Solving Cryptarithmetic Problems with a Sequential Algorithm

Alex Matos Iuasse[#1], Windelimarcos Ferreira de Borba[*2]

[#]*Grupo de Pesquisa em Computação Pura e Aplicada,*
Universidade Federal de Mato Grosso
Barra do Garças - MT, Brasil
[1]alexiuasse@gmail.com
[2]windelimarcos@gmail.com

***Abstract*— In our research for solutions of cryptarithmetic problems we have seen that the largest amount of result come from programs with parallel implementations of genetic algorithms. As we have only small problems to solve and we desire to get a detailed point of view of the performance, by itself. Therefore we decided to implement a sequential algorithm to solve smalls well-known problems of cryptarithmetic.**

***Keywords*— Cryptarithmetic, algorithm, sequential, performance.**

➢ Introduction

Criptoaritmética é um criptograma [1] constituido de um problema de aritmética onde os dígitos foram trocados por letras do alfabeto [1]. É um ramo matemático popular por produzir materiais recreativos tais quais quebra-cabeça, enígmas, dentre outros usualmente publicados em jornais e/ou revistas. Solucionar o criptograma significa transpor os caracteres alfabéticos para dígitos novamente.

Nestes quebra-cabeças, cada letra serve para representar única e exclusivamente um dígito distinto e nenhuma palavra (ou sequencia de dígitos) deve começar com uma letra de valor equivalente a zero (0) [2]. Este conceito será detalhado mais abaixo. O exemplo utilizado neste artigo e no algorítimo foi publicado por Henry Dudeney na Strand Magazine em julio de 1924 [1], mostrado na Fig. 1, e ao lado a sua solução, Fig. 2:

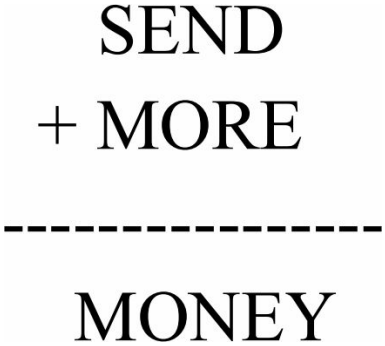

Fig. 1 Exemplo de criptograma baseado em criptoaritmética deveras conhecido, publicado por Henry Dudeney em 1924

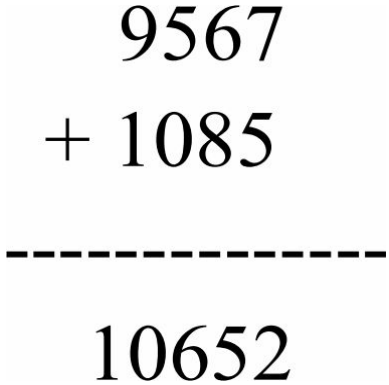

Fig. 2 Solução do exemplo acima

Este artigo tenciona objetivar todos os elementos implementados bem como mostrar, descrever e analisar os resultados obtidos com o algorítimo, desenvolvido pelos alunos Alex Matos Iuasse e Windelimarcos Ferreira de Borba, proposto como solução do trabalho de criptoaritimética da disciplina de Inteligência

[1] um texto cifrado que obedece a um código e uma lógica pré-determinados para decifrar a mensagem.

Artificial do curso de Ciência da Computação da Universidade Federal de Mato Grosso.

O trabalho foi proposto em duas partes tendo a primeira o objetivo de identificar a melhor função de avaliação. As especificações dadas foram as seguintes:

- População inicial de 100 indivíduos
- Número de gerações igual a 100
- Taxa de cruzamento de 60%
- TAxa de mutação de 2%

As funções de avaliação foram especificadas:

- Diferença entre a soma esperada e a obtida no valor total
- Soma das diferenças entre a soma esperada e a obtida dígito a dígito
- Produto das diferenças entre a soma esperada e a obtida dígito a dígito

O método de seleção escolhido foi o de *Seleção por Torneio* com *N* igual a 3 e para a operação de mutação utilizamos o método de permutação.

A segunda etapa do trabalho constituía-se de utilizar a função identificada na primeira etapa para então executar o algorítimo com dezesseis (16) configurações sendo elas uma combinação das especificaões abaixo:

- Tamanho da população 50 ou 500
- Numero de gerações 50 ou 500
- Taxa de cruzamento de 40% ou 80%
- Taxa de mutação de 0% ou 15%
- Tamanho do torneio 2

Uma nota importante para se considerar a partir deste é que, como levantado acima, nenhuma palavra deve começar com 0 pois na ocorrência deste caso nosso criptograma estaria induzindo uma resposta semanticamente incorreta, uma vez que não existe números que começam com 0 exceto os com pontos flutuantes. Cientes disto, deixamos formulado aqui que para o exemplo usado (SEND + MORE = MONEY), utilizando a técnica de codificação onde é excluido as opções que tem 0 como digito referente a uma letra inicial de uma palavra, existe apenas uma solução [3] que é a codificação exibida na Fig. 2. Vale ressaltar que se excluído a verificação explanada acima a quantidade de soluções para o criptograma salta para vinte e quatro (24).

➢ Metodologia

❑ *Formulação do algorítimo*

Processo iniciado, o algorítmo solicita a primeira *string*, ou seja a primeira palavra codificada (no exemplo, SEND), após a mesma ser digitada é solicitado a segunda (MORE, no exemplo) e por fim a *string* resultante (que no caso do exemplo usado para teste é a palavra MONEY). Com todas as palavras necessárias estabelecidas, o algorítimo começa o processo de codificação dos caracteres. Na primeira etapa é engendrado randomicamente os dígitos referentes a cada uma das letras.

Com a população inicial, que foi gerada randomicamente, iremos então executar as 100 gerações que foi estabelecida como regra para que possamos auferir a melhor função de avaliação. Este procedimento ocorre da seguinte forma: para cada função de avaliação, adquirimos o arquivo que contém a população no momento, calculamos sua aptidão na função de avaliação e criamos então um arquivo contendo a avaliação de cada indivíduo da geração. Após, captamos da população 60 cromossomos (taxa de 60% estabelecia previamente) que serão utilizados como fontes para a criação de outros 60 indivíduos (filhos). Este procedimento é executado 10 vezes.

❑ *Processo de análise das funções de avaliação*

Para verificação das funções de avaliação, a cada execução verificamos se foi encontrado o cromossomo que contém o resultado desejado, ou seja se função de avaliação retornou 0 para algum cromossomo. Este procedimento é feito durante 10 vezes. No final de todas as execuções, temos a quantidade total de cromossomos que contém o resultado e em qual geração a população estava quando o mesmo foi obtido.

Por fim decidimos então que a análise das

funções de avaliação seria dado pelo retorno do valor mais precoce alusivo a geração que a população estava quando obitido a solução do criptograma bem como a que retornou a maior quantidade de cromossomos-respostas. Desta forma, a função que retornar o menor valor será então a melhor função, uma vez que precisou de poucas gerações para convergir para a solução desejada.

- *Processo de efetivação do parecer sobre as configurações*

Após um ciclo de execução que é quando o processo executa 10 vezes para cada uma das configurações diferente, temos 16 configurações, logo processo executará 160 vezes, teremos um resultado salvo para cada configuração. No término de um ciclo verifica-se com qual configuração obtivemos maiores quantidades de resultados e qual a geração corrente quando o mesmo foi encontrado.

Assim, em epítome, será retornado a melhor configuração e portanto estará finalizada a tarefa descrita.

➢ Resultados

Os resultados foram sendo obtidos e gravados em um arquivo e então analisados pelos autores do algorítimo. Portanto, todos os valores que serão mostrados aqui podem ser confirmados nos arquivos que seguem em anexo na pasta do programa.

- *Primeira Parte*

Para a primeira parte foram encontrado os seguintes resultados:

Avaliação da função que realiza a diferença entre a soma desejada e a soma real no valor total:

Foram encontrados dos 100.000 (Cem mil) cromossomos gerados, obtidos a partir de 10 execuções com uma população de 100 indivíduos por população durante as 100 gerações, nós obtivemos 26843 indivíduos-respostas que foram encontrados a partir da 2 geração. Ou seja, durante uma execução das dez, na segunda geração alguns cromossomos já convergiram para a resposta. Um notável resultado.

Avaliação da função que realiza o somatório das diferenças entre a soma desejada e a soma real dígito a dígito:

Foram encontrados, nos 100.000 cromossomos gerados nas 10 execuções desta função, 2800 cromossomos respostas. A geração mais prematura onde os mesmo começaram a ser engendrados foi a 3. Ou seja, não só a quantidade de respostas em relação a primeira é consideravelmente menor como também ela demorou mais, ainda que apenas uma, gerações para começar a convergir para o resultado.

Avaliação da função que realiza o produto das diferenças entre a soma esperada e a soma real dígito a dígito:

Esta função, durante a execução que realizamos não encontrou nenhum cromossomo-resultado. Fizemos então a validação da mesma e inferimos que os cálculos realizados pela mesma estavam corretos. Porém fica neste como trabalho futuro a averiguação desta função para entender os possíveis caminhos que levaram até este ponto.

- *Segunda Parte*

A segunda parte foi relativamente mais complicada de se executar a avaliação considerando que os arquivos gerados não contém a avaliação. Abaixo temos os resultados para cada uma das 16 configurações e para ajudar na compreensão, deixamos aqui o que cada termo significa:

  - Primeiro: Tamanho da população
  - Segundo: Número de gerações
  - Terceiro: Taxa de crossover
  - Quarto: Taxa de Mutação
  - Quinto: Como o tamanho do torneio é fixo em 2 não vemos necessidade em colocá-lo.

- *Configuração 50 50 40 0*

Gerados 3910 cromossomos-respostas a partir da 8 geração.

- *Configuração 50 50 80 0*

Gerados 2379 cromossomos-respostas a partir da 10 geração.

- *Configuração 50 50 80 15*

Gerados 2 cromossomos-respostas a partir da 2 geração.

- *Configuração 50 500 40 15*

Gerados 8842 cromossomos-respostas a partir da 4 geração.

- *Configuração 50 500 80 0*

Gerados 1706 cromossomos-respostas a partir da 6 geração.

- *Configuração 500 50 40 0*

Gerados 6569 cromossomos-respostas a partir da 2 geração.

- *Configuração 500 50 40 15*

Gerados 3110 cromossomos-respostas a partir da 1 geração.

- *Configuração 500 50 80 0*

Gerados 9998 cromossomos-respostas a partir da 2 geração.

- *Configuração 500 50 80 15*

Gerados 3073 cromossomos-respostas a partir da 7 geração.

- *Configuração 500 500 40 0*

Gerados 1200240 cromossomos-respostas a partir da 2 geração.

- *Configuração 500 500 40 15*

Gerados 36225 cromossomos-respostas a partir da 2 geração.

- *Configuração 500 500 80 0*

Gerados 1210172 cromossomos-respostas a partir da 1 geração.

- *Configuração 500 500 80 15*

Gerados 42370 cromossomos-respostas a partir da 5 geração.

Com isso terminamos os nosso resultados.

➢ Discussão

Podemos então concluir que o numero de gerações influencia profundamente nos resultados uma vez que todas as configurações que tiveram 500 como número de geração geraram uma quantidade consideravelmente maior de cromossomos-respostas em relação as que tinha apenas 50. Dito isso é necessário também deixar notado que todo o algorítimo esta gerando cromossomos com base aleatória então fica um tanto quanto parcial impor de fato a melhor, pois foi executado apenas uma vez. Ou seja, o correto seria executar os 10 ciclos mais de uma vez para podermos inferir com uma completa certeza.

➢ Considerações Finais

Apesar de levantado na discussão que precisamos executar mais vezes para podermos inferir com certeza absoluta é inegável o fato de que as 10 execuções são suficientes para tirarmos conclusões relacionadas. Desta maneira, podemos dizer que em relação a estas configurações as execuções onde a mesma tinha o número total de geração igual a 500 favoreceram valores muito bons.

Explicitamos novamente o fato de que todo o algorítmo tem por base geração aleatória. Isto pode ser visto olhando os resultados e verificando que alguns conseguiram encontrar o cromossomo-resposta ainda na primeira geração, ou seja, quando gerado aleatório a população inicial, apenas uma execução de cruzamento já foi suficiente para encontrar o cromossomo-resposta.

Logo, podemos dizer que o algorítmo tem uma boa taxa de conversão quando o número de gerações é consideravelmente grande juntamente com o uma população relacionada.

Referências

[1] Abbasian, R. "*Solving Cryptarithmetic Problems Using Parallel Genetic ...*" 2009. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5380487>.

[2] Naoghare, MM, and VM Deshmukh. "*Comparison of parallel genetic algorithm with depth first search algorithm for solving verbal arithmetic problems*." *Proceedings of the International Conference & Workshop on Emerging Trends in Technology* 25 Feb. 2011: 324-329.

[3] http://ken.duisenberg.com/potw/archive/arch97/970710sol.html.