UM NOVO MODELO DE PROGRAMAÇÃO GENÉTICA COM INSPIRAÇÃO QUÂNTICA

Aluno: Camilo Poppe de F. Muñoz

Orientador: Marley M. B. Rebuzzi Vellasco

Co-orientador: Douglas Mota Dias

Introdução

A computação evolutiva é uma área que se propõe a evoluir, de forma automática, soluções sob diversas formas. Dentre elas valores numéricos, matrizes, vetores ordenados, algoritmos, programas, etc. Assim, o usuário é capaz de resolver problemas complexos fornecendo uma quantidade mínima de informações. Por outro lado, a teoria da computação quântica (CQ) já foi usada para o desenvolvimento de diversos algoritmos com inspiração quântica. É da união destes conceitos que surge a proposta de desenvolver este novo modelo de programação genética com inspiração quântica. Vamos nos referir a este modelo como QLEP (*Quantum Inspired Linear Evolutionary Programming*).

O projeto gira em torno destes dois eixos principais: computação quântica e programação genética (PG). Cada um destes conceitos envolve muita teoria e é bastante abrangente. Por isso, antes de começar a explicar a estrutura do QLEP vamos explicitar quais aspectos das teorias, tanto de CQ quanto de PG, vamos adotar e por que.

A inspiração quântica pode permitir que o QLEP obtenha um melhor desempenho comparado aos atuais modelos de inteligência computacional (IC). Alem do mais, como veremos mais adiante, os algoritmos que usam tal inspiração tem demonstrado maior capacidade para resolver problemas mais complexos. Por isso, espera-se que o QLEP apresente resultados competitivos com os modelos clássicos já existentes e consagrados.

Neste relatório explicaremos a estrutura e organização do QLEP. Logo vamos esclarecer a forma como está sendo implementado. Em seguida serão apresentados os primeiros resultados e os ajustes que ainda devem ser feitos para que o sistema seja mais eficiente.

Fundamentos Teóricos

A – Programação Genética

O conceito de programação genética [1] surge em 1990 nos artigos publicados por John R. Koza e Hugo deGaris. Desde então, a PG vem apresentando uma capacidade de resolver uma ampla variedade de problemas, requerendo do usuário uma quantidade mínima de informações especificas do problema, e produzindo resultados competitivos com aqueles produzidos por programadores humanos.

Simplificando, podemos dizer que toda a teoria pode ser abordada a partir de um conceito. As possíveis soluções de um problema são representadas em indivíduos (cromossomos) que serão submetidos a um processo evolutivo e logo avaliados por uma função de avaliação especifica do problema. Em PG as soluções são programas de computador. Sendo assim, a avaliação de cada indivíduo deve necessariamente passar pela sua execução.

Entretanto devemos estabelecer as diferenças de um algoritmo evolutivo e genético para um algoritmo apenas evolutivo. Hoje o QLEP é um modelo puramente evolutivo, pois não possui operadores de troca de material genético entre indivíduos. Para que um algoritmo evolutivo seja genético é necessário implementar cruzamentos entre indivíduos. Caso haja evolução sem cruzamento falamos apenas de um algoritmo evolutivo.

Ao longo dos anos diversas formas de representação foram desenvolvidas. Inicialmente Koza apresenta a representação em forma de árvore [1] onde fica clara a diferença entre função e terminal. Estes são dois conceitos muito importantes, pois cada solução é composta da combinação certa de funções e terminais. Nós não usamos a representação em árvore. Preferimos adotar a representação linear dos indivíduos. A LGP (*Linear Genetic Programming*) que está definida na tese de doutorado de Markus Brameier [2]. Esta forma continua usando os mesmos conceitos teóricos básicos da representação em árvore, porem é a representação mais adequada para a evolução de código de máquina. (Mais adiante explicaremos porque é preferível que o QLEP evolua os programas diretamente código de máquina)

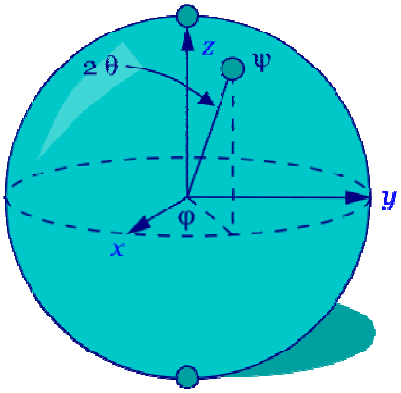
Mesmo trabalhando segundo o modelo de LGP, o QLEP não possui apenas indivíduos “clássicos”. Devemos lembrar que o modelo se inspira também nos princípios da computação quântica. Por isso iremos usar o conceito de indivíduos quânticos. Mas para isso devemos abordar alguns fundamentos teóricos da CQ.

B - Computação Quântica

O computador quântico é um dispositivo teórico que faz uso direto de certos fenômenos da mecânica quântica para realizar operações de dados. Serão, portanto, estudados os processos computacionais que se baseiam nesses fenômenos e que permitem diminuir o esforço e a complexidade para resolver determinados problemas.

O principal fenômeno da mecânica quântica que será usado na computação é o fenômeno do gato de Schrödinger. É desta característica física que surge o conceito de q-bit [3]. Em um computador quântico o q-bit passa a ser adotado como nova unidade de base, assim os processos podem emitir sinais 0, 1 ou uma superposição

probabilística de 0 e 1. Da mesma forma que o q-bit assume, na computação quântica, uma função de unidade de base, o QLEP terá uma unidade de base inspirada no q-bit. É o q-token. Mais adiante este conceito será definido em detalhes. Por enquanto basta notar este como um dos aspectos de influência quântica no modelo QLEP.



Representação em esfera de um q-bit

É necessário lembrar que a computação quântica ainda não é comercialmente viável. Só foi realizada em condições especiais criadas em laboratório e trabalhando com apenas 7 q-bits e a temperaturas baixíssimas. Por isso ao falar em inspiração nos referimos a criar um algoritmo que rodará sempre em computadores clássicos tomando os conceitos quânticos como inspiração para tentar melhorá-lo. Desenvolver algoritmos puramente quânticos pode ser muito difícil, além de não poderem ser implementados em máquinas "clássicas". Mas isto não impede um sucesso das aplicações de algoritmos evolutivos com inspiração quântica [4]. O QEA (*Quantum-Inspired Evolutionary Algorithm*) [5], por exemplo, já foi implementado com sucesso em um modelo evolutivo que também é caracterizado por um cromossomo, uma função de avaliação e uma dinâmica populacional. Além de estes modelos apresentarem respostas corretas, também chegam a uma solução de forma mais veloz e são capazes de resolver problemas mais complexos.

Portanto, este trabalho parte da crença de que há chances concretas de que a utilização da inspiração quântica em programação genética também possa ser bem sucedida em termos de melhoria de desempenho do modelo clássico. É necessário então desenvolver a estrutura deste novo modelo.

Estrutura

A - Q-token

O q-token constitui a unidade básica do QLEP. Portanto é necessário esclarecer o que ele é exatamente.

O q-token é uma superposição de estados. No caso de uma função ele será composto pela probabilidade de ativação de cada função. Caso seja o q-token referente a um terminal, ele age da mesma forma com as opções de terminal que lhe são respectivas.

Visualmente ele poderia ser representado por um diagrama de barras. Assim cada barra representa a probabilidade de determinada função ou terminal ser ativada uma vez que o q-token seja observado.

Como estamos abordando problemas onde existem múltiplas funções e múltiplos terminais para cada função, o q-bit não poderia ser adotado. Foi necessária uma unidade de base que fosse capaz de admitir mais de três dimensões ao mesmo tempo. Por isso falamos de q-token, e também de representação em diagrama de barras. Uma visualização em forma de esfera já não seria possível.

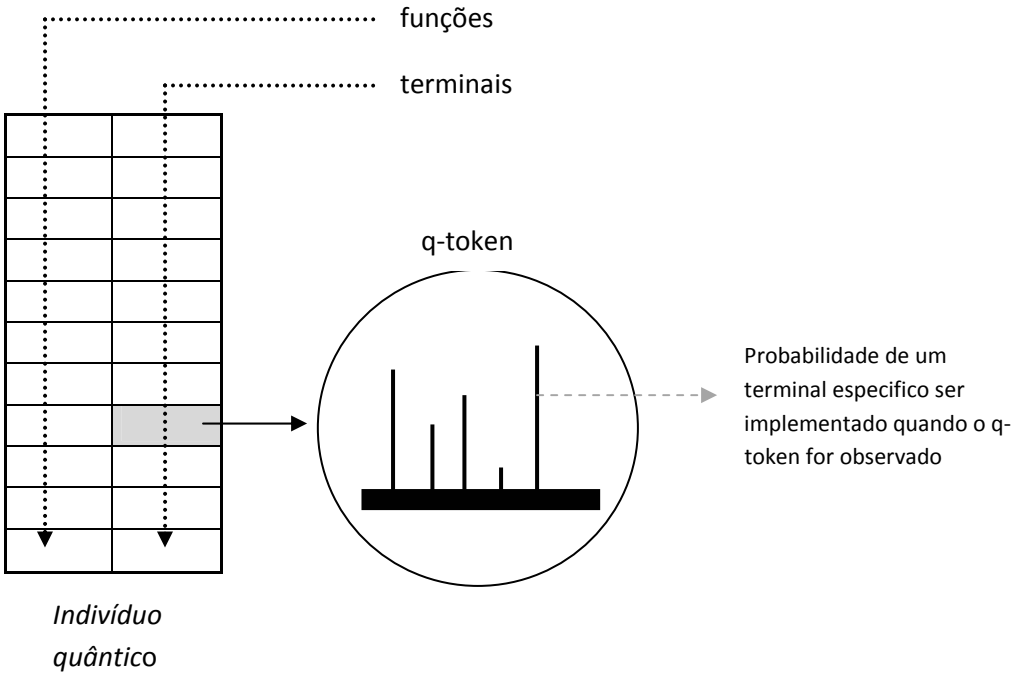
Uma vez que o q-token está definido podemos descrever o indivíduo que será submetido ao processo de evolução.

B - Representação do indivíduo

O indivíduo, no QLEP, é representado em três fases diferentes. Em uma primeira fase, ele é totalmente quântico. O indivíduo quântico é aquele que ainda não foi observado. Após observação obtemos um indivíduo simbólico, o "token individual". Ao ser reescrito em forma linear, este indivíduo simbólico passa a ser o indivíduo "clássico" representado na forma linear como em LGP. Assim ele poderá ser avaliado. Vale apontar que é durante essas três fases que será aplicado o processo evolucionário. Por isso vamos analisar com mais detalhes as características dos indivíduos quântico, simbólico e final, dito "clássico".

a) Indivíduo Quântico

O indivíduo quântico é composto de q-tokens ainda não observados. Ao ser iniciado cada q-token está igualmente calibrado (Existe uma igualdade de probabilidade para cada acontecimento possível). O primeiro indivíduo clássico será gerado a partir desse indivíduo quântico. Por tanto, ao ser iniciado desta forma, garantimos que o primeiro indivíduo clássico será gerado de forma aleatória. Em PG é comum usar populações iniciais criadas aleatoriamente. Porém, é possível manipular as condições iniciais para dar mais ênfase a determinadas características privilegiadas para a solução do problema.



Note que em PG costuma-se trabalhar com populações de indivíduos. Uma das características que o QLEP herda da CQ é a de não precisar de tal dinâmica. O fato de trabalhar com um indivíduo quântico faz com que uma evolução possa ser feita ao re-calibrar os q-tokens que o compõem.

Finalmente, a única mudança que ocorre no indivíduo quântico é a distribuição das probabilidades em cada um de seus q-tokens. Isto ocorre no fim de cada rodada e representa uma grande mudança para os processos que seguem. A observação do indivíduo quântico leva à criação imediata do indivíduo simbólico.

b) Indivíduo simbólico

O indivíduo simbólico é obtido a partir do indivíduo quântico já observado. Esta é uma das características que o QLEP adota das teorias da CQ. Ou ainda, esta propriedade vem da mecânica quântica. É o fenômeno ilustrado pelo gato de Schrödinger.

No momento da observação cada q-token colapsa e adota uma função ou terminal definido. Desta forma, o indivíduo simbólico já pode ser considerado no sentido absoluto da palavra, ele possui características próprias e fixas. É nesse momento que o QLEP deixa de usar conceitos de CQ e passa a trabalhar com os princípios da computação evolucionaria.

É importante ressaltar que todo q-token pode colapsar em um qNop. O qNop é uma função que não opera no QLEP. Trechos de código inoperantes podem ser interessantes para se chegar a uma solução de forma mais rápida. Isto já foi alvo de alguns trabalhos de PG [2]. O fato de ter espaços de código inoperante pode representar uma vantagem ao permitir que dois trechos que antes eram separados por um código ruim se juntem formando uma solução melhor. Naturalmente, estes códigos inoperantes não podem ser excessivos. Muito código inoperante passa a ser lixo e ocupa espaço de memória inutilmente. Mas em PG (diferentemente dos algoritmos genéticos) pequenas variações podem representar mudanças significantes na execução de um programa (solução). As taxas de mutação e todos os parâmetros de regulagem tendem a ser baixos evitando grandes mudanças nas soluções. Foi então necessária a criação do qNop para que o QLEP possa ter um bom desempenho.

A cada nova observação do indivíduo quântico iremos obter um novo indivíduo simbólico. O objetivo é encontrar aquele que represente a melhor solução do modelo para o problema. Para isto, devemos submeter o indivíduo a um processo de avaliação. Porém, os indivíduos simbólicos ainda não podem ser avaliados. Para isso será necessário representá-los em forma linear.

c) Indivíduo Clássico

Em LGP o indivíduo é representado em forma linear. Funções e terminais são ordenados para poderem ser avaliados por uma função pertinente ao problema.

Para poder avaliar o indivíduo simbólico é necessário então transformá-lo em indivíduo clássico. Este será descrito em código de máquina e representado em forma linear como uma seqüência de comandos. O “header” e o “footer” são as seqüências de código de máquina que determinam o inicio e o fim do indivíduo respectivamente.

Existe também outro motivo que levou à necessidade de criação do qNop. Ele desempenha um papel fundamental para o QLEP ao permitir a evolução de indivíduos clássicos de tamanho variável a partir de um indivíduo quântico de tamanho fixo. Conseguimos criar estes indivíduos clássicos flexíveis em tamanho uma vez que os qNop's presentes em um indivíduo simbólico não geram código de máquina no indivíduo clássico. Desta forma quanto mais qNop's tiver o indivíduo simbólico, menor será o código de máquina representado no indivíduo clássico. Em PG estima-se que códigos muito grandes tendem a se distanciar da solução [2]. Esse é mais um motivo para permitir a variação de tamanho do indivíduo clássico.

Este indivíduo clássico constitui a fase final do indivíduo no modelo QLEP. O processo evolutivo será feito a partir das três fases: quântica, simbólica e clássica.

C - Processo evolutivo

O processo evolutivo usado no QLEP é semelhante a um processo usado por Han e Kim ao desenvolver o algoritmo evolucionário com inspiração quântica. Desde já, uma diferença importante é o fato de trabalharmos com apenas um indivíduo quântico.

Ao ser iniciado, este gera o primeiro indivíduo simbólico. Como foi discutido anteriormente, podemos assegurar que este é criado aleatoriamente já que o q-tokens iniciais tem suas probabilidades igualmente distribuídas. O indivíduo simbólico obtido será então transformado em indivíduo clássico para poder ser avaliado. Assim a função de avaliação poderá determinar o grau de pertinência deste. Por ser o primeiro, ele automaticamente será o que obteve melhor avaliação até o momento. Deve ser feito então o reajuste e ele será guardado até que apareça um outro indivíduo com melhor avaliação. O QLEP guarda em memória apenas o indivíduo simbólico referente ao melhor indivíduo clássico avaliado.

O reajuste será feito sempre e usa como parâmetro o melhor indivíduo no momento. Trata-se de redistribuir as probabilidades de cada q-token. O indivíduo simbólico já possui funções e terminais bem definidos, assim os q-tokens que constituem cada posição do indivíduo quântico terão um aumento da probabilidade respectiva á função ou terminal. Em seguida haverá uma normalização do conjunto de probabilidades desse q-token para garantir que a soma destas seja igual a um.

Este processo de reajuste constitui a essência para o processo evolutivo. Ao redistribuir as probabilidades nos q-tokens o indivíduo simbólico tende a convergir. Assim o QLEP se aproxima cada vez mais da solução do problema. Indivíduos com uma avaliação boa tendem a estar mais próximos da solução, por isso é interessante dar mais chances a uma repetição de certas características desse indivíduo. Tratamos as probabilidades com o principio de elitismo. Caso alguma função tenha uma probabilidade superior ou igual a 0,9 um novo processo de reajuste é ativado. Desta vez o reajuste deverá ser feito seguindo o principio de linearização. Desta forma evitamos a presença de superindivíduos (indivíduos tão polarizados que inibem qualquer chance de diversidade).

Um ciclo evolutivo do QLEP pode ser descrito finalmente como a seguinte seqüência de procedimentos:

1. Observação do indivíduo quântico para obter o indivíduo simbólico.
2. Avaliação do indivíduo clássico obtido a partir do indivíduo simbólico.
3. Comparar avaliações do melhor indivíduo guardado e do novo.
4. Guardar o melhor indivíduo simbólico.
5. Reajustar os q-tokens de acordo com o melhor indivíduo simbólico.

Para realizar este processo, algumas variáveis devem ser determinadas. É necessário definir quantos ciclos devem ser executados, quais constantes deverão ser usadas, e quais funções estão permitidas. Alem de definir o nível do reajuste. Isto é crucial já que um reajuste grande demais poderá levar a uma convergência antes de conseguir uma solução satisfatória. E por outro lado, um reajuste fraco demais pode demorar demais ou simplesmente fazer com que o QLEP permaneça como uma busca aleatória.

Existe ainda a possibilidade de realizar diferentes experimentos. Estes são conjuntos de ciclos. A cada novo experimento o indivíduo quântico é reiniciado. Desta forma obtemos os resultados de evoluções que tiveram diferentes indivíduos iniciais. Mesmo realizando vários experimentos o QLEP compara o melhor indivíduo de cada experimento e guarda apenas o melhor indivíduo simbólico.

Estes conceitos evolutivos vêm em grande parte de PG. Porem existem muitos aspectos do desenvolvimento do projeto que não fazem parte da estrutura teórica em si. A implementação é também um fator determinante para o desenvolvimento da teoria.

Implementação

Uma das vantagens do QLEP é a forma como ele é implementado. Para agilizar os processos e ganhar velocidade, o código em C é direcionado para trabalhar diretamente com código de máquina. Os indivíduos (programas) são rodados diretamente na FPU para poder então pular a fase de compilação. Desta forma a avaliação feita a cada ciclo é agilizada.

Para trabalhar desta forma foi necessário usar certas características de PG. A opção de trabalhar com LGP foi feita para permitir um processamento do indivíduo diretamente na FPU do computador. A representação em arvore não permite uma execução seqüencial das funções como é feito no processador. Por outro lado, o QLEP não trabalha com as funções do processador em forma de pilha. O código em C usa acesso direto às posições na FPU.

Desta forma usamos a memória apenas para armazenamento do melhor indivíduo. Esperamos assim obter um rendimento melhor deslocando um numero mínimo de informações. No momento da execução isto é uma grande vantagem do QLEP, porem exige mais trabalho para a programação inicial do sistema. Um mapeamento completo das funções de ponto flutuante deve ser feito. Foi necessário também integrar linguagens como XML, C# e C para poder traduzir as funções do processador para o programa que está escrito em C. Isto causa um trabalho longo de preparo do QLEP mas espera-se que o ganho em tempo de solução dos problemas seja diminuído desta forma.

Uma vez que a questão do trabalho com o código de máquina foi resolvida, resta estudar a implementação do código em linguagem C. Como o modelo é relativamente simples, foi possível estabelecer um algoritmo para evoluir os problemas. A maior preocupação foi em fazer com que o código não ficasse especifico demais. Deixar um grau de flexibilidade foi prioridade. Assim, novos conceitos de PG poderão ser introduzidos. Até hoje abordamos o modelo sem usar dinâmicas populacionais onde ocorrem mutações e cruzamentos de indivíduos. Mas estes mecanismos poderão ser úteis. Então essa flexibilidade se torna necessária. A implementação do modelo na linguagem de programação está sempre sofrendo pequenas modificações pois é no algoritmo que se define o sucesso do modelo. Mas já foi possível consolidar uma base para o modelo que já apresenta alguns resultados. Após recolher dados e analisá-los podemos remanejar o algoritmo.
Mas antes vamos apresentar os primeiros resultados obtidos com o QLEP.

Primeiros resultados

O objetivo é testar o QLEP de forma a poder comparar seu desempenho com os outros modelos existentes. Para isso vamos testá-lo com funções padrão já usadas para testar outros modelos. Entre estas estão as funções “mexican hat” e “distancia” descritas na tese de Markus Brameier [2]. As funções e constantes usadas devem ser as mesmas. Os parâmetros de controle da evolução também devem ser igualmente regulados. Alguns desses parâmetros podem estar presentes em um determinado modelo e não em outro. Mas sempre que houver um parâmetro em comum a regulagem deverá ser a mesma.

Atualmente o QLEP ainda está sendo preparado para ser submetido a esses testes. O mapeamento das funções usadas ainda está sendo feito e traduzido para poder ser implementado pelo código em C. Por este motivo ainda não temos resultados concretos quanto a seu desempenho comparado a outros modelos.

Atualmente o QLEP foi testado com uma mesma função simples a fim de tentar descobrir pequenas variações que podem melhorar o desempenho do programa. Trata-se de resolver pequenas falhas de implementação do código em C. A função usada foi criada arbitrariamente. O modelo foi usado para resolver um problema de regressão. O procedimento de teste será apresentado a seguir.

A) Preparo da experiência

Um problema de regressão pode ser simulado por uma tabela de entradas e saídas onde todos os dados de entrada são usados por uma mesma função para criar os dados de saída. O objetivo deste tipo de problema é achar, a partir desses dados a função que os liga.

Para isto foi necessário criar uma base de dados para que o QLEP possa resolver o problema. Como se trata de uma primeira experiência. O objetivo principal era testar a capacidade do modelo em chegar a uma resposta por um método evolucionário. Desejávamos confirmar a capacidade de convergência do modelo. Alguns parâmetros do teste foram determinados arbitrariamente. A função propriamente dita foi um desses parâmetros.

Para preparar a base de dados, criamos no Excel uma planilha com colunas que representam dados de entrada. Definimos a função que o QLEP deveria ser capaz de encontrar por regressão. E com essa função usamos os dados de entrada para criar a coluna que representa os dados de saída. As configurações foram as seguintes:

- Função: x^2+y^2+3
- Ciclos: 500000
- Comprimento do indivíduo: 64
- Probabilidade de qNop: 0.2
- Constantes: 2 e 3

O arquivo que constitui a base de dados foi criado em Excel com duas variáveis de entrada e uma de saída. Foram criados 25 dados de entrada para cada variável, criados aleatoriamente entre 0 e 1. Consequentemente usando a função definida x^2+y^2+3 obtivemos 25 dados de saída (um para cada dois dados de entrada).

Outro aspecto do preparo da experiência foi a coleta de dados. Para poder colher dados de forma eficiente, um código de impressão teve que ser implementado. Este código foi adicionado ao algoritmo e permitia imprimir os resultados da evolução. Como uma solução precisa de alguns milhares de ciclos para ser encontrada, colher dados não era apenas uma tarefa de impressão. Os arquivos obtidos eram excessivamente grandes e não facilmente manipuláveis. Foi necessário então, fazer uma seleção dos dados. Para isso um procedimento de seleção foi aplicado. Apenas era impressa a avaliação de um indivíduo quando este era melhor que o melhor indivíduo guardado até então. Isto foi feito para um arquivo do Excel onde cada avaliação impressa era escrita em coluna. Cada coluna desse arquivo corresponde a uma experiência. Assim podia ser calculada facilmente a média das experiências.

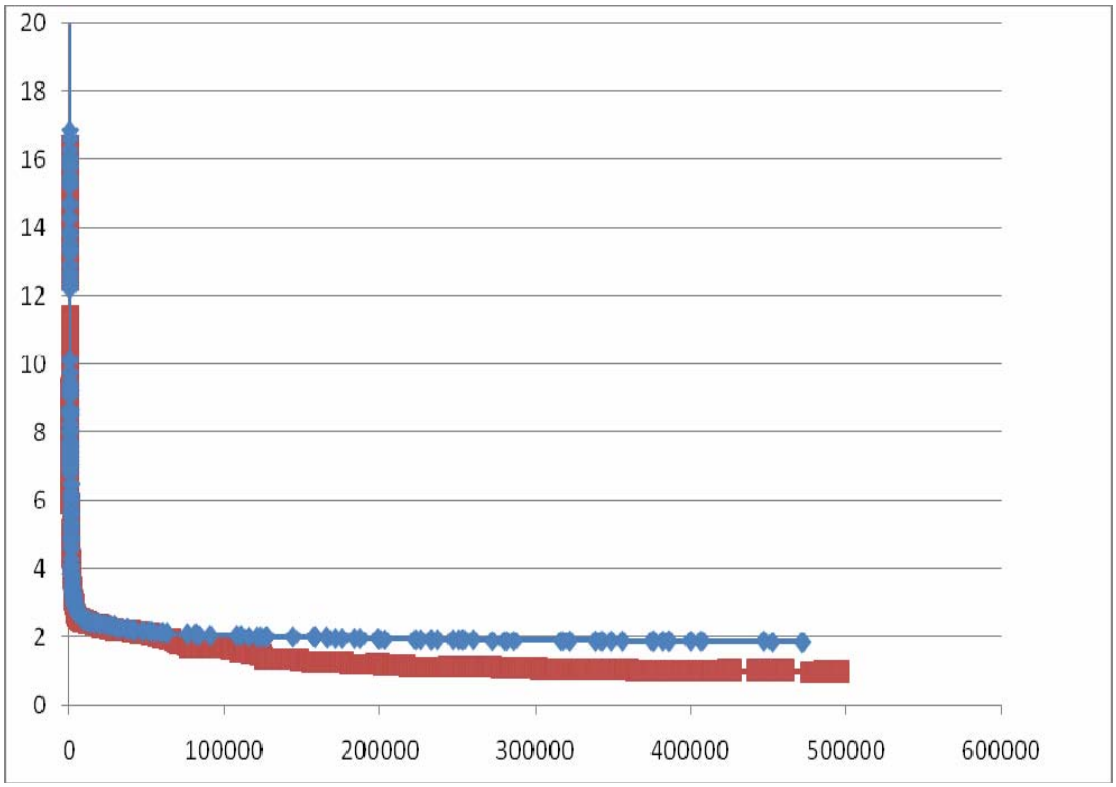
Neste primeiro teste para o QLEP comparamos os resultados com aqueles obtidos por busca aleatória. Consideramos que se ao convergir mais rapidamente que uma busca aleatória o QLEP teria demonstrado que realmente implementa um processo evolutivo.

A busca aleatória usada para testar o QLEP foi realizada simplesmente executando todos os comandos porem sem reajustar as probabilidades do indivíduo quântico. Desta forma, os q-tokens teriam sempre a mesma probabilidade para cada função ou terminal. E assim, o indivíduo simbólico obtido a cada observação do quântico seria totalmente independente do indivíduo simbólico obtido anteriormente. Criamos assim um processo de busca aleatória simples que segue os mesmo parâmetros que o QLEP para solucionar o problema de regressão da função x^2+y^2+3.

B) Interpretação de resultados

Os resultados coletados após este teste não são significantes quanto à competitividade do QLEP frente aos outros modelos evolutivos existentes hoje para PG. O objetivo aqui é reconhecer algum padrão que indique uma convergência e uma aproximação da solução do problema proposto anteriormente.

Ao obter os resultados das experiências rodadas foi feita uma media dos resultados com a ajuda do Excel. Assim obtivemos os dados suficientes para desenhar o seguinte gráfico:



Evolução das soluções apresentadas pelo QLEP (vermelho) e pela busca aleatória (azul).

Como podemos ver a curva referente ao QLEP (curva vermelha) chegou mais perto de uma solução que a busca aleatória (curva azul). Após os 500000 ciclos o QLEP apresentava uma avaliação menor que o processo aleatório. O gráfico traçado em Excel une os pontos que foram obtidos na colheita dos dados. Lembre que a avaliação do indivíduo só era registrada quando ela fosse melhor que a avaliação do indivíduo guardado em memória. Desta forma podemos ver que a curva vermelha é constituída de uma quantidade muito superior de pontos que a curva azul. Isto indica que as melhorias feitas pelo modelo QLEP são feitas com muito mais freqüência que as melhorias obtidas pela busca aleatória.

Note que, ao se tratar de um problema de regressão, quanto mais próxima de zero for a avaliação melhor. A função que avalia este tipo de problema trabalha com um conceito de distancia onde a distancia indica o erro. Por tanto ao se aproximar de zero o erro está diminuindo, o programa está chegando mais perto da solução.

A busca aleatória parece apresentar resultados pouco aleatórios já que ela decresce sem picos, variando de forma parecida com a curva do QLEP. Isto se deve ao fato que apenas são registrados os momentos em que houve uma melhoria da solução. Por tanto é natural que ela decresça sempre e sem picos.

Outro aspecto relevante que pode ser tirado desses resultados é a semelhança entre as curvas. Por serem parecidas, fica a impressão que o QLEP realmente evolui

problemas porem de forma pouco eficiente. É necessário fazer ponderações. Neste teste foram usadas poucas entradas de dados. Quanto mais dados para processar, mais o QLEP é capaz de concertar o erro. Assim ele resolve mais facilmente os problemas. Por outro lado, a busca aleatória tende a ficar cada vez mais longe da resposta ao trabalhar com um volume maior de dados. De certa forma, quanto mais exigente, mais difícil será encontrar uma resposta ao acaso.

Devemos reconhecer também que o QLEP foi usado em seu modo mais simples. Ainda existem ajustes e processos que podem ser adicionados ao modelo para fazer com que ele tenha uma eficiência melhor. Para isso o acompanhamento teórico é constante ao mesmo tempo em que o código é flexível para permitir tanto intervenções pequenas como estruturais.

Novos Ajustes

Já existem alguns ajustes que consideramos. Em alguns modelos parecidos ao QLEP eles já foram aplicados e representaram mudanças significativas na eficiência do programa.

Em sua tese, M. Brameier aborda a influencia que os “introns” exercem no resultado final do programa. Introns são trechos de código que não fazem sentido, eles representam lixo no momento da execução. Dependendo da ordem de certas funções isto pode ocorrer. Sobre tudo ao se tratar de PG onde pequenas mudanças podem anular todo um trecho de código. Para resolver a questão dos introns, Brameier propõe um algoritmo de detecção de introns [2]. Desta forma ele percorre a solução de trás para frente localizando os trechos inoperantes e removendo uma determinada proporção deles. Ele também explica porque não se devem remover todos os introns. Em essência é pelo mesmo motivo que o QLEP usa o qNop. Alguns introns podem ajudar a proteger os trechos de código que são úteis.

É possível que ao introduzir um algoritmo de detecção de introns no QLEP possamos observar algumas melhorias. Mas isso ainda deve ser implementado e testado.

Outra idéia de ajuste é introduzir um princípio de semeadura [6]. Ele consiste em iniciar uma população com indivíduos que já estão mais próximos da solução. Para isso podemos usar uma solução previamente evoluída para iniciar o próximo indivíduo quântico. Se o resultado obtido após todos os ciclos ainda não é satisfatório, podemos iniciar um novo experimento onde o indivíduo quântico começa com seus q-tokens regulados em função do melhor indivíduo do experimento anterior. É possível que isto melhore o desempenho do processo como um todo aproveitando os resultados dos diferentes experimentos. Porem esse princípio pode inibir as possibilidades de diversidade.

Finalmente, existe um terceiro conceito que pode ser importante para o QLEP. É o principio de micro-mutação [2]. Para isso deverá ser introduzido o operador de mutação. Porém, neste caso essas mutações deverão acontecer de forma especifica sobre apenas um q-token. Esse conceito foi criado para LGP e garante que mutações pequenas aplicadas em pequenos trechos do indivíduo tendem a ser mais favoráveis. Com isso é possível incentivar a diversidade do indivíduo sem correr o risco de

transformar grandes trechos. Desta forma é menos provável que o código se torne inoperante, ou simplesmente mais distante da resposta.

Estas são apenas algumas das possibilidades de ajustes para o QLEP. Agora que a capacidade de convergência do modelo foi comprovada, os ajustes e aperfeiçoamentos devem ser estudados com mais detalhe e testados. O modelo já avançou muito desde sua concepção teórica. Alguns aspectos tiveram que ser mudados como, por exemplo, a unidade básica do QLEP que deveria ter sido em essência o q-bit. Desde então já foi criado o q-token abrindo assim um novo modo de implementação. Hoje o modelo já pode confirmar seu aspecto evolutivo. Trabalhamos agora para validar a eficiência e competitividade do QLEP em relação aos outros modelos consagrados presentes no mercado.

Referências:

1 - KOZA, J. R. **Genetic Programming: On the programming of computers by means of natural selection.** Cambridge, MA, MIT Press, 1992.

2 - BRAMEIER, M. **On Linear Genetic Programming.** Dortmund, 2004.

3 - SCHUMACHER, B. **Phys. Rev. A 51, 2738.**

4 - ABS DA CRUZ, A. V. **Algoritmos evolutivos com inspiração quântica para problemas com representação numérica.** PUC-Rio, 2007.

5 - HAN, K., KIM, J. **Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization.** IEEE Transactions on Evolutionary Computation, Vol. 6, n^{o}6, 2002.