Quantum Algorithm to Solve a Maze :
Converting the Maze Problem into a Search Problem

Debabrata Goswami*
dgoswami@iitk.ac.in
Indian Institute of Technology, Kanpur

Niraj Kumar†
kniraj@iitk.ac.in
Indian Institute of Technology, Kanpur

We propose a different methodology towards approaching a Maze problem. We convert the problem into a Quantum Search Problem (QSP), and its solutions are sought for using the iterative Grover's Search Algorithm. Though the category of mazes we are looking at are of the NP complete class[1], we have redirected such a NP complete problem into a QSP. Our solution deals with two dimensional perfect mazes with no closed loops. We encode all possible individual paths from the starting point of the maze into a quantum register. A quantum fitness operator applied on the register encodes each individual with its fitness value. We propose an oracle design which marks all the individuals above a certain fitness value and use the Grover search algorithm to find one of the marked states. Iterating over this method, we approach towards the optimum solution.

I. INTRODUCTION

In 1964, Gordon Moore, the co-founder of Intel, stated that the computing powers of computers would double about every 18 months and this is popularly known as the Moore's law [2]. If this were to continue to happen, then the number of transistors on the silicon chips would increase accordingly and the number of electrons in the transistor would drop as time goes on. In fact, Moores law has been more or less consistent, and presently the number of transistors in silicon chip for Core I7 Extreme Edition reaches 1.3 billion. This implies that only a handful of electrons make up a transistor, thus necessitating the need to work in Quantum regime. Such quantum computers have the potential to be the most powerful computational devices ever created, however, only a few practical applications of these have been devised effectively. One of the problems have been the design of quantum algorithm.

Quantum algorithms are the set of algorithms performed on the quantum bits employing the powerful concepts of quantum superposition and entanglement of the qubits. The interest in this domain was generated ever since physicists claimed that quantum computers use fewer resources (time and space) to perform certain computations as compared to classical computers (Deutsch & Jozsa algorithm) [3]. Shor's algorithm [4], for factoring numbers uses $O(n^2 log(n) log(log(n)))$ operations as compared to classical factoring algorithm which uses $\Theta(n^{1/3} log^{1/3}(n))$ operations and thus provides an exponential speed-up over the best known classical factoring algorithm. Also the Grover's search algorithm [5] of an unsorted data provides a quadratic speed-up, $O(\sqrt{n})$ over the most efficient classical search algorithm $O(n)$. These have led to believe that QAs may have the potential of solving certain problems that seem intractable using Classical algorithms.

Solution to a general n × m Maze [1] is a NP complete problem and so it is an attractive problem to attack through quantum computing principles, which we present here. In context of the Maze problem, we define:

1. *Individual*: One of the members of the population and a candidate for the solution to the Maze problem

2. *Individual Register*: The entire population containing a superposition of all individuals

3. *Fitness*: Measure of how close a particular individual is to the optimal solution. Closer is the individual; higher is its fitness value.

II. OVERVIEW OF STRATEGY USED

1. First step involves the creation of a perfect square maze using the Recursive Backtracker Algorithm[6]

2. Creation of individual register using quantum superposition storing the entire population.

3. Classifying the individuals based upon their closeness to the optimum solution i.e. defining a fitness register storing the fitness values of individuals.

4. Defining an Oracle which marks certain individuals having fitness values greater than a threshold fitness value.

5. Employing Grover's search algorithm in finding out one of the oracle's marked states.

———————
* http://home.iitk.ac.in/∼dgoswami/
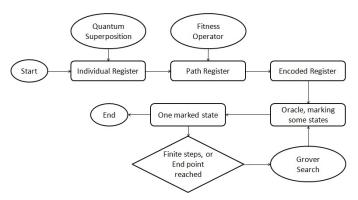† http://kumarniraj.weebly.com

FIG. 1. An outline of the method used in the paper to solve a Maze Problem.

6. Iterating over the above two steps to find either the optimum individual or an individual nearing the optimum value.

Now let us address the individual issues in the next few sections:

III. MAZE CREATION

The maze is a $m \times m$ two dimensional pathway where m is the number of rows and also the number of columns. The maze creation algorithm used in the paper is recursive backtracker algorithm [6].
The concept of a pushdown stack is employed to implement this algorithm. Each grid in the maze is called a Room.

1. An empty stack is defined initially.

2. A random room is chosen and gets pushed onto the stack. This room becomes the starting room.

3. Another room adjacent to the first room is chosen randomly, and the door is opened between the two rooms. This room is the current room.

4. The starting room is pushed onto the stack.

5. While there's at least one room left in the stack, the following steps are repeated:

   (a) If there are any unconnected rooms next to the current room:

      i. A random room is selected and the door between the two is open.
      ii. The current room gets pushed onto the stack and the door between the two is opened.
      iii. Return to the top of loop.

   (b) If there are no unconnected rooms next to the current room:

      i. A room is popped off the stack and becomes the current room.
      ii. We return to the top of the loop.

6. When the stack is empty, the maze gets formed.

7. Each room has one or more of the basis gates, North (N), East (E), South (S), West (W) gates. They are assigned as kets with values: $|N\rangle = |00\rangle$, $|E\rangle = |01\rangle$, $|S\rangle = |10\rangle$, $|W\rangle = |11\rangle$.

8. For example, for a particular room $|grid[i][j]\rangle$ (i = 0 to m-1, j = 0 to m-1), if $|N\rangle$ and $|E\rangle$ are open, then $|grid[i][j]\rangle = |00\rangle + |01\rangle$.

9. Values for each grid is stored in the $|grid[i][j]\rangle$ register.

IV. QUANTUM ALGORITHM

A. Superposition

Let us focus on the creation of Individual Register storing entire population [7].

1. For the basis states $|0\rangle$ and $|1\rangle$, the Hadamard gate on the basis states acts in the following manner:

$$H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{1}$$

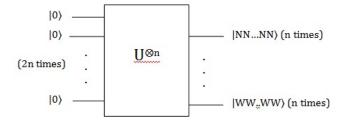$$H\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2}$$

2. The four gates are represented as :

$$|N\rangle = |00\rangle\,, |E\rangle = |01\rangle\,, |S\rangle = |10\rangle\,, |W\rangle = |11\rangle \tag{3}$$

These kets are linearly independent $\langle i\,|\,j\rangle = 1$ for i=j and 0 otherwise. i , j $\in$ { N,E,S,W }

3. An operator U is defined as U = H $\otimes$ H

4. Let the start and end point for the maze be $(i_i\,,\,j_i)$ and $(i_f\,,\,j_f)$ respectively.

5. Each individual in the individual register has a length defined as n = 2 $\times$ (i_f - i_i + j_f - j_i) .
   The length is just taken at hand to ensure that a solution can be reached within that length of an individual.

6. The input $|x_0\rangle = |0\rangle^{\otimes 2n} = |N\rangle^{\otimes n}$ is passed to the operator $U^{\otimes n}$ to obtain the path register $|\Omega\rangle$.

$$|\Omega\rangle = \frac{1}{2^n} \sum_{x\in\{N,E,S,W\}^n}^{4^n-1} |x\rangle \tag{4}$$

B. Fitness Evaluation of Individual Register

1. Under the fitness evaluation part, the *health* of a particular gene is calculated. It is a measure of the closeness of the individual to the optimum solution. Defining an operator for fitness evaluation of individual $|i\rangle$ by operator F such that:
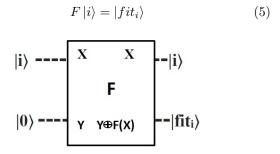
$$F |i\rangle = |fit_i\rangle \qquad (5)$$

FIG. 2. Figure depicting the Fitness operation on individual register.

2. Let the starting point of the maze be (i_i, j_i) . Then for the individual $|i\rangle = |EWNNWE...S\rangle$, the first step thats possible is towards E(east) (i_i+1, j_i) .

3. Current Room: $i_c = i_i$, $j_c = j_i$ and Next Room: $i_n = i_i+1$, $j_n = j_i$.

4. While the gate between Current Room and Next Room is open:
   (a) Current Room = Next Room.
   (b) Next step is W.
   (c) If Current Room = End Room, then break out from the loop.
   (d) Return to top of the loop.

5. $|fit_i\rangle = 2^m - dist(EndRoom - CurrentRoom) = (i_f - i_c)^2 + (j_f - j_c)^2$

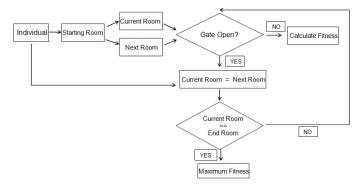6. Fitness = $|fit_i\rangle$ is an m-bit value.

FIG. 3. Flowchart depicting the functionality of Fitness operator.

C. Fitness Evaluation of Path Register

The fitness evaluation discussed above was just for a single individual. But the fitness evaluation is needed for entire population. Quantum Parallelism is an important tool in quantum computation which enables to carry out fitness evaluation in just one step. The transformation is defined by the map:

$$|\Psi\rangle = F |\Omega\rangle = \frac{1}{2^n} \sum_{x \in \{N,E,S,W\}^n}^{4^n-1} |x\rangle \otimes |fit_x\rangle \qquad (6)$$

FIG. 4. Figure depicting the Fitness operation on path register.

D. Finding the maximum

Now that we have $|\Psi\rangle$ containing the superposition of all individuals with their corresponding fitness values, the only task that remains is of finding the maximum fitness state. But, since the number of individual kets in the superposition ket $|\Psi\rangle$ is O(4^n), where n is the individual length, so finding the maximum fitness state directly via Grovers algorithm is not feasible.

Instead, a highly fit individual, one that is close to the optimum solution, is sought for after a finite number of iterations. An Oracle structure is defined which marks the states having fitness values greater than the cutoff fitness and then the Grovers Algorithm is employed to find one of those marked states. This step is carried out for finite number of iterations and then the highly fit individual is looked up.

E. Oracle Structure

Quantum Search Algorithm employs the Grover operator whose first step is defining an Oracle to mark certain individuals based on the fitness criteria [5]. The steps employed in defining an Oracle are these:

1. A random fitness value $cutoff = |fit_x\rangle$ is chosen from the set of fitness values.

2. An oracle O is designed such that it marks all the kets in the superposition ket $|\Psi\rangle$ that have fitness values greater than *cutoff*.

3. The function F operates such that it flips those input states whose fitness is equal to or less than the *cutoff*

fitness, and retains those states whose fitness is above the *cutoff* fitness.

$$|x\rangle \otimes |fit_x\rangle \rightarrow -1^{f(fit_x)} |x\rangle \otimes |fit_x\rangle \qquad (7)$$

$$f(fit_x) = 1, if\, fit_x > cutoff$$
$$= -1, otherwise$$

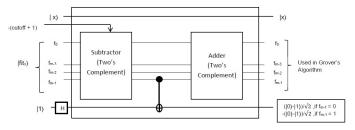4. Defining an Oracle which marks certain individuals having fitness values greater than a threshold fitness value.

FIG. 5. Figure depicting the Fitness operation on path register.

F. Grover's Iteration to find the marked states

After defining the Oracle structure, Grover Operator is used to find one of the marked states. Let j number of fitness states are marked by the oracle which satisfy f(fit_x) = 1.

$$|\Psi\rangle = F|\Omega\rangle = \frac{1}{2^n} \sum_{x \in \{N,E,S,W\}^n}^{4^n - 1} |x\rangle \otimes |fit_x\rangle \qquad (8)$$

$$= \frac{l}{2^n} \sum_{O_k = 1} |x\rangle \otimes |fit_x\rangle + \frac{\sqrt{4^n - l}}{2^n} \sum_{O_k = -1} |x\rangle \otimes |fit_x\rangle \qquad (9)$$

Thus the original register can be written into two different parts, ones for which $O_k = 1$ (the marked ones) and ones for which $O_k = -1$. Also, because multiple individuals may correspond to a single fitness values, hence the value m would in normal case differ from l. And in the above division, it is the m combined state which have $O_k = 1$ and the rest $\sqrt{4^n - l}$ states have $O_k = -1$.

Now in the next iteration, the *cutoff* value is updated by one of the marked state found by the Grover's Algorithm.In subsequent finite number of iterations, a very high fitness state can be found out i.e. approaching the optimum solution.

G. Summary of the Algorithm

1. A path register $|\Omega\rangle$ storing all possible individual states is created.

2. Fitness operator F is applied and $|\Psi\rangle$ state is created with $|\Psi\rangle$ containing the superposition of all possible individuals with their fitness values.

3. (The solution is to be found in m steps)

> i = 0 # Start of Iteration
> **while** $i < m$ **do**
> $cutoff = |fit_{x_0}\rangle$.
>   Oracle O_i marks all $|fit_x\rangle > cutoff$.
>   Grover search to find one marked state $|fit_k\rangle$.
> $cutoff = |fit_k\rangle$.
>   i++
> **end while**

4. At the end of m steps, either the maximum fit gene is obtained or we obtain a highly fit gene close to the optimum solution.

5. On measuring the fitness ket, the individual (or individuals) possessing the given fitness value is obtained because of the entanglement of fitness ket with the individual ket.

V. COMPLEXITY ASSESSMENT

It is known that if k is the length of search space, the complexity of Grovers algorithm is given by O($\sqrt{k}$). But here the population is exponential i.e. $k = 4^n$. So the Grovers algorithm cannot be applied directly to find the optimum solution in polynomial number of steps. Instead, we have defined the oracle and employed the iterative grover search algorithm to look for a high fitness solution in polynomial steps. The algorithm is designed to give a better(higher fitness) solution with each increasing iteration. Hence within m finite steps, it is not necessary that the best solution be reached, but a highly optimum solution can still be reached using this algorithm.

VI. CONCLUSION

Our main focus was on the conversion of a perfect maze into a quantum search problem and to approach towards the optimum solution in finite number of steps using iterative Grover search algorithm. The fitness gate is defined to calculate the ftness of each individuals, which is a criteria in selecting the optimum individuals. An Oracle structure has been proposed to mark certain individual states having fitness levels above the *cutoff* value. Grover's operator iterates subsequently, and with each iteration, the closeness to the optimum solution increases. In the first iteration, if m individuals have fitness levels above the *cutoff*, an O($\sqrt{\frac{n}{m}}$) steps are used in the first iteration, with n being the total number of individuals. Next iteration is performed on the previous marked states, and so on, until the end of iteration is reached, or an optimum solution is found out. Grovers

algorithm provides only the quadratic speedup, hence a solution is not guaranteed in the polynomial iteration steps. Subsequent search algorithms with more speedup would boost the approach towards the optimum solution.

The only limitation lies with the capability of the Grover's algorithm. Subsequent search algorithms, with power more than that of the Grover's algorithm, would boost the approach towards the optimum solution. A point to also note is that, if in future, the search algorithms are found with exponential speed up over the classical algorithms, then the NP-complete problem would have a solution in polynomial steps.

[1] Constructing Optimal Design Trees is NP-Complete. L. Hyafil and R. L. Rivest. *Information Processing Letters.* 1976

[2] Moore's law: past, present and future. Schaller, R.R. Inst. of Public Policy, George Mason Univ., Fairfax, VA, USA IEEE , 0018-9235 , 2002

[3] Quantum Computation and Quantum Information M. A. Nielsen and I. L.Chuang. *New York: Cambridge University Press*, 2000.

[4] Quantum computation and Shor's factoring algorithm Artur Ekert and Richard Jozsa *Rev. Mod. Phys.* 68 (1996) 733753.

[5] A fast quantum mechanical algorithm for database search. L.K. Grover. 3C-404A, AT&T Bell Labs, 600 Mountain Avenue, Murray Hill, NJ.
Proceeding STOC '96 *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing.* 1996

[6] A Multipurpose Backtracking Algorithm. H. A. Priestley and M. P. Ward. *J. Symb. Comput.* 18(1994) 1-40.

[7] From Baker Street to Binary: An Introduction to Computers and Computer Programming. H. F. Legard, P. E. Mcquaid, A. Singer. *New York: McGraw-Hill*, 1983.

[8] Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. Peter Shor. *SIAM J. Computing* 26 (1997) 1484 (quant-ph/9508027).