Self-Assembly Quantum Dots Growth Prediction by Quantum-Inspired Linear Genetic Programming

Douglas Mota Dias
Anderson Pires Singulani
Marco Aurélio C. Pacheco
Patrícia Lustoza de Souza
PUC-Rio
INCT-DISSE
Rio de Janeiro, RJ, Brazil
Email: douglasm,anderson,marco@ele.puc-rio.br
plustoza@cetuc.puc-rio.br

Maurício Pamplona Pires
Physics Institute
UFRJ
INCT-DISSE
Rio de Janeiro, RJ, Brazil
Email: pires@if.ufrj.br

Omar Paranaiba Vilela Neto
Computer Science Department
UFMG
INCT-DISSE
Belo Horizonte, MG, Brazil
Email: omar@dcc.ufmg.br

Abstract—**In this work we present the application of quantum inspired linear genetic programming (QILGP) to the growth of self-assembled quantum dots. Quantum inspired linear genetic programming is a novel model to evolve machine code programs exploiting quantum mechanics principles. Quantum dots are nanostructures that have been widely applied to optoelectronics devices. The method proposed here relies on an existing database of growth parameters with a resulting quantum dot characteristic to be able to later obtain the growth parameters needed to reach a specific value for such a quantum dot characteristic. The computational techniques were used to associate the growth input parameters with the mean height of the deposited quantum dots. Trends of the quantum dot mean height behavior as a function of growth parameters were correctly predicted, improving on the results obtained by artificial neural network and classical genetic programming.**

Index Terms—**Quantum Inspired Linear Genetic Programming, Quantum Dots,Computational Nanotechnology, Growth Prediction.**

I. Introduction

Recent interest in nanoscience and nanotechnology has emerged due to the possibility of developing novel materials with improved and innovative properties. According to scientists, nanotechnology will be able to change the nature of almost everything that has already been done by humans beings, generating impact on areas such as medicine, engineering, telecommunications, energy, computing, etc. This is a truly multidisciplinary area, involving scientists from different fields of knowledge (chemistry, physics, engineering, computer science, biology, medicine, etc.). In this sense, there is a strong interest in understanding the atomic- and molecular-like systems, such as quantum-dots.

The last decade has witnessed the development of semiconductor quantum dot (QD) structures for application in a variety of optoelectronic devices [1] [2] . QDs are structures on the nanometer scale, which confine the electrons in all three dimensions, leading to the full quantization of the electronic energy levels. The performance of many optoelectronic devices can greatly benefit from the fact that the electronic energy levels are fully quantized [2]. In particular, QD structures have shown a great potential to outperform the quantum well structures in the development of infrared photodetectors based on intraband optical transitions, due to the three-dimensional confinement [2] [3]. Efficient coupling of the normal incident light and higher operation temperatures are some of the promised advantages of the QD structures.

Whether the predictions in terms of improved performance of the infrared photodetectors will prove to be realistic or not strongly depends on the growth of QD structures with high dot density, small size and small size dispersion. One of the difficulties in the process of QD growth is to set up the growth parameters to reach a specific characteristic, namely: a certain QD mean height or density or both. Computational intelligence techniques such as artificial neural network (ANN), genetic programming (GP) and genetic algorithm (GA) may be helpful in finding the optimal growth conditions to achieve a desired QD structure. In [4], the use of ANN has already been considered crucial to solve the future challenges of epitaxial growth. However, only recently the first results on the field were published by our group [5].

On the other hand, the superior performance of quantum algorithms in some specific problems lies in the direct use of quantum mechanics phenomena to perform operations with data on quantum computers. This feature has originated a new approach, named Quantum-Inspired Computing, whose goal is to create classic algorithms (running on classical computers) that take advantage of quantum mechanics principles to improve their performance. In this sense, some quantum-inspired evolutionary algorithms have been proposed and successfully applied in combinatorial and numerical optimization problems, presenting a superior performance to that of conventional evolutionary algorithms, by improving the quality of solutions and reducing the number of evaluations needed to achieve them [6] [7] [8]. In this way, Quantum-Inspired Linear Genetic Programming (QILGP) was recently proposed [9] [10].

In this work, we have studied QILGP behavior in predicting trends in the InAs QDs' mean height behavior as a function of different growth parameters. In addition, the results here

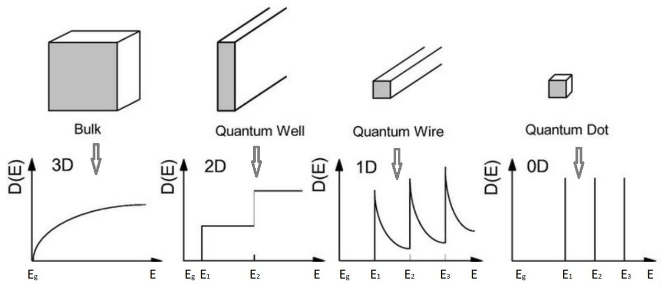

Fig. 1. Schematic morphology and the density of electronic states of bulk materials, quantum wells, quantum wires and quantum dots

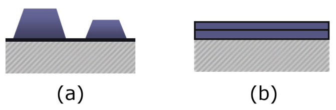

Fig. 2. (a) Volmer-Weber and (b) Frank-van der Merwe (FvdM) growth mode.

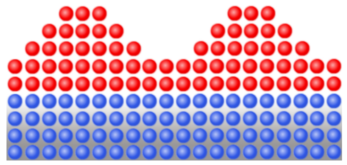

Fig. 3. Stranski-Krastanow (SK) growth mode.

obtained are compared with the prediction of the ANN and a classical linear genetic programming, demonstrating the efficiency of the new method.

This paper is structured as follows: section II describes the quantum dot's features and how they can be synthesized; section III presents the function of the quantum inspired linear genetic programming in detail; section IV describes the problem; section V presents the results and the comparison with other methods; and finally, section VI concludes the work.

II. Quantum-Dots

The research about size quantization effects on semiconductor started in the late 1960s with the creation of novel epitaxial deposition techniques like molecular beam epitaxy (MBE) and somewhat later metalorganic chemical vapor deposition (MOCVD) [1]. More specifically, heteroepitaxy is characterized by the growth of a crystalline film on a crystalline substrate or film of a different material. Within this development, it was possible to insert thin coherent layers of a semiconductor of lower bandgap in a matrix with larger bandgap, restricting carrier movement to only two dimensions. This kind of structure is now know as quantum well [1] .

By the end of the 1980s the main properties of quantum wells were rather well understood and the interest of researchers shifted towards structures with further reduced dimensionality. Thus, quantum wires and quantum dots (QDs) were synthesized, confining the electrons in two and three dimensions, respectively. QDs are structures on the nanometer scale and the confinement of electrons in all three dimensions leads to the full quantization of the electronic energy levels, like in atomic physics [1]. The schematic morphology and the density of electronic states of bulk materials, quantum wells, quantum wires and quantum dots are shown in fig. 1.

The fabrication of quantum dots and hence the confinement of electrons in three dimensions can be achieved in different ways. Among some possibilities we can cite the lithographic techniques, colloidal synthesis and self-organization. The latter, also called self-assembly or self-ordering, has been widely applied in the development of optoelectronic devices. The self-organized heteroepitaxial growth of quantum dots occurs during the growth of strained heterostructures. Different growth modes are well known [11]. In lattice-matched system, i.e., when two crystals have the same lattice constant (length of the unit cell for a cubic lattice), two growth mode can occur, depending only on the interface and surface energies between the two materials. If the sum of the epilayer surface energy and of the interface energy is lower than the energy of the substrate surface, the well-organized Frank-van der Merwe (FvdM) mode occurs. A simple change in the sum described before may drive a transition to the Volmer-Weber (VM) mode, allowing the formation of 3D islands. These two growth modes are shown on fig. 2. On the other hand, in lattice-mismatched systems with small interface energy, initial growth may occur layer by layer, but a thicker layer has a large strain energy and can lower its energy by forming isolated islands in which strain is relaxed. This latter mode is known as Stranski-Krastanow (SK). It was earlier believed that islands formed in the SK growth mode contained dislocation. However, experiments demonstrated the formation of three-dimensional coherently strained islands, i.e. defect-free. Fig. 3 shows in detail the island formation in the SK mode.

QD structures have shown a great potential to outperform the quantum well structures in the development of infrared photodetectors based on intraband optical transitions, due to the three-dimensional confinement [2] [3]. Efficient coupling of the normal incident light and higher operation temperatures are some of the promised advantages of the QD structures. Also, QD can be applied in the development of single-electron transistors, laser, quantum-information processing, etc.

III. Quantum Inspired Linear Genetic Programming

The evolution of programs expressed by one or more tree structures, which are evaluated by an appropriate interpreter, is one of the earliest and most widespread GP types. However, there are other GP types, where programs are represented in different ways (e.g. linear programs). Linear GP [12] is best suited to evolve programs in imperative languages (e.g. C, assembly etc.) and machine code, since each program is a sequence of instructions. The number of instructions can be fixed or variable, which means that different individuals may have different sizes [13].

One of the most recent advances in Evolutionary Computation is represented by the *Quantum-Inspired Evolutionary*

Algorithms (QIEAs) [6]. This approach is motivated by quantum computing, which describes computational processes that are based on making direct use of certain quantum mechanics phenomena (e.g. superposition of states) to perform data operations. These phenomena allow the construction of computers that, in theory, comply with new and more permissive laws of computational complexity [14]. In a quantum computer, the basic unit of information, named *qubit*, can take the states $|0\rangle$, $|1\rangle$ or a superposition of both states. When observed, the qubit is brought to the classical level and the observed state is the value 0 or 1. This superposition of states gives quantum computers an incomparable degree of parallelism that, if properly exploited, allows these computers to perform some impracticable tasks for their classical counterparts, regarding the huge time required for computation.QIEAs aim to create classical algorithms (i.e. running on classical computers), which take advantage of quantum mechanics paradigms in order to improve their performance in solving problems.

An example of a QIEA is the evolutionary algorithm using binary representation applied to multiobjective combinatorial optimization problems with results superior to conventional GAs in terms of convergence time and quality of solutions [7]. In this model, instead of a conventional binary representation, this algorithm uses a special representation which simulates a chromosome consisting of qubits. The evolutionary algorithm for numerical optimization presented in [8] is another example of a QIEA, which is based on the principle of multiple universes of quantum physics. This QIEA uses a representation based on real numbers and has a smaller convergence time for benchmark problems compared to conventional algorithms [8].

Quantum Inspired Linear Genetic Programming (QILGP), whose preliminary version is presented in [9], evolves x86 machine code programs. Using a quantum-inspired approach, QILGP shows best performance for some symbolic regression and binary classification problems when compared in [10] to the reference model, AIMGP [15] (Automatic Induction of Machine Code by Genetic Programming), which is the most successful classical Linear GP model in the evolution of machine code programs. A major motivation to evolve machine code programs is that the most efficient optimization is done at this level. This is the lowest level for optimization of a program and it is also where the highest gains are possible. The optimization can be for speed, space or both [15]. Moreover, this is the fastest GP approach since the evaluation process of each of many individuals is accomplished by the direct execution of the individual by the microprocessor, without requiring intermediate and computationally expensive steps, such as compilation or interpretation.

A. Target Platform

QILGP evolves programs for Intel x86 platform [16], using some instructions of Floating Point Unit (FPU), which can work with data from main memory (m) and/or eight FPU registers ($ST(i) \mid i \in [0\,..\,7]$). This model uses addition, subtraction, multiplication, division, data transfer, trigonometric and arithmetic instructions as the function set for the regression problem of this paper. Table I shows these instructions, their operation, the argument of each instruction (if any) and token values defined by the model, which uniquely identify each instruction. One can note that all instructions in this set have either one or no argument.

A machine code program evolved by QILGP represents a solution. This program reads the input data from main memory, which are composed by the input variables of the problem and by some optional constants supplied by the user. These inputs can be represented by a vector, as exemplified by:

$$I = (V[0], V[1], 1, 2, 3), \tag{1}$$

where $V[0]$ and $V[1]$ contain the two input values of a problem, and where 1, 2 and 3 are the values of three predefined constants. Basically, the AIMGP model is implemented in the same way by the commercial software Discipulus™ [17]. Therefore, this software is also used to perform the comparative experiments presented in this paper.

B. Classical Individuals

QILGP is based on the following entities: the chromosome of a "quantum individual" represents the superposition of all possible programs for the defined search space. It is observed to generate the "classical individual" chromosome, from which the machine code program is generated. That is, the chromosome of a classical individual is the internal representation of a machine code program.

The chromosome of classical individual (CI) represents the functions by a "function token" (FT), which may take integer values from 0 to $(f-1)$, in order to uniquely represent each of f QILGP functions. For the function set of this work, table I shows the tokens' values represented in base 18.

FPU instructions have either one or no argument. In other words, all the set functions have only one terminal, which is represented by a "token terminal" (TT). For a function

TABLE I
FUNCTIONAL DESCRIPTION OF THE INSTRUCTIONS

Instruction	Description	Arg.	Token
NOP	No operation	none	0
FADD m	$ST(0) \leftarrow ST(0) + m$	m	1
FADD ST(0), ST(i)	$ST(0) \leftarrow ST(0) + ST(i)$	i	2
FADD ST(i), ST(0)	$ST(i) \leftarrow ST(i) + ST(0)$	i	3
FSUB m	$ST(0) \leftarrow ST(0) - m$	m	4
FSUB ST(0), ST(i)	$ST(0) \leftarrow ST(0) - ST(i)$	i	5
FSUB ST(i), ST(0)	$ST(i) \leftarrow ST(i) - ST(0)$	i	6
FMUL m	$ST(0) \leftarrow ST(0) \times m$	m	7
FMUL ST(0), ST(i)	$ST(0) \leftarrow ST(0) \times ST(i)$	i	8
FMUL ST(i), ST(0)	$ST(i) \leftarrow ST(i) \times ST(0)$	i	9
FDIV m	$ST(0) \leftarrow ST(0) \div m$	m	A
FDIV ST(0), ST(i)	$ST(0) \leftarrow ST(0) \div ST(i)$	i	B
FDIV ST(i), ST(0)	$ST(i) \leftarrow ST(i) \div ST(0)$	i	C
FXCH ST(i)	$ST(0) \leftrightarrows ST(i)$ (swap)	i	D
FABS	$ST(0) \leftarrow \lvert ST(0) \rvert$	none	E
FSQRT	$ST(0) \leftarrow \sqrt{ST(0)}$	none	F
FSIN	$ST(0) \leftarrow \sin ST(0)$	none	G
FCOS	$ST(0) \leftarrow \cos ST(0)$	none	H

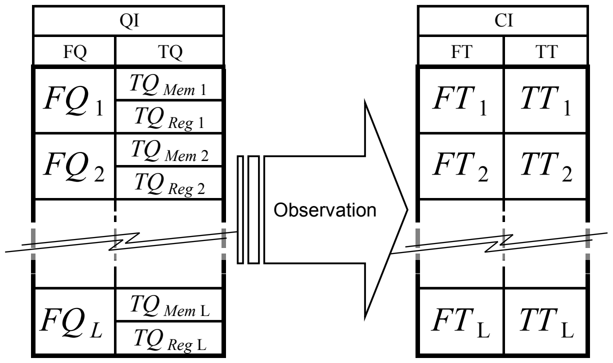

Fig. 4. Creation of a classical individual (CI) by the observation of a quantum individual (QI).

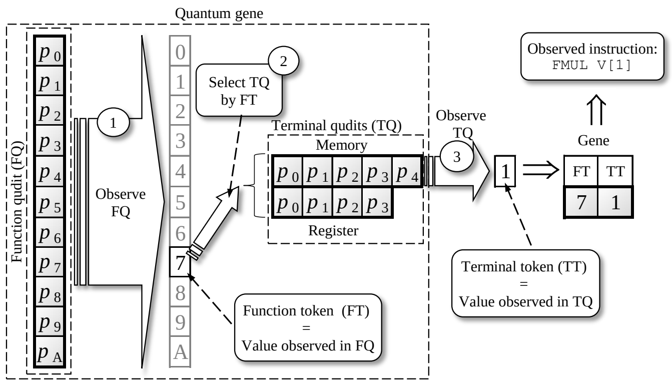

Fig. 5. Creation of a gene by the observation of a quantum gene.

which has no terminal, its corresponding terminal token value is ignored by the model.

Such chromosome can be represented by a structure with $(L \times 2)$ tokens, as shown on the right side of fig. 4, where: L is the maximum program length (in number of instructions); each row represents an instruction i $(1 \leq i \leq L)$; the left column shows the values of function tokens (FT_i) and the right column indicates the values of their respective terminal tokens (TT_i). The execution order of the program is from first to last row. In turn, each row is defined as a "gene". Although this chromosome being fixed-length, the effective program that it represents has variable length. This variation is obtained by adding the `NOP` instruction in the function set. In turn, the process of code generation ignores any gene in which a `NOP` is present, i.e., any gene whose value of its function token is zero is ignored.

C. Quantum Individuals

QILGP is inspired by multilevel quantum systems [18]. Therefore, the basic information unit adopted by QILGP is the qudit. This information can be described by a state vector in a quantum mechanical system of d levels, which equates to a d-dimensional vector space, where d is the number of states in which the qudit can be measured in. That is, d represents the cardinality of the token that will have its value determined by the observation of its respective qudit. The state of a qudit is a linear superposition of d states and may be represented by (2):

$$|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle , \qquad (2)$$

where the $|\alpha_i|^2$ value represents the probability p that qudit is found in state i when observed. The unitary normalization of this state guarantees: $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$.

From the function set of table I, when observing a function qudit whose state is given by (3), such that:

$$|\psi\rangle = \frac{1}{\sqrt{5}}|0\rangle + \frac{1}{\sqrt{4}}|1\rangle + \frac{1}{\sqrt{10}}|2\rangle + \cdots + \frac{1}{\sqrt{8}}|\mathrm{H}\rangle , \qquad (3)$$

the probability of measuring the `NOP` instruction (state "0") is $(1/\sqrt{5})^2 = 0.200$, for `FADD` m (state "1") is $(1/\sqrt{4})^2 = 0.250$, for `FADD ST(i)` (state "2") is $(1/\sqrt{10})^2 = 0.100$ and so on, until finally the probability of measuring the instruction `FCOS` (state "H") is $(1/\sqrt{8})^2 = 0.125$.

The chromosome of a quantum individual (left side of fig. 4) is represented by a list of structures named "quantum genes". A quantum gene is composed by a function qudit (FQ), which represents the superposition of all functions predefined by the function set. It also has two terminal qudits (TQ), since functions can use two different types of terminals. One of the terminal qudits represents the FPU registers (TQ_{Reg}) and the other one represents memory locations (TQ_{Mem}). These registers and memory locations belong to a predefined terminal set. For example, the instruction `FADD ST(0), ST(i)` uses a terminal qudit that represents the superposition of the indexes i of registers `ST(i)`, while terminal qudit for the instruction `FADD` m represents the superposition of memory locations m. Since each quantum gene is observed to generate a classical individual gene (i.e. a complete instruction), both quantum (QI) and classical individuals (CI) have the same length, as shown in fig. 4. The initial length of a quantum individual is controlled by the parameter $p_{0,0}$, whose predetermined value is assigned to p_0 (the probability of observing `NOP`) when it is initialized.

Fig. 5 illustrates the creating process of a gene by the observation of a quantum gene from an example based on both table I and vector case $I = (V[0], V[1], 1, 2, 3)$, exemplified in (1). This process can be explained by the following three basic steps, indicated by numbered circles in fig. 5:

1) The function qudit (FQ) is observed and the resulting value (e.g. 7) is assigned to the function token (FT) of this gene.
2) The function token value determines the terminal qudit to be observed, since each instruction requires a different type of terminal amongst two: register or memory.
3) The terminal qudit (TQ) determined by the function token value is observed and the resulting value (e.g. 1) is assigned to the terminal token (TT) of this gene.

Thus in this example, the observed instruction is `FMUL V[1]`, since "7" is the function token value for this instruction (table I) and "1" is the terminal token value that represents

$V[1]$ in the input vector "I" defined in (1). That is, if a terminal token represents the argument of an instruction which accesses a memory position, the terminal token value indicates the input vector position to be used as its argument. Therefore, in this case, the terminal token values from "0" to "4" indicates that the argument is $V[0]$, $V[1]$, 1, 2 or 3, respectively. However, if the terminal token is for an instruction whose argument is the contents of a FPU register, the terminal token value directly indicates which of the eight registers is the argument. Only the first four registers are used in this example: $ST(0)$ to $ST(3)$.

D. Evaluation of a Classical Individual

Each program evolved by QILGP, as well as AIMGP, is a machine code program consisting of three segments:

- *Header* – Initializes FPU and loads zero into its eight registers.
- *Body* – It is the evolved code itself.
- *Footer* – It transfers $ST(0)$ contents to $V[0]$, since this is the default output of evolved programs and reinitializes FPU. Then executes the return instruction to terminate the program and returns to the evolutionary algorithm main flow.

That is, header and footer are not affected by the evolutionary process.

The evaluation process treats problems caused by instructions such as `FDIV` incurred in dividing by zero or instructions `FSQRT` incurred in calculating a negative number square root, which directly affect the value resulting from the execution of a evolved program. In both cases, the value attributed as a result of such fitness case is zero ($V[0] \leftarrow 0$). This is the same approach adopted by AIMGP, which promotes a neutrality when comparing the results of both models.

E. Quantum Operator

The quantum operator proposed by this model actuates directly on a probability p_i of a qudit, satisfying the normalization condition: $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$, where d is the qudit cardinality and $|\alpha_i|^2 = p_i$. Thus, this operator, here named "P operator", represents the functionality of a quantum gate, performing rotations in a vector which represents a state $|\psi\rangle$ of a qudit in a d-dimensional vector space.

This operator works in two basic steps. First, it increases a given probability of a qudit, as follows:

$$p_i \leftarrow p_i + s(1 - p_i), \tag{4}$$

where s is a parameter named "step size", which can assume any real value between 0 and 1. The second step is to adjust the values of all the probabilities of this qudit to satisfy the normalization condition. Therefore, the P operator modifies the state of qudit increasing the value of its probability p_i by a quantity which, in turn, is directly proportional to the value of step size s.

On the asymptotic behavior of p_i by (4), one can note that a probability never reaches the unit value. This is an important feature of this operator, because it avoids that a probability leads its qudit to collapse, which could cause a premature convergence of the evolutionary search process, damaging the model's performance.

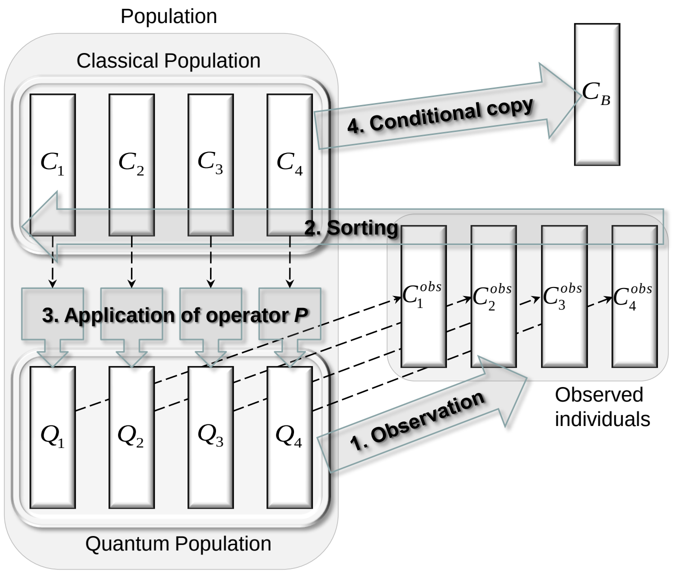

Fig. 6. Basic diagram of the QILGP model.

F. Overall Structure and Operation

The diagram in fig. 6 illustrates the model's structure and its basic operation principles.

Regarding its structure, QILGP has a hybrid population which, in turn, is composed of two populations, one quantum and one classical, both having the same number M of individuals. It also has M auxiliary classical individuals C_i^{obs}, which result from observations of quantum individuals Q_i, where $1 \leq i \leq M$. In the case exemplified by fig. 6, $M = 4$. Besides its structure, the same diagram shows (enumerated) the four basic steps that characterize a "generation" of QILGP, which are described below:

1) Each of the M quantum individuals is observed once, resulting in M classical individuals C_i^{obs}.
2) The individuals of a classical population and the observed individuals are jointly sorted by their evaluations. As a result, the M best individuals from the $2M$ evaluated ones are kept in classical population, ordered from best to worst, from C_1 to C_M. The other M classical individuals remain stored and sorted from best to worst, from C_1^{obs} to C_M^{obs}.
3) The *P* operator is applied to each individual of the quantum population, with reference to their corresponding individuals in classical population. This step is where the evolution actually occurs, since every new generation, the application of this operator increases the probability that quantum individuals observations generate classical individuals more similar to the best ones found so far.
4) If any individual of classical population evaluated in the current generation is better than the best classical individual evaluated so far (compared to previous generations), a copy is stored in C_B, the best classical

individual found by the algorithm so far.

In order to avoid a premature convergence, when the evolution achieves the predefined value of the parameter gwi_{max} (maximum number of generations without improvement), both quantum and classical populations are reinitialized. This process clears all the classical individuals contents and resets the probabilities of all quantum individuals. Next, C_B is copied to C_1 as a seed. Finally, the P operator is applied once in Q_1, taking C_1 as a reference. However, the value of step size (s) used here is different, being determined by the parameter s_r (step size of reinitialization), whose value is some orders of magnitude larger than that of s. This last process can be considered as a "quantum seeding".

IV. Problem Description

The self-assembly growth of quantum-dots was described in section II. As already highlighted, one of the difficulties in the process of QD growth is to set up the growth parameters to reach a specific characteristic, namely: a certain QD mean height or density or both. These concerns increase when considering industrial applications.

The use of predictions tools, such as artificial neural network (ANN) and genetic programming (GP), has already been considered crucial to solve the future challenges of epitaxial growth [4]. The idea in this work is to use an existing database of growth parameters with the resulting QD mean height to be able to supply the set of growth parameters needed to reach a desired QD mean height. Note that any other sample characteristic could have been chosen. In order to do so we use the recently proposed Quantum Inspired Linear Genetic Programing (QILGP) to infer the behavior of the nucleated QDs. The results are compared to a preliminary work using ANN [5] and a well known GP tool [17].

The investigated samples contain one layer of free standing self-assembled InAs QDs deposited by metalorganic vapor phase epitaxy (MOVPE) at 100 mbar on top of InP, InGaAs or InGaAlAs, where the alloys are all lattice matched to the InP substrate. Equivalent structures have been used in the development of QD infrared photodetectors, and are described in great detail elsewhere [19]. The QD height and density of the grown samples were determined by atomic force microscopy (AFM). X-ray diffraction and conventional photoluminescence were used to determine the alloy composition.

We have used a total of 67 sets of growth parameters, obtained from experiments performed in the Semiconductor Laboratory at PUC-Rio. The sets were split in three groups: 47 for training, 10 for validation and 10 for testing.

The six different growth parameters of each set used as input are: the indium flux in the reactor, the growth temperature, the deposition time, the width of the layer on top of which the dots are nucleated, the aluminum and indium contents of this layer material. We have used three different materials for this substrate layer, namely, InP, InGaAs and InAlGaAs. The system output is the mean QD height. Obviously, the QD height depends on other parameters such as pressure and annealing time, for instance. However, in our database these parameters were kept constant, consequently using them as input for the QILGP is unnecessary in this case. When necessary, any other parameter can be introduced in the creation of the prediction system.

For evolutionary algorithms in general, the population of individual solutions can be divided into multiple subpopulations, named as "demes". The migration of individuals between demes promotes the evolution of this population as a whole, which progresses more rapidly than in a single population of equal size. This acceleration inherent to the evolution with demes was confirmed for evolutionary algorithms [20] and in particular for GP [21] [22]. In this work, demes are used first in AIMGP and then in an extended version of QILGP.

V. Results and Discussion

This section aims to present a performance evaluation of the proposed model. To do this, experiments are performed in order to compare the QILGP and the AIMGP (Automatic Induction of Machine Code by Genetic Programming) models. AIMGP is currently the most efficient GP model to evolve machine code and is implemented by the commercial software Discipulus™ [17]. Therefore, this software is used to perform the comparative experiments of this paper.

For symbolic regression problems, as is the case of this work, the fitness value of an individual is determined by computing its mean absolute error (MAE) or its mean absolute

TABLE II
Setup of AIMGP model for the experiments

Parameter	Value
# generations (G)	1,000
Population size (M)	1,000
# demes	10
Migration rate between demes	1%
Initial maximum program length	20
Maximum program length	128
Mutation rate	95%
– Block mutation rate	30%
– Instruction mutation rate	30%
– Data mutation rate	40%
Crossover rate	50%
– Homologous crossover rate	95%
# FPU registers	8

TABLE III
Setup of QILGP model for the experiments

Parameter	Value
# generations (G)	300,000
Population size (M)	6
Max # generations without improvement (gwi_{max})	20,000
Initial probability of NOP ($p_{0.0}$)	0.9
Reinitialization step size (s_r)	1.0
P operator step size (s)	0.004
Maximum program length	128
# FPU registers	8

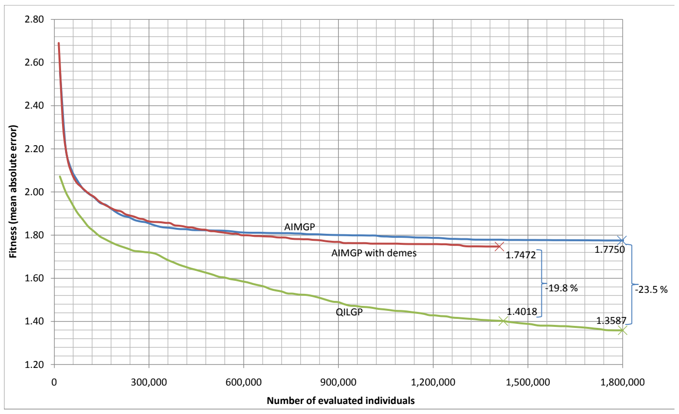

Fig. 7. Fitness curves of the case study.

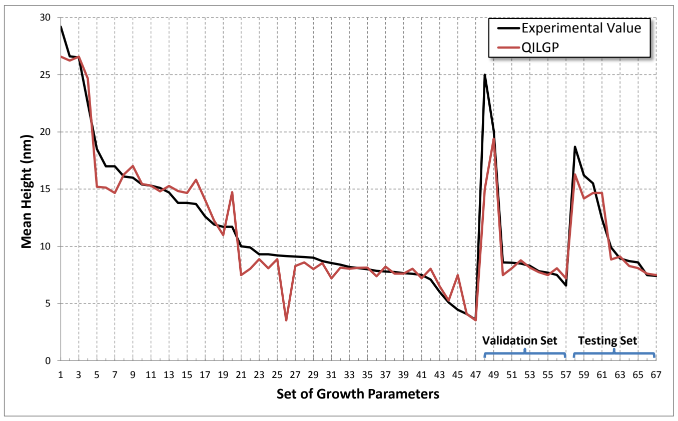

Fig. 8. Experiment 1: Comparison of the ANN prediction for the QD mean height and the obtained experimental data obtained by AFM.

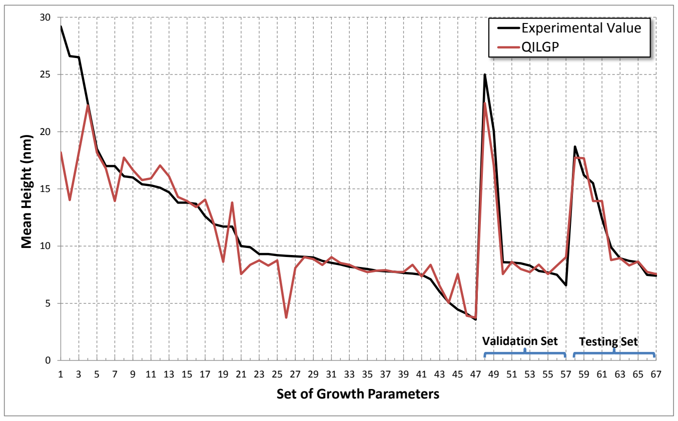

Fig. 9. Experiment 2: Comparison of the ANN prediction for the QD mean height and the obtained experimental data obtained by AFM.

percentage error (MAPE), depending on the experiment. The mean absolute error for n fitness cases is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |t_i - V[0]_i|, \tag{5}$$

where t_i is the target value (true value) of case i and $V[0]_i$ is the output value of the individual for this same case (value of its approximation). In turn, the mean absolute percentage error is calculated by the following equation:

$$\text{MAPE} = 100 \times \frac{1}{n}\sum_{i=1}^{n} \left|\frac{t_i - V[0]_i}{t_i}\right|. \tag{6}$$

The parameters used in the AIMGP and QILGP models are presented in table II and III, respectively.

The number of individuals evaluated by each run of an evolutionary algorithm is usually calculated in a trivial way by multiplying the number of individuals in population (M) by the number of generations (G). However, from experiments conducted in this investigation, one can see that in the case of Discipulus™, the number of individuals evaluated in each run does not match the product $M \times G$. This value varies between executions of the same experiment and one can conclude that the number of individuals evaluated per generation by Discipulus™ is variable and there is no explanation about this fact in any product manual. Thus, we chose the number of generations, G, of the QILGP in order to ensure that the QILGP evaluates at least the same number of individuals than Discipulus™ .

Fig. 7 shows the fitness curves of both models for an average of 100 runs. It also indicates the resulting values from the performance comparison of these models regarding the fitness average of the best individuals per generation. The QILGP solutions present average error values lower than those of the AIMGP, resulting in a superior performance from 19.8% to 23.5%.

As one can observe, the two AIMGPs approaches show an early convergence. Different from this, the QILGP evolution ended without producing a convergence. For that reason and aiming to explore the benefits of demes, we have decided to create an new version of the QILGP using demes and executing a bigger experiment in order to get better results. The new experiment executed 100 QILPG runs of 600,000 generations with 6 demes of 6 individuals each. The results of two different runs are presented below.

Fig. 8 depicts in red (grey) the QD mean height obtained by the quantum inspired linear genetic programming for the 67 different sets of growth parameters. In the same figure we observe in black the experimental values obtained by AFM (Atomic Force Microscope) images. The results obtained by the QILGP follow quite closely the experimental data, except for some bigger errors. In this experiment, the mean average percentage error of the test set is 7.47% compared with 8.30% of the best neural network [5], which represents an error reduction of 10.00%. This is an exceptional result given the relative small number of sets of growth parameters used as input for the creation of the prediction system. The achieved mean average percentage error for the training and validation set was 9.50% and 8.89%, respectively. This experiment was chosen because it provided the smallest error in the validation set.

The results of other selected experiment is shown in Fig. 9.

In this special case, the experiment presents a lower test error, 5.86%, but a greater training and validation error, 11.00% and 10.77%, respectively. It means a 29.40% improvement when compared with the best ANN [5]. The poorer result presented by the training set is due mainly to the three highest mean height data. It is also interesting to observe that most of the other errors in the training set seems the same as those presented in fig. 8. These were also observed in other experiments. We believe that these errors are due to duplicate input data with different output values (occurs due to the uncertainty surrounding the growth in the reactor) and other data without much representation. These data were kept even after a preprocessing, since the database is rather small.

VI. Conclusion

In this work we presented the application of a new quantum inspired evolutionary algorithm in the prediction quantum dots growth parameters. The obtained results demonstrated the potential of quantum inspired linear genetic programming techniques in guiding the growth of QDs. It is encouraging that even with a rather small database we have obtained results with very good accuracy, suggesting that the QILGP can be a reliable approximation of the function that governs the growth of QDs.

Also, the great difference between QILGP and classical genetic programming suggests that the new model can be effective in dealing with small databases. However, new experiments must be conducted and new comparisons, using different databases, should be analyzed before a final conclusion.

Even though the results reported here are specific for a particular QD characteristic, it is clear that the computational technique reported here can be used to predict any desired sample parameter or trend in different samples' characteristics. Another interesting characteristics to be studied would be QD size dispersion and density, which are already underway. Also, other nanostructures rather than quantum dots can be studied.

Computational techniques such as the ones employed in this investigation decrease the number of experiments needed in the laboratory, as they serve as a guide in searching for growth parameters, reducing research costs. The obtained preliminary results are quite promising, in particular for industrial applications, and show how the development of the nanotechnology can greatly benefit from the judicious use of computational techniques.

Acknowledgment

The authors would like to thank CNPq, FAPEMIG, FAPERJ and the Brazilian National Institute of Science and Technology on Semiconductor Nanodevices (DISSE).

References

[1] D. Bimberg, M. Grundmann, and N. Ledentsov, *Quantum dot heterostructures*. New York: Wiley, 2001.

[2] P. Bhattacharya and Z. Mi, "Quantum-dot optoelectronic devices," *Proceedings of the IEEE*, vol. 95, no. 9, pp. 1723–1740, Sep. 2007.

[3] V. Ryzhii, I. Khmyrova, M. Ryzhii, and V. Mitin, "Comparison of dark current, responsivity and detectivity in different intersubband infrared photodetectors," *Semiconductor Science and Technology*, vol. 19, no. 1, pp. 8–16, 2004.

[4] S. Bland, "Future challenges for MOVPE–an industrial perspective," *Journal of Materials Science: Materials in Electronics*, vol. 13, no. 11, pp. 679–682, 2002.

[5] A. Singulani, O. Vilela Neto, M. Aurélio Pacheco, M. Vellasco, M. Pires, and P. Souza, "Computational intelligence applied to the growth of quantum dots," *Journal of Crystal Growth*, vol. 310, no. 23, pp. 5063–5065, 2008.

[6] N. Nedjah, L. dos Santos Coelho, and L. de Macedo Mourelle, Eds., *Quantum Inspired Intelligent Systems*, ser. Studies in Computational Intelligence. Berlin: Springer, 2008.

[7] Y. Kim, J.-H. Kim, and K.-H. Han, "Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 2601–2606.

[8] A. da Cruz, M. Vellasco, and M. Pacheco, "Quantum-Inspired Evolutionary Algorithms applied to numerical optimization problems," in *IEEE Congress on Evolutionary Computation*. IEEE Press, 2010, pp. 1–6.

[9] D. M. Dias and M. A. C. Pacheco, "Toward a quantum-inspired linear genetic programming model," in *IEEE Congress on Evolutionary Computation*. Trondheim, Norway: IEEE Press, May 2009, pp. 1691–1698.

[10] ——, "Quantum-inspired linear genetic programming," *Genetic Programming and Evolvable Machines*, Oct. 2010, submitted.

[11] A. Pimpinelli and J. Villain, *Physics of crystal growth*. Cambridge Univ Pr, 1998.

[12] M. Brameier and W. Banzhaf, *Linear Genetic Programming*, ser. Genetic and Evolutionary Computation. Boston, MA, USA: Springer, 2007, no. XVI.

[13] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Published via http://lulu.com, 2008, (With contributions by J. R. Koza). [Online]. Available: http://www.gp-field-guide.org.uk

[14] L. Spector, *Automatic Quantum Computer Programming – A Genetic Programming Approach*, J. Koza, Ed. Boston, USA: Springer, 2004.

[15] P. Nordin, "AIMGP: A formal description," in *Late Breaking Papers at the Genetic Programming 1998 Conference*, J. R. Koza, Ed. University of Wisconsin, Madison, Wisconsin, USA: Stanford University Bookstore, 1998.

[16] *Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual*, Intel Corporation, 1997. [Online]. Available: http://www.intel.com/design/pentiumii/manuals/243191.htm

[17] F. D. Francone, *Discipulus Owner's Manual, Version 3.0*, Register Machine Learning Technologies, 2004.

[18] B. Lanyon, M. Barbieri, M. Almeida, T. Jennewein, T. Ralph, K. Resch, G. Pryde, J. O'Brien, A. Gilchrist, and A. White, "Quantum computing using shortcuts through higher dimensions," *Arxiv preprint arXiv:0804.0272*, 2008.

[19] P. Souza, A. Lopes, T. Gebhard, K. Unterrainer, M. Pires, J. Villas-Boas, G. Vieira, P. Guimarães, and N. Studart, "Quantum dot structures grown on Al containing quaternary material for infrared photodetection beyond 10 μm," *Applied Physics Letters*, vol. 90, p. 173510, 2007.

[20] R. Tanese, "Distributed Genetic Algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 1989, p. 439.

[21] W. Tackett, "Recombination, selection, and the genetic construction of computer programs," Ph.D. dissertation, University of Southern California, 1994.

[22] D. Andre and J. Koza, "Parallel genetic programming: A scalable implementation using the transputer network architecture," in *Advances in genetic programming*. MIT Press, 1996, pp. 317–337.