Implementando Algorítimo Genético Firefly com Paralelismo em C++

Danilo Sanchez Tuzita
(danilo_st@hotmail.com)

I. Resumo

Esse é um trabalho que tem como objetivo implementar o algorítimo bioinspirado *Firefly* com paralelismo em *C++* usando a *API OpenMP*.

II. Introdução

Algorítimos bioinsparado estão entre as escolhas mais populares para solução de problemas de otimização. Nesse trabalho foi implementado o algorítimo conhecido como *Firefly Algorithm (FA)*, pela sua afinidade de processamento em paralelo.

III. Teoria

Para o entendimento desse trabalho é necessário conhecimentos básicos de Inteligência Artificial.

IV. Proposta e Implementação

O Algorítimo *Firefly*, desenvolvido por YANG em 2009 [1], inspirou seu algorítimo em uma metáfora imaginando-se um enxame de vaga-lumes dispersos no espaço, onde cada vaga-lume é atraído por vaga-lumes mais brilhantes, sendo tal atração proporcional não apenas ao brilho, mas também à distância entre si. É importante considerar que para a proposta do algorítimo, os vaga-lumes não tem sexo, ou seja, todos os vaga-lumes podem atrair todos os outros vaga-lumes.

O brilho de certo vaga-lume é afetado ou determinado pela forma da função objetivo. Para um problema de maximização, o brilho pode ser apenas a função objetivo por exemplo. Outras formas de brilho podem ser definidas de forma similar à função *fitness* dos algorítimos genéticos.

Com esses conceitos, o Algorítimo *Firefly* pode ser sumarizado pelo pseudo-código 1.

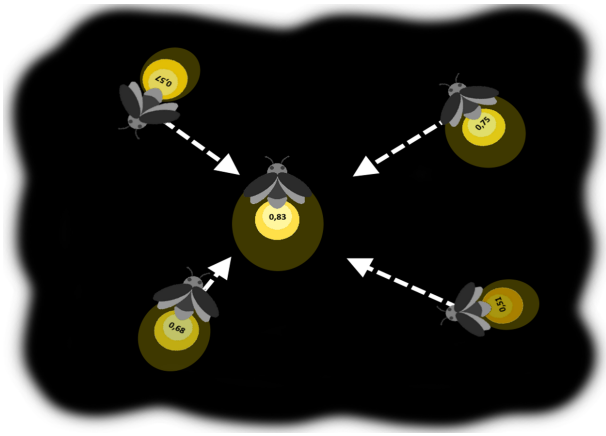

Fig. 1. Representação do comportamento dos vaga-lumes

Algorithm 1 Pseudo-código do Algorítimo *Firefly*

```
Função objetivo: o(x)
Gera vaga-lumes aleatórios: ff
while t < MaxGeneracoes do
    Brilho: I_i de x_i determinado por o(x_i)
    Ranqueia ff pelo brilho
    for i = 1: n todos n vaga-lumes do
        for j = 1: n todos n vaga-lumes do
            if I_j > I_i then
                Move ff_i em direção ff_j
            end if
        end for
    end for
end while
Ranqueia ff pelo brilho
Pós-processa o resultado
```

A. Atratividade e Movimentação

Certo vaga-lume sempre será atraído em direção a um vaga-lume mais brilhante. Porém sua atração depende também de sua distância para tal vaga-lume mais brilhante, sendo quanto mais distante, menos atrativo. Podemos definir a atratividade pela fórmula 1, onde r_{ij} é a distancia euclidiana entre os vaga-lumes i e j, β_{ij} é o coeficiente de atratividade de certo vaga-lume, β_0 é o coeficiente de atratividade para $r = 0$ e γ é o coeficiente de absorção da luz, ou seja o quanto a luz perde intensidade relativo a distância percorrida.

$$\beta_{ij} = \beta_0 \exp(-\gamma r_{ij}^2) \tag{1}$$

$$x_i = x_i + \beta_{ij}\,(x_j - x_i) + \alpha\,(rand - \frac{1}{2}) \tag{2}$$

Com a atração calculada, pode-se mover os vaga-lumes, determinado pela fórmula 2, onde x_i é a posição do vaga-lume i na dimensão x, $rand$ um valor aleatório em $[0, 1]$ e α o coeficiente de aleatoriedade, ou seja, o quanto o vaga-lume i desviará do seu caminho em direção ao vaga-lume j. Esse calculo deve ser feito para todas as dimensões da solução.

B. Paralelismo e Performance

No Algorítimo *Firefly*, muitos desses laços de código podem ser paralelizados, em especial o cálculo do brilho. A cada

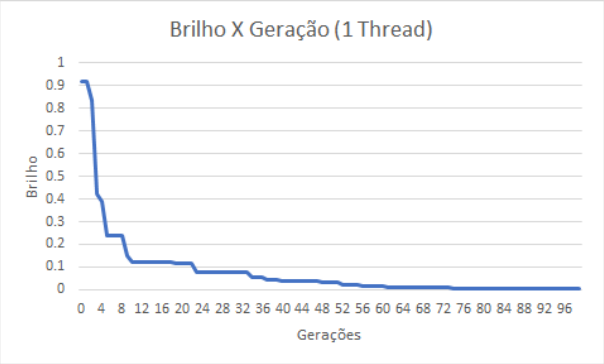

Fig. 2. Comportamento do brilho para execução com uma *thread*

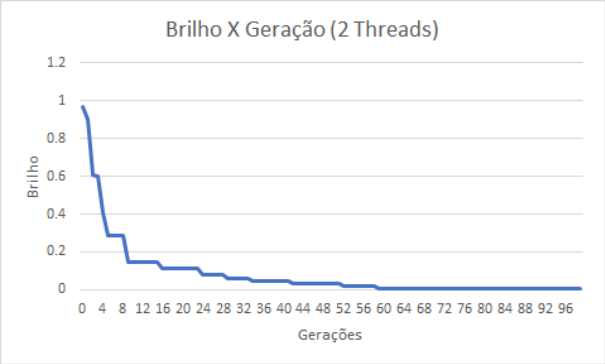

Fig. 3. Comportamento do brilho para execução com duas *thread*

TABLE I
PERFORMANCE DO ALGORÍTIMO VARIANDO A QUANTIDADE DE THREADS

Threads	Tempo médio por Geração	Tempo Total
1	1.685950 s	168.771811 s
2	0.942090 s	94.434781 s
4	0.587915 s	59.005703 s

geração é necessário reavaliar o brilho de cada vaga-lume e por esse processo ser dependente apenas de leitura, esse bloco de código pode ser paralelizado.

V. Experimentos e Resultados

Para testar o método implementado, foi executado o Algorítimo *Firefly* com os seguintes parâmetros:

- **Threads**: Uma, Duas e Quatro *Threads*;
- **Função Objetivo**: Distancia euclidiana para um vaga-lume aleatório que foi gerado antes de ser executado propriamente o Algorítimo *Firefly*;
- **Alpha**: 1.00;
- **Beta0**: 1.00;
- **Gamma**: 0.95;
- **Qtd. Vaga-lumes**: 200;
- **Qtd. Gerações**: 100;
- **Dimensões**: 5;
- **Dominio**: $[0, 5]$;

A Tabela I demonstra a performance do algorítimo para cada uma das execuções de quantidade de *threads*. Podemos observar que mesmo apenas a avaliação do brilho sendo executado em paralelo, a performance é bastante impactada, sendo a execução com 4 *threads* completada em apenas 35% do tempo de processamento para uma única *thread*.

As Figuras 2, 3 e 4, demonstra o comportamento do brilho ao longo das gerações para cada uma das execuções, uma observação importante sobre as figuras é que o brilho diminuí a cada iteração, isso se dá à uma decisão de implementação, onde era mais fácil lidar com o brilho tendendo a zero do que tendendo ao infinito.

Podemos observar um comportamento logarítmico nas curvas das figuras, onde o brilho incia alto e caí bruscamente nas próximas iterações até se estabilizar próximo de zero, um resultado que é esperado para esse algorítimo com essa função objetivo.

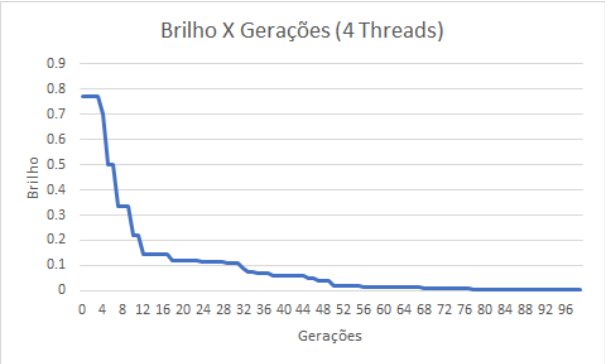

Fig. 4. Comportamento do brilho para execução com quatro *thread*

VI. Conclusão

O Algorítimo *Firefly* é Nesse trabalho foi implementado e validado o Algorítimo *Firefly* de forma paralela. Devemos destacar a assertividade do algorítimo. Em todas as suas execuções em cerca de vinte iterações o brilho já estava próximo de 10^{-1} e finalizando próximo de 10^{-3}, ou seja a distância euclidiana do melhor vaga-lume para o vaga-lume alvo é de aproximadamente 10^{-3} num domínio de 5 dimensões de $[0, 5]$, que é impressionante para o pouco tempo de processamento.

References

[1] X.-S. Yang, “Firefly algorithms for multimodal optimization,” in *SAGA*, 2009.