3
Programação Genética Linear com Inspiração Quântica

Este capítulo apresenta o modelo de programação genética linear com inspiração quântica proposto nesta tese.

3.1
Plataforma

A Programação Genética Linear com Inspiração Quântica (PGLIQ) evolui programas para a plataforma Intel x86 [61], utilizando algumas instruções relativas à sua Unidade de Ponto Flutuante (*Floating Point Unit* – FPU), as quais, por sua vez, podem trabalhar com dados da memória principal (m) e/ou dos oito registradores da FPU ($ST(i)$, onde $i = 0, 1, \ldots, 7$). Neste modelo, são utilizadas instruções para adição, subtração, multiplicação, divisão e transferência de dados, assim como instruções aritméticas e trigonométricas, além de instruções de comparação e de saltos condicionais. Portanto, estas instruções representam o conjunto de funções da PGLIQ para os estudos de caso abordados neste trabalho.

A tabela 3.1 mostra as instruções citadas e suas respectivas funcionalidades, onde CF representa um *status flag* de comparação da FPU, além de mostrar o respectivo argumento de cada instrução, caso haja. Portanto, pode-se notar que todas as instruções deste conjunto possuem apenas um ou nenhum argumento. É importante mencionar que, apesar das instruções `JB` e `JNB` possuírem um argumento, o qual determina o destino dos saltos condicionais, ambos os modelos PGLIQ e AIMGP não consideram seu argumento para fins de evolução. O que ocorre é que estes modelos fazem uso destas instruções apenas para pular (*skip*), ou não, a próxima instrução a ser executada. Portanto, no contexto deste trabalho, o valor do argumento passado a ambas as instruções é fixo e vale 6 *bytes*, que é o valor para o salto de exatamente uma instrução.

Um programa em código de máquina evoluído pela PGLIQ representa uma solução. Este programa lê os dados de entrada a partir da memória principal, os quais são compostos pelas variáveis de entrada do problema e, opcionalmente, por algumas constantes fornecidas pelo usuário. Estas entradas podem ser representadas por um vetor, conforme exemplificado a seguir:

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

Instrução	Descrição	Argumento
NOP	Nenhuma operação	-
FADD m	$ST(0) \leftarrow ST(0) + m$	m
FADD ST(0), ST(i)	$ST(0) \leftarrow ST(0) + ST(i)$	i
FADD ST(i), ST(0)	$ST(i) \leftarrow ST(i) + ST(0)$	i
FSUB m	$ST(0) \leftarrow ST(0) - m$	m
FSUB ST(0), ST(i)	$ST(0) \leftarrow ST(0) - ST(i)$	i
FSUB ST(i), ST(0)	$ST(i) \leftarrow ST(i) - ST(0)$	i
FMUL m	$ST(0) \leftarrow ST(0) \times m$	m
FMUL ST(0), ST(i)	$ST(0) \leftarrow ST(0) \times ST(i)$	i
FMUL ST(i), ST(0)	$ST(i) \leftarrow ST(i) \times ST(0)$	i
FDIV m	$ST(0) \leftarrow ST(0) \div m$	m
FDIV ST(0), ST(i)	$ST(0) \leftarrow ST(0) \div ST(i)$	i
FDIV ST(i), ST(0)	$ST(i) \leftarrow ST(i) \div ST(0)$	i
FXCH ST(i)	$ST(0) \leftrightarrows ST(i)$ (troca conteúdos)	i
FABS	$ST(0) \leftarrow \lvert ST(0) \rvert$	-
FSQRT	$ST(0) \leftarrow \sqrt{ST(0)}$	-
FSIN	$ST(0) \leftarrow \operatorname{sen} ST(0)$	-
FCOS	$ST(0) \leftarrow \cos ST(0)$	-
FCOMI ST(0), ST(i)	$CF \leftarrow (ST(0) < ST(i))$	i
JB 6	Pula próxima instrução se $CF = 1$	-
JNB 6	Pula próxima instrução se $CF = 0$	-

Tabela 3.1: Descrição das instruções.

onde $V[0]$ e $V[1]$ contêm os valores das duas entradas do problema, e onde 1, 2 e 3 são os valores das três constantes.

Basicamente, o modelo AIMGP é implementado da mesma forma pelo *software* comercial Discipulus™ [69]. Portanto, este *software* é utilizado para a execução dos experimentos comparativos deste trabalho.

3.2 Representação

A PGLIQ é baseada nas seguintes entidades, da seguinte forma: o cromossomo do "indivíduo quântico", que representa a superposição de todos os programas possíveis para o espaço de busca predefinido, é observado para se gerar o cromossomo do "indivíduo clássico", a partir do qual o programa em código de máquina é finalmente gerado. Ou seja, o cromossomo de um indivíduo clássico é a representação interna de um programa em código de máquina.

3.2.1 Indivíduo Clássico

O modelo representa internamente as funções por um "token de função" (TF), que pode assumir valores inteiros de 0 a $(f - 1)$, de forma a representar de maneira

única cada uma das f funções da PGLIQ. A tabela 3.2 mostra um exemplo de conjunto de funções definido para um dado problema, onde os valores dos *tokens* são representados na base hexadecimal.

Instrução	Descrição	Valor do *token*
NOP	Nenhuma operação	0
FADD m	$ST(0) \leftarrow ST(0) + m$	1
FADD ST(0), ST(i)	$ST(0) \leftarrow ST(0) + ST(i)$	2
FADD ST(i), ST(0)	$ST(i) \leftarrow ST(i) + ST(0)$	3
FSUB m	$ST(0) \leftarrow ST(0) - m$	4
FSUB ST(0), ST(i)	$ST(0) \leftarrow ST(0) - ST(i)$	5
FSUB ST(i), ST(0)	$ST(i) \leftarrow ST(i) - ST(0)$	6
FMUL m	$ST(0) \leftarrow ST(0) \times m$	7
FMUL ST(0), ST(i)	$ST(0) \leftarrow ST(0) \times ST(i)$	8
FMUL ST(i), ST(0)	$ST(i) \leftarrow ST(i) \times ST(0)$	9
FXCH ST(i)	$ST(0) \leftrightarrows ST(i)$	A

Tabela 3.2: Exemplo de um conjunto de funções e seus *tokens*.

Assim como mostrado na seção 3.1, as instruções de FPU utilizadas têm apenas um ou nenhum argumento. Em outras palavras, todas as funções do conjunto têm apenas um terminal, o qual é representado por um "token de terminal" (TT). No caso de uma função que não tenha terminal, o seu *token* de terminal correspondente tem seu valor ignorado pelo modelo.

Conforme mencionado, um programa é internamente representado pelo cromossomo do indivíduo clássico, que pode ser representado por uma estrutura com $(L \times 2)$ *tokens*, conforme mostrado na figura 3.1, onde: L é o comprimento máximo do programa (em número de instruções); cada linha representa uma instrução i ($1 \leq i \leq L$); a coluna da esquerda contém os valores dos *tokens* de função (TF_i) e a da direita, os valores dos seus respectivos *tokens* de terminal (TT_i). A ordem de execução do programa é da primeira até a última linha. Por sua vez, cada linha é definida como sendo um "gene".

Apesar do cromossomo do indivíduo clássico possuir comprimento fixo, o programa efetivo que ele representa possui comprimento variável, assim como na PG Linear e também, mais especificamente, na AIMGP. No caso da PGLIQ, esta variação é obtida pela inclusão da instrução NOP no conjunto de instruções. Ocorre que o processo de geração de código ignora qualquer gene no qual um NOP esteja presente. Ou seja, é ignorado todo gene cujo valor do seu *token* de função valha zero.

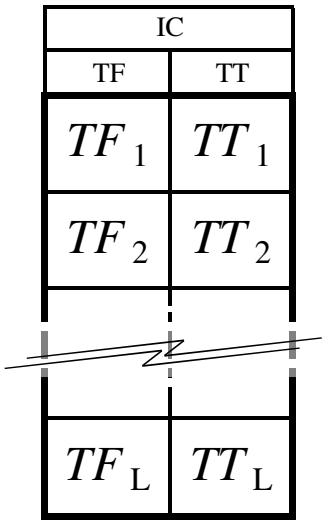

Figura 3.1: Cromossomo de um indivíduo clássico.

3.2.2 Indivíduo Quântico

O modelo proposto neste trabalho é inspirado em sistemas quânticos multiníveis [25]. Portanto, conforme mostrado na seção 2.2.1, a unidade básica de informação adotada pela PGLIQ é o *qudit*. Esta informação pode ser descrita por um vetor de estado em um sistema mecânico quântico de d níveis, o qual equivale a um espaço vetorial d-dimensional, onde d é o número de estados nos quais o *qudit* pode ser medido. Isto é, d representa a cardinalidade do *token* que terá seu valor determinado pela observação do seu respectivo *qudit*.

O estado de um *qudit* é uma superposição linear dos d estados e pode ser representado pela equação 3-2

$$|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i \, |i\rangle \, , \tag{3-2}$$

onde o valor de $|\alpha_i|^2$ representa a probabilidade p de que o *qudit* seja encontrado no estado i quando observado. A normalização unitária do estado garante: $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$.

Tomando-se o conjunto de instruções da tabela 3.2 como exemplo, ao se observar um *qudit* de função cujo estado é dado pela equação 3-3, tal que:

$$|\psi\rangle = \frac{1}{\sqrt{5}} |0\rangle + \frac{1}{\sqrt{4}} |1\rangle + \frac{1}{\sqrt{10}} |2\rangle + \cdots + \frac{1}{\sqrt{8}} |\mathrm{A}\rangle \, , \tag{3-3}$$

a probabilidade de se medir a instrução "NOP" (estado "0") é $(1/5)^2 = 0{,}200$, para "FADD m" (estado "1") é $(1/4)^2 = 0{,}250$, para "FADD ST(i)" (estado "2") é $(1/10)^2 = 0{,}100$ e assim por diante, até que finalmente a probabilidade de se medir a instrução "FXCH ST(i)" (estado "A") é $(1/5)^2 = 0{,}125$.

O estado de um *qudit* é implementado e armazenado em uma estrutura de dados similar a uma roleta. Por exemplo, a figura 3.2 ilustra a implementação do *qudit* relativo à equação 3-3, onde $p'_0 = p_0 = \alpha_0^2$ e, para $i > 0$, $p'_i = p'_{i-1} + p_i$, sendo que $p_i = \alpha_i^2$. Na prática, os valores de p'_i são armazenados nas suas respectivas posições p_i da implementação do *qudit*.

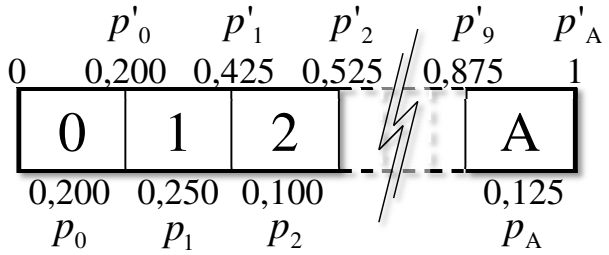

Figura 3.2: Exemplo da implementação de um *qudit*.

Sendo assim, a implementação do processo de observação do *qudit* exemplificado pela equação 3-3 pode ser definida pela função:

$$T(r) = \begin{cases} 0 & \text{se } 0 \leq r < p'_0 \\ 1 & \text{se } p'_1 \leq r < p'_2 \\ 2 & \text{se } p'_2 \leq r < p'_3 \\ \vdots & \vdots \\ \text{A} & \text{se } p'_9 \leq r \leq p'_\text{A}, \end{cases} \tag{3-4}$$

onde r é um valor real entre 0 e 1, o qual é gerado aleatoriamente, com distribuição uniforme, como o primeiro passo da observação, e $T(r)$ retorna o valor observado para o *token*.

O cromossomo do indivíduo quântico, denominado "cromossomo quântico", é representado por uma lista de estruturas denominadas "genes quânticos". Um gene quântico é composto por um *qudit* de função (QF), que representa a superposição de todas as funções predefinidas pelo conjunto de funções. Também possui dois *qudits* de terminal (QT), uma vez que as funções podem utilizar dois tipos diferentes de terminais. Um dos *qudits* de terminal representa os registradores da FPU (QT_{Reg}) e o outro, as posições de memória (QT_{Mem}), sendo que estes registradores e posições de memória pertencem a um conjunto de terminais predefinido. Por exemplo, a instrução "FADD ST(0), ST(i)" utiliza um *qudit* de terminal que representa a superposição dos índices i dos registradores "ST(i)", enquanto que o *qudit* de terminal relativo à instrução "FADD m" representa a superposição das posições m da memória.

Como cada gene quântico é observado para gerar um gene do indivíduo clássico (ou seja, uma instrução completa), ambos os indivíduos, quântico (IQ) e

clássico (IC), possuem o mesmo comprimento, conforme mostra a figura 3.3.

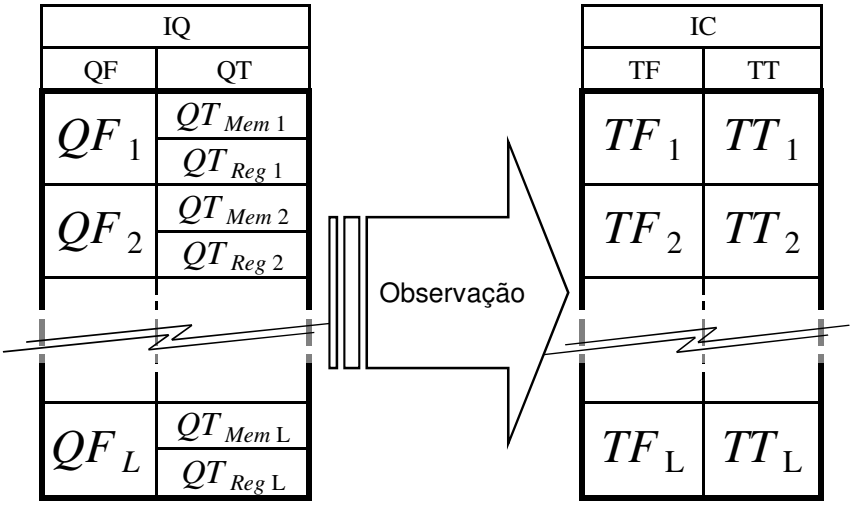

Figura 3.3: Criação de um indivíduo clássico (IC) pela observação de um indivíduo quântico (IQ).

A figura 3.4 ilustra o processo de criação de um gene pela observação de um gene quântico, a partir de um exemplo baseado na tabela 3.2 e no vetor de caso $I = (V[0], V[1], 1, 2, 3)$, exemplificado em 3-1. Este processo pode ser explicado pelos três seguintes passos básicos, indicados por círculos numerados na figura 3.4:

1. O *qudit* de função (QF) é observado e o valor resultante (como, por exemplo, 7) é atribuído ao *token* de função (TF) do gene.

2. O valor do *token* de função determina o *qudit* de terminal a ser observado, uma vez que cada instrução demanda um tipo diferente de terminal dentre dois: registrador ou memória.

3. O *qudit* de terminal (QT) determinado pelo valor do *token* de função é observado e o valor resultante (como, por exemplo, 1) é atribuído ao *token* de terminal (TT) do gene.

Portanto, neste exemplo, a instrução observada é "FMUL V[1]", uma vez que "7" é o valor do *token* de função para esta instrução (tabela 3.2) e "1" é o valor do *token* de terminal que representa $V[1]$ no vetor de entrada I, definido em 3-1. Ou seja, se um *token* de terminal for relativo a uma instrução cujo argumento é um conteúdo de memória, o valor do *token* de terminal indica a posição do vetor de entrada a ser utilizada como seu argumento. Ou seja, neste caso, os valores de um *token* de terminal de "0" a "4" significam que o argumento é $V[0]$, $V[1]$, 1, 2 ou 3, respectivamente. Entretanto, se o *token* de terminal for relativo a uma instrução cujo argumento seja o conteúdo de um registrador da FPU, o valor do *token* de terminal indica diretamente qual dos oito registradores é o argumento, sendo que apenas os quatro primeiros registradores são utilizados neste exemplo: $ST(0)$ a $ST(3)$.

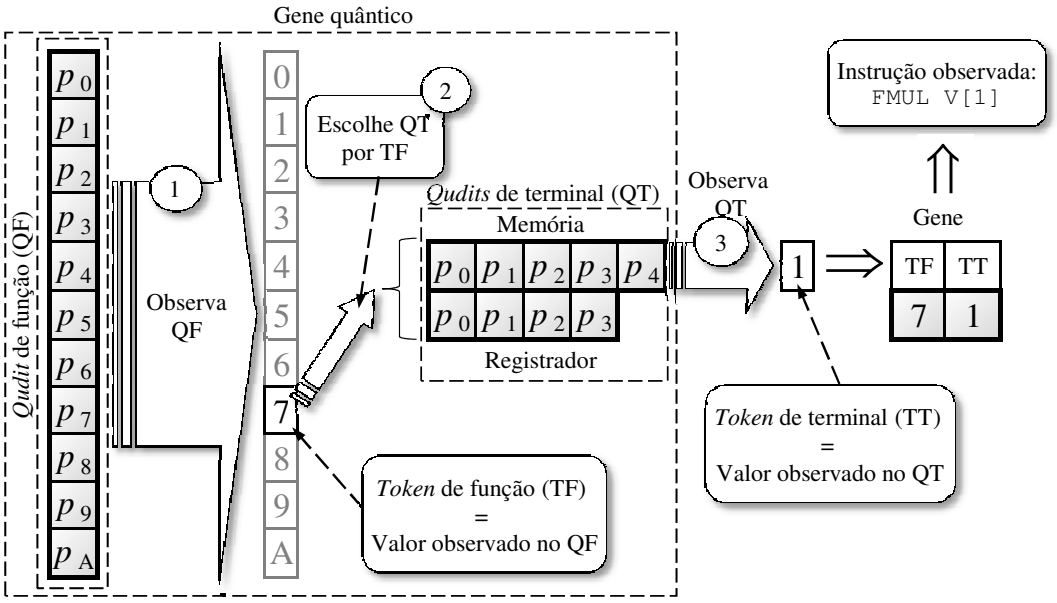

Figura 3.4: Criação de um gene pela observação de um gene quântico.

3.2.3 Programas Evoluídos

Cada programa evoluído pela PGLIQ, assim como aqueles evoluídos pela AIMGP, é um programa em código de máquina composto pelos três seguintes segmentos:

- Cabeçalho (*header*) – Inicia a FPU e carrega o valor zero em cada um dos oito registradores.
- Corpo – É o código evoluído em si.
- Rodapé (*footer*) – Transfere o conteúdo de $ST(0)$ para $V[0]$, uma vez que esta é a saída predefinida do programa, e reinicia a FPU. Em seguida, executa a instrução de retorno para encerrar a execução do programa e retornar ao fluxo principal do algoritmo evolutivo.

Ou seja, nem o cabeçalho, nem o rodapé são afetados pelo processo evolutivo.

3.3 Avaliação de um Indivíduo Clássico

Este processo se inicia com a geração de um programa em código de máquina a partir do indivíduo clássico a ser avaliado. Conforme mostrado na seção 3.2.1, neste processo, o cromossomo do indivíduo clássico é percorrido sequencialmente, gene por gene, *token* por *token* (de função e de terminal), gerando serialmente o código de máquina do corpo do programa relativo àquele indivíduo clássico.

A avaliação de um indivíduo clássico compreende a sua avaliação quanto a todos os casos de aptidão. Sendo assim, a figura 3.5 exibe um trecho de código

em linguagem C, no qual se mostra o funcionamento da avaliação de um caso de aptidão, onde `V` é um vetor que contém os valores do caso de aptidão a ser avaliado. Por sua vez, como descrito na seção 3.1, estes valores são compostos pelos valores das variáveis de entrada do problema e pelos valores das constantes, conforme exemplificado pelo vetor de caso $I = (V[0], V[1], 1, 2, 3)$ em 3-1.

```
void  InitializeModel (void)
{
    /* --------- code --------- */
    int  individualLength ;
    int  numOfBytesOfAnInstruction;
    /* --------- code --------- */

    /* --------- code --------- */
    unsigned char *prog =
        malloc( individualLength  * numOfBytesOfAnInstruction * sizeof(unsigned char));
    /* --------- code --------- */
}

void RunCase(unsigned char *prog, float  *V)
{
    typedef void(*pFunction)(V);

    ((pFunction) prog)(V);

    if  (! _finite (V[0]))
    {
        V[0] = 0;
    }
}
```

Figura 3.5: Trecho de código relativo à avaliação de um caso de aptidão.

Ainda no código da figura 3.5, o vetor `prog` é responsável por armazenar o código de máquina do indivíduo a ser avaliado. A alocação de espaço em memória para `prog` deve garantir que este vetor possua capacidade para armazenar todo o código de máquina. Sendo assim, seu tamanho em *bytes* é calculado pelo produto do número máximo de instruções que um cromossomo clássico pode ter (`individualLength`), pelo número de *bytes* de uma instrução da FPU (`numOfBytesOfAnInstruction`). Esta alocação de memória é feita somente na iniciação do modelo, processo este representado pela função `InitializeModel`.

A avaliação de um caso de aptidão em si é executada pela função `RunCase`. Na primeira linha é declarado um ponteiro para uma função que recebe `v` como argumento. Na linha seguinte, é feito um *typecast* do vetor `prog` para função, o que faz com que o seu conteúdo seja executado como uma função.

Conforme descrito na seção 3.2.3, o resultado do programa avaliado é transferido do registrador $ST[0]$ para a posição $V[0]$ do vetor de caso. Sendo assim, as

últimas linhas da função `RunCase` servem para contornar o problema causado por instruções do tipo `FDIV` que incorram em divisão por zero ou por instruções `FSQRT` que incorram no cálculo da raiz quadrada de um número negativo, e que afetem diretamente o valor resultante da execução de `prog`. Para ambos os casos, o valor atribuído como resultado da avaliação daquele caso de aptidão é nulo. É importante salientar que essa abordagem é a mesma adotada pelo modelo AIMGP. A utilização da mesma abordagem em ambos os modelos visa a preservar a neutralidade quando da comparação dos resultados obtidos pelos mesmos.

3.4 Operador Quântico

O operador quântico aqui proposto atua diretamente nas probabilidades p_i de um *qudit*, satisfazendo a condição de normalização: $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$, onde d é a cardinalidade do *qudit* e $|\alpha_i|^2 = p_i$. Sendo assim, conforme exposto na seção 2.2, este operador, aqui denominado "operador P", representa a funcionalidade de uma porta quântica, efetuando rotações no vetor que representa o estado $|\psi\rangle$ de um *qudit* em um espaço vetorial d-dimensional.

O operador P funciona em dois passos básicos. Primeiramente, o operador incrementa uma dada probabilidade do *qudit*, da seguinte maneira:

$$p_i \leftarrow p_i + s(1 - p_i), \tag{3-5}$$

onde s é um parâmetro denominado "tamanho de passo", que pode assumir qualquer valor real entre 0 e 1. O segundo passo é o ajuste dos valores de todas as probabilidades do *qudit* de forma a satisfazer a condição de normalização.

Portanto, este operador pode ser representado pela função:

$$q' = P\left(q,i,s\right), \tag{3-6}$$

que modifica o estado do *qudit* q, resultando no *qudit* q'. Esta função incrementa o valor da probabilidade p_i do *qudit* q de uma quantidade que, por sua vez, é diretamente proporcional ao valor do tamanho de passo s.

A implementação do operador pode ser descrita pelo pseudocódigo da figura 3.6, onde t_u é o valor superior do *token* representado pelo *qudit* (como, por exemplo, $t_u =$ "A" na equação 3-3).

A figura 3.7 ilustra o funcionamento do operador P, a partir de um exemplo onde é aplicado 5.000 vezes a um *qudit* de $d = 4$ níveis, também denominado *quadrit* [70], aumentando continua e repetidamente o valor da probabilidade p_1. Este *quadrit* pode representar, por exemplo, um *qudit* de terminal relativo a 4 registradores ($ST(0)$ a $ST(3)$), conforme ilustrado pela figura 3.4. Pelo comportamento assintótico de p_1 na figura 3.7, e pela análise da equação 3-5, pode-se perceber que

```
se i = 0 então                                  // incrementa o valor de p_0
    p'_0 ← p'_0 + s(1 − p_0)
    k = 1
senão
    k = i
fim
para j = k até t_u faça                         // incrementa o valor de p_i
    p'_j ← p'_{j−1} + s(1 − p_i)
fim
para j = i até t_u faça                         // normaliza o qudit
    p'_j ← p'_j / p'_{t_u}
fim
```

Figura 3.6: Pseudocódigo da implementação do operador P.

uma probabilidade, na prática, nunca atinge o valor unitário. Esta é uma importante característica do operador, pois evita que uma probabilidade leve seu *qudit* ao colapso, o que poderia provocar uma convergência prematura do processo de busca evolutiva, prejudicando o desempenho do modelo.

3.5 Estrutura e Funcionamento do Modelo

O diagrama na figura 3.8 tem como objetivos ilustrar a estrutura do modelo e auxiliar na introdução ao seu funcionamento básico.

Com relação à estrutura, a PGLIQ possui uma população (híbrida) que, por sua vez, é composta por duas populações, uma quântica e outra clássica, ambas possuindo o mesmo número M[1] de indivíduos. Também possui M indivíduos clássicos auxiliares C_i^{obs}, que resultam das observações dos indivíduos quânticos Q_i, onde $1 \leq i \leq M$. No caso exemplificado pelo diagrama da figura 3.8, $M = 4$.

Além da estrutura, o mesmo diagrama apresenta, enumeradas, as 4 etapas básicas que caracterizam uma "geração" do modelo, as quais são descritas a seguir:

1. Cada um dos M indivíduos quânticos é observado uma vez, resultando nos M indivíduos clássicos C_i^{obs}.

2. É efetuada a ordenação conjunta dos indivíduos da população clássica e dos indivíduos observados quanto às suas avaliações. Como resultado, os M melhores indivíduos clássicos dentre os $2M$ avaliados são mantidos na população clássica, ordenados do melhor para o pior, de C_1 para C_M. Os

[1] Por convenção na bibliografia, a letra M é utilizada para designar o tamanho da população da PG.

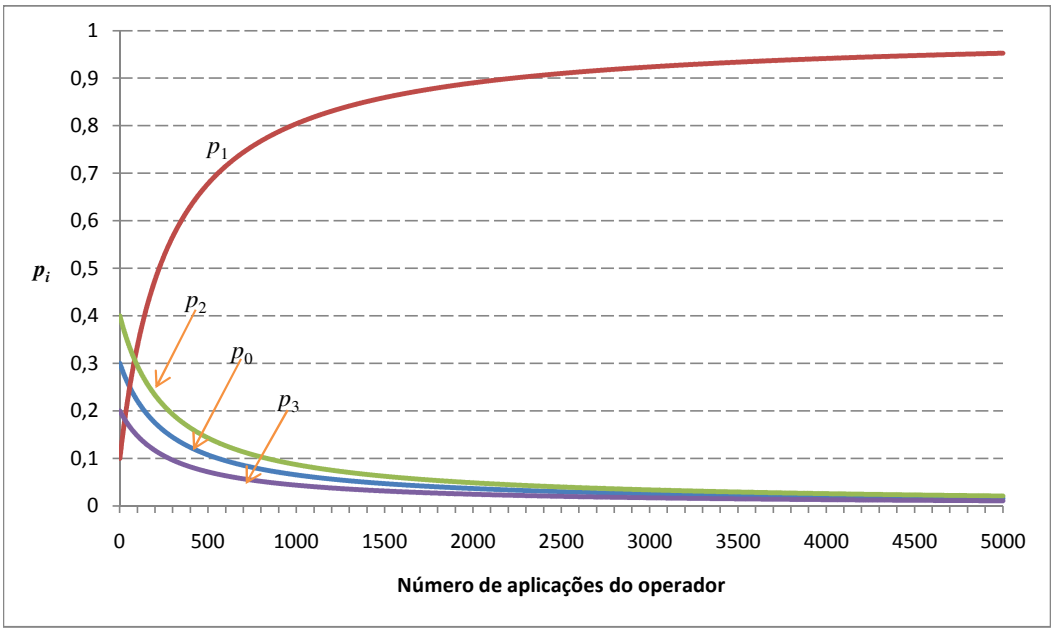

Figura 3.7: Curvas das probabilidades de um *quadrit* durante aplicações sucessivas do operador P.

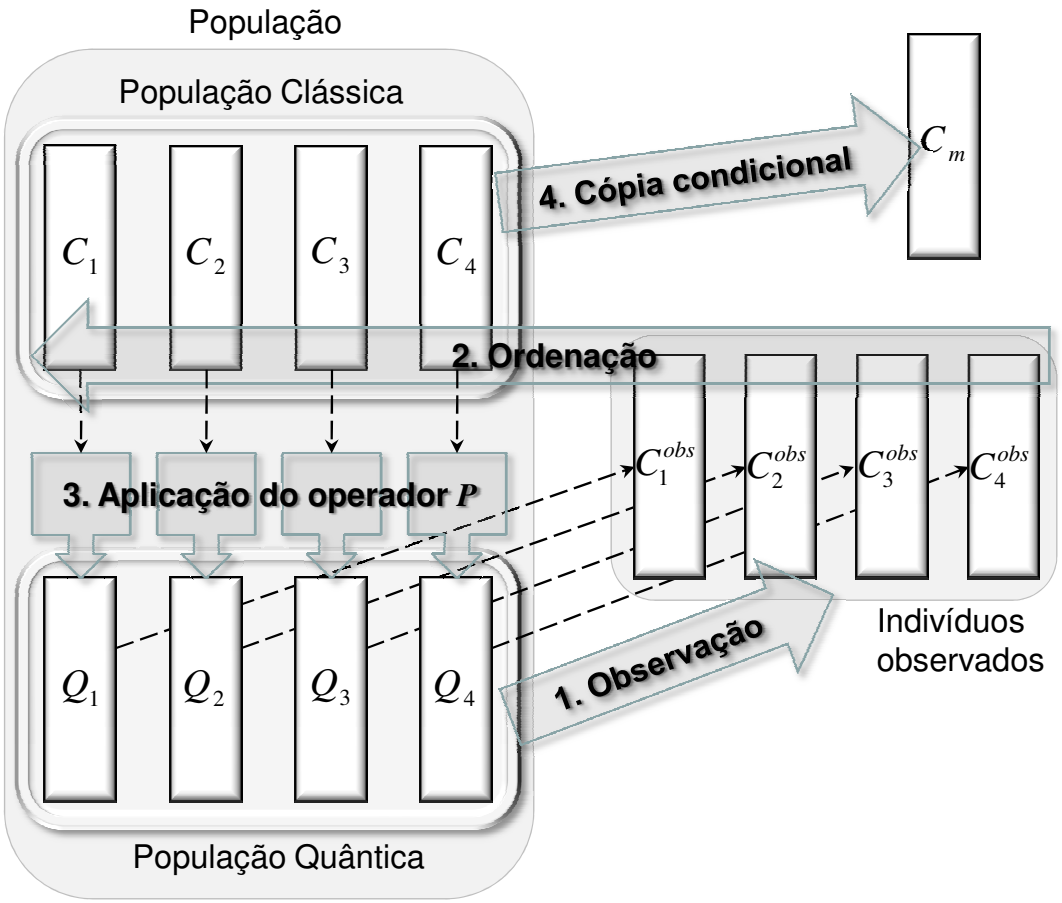

Figura 3.8: Diagrama descritivo básico do modelo PGLIQ.

demais M indivíduos clássicos permanecem armazenados e ordenados, do melhor para o pior, de C_1^{obs} para C_M^{obs}.

3. O operador P é aplicado a cada indivíduo da população quântica, tomando como referência seus indivíduos correspondentes na população clássica. É nesta etapa que ocorre efetivamente a evolução, uma vez que, a cada nova geração, a aplicação do operador aumenta a probabilidade de que a observação dos indivíduos quânticos gerem indivíduos clássicos mais parecidos com os melhores encontrados até então.

4. Caso algum dos indivíduos da população clássica, avaliados na geração atual, seja melhor que o melhor indivíduo clássico avaliado até então (em relação às gerações anteriores), uma cópia do mesmo é armazenada em C_m, o melhor indivíduo clássico encontrado pelo algoritmo até então.

3.6 Algoritmo Evolutivo

Uma vez que os tipos de estudos de casos abordados nesta tese (regressão simbólica e classificação) representam problemas de aprendizado supervisionado, a base de dados é composta por três conjuntos: de treinamento, de validação e de teste. O conjunto de treinamento é o que efetivamente conduz a evolução, enquanto que o conjunto de validação serve para verificar constantemente a qualidade da generalização das melhores soluções encontradas durante a evolução. A aplicação do conjunto de validação em um indivíduo também pode ser denominada "validação" do indivíduo. Finalmente, o conjunto de teste, que não é apresentado ao algoritmo em nenhum momento durante a evolução, é aplicado ao melhor indivíduo obtido como saída de uma execução completa do algoritmo. A aplicação deste conjunto também pode ser denominada "teste" do indivíduo.

O funcionamento do algoritmo evolutivo é descrito pelo pseudocódigo da figura 3.9 e pelas explicações adicionais nas próximas subseções. Sendo assim, todas as citações sobre o pseudocódigo do algoritmo evolutivo da PGLIQ nas próximas subseções, da 3.6.1 à 3.6.7, referem-se ao pseudocódigo descrito pela figura 3.9.

3.6.1 Iniciação da População Quântica

Neste processo, todos os M indivíduos da população quântica são iniciados, conforme mostra o laço nas linhas 1 a 5 do pseudocódigo, sendo que a iniciação de cada indivíduo quântico ocorre na linha 2. Uma vez que um indivíduo quântico é uma lista de genes quânticos, é necessário iniciar cada um dos seus L genes

```
1  para i ← 1 até M faça
2      Inicia indivíduo Q_i da população quântica
3      Inicia indivíduo C_i da população clássica pela observação do
       indivíduo Q_i da população quântica
4      Avalia C_i com o conjunto de treinamento
5  fim
6  C_m.aptidãoTreinamento ← C_m.aptidãoValidação ← ∞
7  gsm ← 0 // Número de gerações sem melhoria
8  para cada geração faça
9      para i ← 1 até M faça
10         Atualiza indivíduo clássico C_i^obs pela observação de Q_i
11         Avalia C_i^obs com o conjunto de treinamento
12     fim
13     Ordena os indivíduos da população clássica em conjunto com os
       indivíduos observados
14     Remove indivíduos da população clássica com mesma aptidão e
       comprimento maior
15     para i ← 1 até M faça
16         Atualiza Q_i aplicando o operador P baseado em C_i
17     fim
18     para i ← 1 até M faça
19         se C_i.aptidãoTreinamento < C_m.aptidãoTreinamento então
20             Avalia C_i com o conjunto de validação
21             se C_i.aptidãoValidação < C_m.aptidãoValidação então
22                 Armazena cópia de C_i em C_m
23                 gsm ← 0
24             senão
25                 gsm ← gsm + 1
26             fim
27         fim
28     fim
       // Verifica se a evolução convergiu
29     se gsm = gsm_máximo então
30         Reinicia populações quântica e clássica
31         gsm ← 0
32     fim
33 fim
```

Figura 3.9: Pseudocódigo do algoritmo evolutivo da PGLIQ.

quânticos. Cada *qudit* de terminal é iniciado pela atribuição do mesmo valor de $1/t$ a todas as suas probabilidades p_i, onde $0 \leq i < t$, e t é o número de terminais que o *qudit* de terminal representa simultaneamente, ou seja, sua cardinalidade.

Entretanto, a iniciação de um *qudit* de função atribui um valor predeterminado $p_{0,0}$ a p_0, de forma a fornecer um controle de comprimento inicial sobre os primeiros programas evoluídos, uma vez que p_0 é a probabilidade de se observar a instrução NOP, a qual é ignorada durante a criação de um programa a partir da leitura de um indivíduo clássico. Portanto, as demais probabilidades p_i, onde $1 \leq i < f$, são iniciadas com o mesmo valor de $(1-p_{0,0})/(f-1)$, onde f é o número de instruções que o *qudit* de função representa simultaneamente. Ou seja, a cardinalidade de um *qudit* de função tem o valor do tamanho do conjunto de funções.

3.6.2 Iniciação da População Clássica

A iniciação da população clássica também ocorre no laço composto pelas linhas 1 a 5 do pseudocódigo, onde são iniciados todos os M indivíduos clássicos desta população. Por sua vez, a iniciação de um individuo clássico C_i, representada pela linha 3 do mesmo pseudocódigo, consiste na iniciação de cada um dos seus L genes pela observação dos L genes quânticos correspondentes que constituem seu indivíduo quântico correspondente Q_i, conforme anteriormente descrito na Seção 3.2.2.

3.6.3 Iniciação do Melhor Indivíduo Clássico

Todo indivíduo clássico possui duas variáveis para armazenar sua aptidão, uma para a aptidão obtida a partir do conjunto de treinamento (*aptidãoTreinamento*) e a outra, a partir do conjunto de validação (*aptidãoValidação*).

Na linha 6 do pseudocódigo, o indivíduo clássico C_m, responsável por armazenar constantemente o melhor indivíduo encontrado ao longo da evolução, é iniciado pela atribuição do valor infinito (disponível na FPU) a ambas as suas variáveis de aptidão. Isto garante que o melhor indivíduo da população clássica seja armazenado desde a primeira geração, uma vez que os problemas aqui abordados são de minimização de erro.

3.6.4 Execução das Gerações

Após a iniciação do modelo, descrita pelas seções anteriores, uma execução completa do algoritmo evolutivo compreende a execução de cada uma das suas gerações, conforme mostrado pelo laço situado nas linhas 8 a 33 do pseudocódigo.

Portanto, as quatro etapas básicas que compõem a execução de uma geração, expostas na seção 3.5, além de outras ainda não mencionadas, encontram-se inseridas neste trecho do pseudocódigo.

Nas linhas 9 a 12, que representam a primeira etapa básica, cada indivíduo clássico C_i^{obs} é atualizado, primeiramente pela observação do indivíduo quântico Q_i correspondente e, em seguida, pela avaliação utilizando o conjunto de treinamento.

A segunda etapa básica é representada pela linha 13, onde é efetuada a ordenação conjunta dos indivíduos da população clássica e dos indivíduos observados, quanto às suas avaliações, resultando no armazenamento dos M melhores indivíduos na população clássica. O método de comparação entre dois indivíduos, empregado por este processo de ordenação, determina o melhor indivíduo como aquele que apresenta a menor aptidão (menor erro). Entretanto, caso ambos apresentem a mesma aptidão, o melhor indivíduo é aquele que possui o menor comprimento efetivo, ou seja, o menor número de instruções, ignorando-se os genes que possuam a instrução NOP. O objetivo deste método é promover a evolução de indivíduos menores sem prejudicar a qualidade das soluções e, dessa forma, evitar a ocorrência de inchaço (*bloat*) nos programas evoluídos. Testes preliminares mostraram que o método, de fato, melhora significativamente o desempenho do modelo, conforme o esperado.

O processo descrito na linha 14 do pseudocódigo complementa e finaliza a segunda etapa. Este processo, que funciona como um pós-processamento da ordenação executada na linha 13, tem como objetivo promover a diversidade da população clássica, uma vez que atua removendo indivíduos que possuam aptidões repetidas. Seu efeito pode ser exemplificado a partir da figura 3.8, onde $M = 4$, e do seguinte conjunto de duplas representando a população clássica (C_1 a C_4): $pop_C = \{\{0{,}1; 20\}, \{0{,}1; 23\}, \{0{,}4; 15\}, \{0{,}4; 17\}\}$, onde cada dupla representa um indivíduo clássico, sendo que o primeiro elemento de uma dupla representa sua aptidão e o segundo, seu comprimento. Supondo-se que o conjunto $pop_O = \{\{0{,}5; 18\}, \{0{,}6; 14\}, \{0{,}8; 11\}, \{0{,}8; 12\}\}$ represente os indivíduos observados (C_1^{obs} a C_4^{obs}), o conteúdo final da população clássica após a execução da linha 14 seria $pop_C = \{\{0{,}1; 20\}, \{0{,}4; 15\}, \{0{,}5; 18\}, \{0{,}6; 14\}\}$, onde pode-se notar que o pós-processamento eliminou todos os indivíduos com aptidões repetidas e comprimentos mais longos. Ou seja, este pós-processamento faz uso dos próprios indivíduos da população clássica e dos indivíduos observados para reduzir ao máximo o número de indivíduos com a mesma aptidão que permanecem na população clássica. Também neste caso, conforme esperado, testes preliminares mostraram a significativa melhoria que este pós-processamento promove no desempenho do modelo.

3.6.5
Aplicação do Operador "P"

Representando a terceira etapa básica ilustrada pela figura 3.8, nas linhas 15 a 17, o operador P é aplicado a cada indivíduo quântico Q_i, tomando como referência seus respectivos indivíduos C_i na população clássica. Por sua vez, a figura 3.10 apresenta um pseudocódigo que descreve como o operador P é aplicado em um indivíduo quântico Q, onde é mostrado que as probabilidades de cada um dos L genes de Q são ajustadas segundo seus respectivos genes g de um indivíduo clássico C. Para o gene quântico, QF representa o *qudit* de função, e QT, o de terminal. Já para o gene clássico, TF representa o *token* de função, e TT, o de terminal. Desta forma, conforme já mencionado, toda vez que é aplicado, o operador P incrementa a probabilidade de Q ser observado em um estado mais parecido com o estado de C.

para $g \leftarrow 1$ **até** L **faça**
  $Q.QF[g] \leftarrow P(Q.QF[g], C.TF[g], s)$
  $Q.QT[g] \leftarrow P(Q.QT[g], C.TT[g], s)$
fim

Figura 3.10: Pseudocódigo relativo à aplicação do operador P em um indivíduo quântico.

3.6.6
Determinação e Cópia do Melhor Indivíduo Clássico

Este processo representa a quarta etapa básica ilustrada na figura 3.8, que é realizado nas linhas 18 a 28 do pseudocódigo, onde todos os indivíduos da população clássica têm suas respectivas aptidões comparadas com a aptidão do melhor indivíduo clássico evoluído até então.

A comparação se inicia na linha 19, onde a aptidão do indivíduo clássico C_i, calculada a partir do conjunto de treinamento, é comparada com a aptidão do melhor indivíduo C_m, também obtida pelo mesmo conjunto. Caso a aptidão de C_i seja menor que a de C_m, isto significa que C_i representa uma solução melhor que C_m com relação ao conjunto de treinamento. Neste caso, C_i tem sua capacidade de generalização avaliada em seguida (linha 20), o que é feito pela aplicação do conjunto de validação. Se, nesta avaliação, também apresentar aptidão menor que a aptidão de C_m obtida previamente pelo mesmo conjunto, o indivíduo clássico C_i passa a ser o melhor indivíduo evoluído até então. Sendo assim, é armazenada uma cópia de C_i em C_m (linha 22).

Como consequência, pode-se notar que a validação das melhores soluções segue um gradiente de aptidão. É desnecessário validar todos os indivíduos durante uma execução completa da programação genética, uma vez que soluções com boa validação, mas que tenham uma aptidão de treinamento comparativamente pior, não são interessantes como resultado. Além disso, estas validações desnecessárias causariam um impacto considerável no custo computacional do algoritmo.

3.6.7 Reinício Condicional da População

Esta heurística, cuja execução condicional encontra-se descrita pelas linhas 29 a 32 do pseudocódigo, visa a evitar a convergência prematura da evolução durante cada execução do modelo.

A variável gsm ("gerações sem melhoria") é responsável por contar há quantas gerações não surge um novo melhor indivíduo em uma execução do algoritmo, sendo iniciada com valor nulo na linha 7. A cada geração em que não ocorre uma melhoria na evolução, a variável gsm é incrementada (linha 25). Por outro lado, quando surge um indivíduo melhor, este contador recebe valor nulo novamente, sendo reiniciado (linha 23). Entretanto, conforme mostram as linhas 29 a 32, se o valor de gsm atingir o valor predefinido do parâmetro $gsm_{máximo}$ (número máximo de gerações sem melhoria), as populações quântica e clássica são submetidas a um processo específico de reinício e, em seguida, gsm é reiniciada com valor nulo. Quanto ao seu valor típico, $1 \ll gsm_{máximo} \ll G$, onde G representa o número máximo de gerações, predefinido, para cada execução do modelo.

Por sua vez, o processo de reinício das populações quântica e clássica, executado na linha 30 do pseudocódigo, pode ser descrito pelo seguinte algoritmo:

1. Todos os indivíduos da população quântica são reiniciados, ou seja, todos os seus *qudits*, tanto de função quanto de terminal, tem suas probabilidades reiniciadas para o mesmo valor $p = 1/c$, onde c é a cardinalidade do *qudit* ao qual p pertence.

2. O operador P é aplicado uma vez em todos os indivíduos da população quântica, tomando como referência o melhor indivíduo clássico C_m. Entretanto, o valor do tamanho do passo utilizado pelo operador nesta etapa é diferenciado, sendo determinado pelo parâmetro s_r (tamanho do passo de reinício). Ou seja, aqui o valor do tamanho do passo $s = s_r$, sendo $s_r = 1{,}0$ o melhor valor obtido experimentalmente. Uma vez que o valor típico de s_r é muito maior que o atribuído a s durante a evolução (tipicamente entre 0,001 a 0,010), a consequência da execução desta etapa é que todos os indivíduos quânticos assumem um estado, cuja configuração aumenta as chances das suas respec-

tivas observações acarretarem em indivíduos clássicos mais parecidos com o melhor indivíduo clássico avaliado até então (C_m). Ou seja, pode-se dizer que aqui é efetuada uma "semeadura quântica".

3. O indivíduo C_1 da população clássica recebe uma cópia de C_m como semente.

4. Os demais indivíduos da população clássica (de C_2 a C_M) são reiniciados pela atribuição do valor infinito (disponível na FPU) às suas respectivas aptidões. Estas atribuições garantem que tais indivíduos sejam excluídos da população clássica, como consequência da próxima ordenação conjunta desta população com os indivíduos clássicos observados (C_1^{obs} a C_M^{obs}).

Conforme pode-se observar no pseudocódigo, após a conclusão deste processo de reinício condicional (linha 32), o algoritmo evolutivo continua a ser executado até que alcance a sua última geração (laço nas linhas 8 a 33).