in which the quantum protocol gives an exponential speed-up over the classical protocol is to be considered, specifically the classical protocol is exponential in N whereas, the quantum protocol is polynomial in N. (For example, in the factorization problem, the aim is to factor a number of the order of 2^N. The classical protocol is exponential in N and, in Shor's algorithm, n_1 and n_2 are linear in N.)

[0387] In describing the protocols as applied to pure states, the first steps are as follows:

[0388] Prepare the system in the initial state:

$$|\psi_0\rangle=|00\ldots0\rangle\oplus|00\ldots0\rangle$$

[0389] Perform a Hadamard transform on the input register, so that the state becomes

$$|\psi_1\rangle=\frac{1}{2^{n_1/2}}\sum_{x=0}^{2^{n_1}-1}|x\rangle\otimes|00\ldots0\rangle$$

[0390] Evaluate the function f: $\langle 0,1\rangle^{n_1}\rightarrow\langle 0,1\rangle^{n_2}$. The state becomes

$$|\psi_2\rangle=\frac{1}{2^{n_1/2}}\sum_{x=0}^{2^{n_1}-1}|x\rangle\otimes|f(x)\rangle.$$

[0391] Now consider the protocol when applied to a mixed state input. Thus, the initial state ρ_0 is

$$\rho=(1-\epsilon)M_{2^n}+\epsilon|\psi_0\rangle\langle\psi_0|,$$

where M_{2^n} is the maximally mixed state in the 2^n dimensional Hilbert space. After the second computational step the state is

$$\rho=(1-\epsilon)M_{2^n}+\epsilon|\psi_2\rangle\langle\psi_2|.$$

[0392] Consider now protocols in which the function $f(x)$ is not constant. Let x_1 and x_2 values of x such that $f(x_1)\neq f(x_2)$. Thus the state $\psi_2\rangle$ can be written as

$$|\psi_2\rangle=\frac{1}{2^{n_1/2}}\{|x_1\rangle|f(x_1)\rangle+|x_2\rangle|f(x_2)\rangle+|\psi_r\rangle\}.$$

where $|\psi_r\rangle$ has no components in the subspace spanned by $|x_1\rangle|f(x_1)\rangle$, $|x_1\rangle|f(x_2)\rangle$, $|x_2\rangle|ff(x_1)\rangle$, $|x_2\rangle|f(x_2)\rangle$. It is convenient to relabel these states and write

$$|\psi_2\rangle=\frac{1}{2^{n_1/2}}\{|1\rangle|1\rangle+|2\rangle|2\rangle+|\psi_r\rangle\},$$

where $|\psi_r\rangle$ has no components in the subspace spanned by $|1\rangle|1\rangle$, $|1\rangle$, $|2\rangle$, $|2\rangle|1\rangle$, $|2\rangle|1\rangle$.

[0393] A necessary condition on ϵ for the state of the system to be separable throughout the computation is obtained by considering projecting each particle onto the subspace spanned by $|1\rangle$ and $|2\rangle$. The state after projection is

$$\rho_2'=\frac{1}{A}\left[\frac{4(1-\varepsilon)}{2^{n_1+n_2}}M_4+\frac{2\varepsilon}{2^{n_1}}\left(\frac{|1\rangle|1\rangle+|2\rangle|2\rangle}{\sqrt{2}}\right)\left(\frac{\langle 1|\langle 1|+\langle 2|\langle 2|}{\sqrt{2}}\right)\right]=,$$

$$=(1-\varepsilon')M_4+\varepsilon'\left(\frac{|1\rangle|1\rangle+|2\rangle|2\rangle}{\sqrt{2}}\right)\left(\frac{\langle 1|\langle 1|+\langle 2|\langle 2|}{\sqrt{2}}\right)$$

-continued

where

$$A=\left(\frac{4(1-\varepsilon)}{2^{n_1+n_2}}+\frac{2\varepsilon}{2^{n_1}}\right)$$

is the normalization factor, M_4 is the maximally mixed state in the four-dimensional Hilbert space spanned by $|1\rangle|1\rangle$, $|1\rangle|2\rangle$, $|2\rangle$, $|1\rangle$, $|2\rangle$|2$\rangle$, and

$$\varepsilon'=\frac{2\varepsilon}{2^{n_1}A}=\frac{\varepsilon}{(1-\varepsilon)2^{-n_2+1}+\varepsilon}.$$

[0394] Now a two qubit state of the form

$$(1-\delta)M_4+\delta\left(\frac{|1\rangle|1\rangle+|2\rangle|2\rangle}{\sqrt{2}}\right)\left(\frac{\langle 1|\langle 1|+\langle 2|\langle 2|}{\sqrt{2}}\right)$$

is entangled for $\delta>1/3$. Therefore, the original state must have been entangled unless

$$\varepsilon'\leq 1/3\Rightarrow\varepsilon\leq\frac{1}{1+2^{n_2}},$$

since local projections cannot create entangled states from un-entangled ones.

[0395] Therefore, a computational protocol (for non-constant f) involves starting with a mixed state and, if the state remains separable throughout the protocol, then

$$\varepsilon\leq\frac{1}{1+2^{n_2}}.$$

[0396] However, even in favorable circumstances, a computation with noise ϵ takes of the order of $1/\epsilon$ repetitions to get the correct answer with probability of the order of one.

[0397] Thus, computational protocols of the sort considered require exponentially-many repetitions. So no matter how efficient the original pure-state protocol is, the mixed-state protocol, which is sufficiently noisy that it remains separable for all N, will not transform an exponential classical problem into a polynomial one.

[0398] When $|\psi\rangle$ is entangled but ρ_{PPS}^{n} is separable, the PPS exhibits pseudo-entanglement. The condition

$$\varepsilon<\frac{1}{1+2^{2n-1}}$$

is sufficient for separability but not necessary. Thus, entanglement will not appear in a quantum unitary computation that starts in a separable PPS whose purity parameter ϵ obeys

$$\varepsilon < \frac{1}{1+2^{2n-1}}.$$

A final measurement in the computational basis will not make entanglement appear either.

[0399] Two examples: the solutions of Deutsch-Jozsa and Simon's problems are now shown without entanglement.

[0400] For the Deutsch-Jozsa problem, given a function f: $\{0,1\}^n \rightarrow \langle 0,1\rangle$ in the form of an oracle (or black-box), assume that either this function is promised to be either constant, $f(x)=f(y)$, or that it is balanced, $f(x)=0$, on exactly half the n-bit strings x. The task is to decide which is the case. A single oracle call (in which the input is given in superposition) suffices for a quantum computing to determine the answer with certainty, whereas no classical computing can be sure of the answer before it has asked $2^{n-1}+1$ questions. More to the point, no information at all can be derived from the answer to a single classical oracle call.

[0401] The QA of Deutsch-Jozsa (DJ) solves this problem with a single query to the oracle by starting with state $|0^n\rangle$|$1\rangle$ and performing a Walsh-Hadamard transform on all n+1 qubits before and after the application entanglement operator (quantum oracle) U_f. A measurement of the first n qubits is made at the end (in computational basis), yielding classical n-bit string z.

[0402] By virtue of phase kick-back, the initial Walsh-Hadamard transforms and the application of U_f results in the following state:

$$|0^n\rangle|1\rangle \xrightarrow{H} \left(\frac{1}{\sqrt{2^n}}\sum_x |x\rangle\right)|-\rangle \xrightarrow{U_f} \left(\frac{1}{\sqrt{2^n}}\sum_x (-1)^{f(x)}|x\rangle\right)|-\rangle.$$

[0403] Then, if f is constant, the final Walsh-Hadamard reverts the state back to $\pm|0^n\rangle|1\rangle$, in which the overall phase is "+" if $f(x)=0$ for all x and "−" if $f(x)=1$ for all x. In either case, the result of the final measurement is necessarily z=0. On the other hand, if f is balanced, the phase of half the $|x\rangle$ in the above expression is + and the phase of the other half is −. As a result, the amplitude of $|0^n\rangle$ is zero after the final Walsh-Hadamard transforms because each $|x\rangle$ is sent to

$$+\frac{1}{\sqrt{2^n}}|0^n\rangle + \ldots$$

by those transforms.

[0404] Therefore, the final measurement cannot produce z=0. It follows from the promise that if z=0 it can be concluded that f is constant and if $z\neq 0$, then it can be concluded that f is balanced. Either way, the probability of success is 1 and the QA provides full information on the desired answer.

[0405] On the other hand, due to the special nature of the DJ-problem, a single query does not change the probability of guessing correctly whether the function is balanced or constant. Therefore, the following proposition holds: When restricted to a single DJ-oracle call, a classical computing algorithm learns no information about type of f. In sharp contrast, the advantage of quantum computing even without entanglement: When restricted to a single DJ-oracle call, a quantum computing whose state is never entangled can learn a positive amount of information about the type of f.

[0406] In this case, starting with a PPS in which the pure part is $|0^n\rangle$|$1\rangle$ and its probability is ϵ, one can still follow the DJ-strategy, but now it becomes a guessing game. One can obtain the correct answer with different probabilities depending on whether f is constant or balanced: If f is constant, then z=0 with the probability

$$P(z=0 \mid f \text{ is constant}) = \varepsilon + \frac{1-\varepsilon}{2^n}$$

because the algorithm started with state $|0^n\rangle$|$1\rangle$ with probability ϵ, in which case DJ-QA is guaranteed to produce z=0 since f is constant, or it started with a completely mixed state with complementary probability $1-\epsilon$, in which case DJ-QA produces a completely random z whose probability of being zero is 2^{-n}.

[0407] Similarly,

$$P(z\neq 0 | f \text{ is constant}) = (1-\varepsilon)\frac{2^n-1}{2^n}.$$

[0408] If f is balanced one obtains a non-zero z with probability

$$P(z\neq 0 | f \text{ is balanced}) = \varepsilon + (1-\varepsilon)\frac{2^n-1}{2^n},$$

and z=0 is obtained with probability

$$P(z=0 | f \text{ is balanced}) = \frac{1-\varepsilon}{2^n}.$$

[0409] Therefore, for all positive ϵ and all n, an advantage is observed over classical computing.

[0410] In particular, this is true for

$$\varepsilon < \frac{1}{1+2^{2n-1}},$$

in which case the state remains separable throughout the entire computation in

$$\varepsilon < \frac{1}{1+2^{2n-1}}$$

with n+1 qubits.

[0411] An information analysis of the DJ problem without entanglement begins by assuming the a priori probability of f being constant is p (and therefore, the probability that it is balanced is 1−p). The following diagrams describe the probability that zero (or non-zero) is measured, given a constant (or balanced) function, in pure and the totally mixed cases.

[0412] The case of pseudo-pure state is the weighted sum of the previous cases. The details of the pseudo-pure case are summarized in the joint probability Table 5.5.

TABLE 5.5

Joint probability of function type (X) and measurement (Y)

X	y = zero	y = non-zero
constant	$p\left(\varepsilon+\frac{1-\varepsilon}{2^n}\right)$	$p(1-\varepsilon)\left(1-\frac{1}{2^n}\right)$
balanced	$(1-p)\frac{1-\varepsilon}{2^n}$	$(1-p)\left(1-\frac{1-\varepsilon}{2^n}\right)$
P(Y = y)	$p_0 = p\varepsilon+\frac{1-\varepsilon}{2^n}$	$1-p_0$

[0413] Thus, the probability p_0 of obtaining $z=0$ is

$$\varepsilon\cdot p+\frac{1-\varepsilon}{2^n}.$$

To quantify the amount of information gained about the function, given the outcome of the measurement, calculate the mutual information between X and Y, where X is a random variable signifying whether f is constant or balanced, and Y is a random variable signifying whether z=0 or not. Let the entropy function of a probability q be h(q)≡−q log q−(1−q)log(1−q). The marginal probability of Y and X may be calculated from that table, and using Bayes rule,

$$P(X|Y)=\frac{P(Y|X)P(X)}{P(Y)},$$

the conditional probabilities are

$$P(X=\text{constant}|Y=\text{zero})=\frac{p}{P_0}\left(\varepsilon+\frac{1-\varepsilon}{2^n}\right),$$

$$P(X=\text{constant}|Y=\text{non-zero})=\frac{p(1-\varepsilon)}{1-P_0}\left(1-\frac{1}{2^n}\right)$$

where

-continued

$$p_0=P(Y=\text{zero})=p\varepsilon+\frac{1-\varepsilon}{2^n}.$$

[0414] The conditional entropy is

$$H(X|Y)=\sum_y P(Y=y)h(P(X=\text{constant}|Y=y))$$

$$=p_0 h\left(\frac{p}{p_0}\left[\varepsilon+\frac{1-\varepsilon}{2^n}\right]\right)+(1-p_0)h\left(\frac{p(1-\varepsilon)}{1-p_0}\left[1-\frac{1}{2^n}\right]\right).$$

[0415] Then, the mutual information gained by a single quantum query is

$$I(X;Y)=H(X)-H(X|Y)$$

$$=h(p)-p_0 h\left(\frac{p}{p_0}\left[\varepsilon+\frac{1-\varepsilon}{2^n}\right]\right)-$$

$$(1-p_0)h\left(\frac{p(1-\varepsilon)}{1-p_0}\left[1-\frac{1}{2^n}\right]\right).$$

[0416] The mutual information is positive for every ϵ>0, unless p=0 or p=1. This is more than the zero amount of information gained by a single classical query. For p=1/2 this reduced into

$$1-\frac{1+\varepsilon(2^{n-1}-1)}{2^n}h\left(\frac{1+\varepsilon(2^n-1)}{2(1+\varepsilon(2^n-1))}\right)-\frac{2^n-1-\varepsilon(2^{n-1}-1)}{2^n}h\left(\frac{(\varepsilon-1)(2^n-1)}{2(1+\varepsilon(2^n-1)-2^n)}\right)$$

and, for very small

$$\varepsilon\left(\varepsilon\square\frac{1}{2^n}\right),$$

using the fact that

$$h\left(\frac{1}{2}+x\right)=1-\frac{2x^2}{\ln 2}+O(x^4),$$

this expression may be approximate by

$$I(X;Y)=1-p_0 h\left(\frac{1}{2}+\frac{2^n\varepsilon}{4}+O(2^n\varepsilon^2)\right)-(1-p_0)h$$

-continued

$$\left(\frac{1}{2}+\frac{\varepsilon}{1-\frac{4}{2^n}}+O(2^n\varepsilon^2)\right)$$

$$=\frac{2^{2n}\varepsilon^2}{8(2^n-1)\ln 2}+O(2^n\varepsilon^2)>0$$

[0417] Consider, for example, the case when

$$p=\frac{1}{2},\ n=3 \text{ and } \varepsilon=\frac{1}{1+2^{2n+1}}=\frac{1}{129}.$$

In this case, I(X; Y)=0.0000972 bits of information are gained. Therefore, some information is gained even for separable PPSs, in contrast to the classical case where the mutual information is always zero. Furthermore, some information is gained even when ϵ is arbitrarily small.

[0418] It is possible to improve the expected amount of information that is obtained by a single call to the oracle by measuring the (n+1)-st qubit and take it into account. Indeed, this qubit should be $|1\rangle$ if the configuration comes from the pure part. Therefore, if that extra bit is $|0\rangle$, which happens with probability

$$\frac{1-\varepsilon}{2},$$

it is known that the PPS contributes the fully mixed part, hence no useful information is provided by z and the situation is better than in the classical case. Indeed, when that extra bit is $|1\rangle$, which happens with probability

$$\frac{1+\varepsilon}{2},$$

the probability of the pure part is enlarged from ϵ to

$$\hat{\varepsilon}=\frac{2\varepsilon}{1+\varepsilon},$$

and the probability of the mixed part is reduced from

$$1-\varepsilon \text{ to } 1-\hat{\varepsilon}=\frac{1-\varepsilon}{1+\varepsilon}.$$

The probability of z=0 changes to

$$\hat{p}_0=p\hat{\varepsilon}+\frac{1-\hat{\varepsilon}}{2^n}$$

and mutual information to

$$I(X;Y)=$$

$$\frac{1+\varepsilon}{2}\left[h(p)-\hat{p}_0 h\left(\frac{p}{\hat{p}_0}\left[\hat{\varepsilon}+\frac{1-\hat{\varepsilon}}{2^n}\right]\right)-(1-\hat{p}_0)h\left(\frac{p(1-\hat{\varepsilon})}{1-\hat{p}_0}\left[1-\frac{1}{2^n}\right]\right)\right]$$

which, for

$$p=\frac{1}{2}$$

and very small ϵ, gives:

$$I(X;Y)=\frac{2^{2n}\varepsilon^2}{4(2^n-1)\ln 2}+O(2^n\varepsilon^3)>0.$$

[0419] This is essentially twice as much information as in the above case.

[0420] For the specific example of

$$p=\frac{1}{2},\ n=3 \text{ and } \varepsilon=\frac{1}{129},$$

this is 0.000189 bits of information.

[0421] In the Simon algorithm, an oracle calculates a function $f(x)$ from n bits to n bits under the promise that f is a two-to-one function, so that for any x there exists a unique y≠x such that $f(x)=f(y)$. Furthermore, the existence of an s≠0 is promised such that $f(x)=f(y)$ for y≠x iff y=x⊕s. The goal is to find s, while minimizing the number of times f is calculated. Classically, even if one calls function f exponentially many times, say $\sqrt[4]{2^n}$ **times, the probability of finding s is still exponentially small with n that is less than**

$$\frac{1}{\sqrt{2^n}}.$$

However, there exists a QA that requires only O(n) computations of f. The algorithm, due to Simon, is initialized with $|0^n\rangle|0^n\rangle$. It performs a Walsh-Hadamard transform on the first register and calculates f for all inputs to obtain

$$|0^n\rangle|0^n\rangle \xrightarrow{H} \frac{1}{\sqrt{2^n}}\sum_x |x\rangle|0^n\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle,$$

which can be written as

$$\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle = \frac{1}{\sqrt{2^n}}\sum_{x<x\oplus s} (|x\rangle + |x\oplus s\rangle)f(x)\rangle.$$

[0422] Then, the Walsh-Hadamard transform is performed again on the first register (the one holding the superposition of all $|x\rangle$), which produces state

$$\frac{1}{\sqrt{2^n}}\sum_{x<x\oplus s}\sum_j ((-1)^{j\cdot x}+(-1)^{j\cdot x\oplus j\cdot s})|j\rangle|f(x)\rangle.$$

[0423] Finally, the first register is measured.

[0424] The outcome j is guaranteed to be orthogonal to s ($j\cdot s=0$) since otherwise, $|j\rangle$'s amplitude $(-1)^{j\cdot x}(1+(1-)^{j\cdot s})$ is zero. After an expected number of such queries in O(n), one obtains n linearly independent j s that uniquely define s.

[0425] For example, let S be the random variable that describe parameter s, and let J be a random variable that describes the outcome of a single measurement. To quantify how much information about S is gained by a single query, assume that S is distributed uniformly in the range [1 . . . 2^n-1], its entropy before the first query is $H(S)=1g(2^n-1)\approx n$. In the classical case, a single evaluation of f gives no information about S: the value of $f(x)$ on any specific x says nothing about its value in different places, and therefore, nothing about s. However, in the case of the QA, one is assured that s and j are orthogonal. If the measured j is zero, s could still be any one of the (2^n-1) non-zero values and no information is gained. But in the overwhelmingly more probable case that j is non-zero, only $(2^{n-1}-1)$ values for s are still possible. Thus, given the outcome of the measurement, the entropy of S drops to approximately n−1 bits and the expected information gain is nearly one bit.

[0426] In order to estimate the entropy, let S be a random variable that represents the sought-after parameter of Simon's function, so that $\forall x$: $f(x)=f(x\oplus s)$. Assume that S is distributed uniformly in the range [1 . . . 2^n-1]. Given that S=s, and starting with a PPS whose purity is ϵ, one can find the distribution of the measurement after a single query. With probability ϵ, one starts with the pure part and measures a j that is orthogonal to s. With probability $1-\epsilon$ one starts with the totally mixed state and measures a random j . Thus, for j so that

$$j\cdot s=0,\ P(J=j\mid S=s)=\varepsilon\frac{2}{2^n}+\frac{(1-\varepsilon)}{2^n},$$

and for j so that

$$j\cdot s=1,\ P(J=j\mid S=s)=\frac{(1-\varepsilon)}{2^n}.$$

-continued

$$P(J=j\mid S=s)=\begin{cases}\frac{1+\varepsilon}{2^n} & \text{if } j\cdot s=0\\ \frac{1-\varepsilon}{2^n} & \text{if } j\cdot s=1\end{cases}.$$

Putting this together,

[0427] The marginal probability of J for any j≠0 is

$$\begin{aligned}P(J=j)&=\sum_s P(s)P(j\mid s) \quad ,\\ &=\frac{1}{2^{n-1}}\left(\sum_{s\cdot j} P(j\mid s)+\sum_{s\neg\cdot j} P(j\mid s)\right)\\ &=\frac{(2^{n-1}-1)\frac{1+\varepsilon}{2^n}+2^{n-1}\frac{1-\varepsilon}{2^n}}{2^n-1}\\ &=\boxed{\frac{1-\frac{1+\varepsilon}{2^n}}{2^n-1}}\end{aligned}$$

while for J=0, all values of s are orthogonal, and

$$\begin{aligned}P(J=0)&=\sum_s P(s)P(J=0\mid s) \quad .\\ &=\frac{1}{2^n-1}\sum_{s\perp j}' P(J=0\mid s)\\ &=\frac{1}{2^n-1}(2^{n-1}-1)\frac{1+\varepsilon}{2^n}\\ &=\boxed{\frac{1+\varepsilon}{2^n}}\end{aligned}$$

[0428] By the definition, the entropy of the random variable J is

$$H(J)=-\sum_j P(J=j)1gP(J=j)=-\left(1-\frac{1+\varepsilon}{2^n}\right)1g\left(\frac{1-\frac{1+\varepsilon}{2^n}}{2^n-1}\right)-\frac{1+\varepsilon}{2^n}1g\frac{1+\varepsilon}{2^n},$$

and the conditional entropy of J given S=s is

$$\begin{aligned}H(J\mid S=s)&=-\sum_j P(J=j\mid S=s)1gP(J=j\mid S=s)\\ &=-2^{n-1}\frac{1+\varepsilon}{2^n}1g\left(\frac{1+\varepsilon}{2^n}\right)-2^{n-1}\frac{1-\varepsilon}{2^n}1g\left(\frac{1-\varepsilon}{2^n}\right)\\ &=\boxed{-\frac{1+\varepsilon}{2}1g\left(\frac{1+\varepsilon}{2^n}\right)-\frac{1-\varepsilon}{2}1g\left(\frac{1-\varepsilon}{2^n}\right)}\end{aligned}$$

[0429] Since the above mentioned expression is independent of the specific values s, it also equals to H(S|J), which is

$$\sum_s P(S=s)H(J\mid S=s).$$

Finally, the amount of knowledge about S that is gained by knowing J is their mutual information:

$$\begin{aligned} I(S;J) &= I(J;S) \\ &= H(J) - H(J\mid S) \\ &= -\left(1-\frac{1+\varepsilon}{2^n}\right)1g\left(\frac{1-\frac{1+\varepsilon}{2^n}}{2^n-1}\right) + (2^{n-1}-1)\frac{1+\varepsilon}{2^n}1g \\ &\quad \frac{1+\varepsilon}{2^n} + \frac{1-\varepsilon}{2}1g\left(\frac{1-\varepsilon}{2^n}\right), \end{aligned}$$

[0430] Consider two extremes: in the pure case (=1), I(S;J)=1−O(2^{-n}) and in the totally mixed case (ϵ=0), I(S;J)=0.

[0431] Finally, it can be shown that for small ϵ the value

$$I(S;J) = \frac{(2^n-2)\varepsilon^2}{2(2^n-1)1n2} + O(\varepsilon^3).$$

[0432] More formally, based on the conditional probability

$$P(J=j\mid S=s) = \begin{cases} \frac{2}{2^n} & \text{if } j\cdot s=0 \\ 0 & \text{if } j\cdot s=1 \end{cases},$$

it follows that the conditional entropy H(J|S=s)=n−1, which does not depend on the specific s and, therefore, H(J|S)=n−1 as well. In order to find the a proiri entropy of J, calculate its marginal probability

$$P(J=j) = \sum_s P(s)P(j\mid s) = \begin{cases} \frac{1-\frac{2}{2^n}}{2^n-1} & \text{if } j\neq 0 \\ \frac{2}{2^n} & \text{if } j=0 \end{cases}.$$

[0433] Thus,

$$\begin{aligned} H(J) &= -\sum_j P(J=j)1gP(J=j) \\ &= -\left(1-\frac{2}{2^n}\right)1g\frac{1-\frac{2}{2^n}}{2^n-1} - \frac{2}{2^n}1g\frac{2}{2^n} \end{aligned}$$

-continued

$$= \left(1-\frac{2}{2^n}\right)\left(n+1g\frac{2^n-1}{2^n-2}\right) + \frac{n-1}{2^{n-1}}$$

and the mutual information

$$I(S;J) = 1 - \frac{2-(2^n-2)\varepsilon^2}{2^n}1g\frac{2^n-1}{2^n-2} = 1-O(2-n)$$

is almost one bit.

[0434] In contrast, a single query to a classical oracle provides no information about s. When restricted to a single oracle call, a classical computing algorithm learns no information about Simon's parameter s. Again in sharp contrast, the following result shows the advantage of quantum computing without entanglement, compared to classical computing. When restricted to a single oracle call, a quantum computing algorithm whose state is never entangled can learn a positive amount of information about Simon's parameter s.

[0435] For example, starting with a PPS in which the pure part is $|0^n\rangle|0^n\rangle$, and its probability is ϵ, the acquired j is no longer guaranteed to be orthogonal to s. In fact, an orthogonal j is obtained with probability

$$\frac{1+\varepsilon}{2}$$

only. For any value of S, the conditional distribution of J as above mentioned is

$$P(J=j\mid S=s) = \begin{cases} \frac{1+\varepsilon}{2^n} & \text{if } j\cdot s=0 \\ \frac{1-\varepsilon}{2^n} & \text{if } j\cdot s=1 \end{cases}$$

from which it is calculated that the information gained about S given the value of J is

$$I(S;J) = -\left(1-\frac{1+\varepsilon}{2^n}\right)1g\left(\frac{1-\frac{1+\varepsilon}{2^n}}{2^n-1}\right) + (2^{n-1}-1)\frac{1+\varepsilon}{2^n}1g\frac{1+\varepsilon}{2^n} + \frac{1-\varepsilon}{2}1g\left(\frac{1-\varepsilon}{2^n}\right).$$

[0436] The amount of information is larger than the classical zero for every ϵ>0. This result is true even for ϵ as small as

$$\frac{1}{1+2^{2(2n)-1}},$$

in which case the state of the computing is never entangled throughout the computation.

[0437] When n=3 and

$$\varepsilon = \frac{1}{1+2^{4\cdot 3-1}} = \frac{1}{2049},$$

147×10^{-9} bits of information are gained.

5.3. Quantum Computing for Design of Robust Wise Control

[0438] Decomposition of the optimization process in design of a robust KB for an intelligent control system is separated in two steps: (1) global optimization based on a Quantum Genetic Search Algorithm (QGSA); and (2) a learning process based on a QNN for robust approximation of the teaching signal from a QGSA.

[0439] **FIG. 40** shows the interrelations between Soft Computing and Quantum Soft Computing for simulation, global optimization, quantum learning and the optimal design of a robust KB in intelligent control systems. The main problem of KB-optimization based on soft computing lies in the design process using one solution space for global optimization. As an example, consider a design of a KB for a fixed class of stochastic excitations on a control object. If the design process is based on many solution spaces with different statistical characteristics of stochastic excitations of the control object, then the GA cannot necessarily find a global solution for an optimal KB. In this case, for global optimization, a QGSA is used to find the KB. In one embodiment, optimization methods of intelligent control system structures (based on quantum soft computing) use a modification of simulation methods for quantum computing.

Quantum Control Algorithm for Robust KB-FC Design.

[0440] **FIG. 41***a* is a block diagram of the structure of an intelligent control system based on a PD-fuzzy controller (PD-FC). In **FIG. 41***a*, a conventional PD (or PID) controller **4102** controls a plant **4103**. A control output from the controller **4102** and an output from the plant **4103** are provided to a QGSA **4101**. A globally optimized KB from the QGSA **4101** is provided to a Fuzzy Controller (FC) **4104**. Gain schedules from the FC **4104** are provided to the PD controller **4102**. An error signal, computed as a difference between an output of the plant **4103** and an input signal is provided to the FC **4104** and to the PD controller **4102**.

[0441] Using a soft computing optimizer, it is possible to design partial KB(i) for the FC **4104** from simulation of control object behaviour using different classes of stochastic excitations. For many cases this KB(i) is not robust if another type of stochastic excitations is applied to the control object (plant) **4103** or if the reference signal is changed. The problem lies in design of a unified robust KB from a number of finite number KB(i) look-up tables created by soft computing and finding a globally optimized KB for intelligent fuzzy control under stochastic excitations.

[0442] The KB can be considered as an ordered DB containing control laws of coefficient gains for a traditional PID controller. The superposition operator is used for design of relations between coefficient gains of the PID-FC. Grover's QSA is used for searching of solutions and max operation between decoding states is analogy of the measurement process of solution search.

[0443] As described above, in an entanglement-free quantum computation no resource increases exponentially. The concrete example below shows that it is possible to design a robust intelligent globally-optimzed KB using a superposition of non-robust KBs. In this case, the quality of control based on the globally optimized KB is more effective than the non-robust KBs obtained by local optimization. In this case, wise robust control is introduced, where wise≡intelligent⊕smart. This situation is similar to the Parrondo Paradox in a quantum game. In design process of wise control, entanglement is not used and thus, it is different from Parrondo Paradox.

[0444] For an entanglement-free quantum control algorithm for design of a robust wise KB-FC, consider one of the examples of quantum computing approach to design robust wise quantum control. As described, **FIG. 41***a* shows the structure of an intelligent control system based on a fuzzy PD-controller (PD-FC). A soft computing optimizer is used to a group of partial knowledge bases KB(i) for the PD-FC from fuzzy simulation of behavior of the plant **4103** using different class of stochastic excitations. For many cases, these KB(i) are not robust used with different type of stochastic excitations, changing initial states, or changing the type of reference signals. The problem lies in design of a unified robust globally optimized KB from the KB(i) look-up tables created by soft computing.

[0445] The entropy of an orthogonal matrix provides a new interpretation of Hadamard matrices as those that saturate the bound for entropy. This definition plays a role in QAs simulation, while the Hadamard matrix is used for preparation of superposition states and in entanglement-free QAs. The entropy of orthogonal matrices and Hadamard matrices (appropriately normalized) saturate the bound for the maximum of the entropy. The maxima (and other saddle points of the entropy function have an intriguing structure and yield generalizations of Hadamard matrices.)

[0446] Consider n random variables with a set of possible outcomes i=1, . . . , n having probabilities p_i, i=1, . . . , n. Then

$$\sum_{i=1}^{n} p_i = 1$$

and the Shannon entropy

$$S^{Sh}(p_i) = -\sum_{i=1}^{n} p_i \ln p_i.$$

[0447] Now define entropy of an orthogonal matrix O^i_n, i, j=1, . . . , n. Here O^i_j are real numbers with the constraint

$$\sum_{i=1}^{n} O_j^i O_k^i = \delta_{jk}.$$

In particular, the j th row of the matrix is a normalized vector for each i=1, . . . , n. It is possible to associate probabilities $p_j^{(i)}=(O^i_j)^2$ with the i th row, as

$$\sum_{j=1}^{n} p_j^{(i)} = 1$$

for each i. Define the Shannon entropy for the orthogonal matrix as the sum of the entropies for each row:

$$S^{Sh}(O_j^i) = -\sum_{i,j=1}^{n} (O_j^i)^2 \ln(O_j^i)^2.$$

[0448] The minimum value zero is attained by the identity matrix $O^i_j=\delta_j^i$ and related matrices obtained by interchanging rows or changing the signs of the elements. The entropy of the i th row can have the maximum value In n, which is attained when each element of the row is

$$\pm\frac{1}{\sqrt{n}}.$$

This gives the bound, $S^{Sh}(O^i_j)\leqq$n 1n n.

[0449] In general the entropy of an orthogonal matrix cannot attain this bound because of the orthogonality constraint

$$\sum_{i=1}^{n} O_j^i O_k^i = \delta_{jk}$$

which constraints $p_j^{(i)}$ for different rows. In fact the bound is obtained only by the Hadamard matrices (rescaled by

$$\frac{1}{\sqrt{n}}\Big).$$

This yields the criterion for the Hadamard matrices (appropriately normalized): those orthogonal matrices which saturate the bound for entropy.

[0450] The entropy is large when each element is as close to

$$\pm\frac{1}{\sqrt{n}}s$$

possible, i.e., to a main diagonal. Thus, maximum entropy is similar to the maximum determinant condition of the Hadamard. The peaks of the entropy are isolated and sharp in contrast to the determinant.

[0451] For, example, a matrix that maximizes the entropy for n=3 is

$$n=3\Rightarrow\begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{pmatrix};$$

$$n=5\Rightarrow\begin{pmatrix} -\frac{3}{5} & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} \\ \frac{2}{5} & -\frac{3}{5} & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{2}{5} & -\frac{3}{5} & \frac{2}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & -\frac{3}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & \frac{2}{5} & -\frac{3}{5} \end{pmatrix}.$$

[0452] For n =5, the result is similar as in the case n=3: the magnitudes of the elements in each row are

$$\frac{2}{5}$$

repeated 4 times and a diagonal element is as

$$\Big(-\frac{3}{5}\Big).$$

[0453] This set can be generalized for any n. The matrix with

$$-\frac{n-2}{n}$$

along the diagonal and each off-diagonal as

$$\frac{2}{n}$$

is orthogonal. Each row is normalized as a consequence of the identity:

$$n^2=(n-2)^2+2^2(n-1).$$

[0454] For each n, there are saddle points apart from maxima and minima.

[0455] For n=3 there is a saddle point and the corresponding matrix is

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{pmatrix}.$$

[0456] The entropy peaks sharply at extrema. Thus, the entropy has a rich set of sharp extrema.

[0457] This result shows the role of the Hadamard operator in an entanglement-free QA: with the Hadamard transformation it is possible to introduce maximally-hidden information about classical basis independent states, and the superposition includes this maximal information. Thus, with superposition operator, it is possible to create a new QA without entanglement, while the superposition includes information about the property of the function f.

[0458] **FIG. 42** shows the structure of the design process for using the above approach in design of a robust KB for fuzzy controllers. The superposition operator used is the particular case of a QFT—the Walsh-Hadamard transform. The KB(i) of the PD-FC includes the set of coefficient gains $K=\langle k_P(t), k_D(t)\rangle$ laws received from soft computing simulation using different types of random excitations on the plant **4103**. **FIG. 43** shows the structure of a quantum control algorithm for design of a robust unified KB-FC from two KBs created by soft computing optimizer for Gaussian (KB(1)) and non-Gaussian (with Rayleigh probability density function)—KB(2) noises.

[0459] The algorithm includes the following operations:

[0460] 1. Prepare two registers of n qubits in the state $|0 \ldots 0\rangle \epsilon H_N$.

[0461] 2. Apply H over the first register.

[0462] 3. Apply diffusion (interference) operator G over the whole quantum state.

[0463] 4. Apply max operation over the first register.

[0464] 5. Measure the first register and output the measured value.

[0465] Normalized real simulated coefficient gains $\langle K_P(t),K_D(t)\rangle$ can be calculated using the values of virtual coefficient gains $\langle k_P^Q(t),k_D^Q(t)\rangle$ as logical negation: $\langle k_P^Q(t),k_D^Q(t)\rangle=1-\langle k_P(t), k_D(t)\rangle$. For example, if the value of the proportional coefficient gain, $k_P(t_i)$, is $k_P(t_i)=0,2$, then $k_P^Q(t_i)=1-0,2=0,8$.

[0466] **FIG. 41***b* shows the geometrical interpretation of this computational process.

[0467] **FIG. 42** shows the logical description of superposition between real and virtual values of coefficient gains created by soft computing simulation. For this case four classical states are joint in one non-classical superposition state with amplitude probability

$$\frac{1}{2}.$$

[0468] For the above described example, the following coding result: $|0_1\rangle \rightarrow 0.2$, $|1_1\rangle \rightarrow 0.8$ is obtained.

[0469] In one embodiment, the computational control algorithm includes the following operations:

[0470] 1. The current values (for fixed time t_i) of the coefficient gains are coded as real values.

[0471] 2. Hadamard matrices are created for superposition between real simulated and virtual classical states. The virtual classical state is calculated from the normalized scale [0,1] (the complementary quantum law is the logical negation of the real simulated value). The Hadamard transform joins two classical states in one non-classical state as a superposition:

$$\frac{1}{\sqrt{2}}[|0_1\rangle+|1_1\rangle] = \frac{1}{\sqrt{2}}[|\text{Yes}\rangle+|\text{No}\rangle]$$

that it is not found in classical mechanics. This operation creates the possibility of extraction of hidden quantum information from classical contradictory states.

[0472] 3. Grover's diffusion operator is used to provide an interference operation search for the solution.

[0473] 4. The Max operation is applied to the classical states in the superposition after the decoding of results.

[0474] The results of the quantum computation are used in new control laws (new coefficient gains) from two KB(i), i=1,2 created from soft computing technology

$$\ddot{x}+(x^2-1)\dot{x}+x=k_P(t)e+k_D(t)\dot{e}+\xi(t) \qquad (4.1)$$

under Gaussian random white noise $\xi(t)$.

[0475] **FIG. 44***b* shows the initial control laws of the coefficient gains $\langle k_P(t),k_D(t)\rangle$ in a PD-FC created from soft computing technology for similar essentially non-linear control object such as a Van der Pol oscillator under non-Gaussian random noise with Rayleigh probability distribution.

[0476] **FIG. 44***c* shows the computational results of new coefficient gains of PD-FC based on the quantum control algorithm for similar essentially non-linear control objects such as the Van der Pol oscillator using KB's created from soft computing technology. **FIG. 44***d* shows the results of simulation of the dynamic behavior of the Van der Pol oscillator using PD-FC with different KBs.

[0477] The comparison of simulation results represented in **FIG. 44***d* shows the more robustness degree of quantum PD-FC than in similar classical soft computing cases as a new effect in intelligent control system design. From two

non-robust KBs of PD-FCs, one robust KB of PD-FC with quantum computation approach can be designed. This effect is similar to the effect in the above mentioned quantum Parrondo Paradox in quantum game theory, but without using of entanglement.

[0478] The comparison of simulation results represented in **FIG. 45** shows the higher degree of robustness in quantum PD-FC than in similar classical soft computing cases as a new effect in intelligent control system design.

6. Model Representations of Quantum Operators in Fast QAs

[0479] In some cases, the speed of the QA simulation can be improved by using a model representation of the quantum operators. This approach is based on using new operations or adding to existing quantum operators in the QSA structure, and/or structural modifications of the quantum operators in QSA. Grover's algorithm is used as an example herein. One of ordinary skill in the art will recognize that the model representation technique is not limited to Grover's algorithm.

6.1 Grover's QSA Structure with New Additional Quantum Operators

[0480] **FIG. 46** shows the addition of a new Hadamard operator, for example, between the oracle (entanglement) and the diffusion operators in Grover's QSA. The new Hadamard operator is applied on a workspace qubit (for complementing superposition and changing sign) to produce an algorithm labeled QSA1. Let M denote the number of matches within the search space such that $1 \leqq M \leqq N$, and for simplicity, and without loss of generality, assume that $N=2^n$. For this case one can describe the steps of the algorithm as follows.

Step	Computational operation
1	Register preparation: Prepare a quantum register of n + 1 qubits all in state $\|0\rangle$, where the extra qubit is used as a workspace for evaluating the oracle $U_f : \|W_0\rangle = \|0\rangle^{\otimes n}\|0\rangle$.
2	Register initialization: Apply Hadamard gate on each of the first n qubits in parallel, so they contain the 2^n states, where i is the integer representation of items in the list: $\|W_1\rangle = (H^{\otimes n} \otimes I)\|W_0\rangle = \left(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\|i\rangle\right)\otimes\|0\rangle$, $N = 2^n$.
3	Applying oracle: Apply the oracle U_f to map the items in the list to either 0 or 1 simultaneously and store the result in the extra workspace qubit: $\|W_2\rangle = U_f\|W_1\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle\otimes\|0 \oplus f(i)\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle \otimes \|f(i)\rangle)$.
4	Completing superposition and changing sign: Apply a Hadamard gate on the workspace qubit. This will extend the superposition for the n + 1 qubits with the amplitudes of the desired states with negative sign as follows:

-continued

Step	Computational operation
	$\|W_3\rangle = (I^{\otimes n} \otimes H)\|W_2\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle \otimes \left\|\frac{\|0\rangle + (-1)^{f(i)}\|1\rangle}{\sqrt{2}}\right\|)$, $P = 2N = 2^{n+1}$.
5	Inversion about the mean: $D = H^{\otimes n+1}(2\|0\rangle\langle 0\| - I)H^{\otimes n+1} = 2\|\psi\rangle\langle\psi\| - I, \|\psi\rangle = \frac{1}{\sqrt{P}}\sum_{k=0}^{P-1}\|k\rangle$, $\|W_4\rangle = D\|W_3\rangle = b\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|0\rangle) + a\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|1\rangle) + b\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|0\rangle) + a\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|1\rangle)$, $a = \frac{1}{\sqrt{P}}\left(3 - \frac{4M}{P}\right)$; $b = \frac{1}{\sqrt{P}}\left(1 - \frac{4M}{P}\right)$; $Ma^2 + (P-M)b^2 = 1$.
6	Measurement: Measure the first n qubits, to obtain the desired solution after first iteration with probability $P_s^{(1)}$ to find a match out of the M possible matches as follows: $P_s = M(a^2 + b^2) = 5r - 8r^2 + 4r^3$, $r = \frac{M}{N}$; with probability P_{ns} to find undesired result out of the states as follows: $P_{ns} = (P - 2M)b^2$, where $P_s + P_{ns} = M(a^2 + b^2) + (P - 2M)b^2 = 1$.

[0481] Consider the particular properties of QSA1. In Step 5 of QSA1 it is assumed that indicates a sum over all i, which are desired matches (2M states), and Σ_2 indicates a sum over all i, which are undesired items in the list. Thus, the state $|W_3\rangle$ of QSA1 can be rewritten as follows:

$$\begin{aligned}|W_3\rangle &= \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}(|i\rangle\otimes|0\rangle + (-1)^{f(i)}|1\rangle)\\ &= \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}(|i\rangle\otimes[|0\rangle - |1\rangle]) + \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}{}_2(|i\rangle\otimes[|0\rangle + |1\rangle])\\ &= \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}{}_1(|i\rangle\otimes|0\rangle) - \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}{}_1(|i\rangle\otimes|1\rangle) +\\ &\quad \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}{}_2(|i\rangle\otimes|0\rangle) + \frac{1}{\sqrt{P}}\sum_{i=0}^{N-1}{}_1(|i\rangle\otimes|1\rangle)\end{aligned}$$

There are M states with amplitude

$$\left(-\frac{1}{\sqrt{P}}\right)$$

where f (i)=1, and (P−M) states with amplitude

$$\left(\frac{1}{\sqrt{P}}\right).$$

[0482] Applying the Hadamard gate on the extra qubit splits the |i> state (solution states), to M states

$$\left(\sum_{i=0}^{N-1}(|i\rangle\otimes|0\rangle)\right)$$

with positive amplitude

$$\left(\frac{1}{\sqrt{P}}\right)$$

and Mstates

$$\left(\sum_{i=0}^{N-1}(|i\rangle\otimes|1\rangle)\right)$$

with negative amplitude

$$\left(-\frac{1}{\sqrt{P}}\right).$$

[0483] In step 5, the effect of applying the (Grover's) diffusion operator D on the general state

$$\sum_{k=0}^{P-1}\alpha_k|k\rangle \text{ produces } \sum_{k=0}^{P-1}[-\alpha_k+2\langle\alpha\rangle]|k\rangle,$$

where

$$\langle\alpha\rangle=\frac{1}{P}\sum_{k=0}^{P-1}\alpha_k$$

(operation of inversion about the mean) is the mean of the amplitudes of all states in the superposition; i.e., the amplitudes α_k will be transformed according to the following relation: $\alpha_k \rightarrow [-\alpha_k+2\langle\alpha\rangle]$. In the discussed case, there are M states with amplitude

$$\left(-\frac{1}{\sqrt{P}}\right)$$

and (P−M) states with amplitude

$$\left(\frac{1}{\sqrt{P}}\right),$$

so the mean <α> is as follows:

$$\langle\alpha\rangle=\frac{1}{P}\left(M\left(-\frac{1}{\sqrt{P}}\right)\right)+(P-M)\left(\frac{1}{\sqrt{P}}\right).$$

So, applying D on the system $|W_3\rangle$, described in step 5 of QSA1, can be understood as follows:

[0484] (i) The M negative sign amplitudes (solutions) will be transformed from

$$\left(-\frac{1}{\sqrt{P}}\right)$$

to α, where α is calculated as follows:

$$a=-\left(-\frac{1}{\sqrt{P}}\right)+\frac{2}{P}\left[M\left(-\frac{1}{\sqrt{P}}\right)+(P-M)\left(\frac{1}{\sqrt{P}}\right)\right]$$

$$=\boxed{\frac{1}{\sqrt{P}}\left(3-\frac{4M}{P}\right)}.$$

[0485] (ii) The (P−M) positive sign amplitudes will be transformed from

$$\left(\frac{1}{\sqrt{P}}\right)$$

to b, where b is calculated as follows:

$$b=\left(\frac{1}{\sqrt{P}}\right)+\frac{2}{P}\left[M\left(\frac{1}{\sqrt{P}}\right)+(P-M)\left(\frac{1}{\sqrt{P}}\right)\right]$$

$$=\boxed{\frac{1}{\sqrt{P}}\left(1-\frac{4M}{P}\right)}.$$

[0486] Then, a>b after applying D, and the new system state $|W_4\rangle$ can be written as step 5 of QSA1. If no matches exist within the superposition (i.e., M=0), then all the amplitudes will have a positive sign and applying the diffusion operator D will not change the amplitudes of the states as follows:

[0487] Substituting

$$\alpha_k = \frac{1}{\sqrt{P}} \text{ and}$$

$$\langle\alpha\rangle = \frac{1}{P}\left(P\left(\frac{1}{\sqrt{P}}\right)\right)$$

in the relation $\alpha_k \rightarrow [-\alpha_k + 2\langle\alpha\rangle]$ gives

$$\alpha_k + 2\langle\alpha\rangle \rightarrow \frac{1}{\sqrt{P}} + \frac{2}{P}\left(P\left(\frac{1}{\sqrt{P}}\right)\right) = \frac{1}{\sqrt{P}} = \alpha_k.$$

[0488] It is possible to produce a second quantum algorithm QSA2 by modifying the structure of the diffusion operator $D \rightarrow D_{part}$ in step 5 of the modified QSA1 on the partial diffusion operator D_{part} which can work similar to the well-known Grover's operator D except that it performs the inversion about the mean operation only on a subspace of the system. The diagonal representation of the partial diffusion operator D_{part}, when applied on n+1 qubits system, can take this form: $D \rightarrow D_{part} = H^{\otimes n+1} \hat{\times} I(2|0\rangle\langle 0| - I) H^{\otimes n+1} \hat{\times} I$, where the vector $|0\rangle$ used in this operation is a vector of length $P = 2N = 2^{n+1}$. **FIG. 47** shows the steps of QSA2.

[0489] The steps of the modified QSA2 can be understood as follows:

Step	Computational operation
1	Register preparation: Prepare a quantum register of n + 1 qubits all in state $\|0\rangle$, where the extra qubit is used as a workspace for evaluating the oracle U_f : $\|W_0\rangle = \|0\rangle^{\otimes n}\|0\rangle$.
2	Register initialization: Apply Hadamard gate on each of the first n qubits in parallel, so they contain the 2^n states, where i is the integer representation of items in the list: $\|W_1\rangle = (H^{\otimes n} \otimes I)\|W_0\rangle = \left(\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}\|i\rangle\right)\otimes\|0\rangle\rangle$, $N = 2^n$.
3	Applying oracle: Apply the oracle U_f to map the items in the list to either 0 or 1 simultaneously and store the result in the extra workspace qubit: $\|W_2\rangle = U_f\|W_1\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle\otimes\|0 \oplus f(i)\rangle) = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle\otimes\|0\rangle) + \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}(\|i\rangle\otimes\|1\rangle)$
4	Partial diffusion: Applying D_{part} on $\|W_2\rangle$ will result in a new system described as follows:

-continued

Step	Computational operation
	$\|W_3\rangle = D_{part}\|W_2\rangle = a_1\sum_{i=0}^{N-1}(\|i\rangle\otimes\|0\rangle) + b_1\sum_{i=0}^{N-1}(\|i\rangle\otimes\|0\rangle) + c_1\sum_{i=0}^{N-1}(\|i\rangle\otimes\|1\rangle)$, $a_1 = 2\langle\alpha_1\rangle - \frac{1}{\sqrt{N}}$; $b_1 = 2\langle\alpha_1\rangle$; $c_1 = -\frac{1}{\sqrt{N}}$; $\langle\alpha_1\rangle = \left(\frac{N-M}{N\sqrt{N}}\right)$, and $(N-M)a_1^2 + Mb_1^2 + Mc_1^2 = 1$
5	Measurement: Measure the first n qubits, to obtain the desired solution after the iteration 1. with probability $P_s^{(1)}$ to find a match out of the M possible matches as follows: $P_s^{(1)} = M(b_1^2 + c_1^2) = 5r - 8r^2 + 4r^3$, $r = \frac{M}{N}$; 2. with probability P_{ns} to find undesired result out of the states as follows: $P_{ns}^{(1)} = (N-M)a_1^2$, where $P_s^{(1)} + P_{ns}^{(s)} = 1$.

[0490] One aspect of using the partial diffusion operator in searching is to apply the inversion about the mean operation only on the subspace of the system that includes all the states which represent the non-matches and half the number of the states which represent the matches, while the other half will have the sign of their amplitudes inverted. This inversion to the negative sign prepars them to be involved in the partial diffusion operation in the next iteration so that the amplitudes of the matching states get amplified partially in each iteration. The benefit of this is to keep half the number of the states, which represent the matches as a stock each iteration to resist the de-amplification behavior of the diffusion operation when reaching the turning points as seen when examining the performance of the modified QSA2. In step 5 of modified QSA2 applying D_{part} can be understood as follows: without loss of generality, the general system

$$\sum_{k=0}^{P-1}\delta_k|k\rangle,\ |\delta_k|^2 = 1$$

can be rewritten as

$$\sum_{k=0}^{P-1}\delta_k|k\rangle = \sum_{j=0}^{N-1}\alpha_j(|j\rangle\otimes|0\rangle) + \sum_{j=0}^{N-1}\beta_j(|j\rangle\otimes|1\rangle),$$

where $\langle\alpha_j = \delta_k : k \text{ even}\rangle$ and $\langle\beta_j = \delta_k : k \text{ odd}\rangle$, and then applying D_{part} on the system gives

$$D_{part}\left(\sum_{k=0}^{P-1}\delta_k|k\rangle\right) = [H^{\otimes n+1}\otimes I(2|0\rangle\langle 0| - I)H^{\otimes n+1}\otimes I]\left(\sum_{k=0}^{P-1}\delta_k|k\rangle\right)$$

$$= 2[H^{\otimes n+1}\otimes I(2|0\rangle\langle 0|)H^{\otimes n+1}\otimes I]\left(\sum_{k=0}^{P-1}\delta_k|k\rangle\right) - \left(\sum_{k=0}^{P-1}\delta_k|k\rangle\right)$$

$$= \boxed{\sum_{j=0}^{N-1}[2\langle\alpha\rangle - \alpha_j](|j\rangle\otimes|0\rangle)} - \sum_{j=0}^{N-1}\beta_j(|j\rangle\oplus|1\rangle),$$

where

$$\langle\alpha\rangle = \frac{1}{N}\sum_{j=0}^{N-1}\alpha_i$$

is the mean of the amplitudes of the subspace

$$\sum_{j=0}^{N-1}\alpha_j(|j\rangle\otimes|0\rangle);$$

i.e., applying the operator D_{part} will perform the version about the mean only on the subspace

$$\sum_{j=0}^{N-1}\alpha_j(|j\rangle\otimes|0\rangle)$$

and will only change the sign of the amplitudes for the rest of the system as

$$\sum_{j=0}^{N-1}\beta_j(|j\rangle\otimes|1\rangle).$$

[0491] **FIG. 48** shows one embodiment of a circuit implementation using elementary gates. The probability of finding a solution varies according to the number of matches M≠0 in the superposition.

[0492] Consider the performance of the modified QSA1 and QSA2 after iterating the algorithm once. Table 6.1 shows the results of probability calculations. The maximum probability is always 1, and minimum probability (worst case) decreases as the size of the list increases, which is expected for small M≠0 because the number of states will increase, and the probability is distributed over more states, while the average probability increases as the size of the list increases.

TABLE 6.1

Algorithm performance with different size search space

n, N = 2^n	Max probability	Min probability	Average probability
2	1	0.8125	0.875
3	1	0.507812	0.93750
4	1	0.282227	0.96875
5	1	0.148560	0.984375
6	1	0.076187	0.992187

[0493] In the measurement process in step 6 of QSA1, for the first iteration,

$$P_{s1}^{(1)} = M(a_1^2 + b_1^2)$$

$$= \frac{M}{2N}\left(10 - 16\left(\frac{M}{N}\right) + 8\left(\frac{M}{N}\right)^2\right)$$

$$= 5r - 8r^2 + 4r^3,\ r$$

$$= \frac{M}{N}.$$

The above equation implies that the average performance of the algorithm to find a solution increases as the size of the list increases. Taking into account that the oracle U_f is taken as a black, box, one can define the average probability of success average(P_s) of the algorithm as follows:

$$\text{average}(P_s) = \frac{1}{2^N}\sum_{M=1}^{N}{}^{N}C_M P_s = \frac{1}{2^N}\sum_{M=1}^{N}\frac{N!}{M!(N-M)!}\cdot M(a^2+b^2)$$

$$= \frac{1}{2^{N+1}N^3}\sum_{M-1}^{N}\frac{N!}{(M-1)!(N-M)!}\cdot$$

$$(10N^2 - 16MN + 8M^2) = \boxed{1 - \frac{1}{2N}},$$

where

$${}^{N}C_M = \frac{N!}{M!(N-M)!}$$

is the number of possible cases for M matches. As the size of the list increases (N→∞), average (P_s) tends to 1.

[0494] For QSA2 in step 5, the following relations hold:

$$\text{average}(P_{s2}^{(1)}) = \frac{1}{2^N}\sum_{M=1}^{N}{}^{N}C_M P_s = \frac{1}{2^N}\sum_{M=1}^{N}\frac{N!}{M!(N-M)!}\cdot M(b_1^2 + c_1^2)$$

$$= \frac{1}{2^{N+1}N^3}\sum_{M=1}^{N}\frac{N!}{(M-1)!(N-M)!}\cdot$$

$$(10N^2 - 16MN + 8M^2) = \boxed{1 - \frac{1}{2N}}$$

where

-continued

$$ {}^{N}C_{M} = \frac{N!}{M!(N-M)!} $$

is the number of possible cases for M matches. As the size of the list increases (N→∞), average (P_s) for both QSA ½ tends to 1.

[0495] Classically, one can try to find a random guess of the item, which represents the solution (one trial guess), and succeed to find a solution with probability

$$ P_s^{(Classical)} = \frac{M}{N}. $$

The average probability can be calculated as follows:

$$ \begin{aligned} \text{average}(P_s^{(Classical)}) &= \frac{1}{2^N}\sum_{M=1}^{N} {}^{N}C_{M}\, P_s^{(Classical)} \\ &= \frac{1}{2^N}\sum_{M=1}^{N} \frac{M \cdot N}{M!(N-M)!} \\ &= \boxed{\frac{1}{2}}. \end{aligned} $$

This means that there is an average probability of one-half to find (or not to find) a solution by a single random guess, even with the increase in the number of matches.

[0496] Grover's QSA has an average probability one-half after an arbitrary number of iterations. The probability of success of Grover's QSA after l iterations is given by:

$$ P_s^{(Gr[l])} = \sin^2((2l+1)\theta), \text{ where } 0 < \theta < \frac{\pi}{2} \text{ and } \sin\theta = \sqrt{\frac{M}{N}}. $$

The average probability of success of Grover's QSA after an arbitrary number of iteration can be calculated as follows:

$$ \begin{aligned} \text{average}(P_s^{(Gr[l])}) &= \frac{1}{2^N}\sum_{M=1}^{N} {}^{N}C_{M}\sin^2((2l+1)\theta) \\ &= \boxed{\frac{1}{2}}. \end{aligned} $$

[0497] **FIG. 49** shows the probability of success of the three algorithms as a function of the ratio

$$ r = \frac{M}{N} $$

for the first iteration of Grover's QSA. **FIG. 49** shows that the probability of success of the modified QSA1 is always above that of the classical guess technique. Grover's QSA solves the case where

$$ M = \frac{N}{4} $$

with certainty, and the modified QSA1 solves the case where

$$ M = \frac{N}{2} $$

with certainty. The probability of success of Grover's QSA will start to go below one-half for

$$ M > \frac{N}{2}, $$

while the probability of success of the modified QSA1 will stay more reliable with a probability of at least 92.6%.

[0498] **FIG. 50** shows the iterating version of the algorithm QSA1 that works as follows:

Step	Computational algorithm
1	Initialize the whole n + 1 qubits system to the state |0>.
2	(i) Apply Hadamard gate on each of the first n qubits in parallel.
3	Iterate the following, for iteration k: Apply the oracle U_f taking the first n qubits as control qubits and the k th qubit workspace as the target qubit exclusively (ii) Apply Hadamard gate on the k th qubit workspace (iii) Apply diffusion operator on the whole n + k qubit system inclusively
4	Apply measurement on the first n qubits

[0499] The second iteration modifies the system as follows:

Step	Results after second QSA1-iteration
1	Append second qubit workspace to the system: $\|W_1^{(2)}\rangle = b_0^{(1)}\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|0\rangle)\Big)\otimes\|0\rangle + a_0^{(1)}\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|1\rangle)\Big)\otimes\|0\rangle +$ $b_0^{(1)}\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|0\rangle)\Big)\otimes\|0\rangle + b_0^{(1)}\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|1\rangle)\Big)\otimes\|0\rangle$
2	Apply U_f as shown in Step 3-(i) of QSA1: $\|W_2^{(2)}\rangle = b_0^{(1)}\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|0\rangle)\Big)\otimes\|1\rangle + a_0^{(1)}\sum_{i=0_1}^{N-1}(\|i\rangle\otimes\|1\rangle)\Big)\otimes\|1\rangle +$ $b_0^{(1)}\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|0\rangle)\Big)\otimes\|0\rangle + b_0^{(1)}\sum_{i=0_2}^{N-1}(\|i\rangle\otimes\|1\rangle)\Big)\otimes\|0\rangle$

-continued

Step	Results after second QSA1-iteration
3	Apply Hadamard gate on second qubit workspace ($I^{\otimes n+1} \otimes H$): $\|W_3^{(2)}\rangle =$ $\frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|1\rangle - \frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|1\rangle +$ $\frac{1}{\sqrt{2}} a_0^{(1)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|0\rangle - \frac{1}{\sqrt{2}} a_0^{(1)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|1\rangle +$ $\frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|0\rangle + \frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|1\rangle +$ $\frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|0\rangle + \frac{1}{\sqrt{2}} b_0^{(1)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|1\rangle$
4.	Apply diffusion operator as shown in Step 3-(iii) of QSA1: $\|W_4^{(2)}\rangle = b_0^{(2)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|0\rangle + b_1^{(2)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|1\rangle +$ $a_0^{(2)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|0\rangle + a_0^{(2)} \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|1\rangle +$ $b_0^{(2)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|0\rangle + b_0^{(2)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) \otimes \|1\rangle +$ $b_0^{(1)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|0\rangle + b_0^{(2)} \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|1\rangle) \otimes \|1\rangle$

[0500] Where the mean of the amplitudes to be used in the diffusion operator is calculated as follows:

$$\langle \alpha_2 \rangle = \frac{1}{2^{n+2}} \left[(2^{n+2} - 4M) \frac{b_0^{(1)}}{\sqrt{2}} \right] = \frac{b_0^{(1)}}{\sqrt{2}} (1 - 4M).$$

[0501] To clear ambiguity, a and b used in the above section for first iteration are denoted as $a_0^{(1)}$ and $b_0^{(1)}$ respectively, where the superscript index denotes the iteration and subscript index is used to distinguish amplitudes.

[0502] The new amplitudes $a_0^{(2)}$, $a_1^{(2)}$ $b_0^{(2)}$, $b_1^{(2)}$ are calculated as follows:

$$a_0^{(2)} = 2\langle \alpha_2 \rangle - \frac{1}{\sqrt{2}} a_0^{(1)}; \; a_i^{(2)} = 2\langle \alpha_2 \rangle + \frac{1}{\sqrt{2}} a_0^{(1)}$$

$$b_0^{(2)} = 2\langle \alpha_2 \rangle - \frac{1}{\sqrt{2}} b_0^{(1)}; \; b_1^{(2)} = 2\langle \alpha_2 \rangle + \frac{1}{\sqrt{2}} b_0^{(1)}.$$

[0503] The probability of success is: $P_s^{(2)} = M[(a_0^{(2)})^2 + (a_1^{(2)})^2 + (b_0^{(2)})^2 + (b_1^{(2)})^2]$.

[0504] In general, after e iterations, the recurrent relations representing the iteration can be written as follows: for the initial conditions

$$a_0^{(0)} = b_0^{(0)} = \frac{1}{\sqrt{N}},$$

[0505] 1. The mean to be used in the diffusion operator is:

$$\langle \alpha_2 \rangle = \frac{b_0^{(l-1)}}{\sqrt{2}} (1 - 4M); \; l \geq 1$$

[0506] 2. The new amplitudes of the system are:

$$a_0^{(1)} = 2\langle \alpha_2 \rangle + \frac{1}{\sqrt{2}} a_0^{(0)}; \; a_{0 \to 2^{l-1}-1}^{(2)} = 2\langle \alpha_l \rangle \mp \frac{1}{\sqrt{2}} a_{0 \to 2^{l-2}-1}^{(l-1)}; \; l \geq 2$$

$$b_0^{(1)} = 2\langle \alpha_2 \rangle - \frac{1}{\sqrt{2}} b_0^{(0)}; \; b_{0 \to 2^{l-1}-1}^{(2)} = 2\langle \alpha_l \rangle \mp \frac{1}{\sqrt{2}} b_{0 \to 2^{l-2}-1}^{(l-1)}; \; l \geq 2$$

[0507] 3. The probability of success for $l \geqq 2$ is:

$P_s^{(l)} = M[(a_i^{(l)})^2 + (b_i^{(l)})^2]; i = 0,1,2, \ldots, 2^{-1}-1$

or, using mathematical induction, the probability of success can take the following form:

$$P_s^{(l)} = \left(\frac{M}{N} - 1\right)\left(1 - \frac{M}{N}\right)^{2l} + 1, \; l \geq 1.$$

[0508] **FIG. 51** shows the iterating version of the QSA2 algorithm. The iterating block applies the oracle U_f and the operator D_{part} on the system in sequence. Consider the system after the first iteration, a second iteration modifies the system as follows:

Step	Results after second QSA2-iteration
1	Apply the oracle U1 will swap the amplitudes of the states which represent only the matches; i.e., states with amplitudes b_1 will be with amplitudes c_1, and states with amplitudes c_1 will be with amplitudes b_1, so the system can be described as: $\|W_4\rangle = a_1 \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) + c_1 \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) + b_1 \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle)$
2	Applying the operator D_{part} will change the system as follows: $\|W_5\rangle = a_2 \sum_{i=0_2}^{N-1} (\|i\rangle \otimes \|0\rangle) + b_2 \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|0\rangle) + c_2 \sum_{i=0_1}^{N-1} (\|i\rangle \otimes \|1\rangle)$, where the mean used in the definition of partial diffusion operator D_{part} is: a: $\langle \alpha_2 \rangle = \frac{1}{N}[(N-M)a_1 + Mc_1]$

-continued

Step	Results after second QSA2-iteration
	and a_2, b_2, c_2 used in this Step 2 of the second iteration are calculated as follows: $a_2 = 2\langle\alpha_2\rangle - a_1$; $b_2 = 2\langle\alpha_2\rangle - c_1$; $c_2 = -b_1$

[0509] And for the third iteration

Step	Results after third QSA2-iteration
1	Apply the oracle U_f will swap the amplitudes of the states which represent only the matches as: $U_f\|W_5\rangle = \|W_6\rangle = a_2\sum_{i=0}^{N-1}{}_2(\|i\rangle\otimes\|0\rangle) + c_2\sum_{i=0}^{N-1}{}_1(\|i\rangle\otimes\|0\rangle) + b_2\sum_{i=0}^{N-1}{}_1(\|i\rangle\otimes\|1\rangle)$
2	Applying the operator D_{part} will change the system as follows: $D_{part}\|W_6\rangle = \|W_7\rangle = a_3\sum_{i=0}^{N-1}{}_2(\|i\rangle\otimes\|0\rangle) + b_3\sum_{i=0}^{N-1}{}_1(\|i\rangle\otimes\|0\rangle) + c_3\sum_{i=0}^{N-1}{}_1(\|i\rangle\otimes\|1\rangle)$, where the mean used in the definition of partial diffusion operator D_{part} is as: $\langle\alpha_3\rangle = \frac{1}{N}[(N-M)a_2 + Mc_2]$, and a_3, b_3, c_3 in this Step 2 of the third iteration are calculated as follows: $a_3 = 2\langle\alpha_3\rangle - a_2$; $b_3 = 2\langle\alpha_3\rangle - c_2$; $c_3 = -b_2$

[0510] In general, the system of QSA2 after l≧2 iterations can be described using the following recurrence relations:

$$|W^{(l)}\rangle = a_l\sum_{i=0}^{N-1}{}_2(|i\rangle\otimes|0\rangle) + b_l\sum_{i=0}^{N-1}{}_1(|i\rangle\otimes|0\rangle) + c_l\sum_{i=0}^{N-1}{}_1(|i\rangle\otimes|1\rangle),$$

where the mean to be used in the definition of the partial diffusion operator D_{part} is as follows:

$$\langle\alpha_l\rangle = [ya_{l-1} + (1-y)c_{l-1}],\ y = 1-r,\ r = \frac{M}{N},$$

and $a_l = s(F_l - F_{l-1})$, $b_l = sF_l$,

$$c_l = -sF_{l-1} \text{ and } F_l(y) = \frac{\sin([l+1]\theta)}{\sin(\theta)},\ s = \frac{1}{\sqrt{N}},$$

where $F_l(y)$ is the Chebyshev polynomials of the second kind.

[0511] The probabilities of the system are:

$$P_s^{(l)} = (1-\cos(\theta))[F_l^2 + F_{l-1}^2],$$

$$P_{ns}^{(l)} = \cos(\theta)[F_l - F_{l-1}]^2,\ y = \cos(\theta),\ 0 \le \theta \le \frac{\pi}{2},$$

such that $P_s^{(1)}+P_{ns}^{(1)}=1$.

[0512] It is instructive to calculate how many iterations, l, are required to find the matches with certainty or near certainty for different cases of 1≦M≦N. To find a match with certainty on any measurement, then $P_s^{(l)}$ must be as close as possible to certainty.

[0513] For interations of the algorithm QSA1, consider the following cases using equation

$$P_s^{(l)} = \left(\frac{M}{N} - 1\right)\left(1 - \frac{M}{N}\right)^{2l} + 1,\ l \ge 1.$$

The number of iterations W in terms of the ratio

$$r = \frac{M}{N}$$

is represented using Taylor's expansion as follows:

$$\boxed{l \ge \frac{P_s^{(l)} - r}{4r(1-r)}},$$

$$r = \frac{M}{N}.$$

[0514] The cases where multiple instances of a match exist within the search space are listed as follows:

1	The case where $M = \frac{1}{2}N$: The algorithm can find a solution with certainty after arbitrary number of iterations (one iteration is enough)
2	The case where $M > \frac{1}{2}N$: The probability of success is, for instance, at least 92.6% after the first iteration, 95.9% after second iteration, and 97.2% after third iteration
3	For iterating the algorithm once ($\ell = 1$) and to get a probability of at least one-half, so, M must satisfy the condition $M > \frac{1}{8}N$

[0515] For the case where l≧1, the following conditions must be satisfied: n≧4 and

$$1 \le M \le \frac{1}{8}N.$$

This means that the first iteration will cover approximately 87.5% of the problem with a probability of at least one-half; two iterations will cover approximately 91.84% and three iterations will cover 94.2%. The rate of increase of the coverage range will decrease as the number of iterations increases.

[0516] For the algorithm QSA2 to find a match with certainty on any measurement, then $P_s^{(l)}$ must be as close as possible to certainty. In this case, consider the following relation: $P_s^{(l)}=1=(1-\cos(\theta))[F_1^2+F_{1-1}^2]$,

$$y = \cos(\theta),\ 0 \le \theta \le \frac{\pi}{2}.$$

$$\text{Then, } l = \frac{\pi - \theta}{2\theta} \text{ or } \theta = \frac{\pi}{2}.$$

Using this result, and since the number of iterations must be an integer, then the required number of iterations is

$$l = \left\lfloor \frac{\pi}{2\sqrt{2}}\sqrt{\frac{N}{M}} \right\rfloor,$$

where $\lfloor\ \rfloor$ is the floor operation. The algorithm runs in

$$O\left(\sqrt{\frac{N}{M}}\right).$$

[0517] The probability of success of Grover's QSA is as follows: $P_S^{(1\text{-}Gr)}=\sin^2[(2l_{Gr}+1)\theta]$, where

$$\sin^2(\theta) = \frac{M}{N};\ 0 \le \theta \le \frac{\pi}{2}$$

and the required l_{Gr} is

$$l_{Gr} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor.$$

[0518] **FIG. 52** shows the probability of success of the iterative version of the algorithm QSA1 where l=1, 2, . . . , 6. This algorithm needs

$$O\left(\frac{N}{M}\right)$$

iterations for n≧4 and

$$1 \le M \le \frac{1}{8}N,$$

which is similar to classical algorithms behavior. This leads to the conclusion that the first few iterations of the algorithm will provide the best performance and that there will be no substantial gain from continuing to iterate the algorithm.

[0519] By contrast, Grover's QSA needs

$$O\left(\sqrt{\frac{N}{M}}\right)$$

to solve the problem, but its performance decreases for

$$M \ge \frac{1}{2}N.$$

Thus, for the case when the number of solutions M is known in advance, for

$$1 \le M \le \frac{1}{8}N,$$

one can use Grover's QSA with

$$O\left(\sqrt{\frac{N}{M}}\right);$$

and if

$$\frac{1}{8}N \le M \le N$$

use QSA1 with O(1).

[0520] **FIG. 53** shows that Grover's QSA is faster in the case of fewer instances of the solution

$$\left(\text{ratio } r = \frac{M}{N} \text{ is small}\right)$$

and the algorithm QSA1 is more stable and reliable in case of multiple instances of the solution.

[0521] Thus, Grover's QSA performs well in the case of fewer instances of the solution, and the performance decreases as the number of solutions increase within the search space; the algorithm QSA1 in general performs better than any pure classical or QSA and still has O$\sqrt{N}$ for the hardest case and approximately O(1) for

$$M \geq \frac{1}{8}N.$$

[0522] For QSA2, the probability of success is as follows:

$$P_s^{(l)} = (1-\cos(\theta))[F_l^2 + F_{l-1}^2],\ F_l(y) = \frac{\sin([l+1]\theta)}{\sin(\theta)},$$

and

$$P_s^{(l)} = (1-\cos(\theta))[F_l^2 + F_{l-1}^2] = (1-\cos(\theta))\left[\frac{\sin^2([l+1]\theta) + \sin^2(l\theta)}{\sin^2(\theta)}\right],$$

where

$$\cos(\theta) = 1 - \frac{M}{N};\ 0 \leq \theta \leq \frac{\pi}{2}$$

and the required l is

$$l = \left\lfloor \frac{\pi}{2\sqrt{2}}\sqrt{\frac{N}{M}} \right\rfloor.$$

[0523] **FIG. 54** shows the probability of success as a function of the ratio

$$r = \frac{M}{N}$$

for both algorithms. For QSA2 the probability will never return to zero once started, and the minimum probability will increase as M increases because of the use of the partial diffusion operator D_{part}, which will resist the de-amplification when reaching the turning points as explained in the definition of the partial diffusion operator D_{part}; i.e., the problem becomes easier for multiple matches, whereas for Grover's QSA, the number of cases (points) to be solved with certainty is equal to the number of cases with zero-probability after arbitrary number of iterations.

[0524] **FIG. 55** shows the probability of success as a function of the ratio

$$r = \frac{M}{N}$$

for both algorithms by inserting the calculated number of iterations l_{Gr} and l in $P_S^{(l-Gr)}$ and $P_S^{(l)}$, respectively. The minimum probability that Grover's QSA can reach is approximately 17.5% when

$$r = \frac{M}{N} = 0.617,$$

while for QSA2, the minimum probability is 87.7% when

$$r = \frac{M}{N} = 0.31.$$

The behavior of QSA2 is similar in this case to the behavior of this algorithm of the first iteration shown in **FIG. 55** for

$$r = \frac{M}{N} > 0.31,$$

which implies that if

$$r = \frac{M}{N} > 0.31,$$

then QSA2 runs in O(1), i.e.: the problem is easier for multiple matches.

[0525] Thus, using modifications in the quantum operators of Grover's QSA structure, both QSA1 and QSA2, based on QAG-approach, perform more reliably than Grover's QSA in the case of fewer matches (e.g., relatively hard cases) and runs in O(1) in the case of multiple matches (e.g., relatively easy cases).

[0526] 6.2. Modification of the Superposition Operator in Grover's QSA: Wavelet QSA with Partial Information.

[0527] Before applying of Grover's QSA, a bisection between a database and quantum states is necessary. If a superposition of N states is initially prepared, the Grover's QSA amplifies the amplitude of the target state up to around one, while those of other states dwindle down to nearly zero. The amplitude amplification is perfomed by two inversion operations: inversion about the target by the oracle and inversion about the initial state by the Fourier transform. Two simultaneous reflections about two mirrors crossing by an angle α induces a 2α rotation. One can imagine that the inversion in the Grover's QSA rotates the initial state around the target state. If the target state and initial state are denoted by $|w\rangle$ and $|\psi\rangle$, respectively (here the initial state is prepared by the Fourier transform of a state $|k\rangle$, i.e.; $|\psi\rangle = (FT)|k\rangle$), the inversion operators are expressed as $O_{|w\rangle} = I - 2|w\rangle\langle w|$, $J_{|w\rangle} = I - 2|\psi\rangle\langle\psi|$. Since $J_{|w\rangle} = (FT)J_{|k\rangle}(FT)^{\dagger}$, the

Grover operator is written as $G=(FT)J_{|k>}(FT)^{\dagger}O_{|w>}$.Then, after applying the operator $O(\sqrt{N})$ times, the final state comes to $|\psi_{fin}>=G^{O(\sqrt{N})}(FT)|k>$. The probability to obtain the target state is $Pr(w)=|<w||\psi_{fin}>>|^2$, which is $1-\epsilon^2$, $\epsilon \square 1$. The query complexity of this QSA, the number of callings of the oracle, is therefore $O(\sqrt{N})$. The running time has nothing to do with the choice of $|k>$.

[0528] When partial information is given in an unstructed database, one can replace the Fourier transform in Grover's QSA with the Haar wavelet transform. In this case, if a partial information L is given to an unstructed database of size N, then there is an improved speed-up of

$$O\left(\sqrt{\frac{N}{L}}\right).$$

[0529] Grover's QSA cannot benefit from the partial information. The fast wavelet WQSA, which is a modification of Grover's QSA can solve this problem by replacing the Fourier transform with the Haar wavelet transform.

[0530] The state $W^{\dagger}|2^{\lambda-1}+j>$ is a superposition of

$$\frac{N}{L}$$

states, where $L=2^{\lambda-1}$ (λ is given by k) is the partial information about an initial state, while the state $(FT)|k>$ is a superposition of N states. Since the operator is composed of wavelet transforms, the initial state is prepared by applying the inverse wavelet transform $W^{\dagger}$ to a state $|k>$. The initial state is now $|\psi>=W^{\dagger}|k>$. The power of the WQSA appears in the initialization procedure.

[0531] The Haar wavelet transform W is represented by the sequence of sparse matrices $W=W_nW_{n-1}\ldots W_1$, where

$$W_k=\begin{bmatrix} H_{2^{n-k+1}} & O_{2^{n-k+1}\times(2^n-2^{n-k+1})} \\ O_{(2^n-2^{n-k+1})\times 2^{n-k+1}} & I_{2^n-2^{n-k+1}} \end{bmatrix},$$

and

$$H_{2^n}=\left[\begin{array}{cc|cc|ccc} 1 & 1 & 0 & 0 & \cdots & 0 & \\ 0 & 0 & 1 & 1 & 0 & \vdots & \\ \hline \vdots & \vdots & & \ddots & 0 & 0 & \\ \hline 0 & 0 & \cdots & 0 & 1 & 1 & \\ 1 & -1 & 0 & 0 & \cdots & 0 & \\ \hline 0 & 0 & 1 & -1 & 0 & & \\ \hline \vdots & \vdots & & \ddots & 0 & 0 & \\ \hline 0 & 0 & \cdots & 0 & 1 & -1 & \end{array}\right]_{2^k\times 2^k},$$

where H_{2^n} is the Haar 1-level decomposition operator, I_n is used as the n×n unit matrix, and $O_{n,m}$ as n×m zero matrix. The wavelet transform W is unitary, since the operator H_{2^n} is unitary.

[0532] One of ordinary skill in the art will recognize that other wavelet transforms can be applied to the WQSA. The Haar wavelet transform is described by sparse matrix, and it is observed that the first half of the Haar wavelet basis differs from the second half of the wavelet basis by the phase $\exp\langle i\pi\rangle$. This implies that the destructive and constructive interference between states accepts a set of states containing the target and rejects the other states.

[0533] In this sense, other known wavelet bases, e.g., Daubechies's, the discrete Hartle transform as

$$A_N=\left(\frac{1-i}{2}\right)(FT)_N+\left(\frac{1+i}{2}\right)(FT)_N^3$$

or the fractional discrete Fourier transform as an α-th root of $(FT)_N$ is

$$F_{N,\alpha}=a_0(\alpha)\cdot 1_N+\ldots+a_3(\alpha)\cdot(FT)_N^3,$$

$$a_0(\alpha)=\frac{1}{2}(1+e^{i\alpha})\cos\alpha,\ a_1(\alpha)=\frac{1}{2}(1-ie^{i\alpha})\sin\alpha,$$

$$a_2(\alpha)=\frac{1}{2}(-1+e^{i\alpha})\cos\alpha,\ a_3(\alpha)=\frac{1}{2}(-1-ie^{i\alpha})\sin\alpha$$

are not appropriate to play the role of selecting a subset of the N states.

[0534] The operator $G^{(W)}=-W^{\dagger}J_{|k>}WO_{|W>}$ is one iteration of the WQSA. The expected runing time is

$$O\left(\sqrt{\frac{N}{L}}\right).$$

[0535] For example, consider the problem of finding a desired one in the set $A=\langle|a>|a=1,2,3,\ldots,2^{n-1}\rangle$. Given a partial information that the target state is in the subset $\Lambda_{\lambda}^{j}=\langle|z>|(j-1)2^{n-\lambda}\leqq z\leqq j2^{n-\lambda}-1,1<j\leqq 2^{\lambda}\rangle$, one can complete the search task in $O(\sqrt{2^{n-\lambda+1}})$ times by choosing the initial state as $W^{\dagger}|2^{\lambda-1}+j>$. Only the λ-number is correctly labeled. The partial information may save this problem. Thus, the power of WQSA appears in the initialization procedure.

[0536] Consider the case of partial information about k as $k\neq 0,1$. Choosing the initial state as $|\psi>=W^{\dagger}|k>$, $k\neq 0,1$ when the target state exists in the restricted domain of the

$$\frac{N}{L}$$

states gives an improved speed-up with the partial information.

[0537] Since $k\epsilon\langle 2,3,4,\ldots,N(=2^n)-1\rangle$, by setting k=2^&lgr;-1+j,1≦j≦2^&lgr; and $\lambda\geqq 1$, and

$$N_1 = \frac{N}{L} = 2^{n-\lambda+1},$$

the initial state $|\psi\rangle = W^{\dagger}|k\rangle$, $k \neq 0,1$ is explicitly.

$$W^{\dagger}|k\rangle = \sum_{\alpha=(j-1)N_1}^{(j-1)N_1+\frac{N_1}{N_2}-1} |\alpha\rangle - \sum_{\beta=(j-1)N_1+\frac{N_1}{N_2}}^{jN_1-1} |\beta\rangle.$$

[0538] Let the target state be $|w\rangle \epsilon A_{\lambda}^{j}$ and the initial state be $W^{\dagger}|2^{\lambda-1}+j\rangle$.It suffices to show that it takes $O(\sqrt{2^{n-\lambda+1}})$ **times for the WQSA to find the target state with the following setting.**

[0539] Let

$$N_1 = \frac{N}{L} = 2^{n-\lambda+1}$$

[0540] and the wavelet search operator is $G^{(W)} = -W^{\dagger}J_{|k\rangle}WO_{|w\rangle}$, where $W^{\dagger}$ is the Haar wavelet transform.

Step	Computational wavelet algorithm
1	Applying the operator $W^{\dagger}$ to the$|k\rangle$, gives the initial state $|\psi\rangle = W^{\dagger}|k\rangle = \sum_{\alpha=(j-1)N_1}^{(j-1)N_1+\frac{N_1}{2}-1} |\alpha\rangle - \sum_{\beta=(j-1)N_1+\frac{N_1}{2}}^{jN_1-1} \left|\beta\right\rangle$, which can be written as follows: $|\psi\rangle =$ $\frac{\varepsilon_w}{\sqrt{N_1}}|w\rangle + \varepsilon_r\sqrt{\frac{N_1-1}{N_1}}|r\rangle$, where $\varepsilon_i \epsilon\{\pm 1\}$ and the state $|r\rangle =$ $\sqrt{\frac{1}{N_1-1}}\sum_{\gamma\neq w}\varepsilon_\gamma\left|\gamma\right\rangle$ is orthogonal complement of the target state.
2	The m iterations of the operator $G^{(W)} = -W^{\dagger}J_{|k\rangle}WO_{|w\rangle}$ create the following state: $|\psi_m\rangle = G_m^{(W)}|\psi\rangle$
3	The probability to obtain the target state after the m iterations is $P_m = |\langle w|\psi_m\rangle|^2 = \cos^2(m\theta-\varphi)$, where $\theta = \sin^{-1}\left(2\varepsilon_w\varepsilon_r\frac{\sqrt{N_1-1}}{N_1}\right)$ and $\varphi = \cos^{-1}\left(\frac{\varepsilon_w}{N_1}\right)$.

[0541] Thus, the total number of iterations is $O(\sqrt{2^{n-\lambda+1}})$. **If we denote** $N=2^n$ and $L=2^{\lambda-1}$, then the running time is written as

$$O\left(\sqrt{\frac{N}{L}}\right).$$

[0542] The partial information that the λ-th number j is correctly labeled leads to the application of the WQSA so that the reference section is filled in time. However, note that there is no improvement in running time when the initial state is $W^{\dagger}|0\rangle$ or $W^{\dagger}|1\rangle$, since, in this case, the initial state is still a superposition of N states. Therefore, from the proposition, one can complete the submission in time if the λ is larger than 2.

[0543] The described construction provides a way for a quantum search to benefit from partial information. Since the running time of the Grover's QSA has nothing to do with the choice of the unitary operator, the complexity of the WQSA is the same as the Grover QSA. However, the speed-up obtained from the WQSA is

$$O\left(\sqrt{\frac{N}{L}}\right)$$

and is obtained by preparing the initial state as follows: $|\psi\rangle = W^{\dagger}|k\rangle$. The running time of the WQSA depends on the choice of k, while that the Grover's QSA does not. This is because the state $|\psi\rangle = W^{\dagger}|k\rangle$ is a superposition of states in the restricted domain of

$$\frac{N}{L}$$

states. Therefore, given a partial information L to a unstructured database of size N, there is an improved speed-up of

$$O\left(\sqrt{\frac{N}{L}}\right).$$

7. Comparison of Different QA Simulation Approaches

[0544] **FIG. 56** shows comparison of the developed approaches of QA simulation. In case of Grover's QSA **FIG. 56*a***, shows results from four simulation methods. It is clear that simulation results according with each method are same, but temporal complexity and size of the data base may vary depending on the approach. Direct matrix based approach is more simple, but the qubit number is limited to 12 qubits, since operator matrices are allocated in PC memory. The second approach with algorithmic replacement of the quantum gates permits an increase in the degree of the analyzed function (number of qubits) up to 20 or more. The problem-oriented approach permits quantum gate applications operating directly with the state vector. This permits an exponential decrease in the number of multiplications, and as a result, allows running of Grover's algorithm on a PC.

With this approach, it is possible to allocate in PC memory a state vector containing 25-26 qubits. An extreme version of the Grover's QSA is an approach when the state vector is allocated as a sparse matrix, taking in consideration that with an absence of decoherence, most of the values of the probability amplitudes are equal, and as a result there is no need to store of all of the state vector, but only the different parts, which is equal to number of the searched elements +1. Thus, excluding memory limitations, one can simulate up to 1024 qubits or more, with only limitation caused by floating point number representations (with larger number of qubits, probability amplitudes after superposition approach to machine zero).

[0545] In the case of Deutsch-Jozsa's algorithm simulation, **FIG. 56***b* shows three simulation approaches. In this case, the direct matrix based approach has the same limitations as in Grover's algorithm, and a PC permits an order up to 11 qubits. With the algorithmic approach, up to 20 qubits or more qubits is possible. The problem-oriented approach with compression gives the same result as in case of Grover's algorithm.

[0546] In case of Simon and Shor's quantum algorithms, **FIG. 56***c* shows different algorithm structure. The matrix based approach and algorithmic approach are shown. The matrix based approach permits simulation up to 10 qubits, and the algorithmic approach permits simulation up to 20 qubits, or more.

[0547] **FIG. 57** shows analysis of the quantum algorithms dynamics from the Shannon information entropy viewpoint. **FIG. 57***a* shows the relation between Shannon information entropy of the state vector of the Grover's QSA for different parameters of the data base. This analysis permits estimation of the number of algorithm iterations required for database search regarding database size. This estimation is shown in **FIG. 58**.

[0548] The results of Shannon entropy behavior are presented in the **FIGS. 57***b* for Deutsch-Jozsa's algorithm, in **FIG. 57***c* for Simon QA and in **FIG. 57***d* for Shor's QA.

[0549] **FIG. 59** shows the screen shot of the Grover's QSA problem oriented simulator with sparse allocation of the state vector. The result of the simulation for 1000 qubits is presented.

[0550] **FIG. 60** summarizes the above approaches to QA simulation. The high level structure of the quantum algorithms can be represented as a combination of different superposition entanglement and interference operators. Then depending on algorithm, one can choose corresponding model and algorithm structure for simulation. Depending on the current problem, one can choose (if available) one of the simulation approaches, and depending on approach one can simulate different orders of quantum systems.

[0551] Although various embodiments have been described, other embodiments will be apparent to those of ordinary skill in the art. Thus, the present invention is limited only by the claims.

What is claimed is:

1. A method for simulating a quantum algorithm on a classical computer, comprising:

applying a unitary matrix quantum gate G to an initial vector to produce a basis vector;

measuring said basis vector, wherein elements of said quantum gate G are computed on an as-needed basis;

repeating said steps of applying and measuring k times, where k is selected to minimize Shannon entropy of said basis vector; and

decoding said basis vectors, said decoding including translating said basis vectors into an output vector.

2. The method of claim 1, wherein said quantum gate G describes an entanglement-free quantum algorithm.

3. The method of claim 1, wherein said elements of said basis vector comprise one of two pre-computed values.

4. An intelligent control system comprising a quantum search algorithm configured to minimize Shannon entropy comprising: a genetic optimizer configured to construct one or more local solutions using a fitness function configured to minimize a rate of entropy production of a controlled plant; and a quantum search algorithm configured to search said local solutions to find a global solution using a gate G expressing a fitness function configured to minimize Shannon entropy, said gate G corresponding to an entanglement-free quantum algorithm for efficient simulation, and wherein elements of said gate G are computed on an as-needed basis.

5. The intelligent control system of claim 4, wherein said global solution comprises weights for a fuzzy neural network.

6. The intelligent control system of claim 4, wherein said fuzzy neural network is configured to train a fuzzy controller, said fuzzy controller configured to provide control weights to a proportional-integral-differential controller, said proportional-integral-differential controller configured to control said controlled plant.

7. The intelligent control system of claim 4, wherein said fitness function is step-constrained.

8. The intelligent control system of claim 4, wherein each element of a state vector of said quantum search algorithm comprises one of a finite number of pre-computed values.

9. The intelligent control system of claim 4, wherein said quantum search algorithm operates on pseudo-pure states.

10. A method for global optimization to improve a quality of a sub-optimal solution comprising the steps of: selecting a first gate G corresponding to a first quantum process, modifying said first gate G into a second gate G corresponding to a second quantum process; having pseudo-pure states; applying a first transformation to an initial state to produce a coherent superposition of basis states; applying a second transformation to said coherent superposition using a reversible transformation according to said second gate G to produce coherent output states; applying a third transformation to said coherent output states to produce an interference of output states; and selecting a global solution from said interference of output states.

11. The method of claim 10, wherein said first transformation is a Hadamard rotation.

12. The method of claim 10, wherein each of said basis states is represented using qubits.

13. The method of claim 10, wherein said second transformation is a solution to Shrodinger's equation.

14. The method of claim 10, wherein said third transformation is a quantum fast Fourier transform.

15. The method of claim 10, wherein said pseudo-pure states are entanglement-free.

16. The method of claim 10, wherein said superposition of input states comprises a collection of local solutions to a global fitness function.

17. A method for terminating iterations of a quantum algorithm, comprising:

performing an interation of a quantum algorithm to produce a measurement vector;

computing a Shannon entropy of said measurement vector;

selecting a termination condition from at least one of: a first local Shannon entropy minimum, a lowest Shannon entropy within a predefined number of iterations; a predefined level of acceptable Shannon entropy; and

repeating said performing and computing until said termination condition is satisfied.

18. The method of claim 17, further comprising measuring a final output result.

19. The method of claim 17, further comprising measuring an output result at each iteration.

20. A method for intelligent control comprising a quantum search algorithm corresponding to a quantum system on entanglement-free states configured to minimize Shannon entropy comprising: optimizing one or more local solutions using a fitness function configured to minimize a rate of entropy production of a controlled plant; and searching, using a quantum search algorithm to search said local solutions to find a global solution using a fitness function to minimize Shannon entropy.

21. The method of claim 20, wherein said global solution comprises weights for a fuzzy neural network.

22. The method of claim 21 further comprising: training a fuzzy controller, providing control weights from said fuzzy controller to a proportional-integral-differential controller, and using said proportional-integral-differential controller to control said controlled plant.

23. The method of claim 20, wherein said quantum search algorithm iterates until a first local Shannon entropy minimum is found.

24. The method of claim 20, wherein said quantum search algorithm iterates until a lowest Shannon entropy is found within a predefined number of iterations.

25. A global optimizer to improve a quality of a sub-optimal solution, said optimizer comprising of a computer software loaded into a memory, said software comprising: a first module for applying a first transformation to an initial state to produce a coherent superposition of basis states; a second module for applying a second transformation to said coherent superposition using a reversible transformation to produce one or more entanglement-free output states; a third module for applying a third transformation to said one or more coherent output states to produce an interference of output states; and a fourth module for selecting a global solution from said interference of output states.

26. The optimizer of claim 25, wherein said first transformation is a Hadamard rotation.

27. The optimizer of claim 25, wherein each of said basis states is represented using qubits.

28. The optimizer of claim 25, wherein said second transformation is based on a solution to Shrodinger's equation.

29. The optimizer of claim 25, wherein said third transformation is a quantum fast Fourier transform.

30. The optimizer of claim 25, wherein said fourth module is configured to find a maximum probability.

31. The optimizer of claim 25, wherein said superposition of input states comprises a collection of local solutions to a global fitness function.

32. The optimizer of claim 25, wherein elements of a quantum gate are computed on an as-needed basis.

33. The optimizer of claim 25, wherein a state vector describing said output states is stored in a compressed format.

* * * * *