TGAlgorithm - Componente delphi para algoritmos genéticos

Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio.
Rio de Janeiro – Brasil
Asiel Aldana Ortiz*,
asiel.aldana89@gmail.com

***Resumo*: Os Algoritmos Genéticos são muito empregados em técnicas de busca e otimização e têm como natureza manter uma analogia à genética natural. Devido a esta característica, sua implementação mantém estruturas em comum nos mais diversos tipos de problemas. Com o objetivo de promover a reusabilidade destas características, o presente trabalho propõe um componente não-visual desenvolvido em linguagens de programação Object Pascal, para Algoritmos Genéticos denominado "TGAlgorithm". Desenvolvido em uma estrutura totalmente orientada a objetos, o Componente explora esta característica para facilitar a sua extensão para algoritmos genéticos e problemas específicos, sejam eles mono-objetivos. Neste trabalho é apresentado um exemplo da utilização do "TGAlgorithm"para uma implementação de um algoritmo genético mono-objetivo.**

Palavras chaves: Algoritmos Genéticos, Componente, Programação.

***Abstract:* Genetic Algorithms are widely used in search and optimization techniques and have the nature to maintain an analogy to natural genetics. Due to this characteristic, its implementation maintains structures in common in the most diverse types of problems. With the objective of promoting the reusability of these characteristics, the present work proposes a non-visual component developed in Object Pascal programming languages, for Genetic Algorithms called "TGAlgorithm". Developed in a fully object-oriented framework, the Component exploits this feature to facilitate its extension to genetic algorithms and specific problems, be they single-purpose. In this work an example of the use of "TGAlgorithm" for an implementation of a genetic algorithm mono-objective is presented.**

Keywords: Genetic Algorithms, Component, Programming.

I. INTRODUÇÃO

Algoritmos Genéticos (GAs - Genetic Algorithms [1]) constituem uma técnica de busca e otimização, altamente paralela, inspirada no princípio Darwiniano de seleção natural e reprodução genética [2]. Os Algoritmos Genéticos utilizam elementos como a sobrevivência dos mais aptos e a troca de informações genética de uma forma estruturada [1]. Embora o Algoritmo Genético use um método heurístico e probabilístico para obter os novos elementos, ele não pode ser considerado uma simples busca aleatória, uma vez que explora inteligentemente as informações disponíveis de forma a buscar novos indivíduos. Os benefícios do uso de um componente, além do reuso, da facilidade de manutenção e da inversão de controle [1], são a capacidade de extensão e a modularidade.

Algoritmos genéticos necessitam de informação do valor de uma função objetivo para cada membro da população, que deve ser um valor não-negativo. Nos casos mais simples, usa-se justamente o valor da função que se quer otimizar. A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ao ambiente ele está, ou seja, suas chances de sobreviver no ambiente e reproduzir-se, passando parte do seu material genético a gerações posteriores.
GAs [3] são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética, que exploram informações históricas para encontrar pontos onde são esperados os melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração. Durante cada iteração, os princípios de seleção e reprodução são aplicados a uma população de candidatos. Através da seleção, se determina quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo seu índice de aptidão. Em outras palavras, os indivíduos com maior adaptação relativa têm maiores chances de se reproduzir.

Este trabalho tem como objetivo, propor um componente [4] para Algoritmos Evolutivos denominado "TGAlgorithm". O modelo apresentado se adequa bem aos sistemas de otimização que utilizam a metaheurística de Algoritmos Genéticos.

II. MATERIAIS E MÉTODOS

Os algoritmos genéticos procuram privilegiar indivíduos com melhores aptidões, com isto tentam dirigir a busca para regiões do espaço de busca onde é provável que os pontos ótimos estejam, o presente projeto foi desenvolvido em Object Pascal, Delphi Rio, aproveitando as vantagens oferecidas pela Programação Orientada a Objetos [5], embora o código seja compatível com versões do Rad Studio superiores ao XE7, a partir das oportunidades oferecidas pelo IDE do Delphi o código exibido a partir do componente pode ser utilizado em aplicativos multiplataforma, Android, iOS, Mac, Windows e nas versões mais recentes do Firemonkey para LINUX.

*****Estudiante de mestrado em Engenheira Eléctrica

O fluxo geral do componente “TGAlgorithm” é ilustrado na Figura 1.

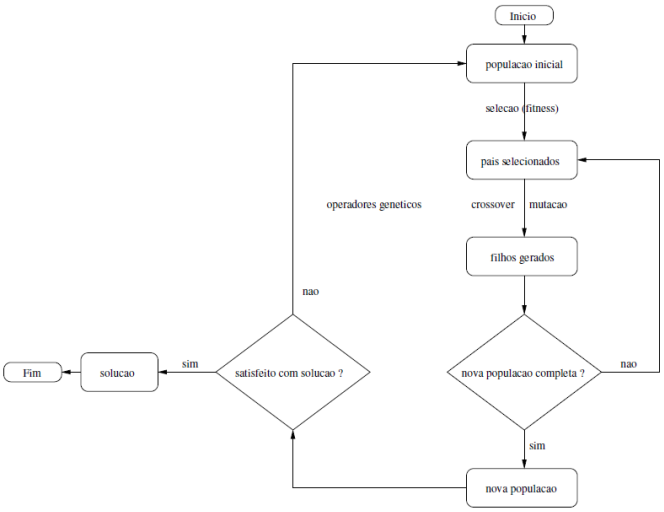

FIGURA NO.1 Fluxo geral do componente “TGAlgorithm”.

INICIALIZAÇÃO:

Como a representação do espaço de busca deve ser a mais sensível possível, a inicialização deste requer algumas ponderações, para este componente, a inicialização escolhida foi a aleatória, onde os os indivíduos da população são gerados de forma aleatória.

AVALIAÇÃO:

Algoritmos genéticos necessitam de informação do valor de uma função objetivo para cada membro da população, que deve ser um valor não-negativo. Nos casos mais simples, usa-se justamente o valor da função que se quer otimizar. A avaliação de cada indivíduo resulta num valor denominado aptidão (**Fitness**).

SELEÇÃO:

Vários métodos de seleção têm sido propostos. A maioria deles procura favorecer indivíduos com maiores valores de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. O TGAlgorithm implementa apenas dois) (roleta e ranking).

CROSSOVER:

O processo de recombinação é um processo sexuado - ou seja, envolve mais de um inivíduo que emula o fenômeno de crossover, a troca de fragmentos entre pares de cromossomos. (Uniforme e N-pontos).

MUTAÇÃO:

O processo de mutação em algoritmos genéticos é equivalente à busca aleatória. Basicamente, seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondentealeatoriamente para um outro possível.

A figura No.2 mostra a estrutura geral do componente “TGAlgorithm”

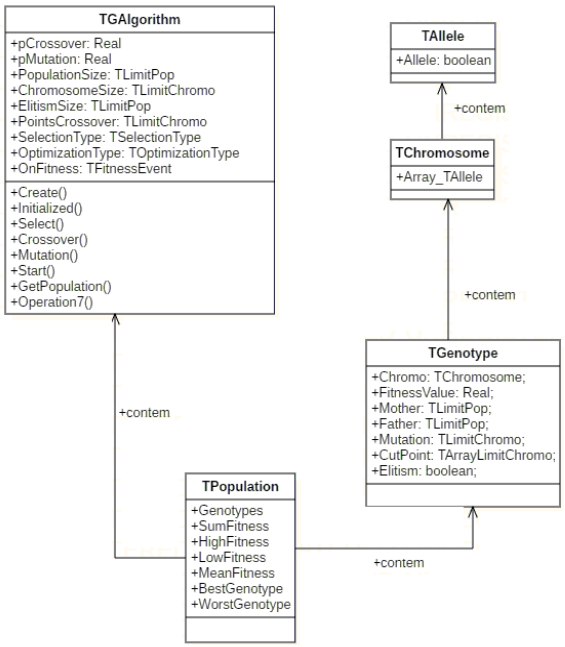

FIGURA NO.2 Estrutura geral do componente “TGAlgorithm”

TGAlgorithm é a principal classe do Componente e sua função está diretamente relacionada ao processo de evolução da população. Esta classe possui os principais parâmetros de controle do algoritmo genético e é responsável pela execução do processo evolutivo. A seguir estão descritos os principais atributos da classe **TGAlgorithm**:

- pCrossover: Probabilidade de Crossover.
- pMutation: Probabilidade de mutação.
- PopulationSize: Tamanho da população

- ChromosomeSize: Tamanho do Cromosomo
- ElitismSize: Número de indivíduos mantidos por Elitism.
- PointsCrossover: Pontos de cruzamento.
- SelectionType: Tipo de seleção.
- OptimizationType: Tipo de otimização
- OnFitness: Função de validação

As funções públicas desta classe de amostra abaixo:

- Create()
- Initialized()
- BinToInt()
- BinToReal()
- ChromoToString()
- Select()
- Crossover()

- Mutation()
- Start()

O elemento mais básico da estrutura é a variável TAllele composta de um campo binário:

TAllele : Boolean

A variável **TChromosome** é definida como uma Arreglo de **TAllelo**.

Uma estrutura **TGenotype** contém uma estrutura TChromosome para identificar cada indivíduo, assim como outros atributos que definem características de um indivíduo em particular. O registro **TPopulation** refere-se a uma matriz de genótipos e armazena as estatísticas de cada população. A figura 3 mostra a fase de inicialização do componente.

Quando o método "Initialized" é verificado Se o método "**OnFitness**" (Ponteiro ao método) tiver sido atribuído a um ponteiro de procedimento, será criada uma população aleatória se ela não existir e o prodecimento "Initialized" e o procedimento de retorno é 1 em caso de sucesso e zero caso contrário

```
with(GA) do
 if(Initialized)then
    begin

        for j := 1 to NGenerations do
          begin

            Start;
            TSeriesA.AddXY(j,GetPopulation.MeanFitness);
          end;

     for i := 1 to GA.PopulationSize do
       begin


          Str := 'Ind' + IntToStr(i) ;
          Series1.Add(GetPopulation.Genotypes[i].FitnessValue,Str);

       end
```

FIGURA NO.3 Mostra a fase de inicialização do componente

PEARTICULARITIES DO COMPONENTE:

- Indivíduos binários
- TSelectionType = (stRoulette, stRank);
- TOptimizationType = (otMaximum, otMinimum);
- MaxPopSize = 1000;
- MaxChromoSize = 100
- OnFitness : TFitnessEvent
- crossover (Uniforme,n)

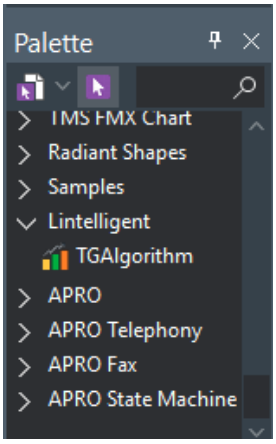

FIGURA NO.4 **TGAlgorithm** em Paleta de componentes Delphi.

Para criar uma instância da classe TGAlgorithm, basta arrastar o componente para a área de trabalho Delphi. Depois que isso é feito, o componente chama o método Create () para inicializar uma instância da classe com seus valores padrão.

III. PROVAS FEITAS

Para realizar os testes no componente, um aplicativo "Optimus" foi implementado sob o sistema operacional Windows, a partir do qual foi definida uma função objetivo para, assim, avaliar como o componente foi capaz de responder às modificações dos parâmetros de ajuste

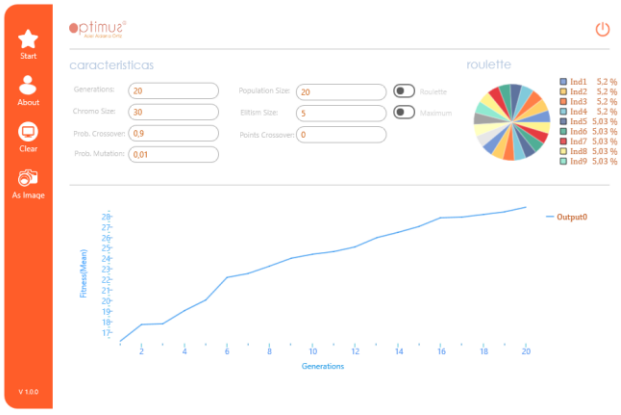

Figura no.5 Aplicação para TESTE, "Optimus"

A função de validação usada é mostrada a continuação, será a soma de todos os 1s encontrados em cada um dos vetores cromossômicos

```
procedure TfrMain.GAFitness(NewChromosome: TChromosome; var
FitnessValue: Real);
var
    i:integer;
    Sum:real;
begin
    Sum:=0;
    for i:=1 to GA.ChromosomeSize do
       if NewChromosome[i] then Sum:=Sum+1;
     FitnessValue:=Sum;

end;
```

Para realizar o teste, os seguintes parâmetros foram introduzidos:

pCrossover= 0,9
pMutation=0,1
PopulationSize=20
ChromosomeSize=30
ElitismSize=5
PointsCrossover=0
SelectionType="Roulette"
OptimizationType="Maximun"

Os resultados obtidos foram os seguintes:

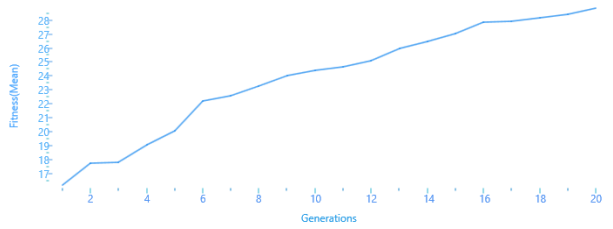

Figura no.6 Fitness Medium versus número de gerações.(Max)

Como pode ser visto para um tamanho de cromossomo igual a 30 que a soma máxima de 1s que o algoritmo pode encontrar é precisamente 30, o gráfico obtido na figura 6 mostra uma convergência do algoritmo em direção a este valor, demonstrando que para este tipo de casos o modelo obtém boas respostas.

Analisando o mesmo exemplo mas desta vez em um problema de minimização, a resposta é a seguinte:

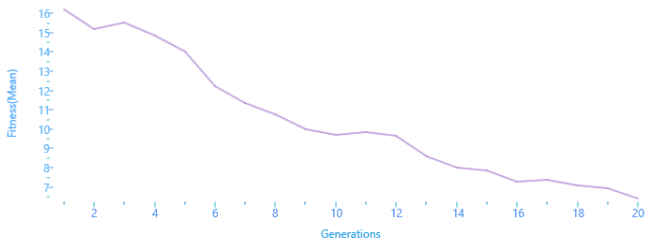

Figura no.7 Fitness Medium versus número de gerações(Min).

Como pode ser visto na figura 7, a convergência da série é em direção a zero quando o sistema é minimizado.

IV. CONCLUSÕES

O Componente TGAlgorithm mostrou-se eficaz no que diz respeito à extensibidade e reusabilidade do código no desenvolvimento de aplicações. Com isto promoveu a simplicidade e a facilidade de manutenção no código das aplicações desenvolvidas. A eficácia do algoritmo implementado foi demonstrada, e recomenda-se incorporar novos tipos de indivíduos, bem como outros algoritmos para mutação e cruzamento.

V. BIBLIOGRAFIA

[1] F. GUIMARÃES e M. RAMALHO, "Implementação de Um Algoritmo Genético.," *UFMG, Belo Horizonte, .*, 2001.

[2] D. GOLDBERG, "Genetic Algorithms in Search Optimization and Machine Learning,," *Addison Wesley* , 1989.

[3] A. SILVA, " Implementação de um Algoritmo Genético utilizando o modelo de Ilhas.," *COPPE – UFRJ, Rio de Janeiro – RJ,* , 2005.

[4] M. C. FAYAD, D. C. SCHMIDT e R. E. JOHNSON, "Building Application Frameworks – Object-Oriented foundations of frameworks design. Estados Unidos:," *Wiley,*, vol. 1999.

[5] P. COAD, " Object-Oriented Patterns. Communications of the ACM,," vol. V. 35, nº nº 9, pp. p 152-159,, setembro 1992..