Evolved Quantum Computing Based Problem Solving

***Abstract*—Quantum Computing (QC) has often been touted as an esoteric and terrifying field of computing research. However, the possible advantages offered by the inherent quantum fundamentals beseeches extensive additional ventures into this field. Just as QP offers exciting ideas in the field of computing, Genetic Programming (GP) offers an application oriented optimization route. GP uses Darwinian theories to maintain a set of candidate solutions, apply multiple operations on the candidates and eventually declare a global winner. In this paper, we combine QC and a flavour of GP to create a new interdisciplinary front of computational intelligence.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Majority of the problems faced in any aspect of computer science, in one way or another, involves a catch-22 of multi-functional optimization. Entire fields have been dedicated to solving a generic version of this issue. The most common method of dealing with optimization problems is to be in collaboration with another interdisciplinary frontier acting in the capacity of a helper function.

A. Computing Methodologies

Most form of computational algorithms in the present day, are executed on a conventional computer. The fundamental notation used for differentiating Classical Computing (CC) and QC [1] is the basic unit of information. While classical computers use bits 0 and 1, quantum computers use "one of" two computational basis states. The label awarded to these states is "bra-ket" notation i.e, state $|0\rangle$ or $|1\rangle$. Bits are assigned states 0 or 1 deterministically and independently. However, qbuits can exist in a superposition state of $\alpha_0|0\rangle$ + $\alpha_1|1\rangle$, where $\alpha 0$ and $\alpha 1$ are complex numbers, such that $|\alpha_0|^2$ + $|\alpha_1|^2$ = 1.

B. Why go Quantum?

The field of applied quantum mechanics is still unexplored for the best part. However, there are certain applications in which QC outperforms CC. Consider Shor's algorithm [2] and RSA [3], both of which are used for encryption. Basically, the encryption of any form of data transmitted over the Internet relies immensley on factorization of a huge number. This process is extremley ardous for a non-quantum computer, with the best known factoring technique requiring an amount of time proportional to $2^{n^{\frac{1}{3}}\log(n)^{\frac{2}{3}}}$, where *n* is the number of digits in the number to be factored [4]. Meanwhile Shor's Quantum algorithm requires time proportional to only $n^2 log(n) log(log(n))$ [5].

QC takes advantage of major quantum phenomena such as superposition, quantum entanglment, principle of uncertainity among others [6], for improving existing searchc and optimization techniques. Humans inherently stick to definitive physical concepts such as deterministic state transition, state duality and temporal static behaviour of particles.

According to Moore's Law, the size of computational units shrinks at an exponential rate as the number of transistors on a chipset increases every year. Even after accounting for physical constraints, a time will come when operations will be conducted on an atomic scale. On such a level, atomic forces overpower particle physics naturally. As a result, understanding QC and applying them in a virtual landscape to get a sense of their possibilities is a must.

C. Using Evolutionary Algorithms

Evolutionary Algorithms (EA) [7] encompass a wide array of research ideas stemming from general principles of genetics and the theory of evolution. Models are developed to illustrate the behaviour of naturally occuring phenomena, develop these algorithms and test out the application of the corresponding theory. The general process of any EA is depicted in figure 1. In a nutshell, an initial population of candidate solutions is generated and each solution is labelled according to their performance on a set fitness function. Further down the line, these candidates are improved upon by conducting a set of operations on them for a predetermined number of generations. After some stopping criterias are achieved, the process strops and the best candidate is hypothesized as the global solution. The advantages offered by this technqiue include but not limited to resilience to noise, in-built support for parallelism and distributed learning, multi-pronged attack and handling complex problems.

Our paper provides an insight into the ensemble of both the previously mentioned techniques and preparing algorithms for a future where quantum computers replace traditional computers as commonplace machines.

II. Methodology

One of the most famous quantum effect invovles "superposition" [8], which allows a quantum entity to exist simultaneously. As a result memory conservation capabilities of any quantum system increases exponentially. Coupling this idea with multiple evolutionary aspects, we present a novel computational frontier.

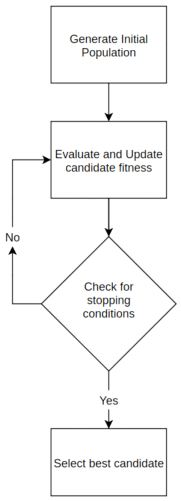
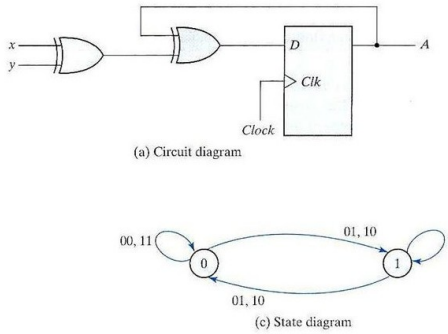


Fig. 1. General Flow of Evolutionary Algorithms

A. Existing Convention

Consider the following problem; Assume the position of a process *P* in a finite state system of *n* states at time *t* to be x_i. Create a general procedure to reach position x_j at time *t+1*.

The conventional process would involve declaration of a register of size *n* and listing out all possible combinations for reaching state x_j along with the costs. This would be followed by creation of a logical circuitry of gates mapping the input state to the output state using boolean primitives. The entire process could still be automated using certain advanced algorithms. However, the memory utilization will not be optimal as all possible state transitions must be considered before building the circuit. Figure 2 displays the state table, state diagram and accompanying circuit diagram for a D flip flop. Notice that all possible states for the input have been mentioned in the state table. This constitues a huge waste of memory as majority of the inputs are never encountered and some states could be invalid. QC aims to reduce this inefficiency between making all the states dependent on each other.



Present state	Inputs		Next state
A	*x*	*y*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table

Fig. 2. States for D flip flop

B. Applied Quantum Mechanics

Classicial state transitions cannot fathom superposition and hence will always be fixated on one fixed state. For the D flip flop example, the number of inputs are two and hence the number of output states is 2^2 for each starting state. The 4 possible states can be represented as an one-hot encoded vector with each bit representing the state of the output. For example, the vector shown below represents state 00 for an input state of 0.

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

Quantum Mechanics is a generalized extension of classical method. The first generalization comes in the form of the temporal values of the elements of the matrix. In QC qubits are utilized instead of bits and as a result the state vectors are variables. Instead of 0 and 1, state vector can be constituted of complex numbers which meet the condition of unitarity. For any matrix *M* to be an unitarity matrix, the only necessary condition is shown in Equation 1 where $M^\dagger$ is the Hermitean adjoint of *M* and *I* is the identity matrix.

$$M^\dagger M = MM^\dagger = I \tag{1}$$

To facilitate the design of circuitry of state diagrams, QC introduces a set of gates to utilize the power of quantum mechanics [9]. The noteworthy ones are mentioned below:

- QNOT: This gate is the quantum extension of the controlled NOT gate in CC. A controlled NOT gate performs the inversion operation on the right-most bit of a vector iff the left-most bit is 1. A 1-bit QNOT gate is depicted as follows:
$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$
- SRN: The application of Square Root of NOT puts a qubit into a state of equal superposition through controlled randomization based on the qubit's past state. A 1-bit SRN is shown below:
$$\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}$$
- Swap: As the name suggests, this gate is used to swap the states of two qubits and is represented as:
$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$
- Parameterized Rotation: This is a family of 1-qubit rotations parameterized by angle θ and is given by:
$$\begin{bmatrix}cos(\theta) & sin(\theta)\\ -sin(\theta) & cos(\theta)\end{bmatrix}$$

C. Applied Evolutionary Operators

EA have been used extensively for complex optimization problems due to quality results even with a lack of mathematical representation, convexity and continuity. They apply a stochastic method and hence expedite the search process.

Algorithm 1 GWO pseudocode

Input: Population size n, random vectors $\vec{r_1}$, $\vec{r_2} \in [0,1]$, initial prey location $\vec{D}$, number of iterations I, fitness function f, coefficient vectors $\vec{A}$, $\vec{C}$ and $\vec{a}$
Set $t = 0$
for i $\in$ [1,n] **do**
  Generate wolf pack population $X_i(t)$ at instance t
  Evaluate each individual against the fitness function
end for
Assign α, β, δ titles to the top three solutions
Evaluate $\vec{D} = |\vec{C} \cdot \vec{X_p}(t) - \vec{X}(t)|$
for i in I **do**
  for Each individual in n **do**
    $\vec{X_1} = \vec{X_\alpha} - \vec{A_1} \cdot \vec{D_\alpha}$
    $\vec{X_2} = \vec{X_\beta} - \vec{A_2} \cdot \vec{D_\beta}$
    $\vec{X_3} = \vec{X_\delta} - \vec{A_3} \cdot \vec{D_\delta}$
    Evaluate $\vec{X}(t+1) = \frac{\vec{X_1}+\vec{X_2}+\vec{X_3}}{3}$
  end for
  Update the coefficient vectors $\vec{A}$ and $\vec{C}$
  $\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a}$
  $\vec{C} = 2\vec{r_2}$
  Linearly decrease $\vec{a}$ from 2 to 0
  Perform Crossover, Selection and Mutation
  Increment t
end for
$\vec{X_\alpha}$ corresponds to the global optimum.

However, EA are not without their flaws which include a probabilistic convergence to the global optima.

In this paper, we follow a genetic modified Grey Wolf Optimization (GWO) algorithm [10] to obtain and update candidate solutions. GWO follows a hierarchical structure of candidate solution which imitates the behaviour of Grey Wolves as observed in nature. The structure of α, β, δ and ω labelled wolves. The α wolves are responsible for finding the global optimum whilst being guided by the β wolves. The δ and ω provide secondary assistance to the leaders and hence maitain the dominant structure. After each search iteration, genetic operators like selection, crossover and mutation are applied to every candidate in the structure. The entire process is discribed in Algorithm 1

The genetic operators for a sample candidate in the form of a genomic are explained here following which the process is depicted in Figure :

- Selection: Given a population of potential solutions, selection involves grading these solutions against a fitness function. The selected solutions are cast into the crossover stage.
- Crossover: From the selected set of solutions, randomly two solutions are selected and the next generation chromosome is created which shares the best attributes of it's parents.
- Mutation: The selected solutions undergo random changes in their composition to promote diversity and increase their changes of converging to the global optimum.

III. Evolved Quantum Computing

Evolved self-organizing computer programs utilize the techniques described in the previous section to solve problems. In this paper the following general problem structure is considered; How to design a circuit to travel from state *A* to state *B*? These problem types are called "Oracle Analysis" as they require we to determine some property of a valid quantum gate with no prior information on the intermediate states and what combination of gates results in what output. Some of the key terminologies used in the following texts are explained in Table *II*

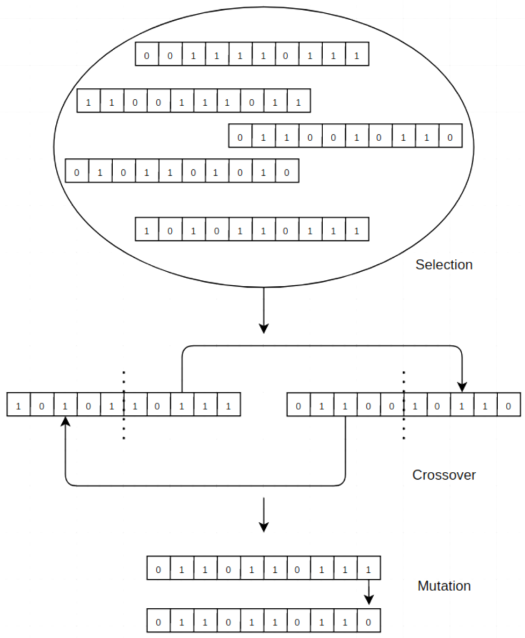


Fig. 3. Genetic Operators

TABLE I
Basic Terminology

Symbol	Description
ψ_{i}	Current State
ψ_{o}	Desired State
ψ_{best}	Current Best State
T	Circuit operator
F	Fitness function
M	Number of chromosomes
G	Number of generations
sel_tech	Selection Techniques
cross_tech	Crossover Techniques
mut_tech	Mutation Techniques
n	Dimensionality of chromosome
$U(n)$	Space of all n dimensional matrices
H_{cur}	Current Gate Matrix
$H_{cur}{}^{\dagger}$	Hermitean Adjoint of H_{cur}
$\|A_{\mathrm{k}}\rangle\ (\psi)$	Orthonormal basis of the environment Hilbert space
p_{k}	Input mapped to Hilbert space
q_{k}	Output mapped to Hilbert space

A. Problem Solving Technique

The process of achieving the target state is discussed in this section. For any test function provided, a set of M chromosomes are initialized and based on the input, $U(n)$ is laid out. Consider applying set of quantum operators Q_μ on the quantum register of chromosomes for $\mu \in [1, o]$. The output state μ for an input qubit chromosome $|\psi_{\mathrm{i}}\rangle$ using operator

Q_μ, can be determined in a probabilistic manner as shown in Equation 2

$$Pr(\mu) = \langle\psi| {Q_\mu}^\dagger Q_\mu |\psi\rangle \tag{2}$$

Similarly the probability of a required state μ is given in Equation 3

$$|\mu\rangle = \frac{Q_\mu |\psi\rangle}{\sqrt{Pr(\mu)}} \tag{3}$$

Since the states can be unitary i.e $H_{cur} \cdot {H_{cur}}^\dagger = I$, elements of *U(n)* will have a definite form. For example, any element of *U(2)*, say *T*, will have the following format for complex inputs z_1 and z_2 seperated by an angle θ:

$$\begin{bmatrix} z_1 & z_2 \\ -e^{i\theta} {z_1}^* & -e^{i\theta} {z_2}^* \end{bmatrix}$$

Now the problem boils down to selection of a particular configuration of these gates. EA uses this subspace and evolutionary techniques to get from ψ_i to ψ_o in *G* generations. A grid of *sel_tech*, *cross_tech* and *mut_tech* is created and all the matrices are graded according to *F*. This step leads into Algorithm 1 which provides the optimal circuit configuration to reach ψ_best, that is closest to ψ_2. The entire process is descibed in Algorithm 2.

B. Key Points

Some of the important points governing the flow of the algorithm are mentioned below:

- The input chromosomes provided are in the form of a bit vector, which is later translated to a quantum register i.e, $\psi = \psi_1 \oplus \psi_2 \oplus \ldots \psi_\text{n}$. The $\oplus$ operator is called the tensorial product in the Hilbert space.
- The most important characteristic of this endeavour is the ability of both the sub-algorithm to retain information. For the evolutionary part, it is the lower swarm particles who memorize the pitfalls. Information retention in QC is guarenteed by the No-Hiding Theorem. The No-Hiding Theorem in an enlarged Hilbert space is given in Equation 4. Due to the correlation between subsystems, quantum information cannot be lost.

$$\begin{aligned} |\mu\rangle \otimes |A\rangle &\rightarrow \sum_k \sqrt{p_\text{k}} |k\rangle \otimes |A_\text{k}(\psi)\rangle \\ &= \sum_k \sqrt{p_\text{k}} |k\rangle \otimes (|q_\text{k}\rangle \otimes |psi\rangle \otimes 0) \end{aligned} \tag{4}$$

- The unused dimension of the Hilbert space are filled by $\otimes 0$ vectors. These unused dimensions can be utilized to completely hide a subsystem and hence delete the information.

Algorithm 2 Evolutionary Quantum Algorithm

1: **Input:** ψ_i, ψ_o, *F*, *M*, *G*, *sel_tech*, *cross_tech* and *mut_tech*
2: Identify *n* as dimension of input
3: Create a grid encompassing all combinations of *sel_tech*, *cross_tech* and *mut_tech*
4: **for** Each generation *G* **do**
5: Evaluate the chromosomes against *F*
6: Call Algorithm 1 with the grid, ψ_i and ψ_o
7: Obtain returned ψ_best
8: **end for**
9: Select the optimal ψ_best

C. Fitness Selection

The evolutionary algorithm depends greatly on the quality of the fitness function. It forms the basis for the next generation. In CC, tailor-made fitness function might suffice for each problem. But for a self-organizing QC system, the main program would have to run multiple times to determine the fitness from each output. Even if a real quantum computer was used, we would get one output and it's associated probability from each epoch. The probability of error so obtained could then be utilized to function as a fitness function. The only drawback in this method would be the absense of a quantum computer and the result exponential computational resources.

The probability of error, in itself is inadequete to function as a fitness function for a QC system. This is due to the extreme ease with which a program could be produced with a 50% best case accuracy that is no where close to the real output. As an analogy, construction of such a program to decide the circuit would be like tossing a coin to decide the next gate in the circuit.

The function used in this paper is a combination of the probability of error and a count of the number of fitness cases for which the program produces better results in more than 50% of the cases. Multiple calls can be made to the "Oracle" to minimize the number of gates in the circuit. Other quantum mechanical properties could also be utilized for improved fitness functions.

D. Schema Theorem

The Schema Theorem [11] dictates which chromosomes move on through the selection phase. If we consider each chromosome as a binary string, there are three possibilities for each bit. It can either be a 1, or a 0 or a * (wildcard) value. The wildcard bit can have either a 1 or a 0. There are two defining characteristics which dictate the goodness of fit.

- Order: Number of non-wildcard bits
- Defining Length: Distance between two furthermost non-wildcard bits

For eg, a chromosome '*10**01' will have an order of 4 and defining length of 5. This particular chromosome represents the set of chromosomes have a 0 or 1 in the leftmost bit, followed by a 10, then two wildcards and ends with a 01. The initial population corresponds to multiple schemata and over the course of applying the genetic operators, some survive while others survive. According to the schema theorem, schemata to low order, high fitness and short defining length reappear in future generations. For a better understanding, an analogy can be drawn between the schema and matching expressions using regular expersions in programming constructs.

IV. Results and Discussion

Due to the unavailability of a quantum computer, an artificial environment was created and simulations were conducted in it. The experiment was set up on a 64-bit Machine with Ubuntu 20.04, Intel® CoreTM i5-7200U CPU @ 2.50GHz × 4, with a GeForce 940MX/PCIe/SSE2 graphic card.

In order to design a quantum state system, we provide the program with a set of input qubits mapped to set of ouput qubit. As discussed previously the fitness function is chosen as a lexicographic combination of the number of misses and probability of error as given in Equation 5, where α is a scaling factor, m is number of misses and e is probability of error. Lower the value returned by the fitness function, higher is the solution's quality.

$$f = \alpha m + e \tag{5}$$

Following this, an initial set of chromosomes corresponding to the ideal circuit were created and gently evolved for set number of generations. The generation progression is composed of the previously mentioned genetic operators. A global count of the best fitness achieved for far is mentioned and the matrix corresponding to that score is declared as the optimal circuit design. The results have been composed for 100 generations in 10 test runs and displayed in Table II.

As we can see, the results for test run 2, 3, 4, 8 and 10 are extremely promising. Further fine tuning could be applied to narrow down the chromosome population. These results act as proof-of-concept when it comes to combing the field of computational intelligence with quantum mechanics.

The main area of improvment would involve identifying and realizing any inherent parallelism which would scale this project exponentially. Genetic computing methods have often been touted as "embarrasingly parallel" due to the increase in throughput and overall performance when deployed on large computing clusters. The reasoning provided is based on the fact that evolutionary computing deals with population batches which are divided into sub-populations called demes, which perform the same function. Similarly, for QC parallelization is important as the applications call for similar computer simulations in the fitness test. While quantum mechanics dwells in the field of decreasing memory requirement, parallelism is well equipped to manage the allocated memory optimally.

REFERENCES

[1] A. Narayanan, "Quantum computing for beginners," Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 1999, pp. 2231-2238 Vol. 3, doi: 10.1109/CEC.1999.785552.

[2] S. Nagaich and Y. C. Goswami, "Shor's Algorithm for Quantum Numbers Using MATLAB Simulator," 2015 Fifth International Conference on Advanced Computing Communication Technologies, Haryana, 2015, pp. 165-168, doi: 10.1109/ACCT.2015.16.

[3] Xin Zhou and Xiaofei Tang, "Research and implementation of RSA algorithm for encryption and decryption," Proceedings of 2011 6th International Forum on Strategic Technology, Harbin, Heilongjiang, 2011, pp. 1118-1121, doi: 10.1109/IFOST.2011.6021216.

[4] N. Guo, L. T. Yang, M. Lin and J. P. Quinn, "A Parallel GNFS Integrated with the Block Wiedemann's Algorithm for Integer Factorization," 2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, Indianapolis, IN, 2006, pp. 45-50, doi: 10.1109/DASC.2006.9.

[5] S. M. Hamdi, S. T. Zuhori, F. Mahmud and B. Pal, "A Compare between Shor's quantum factoring algorithm and General Number Field Sieve," 2014 International Conference on Electrical Engineering and Information Communication Technology, Dhaka, 2014, pp. 1-6, doi: 10.1109/ICEEICT.2014.6919115.

[6] A. Zeilinger, "Quantum entanglement and information," Quantum Electronics and Laser Science Conference (QELS 2000). Technical Digest. Postconference Edition. TOPS Vol.40 (IEEE Cat. No.00CH37089), San Francisco, CA, USA, 2000, pp. 163-.

[7] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, 2016, pp. 261-265, doi: 10.1109/ICGTSPICC.2016.7955308.

[8] R. Jozsa, "Computation And Quantum Superposition," Workshop on Physics and Computation, Dallas, TX, USA, 1992, pp. 192-194, doi: 10.1109/PHYCMP.1992.615537.

[9] P. I. Hagouel and I. G. Karafyllidis, "Quantum computers: Registers, gates and algorithms," 2012 28th International Conference on Microelectronics Proceedings, Nis, 2012, pp. 15-21, doi: 10.1109/MIEL.2012.6222789.

[10] D. Jitkongchuen, P. Phaidang and P. Pongtawevirat, "Grey wolf optimization algorithm with invasion-based migration operation," 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, 2016, pp. 1-5, doi: 10.1109/ICIS.2016.7550769.

[11] Liang Ming, Yu-Ping Wang and Yu-ming Cheung, "A new schema theorem for uniform crossover based on ternary representation," Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004., Melbourne, Vic., Australia, 2004, pp. 235-239, doi: 10.1109/ISSNIP.2004.1417468.

TABLE II
RESULTS

Experiment No.	Input Qubit	Output Qubit	Number of Chromosomes	Best Fitness Value
1	[0 1 1 1 0 1 0 0]	[0 0 0 0 1 1 1 1]	10	0.5081
	[0 1 1 1 0 1 0 0]	[0 0 0 0 1 1 1 1]	50	0.5134
	[0 1 1 1 0 1 0 0]	[0 0 0 0 1 1 1 1]	100	0.5106
2	[0 1 1 0 0 1 1 1]	[0 0 1 0 1 0 1 1]	10	0.4366
	[0 1 1 0 0 1 1 1]	[0 0 1 0 1 0 1 1]	50	0.4821
	[0 1 1 0 0 1 1 1]	[0 0 1 0 1 0 1 1]	100	0.4911
3	[1 0 1 1 1 1 1 1]	[0 1 1 0 0 1 1 0]	10	0.3985
	[1 0 1 1 1 1 1 1]	[0 1 1 0 0 1 1 0]	50	0.3672
	[1 0 1 1 1 1 1 1]	[0 1 1 0 0 1 1 0]	100	0.3725
4	[1 0 1 1 1 1 0 1]	[1 1 0 1 0 1 0 0]	10	0.4786
	[1 0 1 1 1 1 0 1]	[1 1 0 1 0 1 0 0]	50	0.4332
	[1 0 1 1 1 1 0 1]	[1 1 0 1 0 1 0 0]	100	0.4189
5	[1 0 0 1 1 0 0 0]	[0 1 0 1 0 1 1 0]	10	0.6444
	[1 0 0 1 1 0 0 0]	[0 1 0 1 0 1 1 0]	50	0.5842
	[1 0 0 1 1 0 0 0]	[0 1 0 1 0 1 1 0]	100	0.6179
6	[1 0 0 1 1 0 0 0]	[1 1 1 1 0 0 0 0]	10	0.6444
	[1 0 0 1 1 0 0 0]	[1 1 1 1 0 0 0 0]	50	0.5842
	[1 0 0 1 1 0 0 0]	[1 1 1 1 0 0 0 0]	100	0.6179
7	[0 0 1 0 1 1 1 0]	[0 1 0 0 0 0 0 1]	10	0.5895
	[0 0 1 0 1 1 1 0]	[0 1 0 0 0 0 0 1]	50	0.6294
	[0 0 1 0 1 1 1 0]	[0 1 0 0 0 0 0 1]	100	0.5574
8	[0 0 1 1 0 1 0 1]	[1 1 1 0 0 1 1 0]	10	0.4689
	[0 0 1 1 0 1 0 1]	[1 1 1 0 0 1 1 0]	50	0.4355
	[0 0 1 1 0 1 0 1]	[1 1 1 0 0 1 1 0]	100	0.4476
9	[0 0 0 0 0 1 1 0]	[1 0 1 1 1 1 0 1]	10	0.5610
	[0 0 0 0 0 1 1 0]	[1 0 1 1 1 1 0 1]	50	0.5812
	[0 0 0 0 0 1 1 0]	[1 0 1 1 1 1 0 1]	100	0.4877
10	[0 1 1 1 0 0 0 0]	[1 1 1 1 1 0 0 1]	10	0.4919
	[0 1 1 1 0 0 0 0]	[1 1 1 1 1 0 0 1]	50	0.4682
	[0 1 1 1 0 0 0 0]	[1 1 1 1 1 0 0 1]	100	0.4158