

Morcego em GPU

Contextualização

Aplicar o algoritmo do morcego em GPU.

Separar a população paralelamente (DAO, 2015).

C CUDA 8 na Amazon GRID K520.

Algoritmos em GPU

- ▶ Sim: Bactérias
- ▶ Sim: Busca da harmonia
- ▶ Sim: Celular automata
- ▶ Sim: Computação de membrana
- ▶ Sim: Cuco
- ▶ Sim: Formiga
- ▶ Sim: Genéticos
- ▶ Sim: Partículas
- ▶ Sim: Procura por alimentos de abelhas
- ▶ Sim: Pulo do Sapo
- ▶ Sim: Redes Neurais
- ▶ Sim: Vaga-lume
- ▶ Sim: Acasalamento de abelhas
- ▶ Sim: Braimstorming process
- ▶ Sim: CMAE
- ▶ Não: Morcego
- ▶ Não: Polinização de Flores NC
- ▶ Não: Sistema Imunológico Artificial
- ▶ Não: Mosquito
- ▶ Não: Tubarão
- ▶ Não: Vespa
- ▶ Não: Infestação de baratas

Modelagem

- ▶ Agente: Morcego
- ▶ Medida de desempenho: Maior/menor posição já encontrada no espaço dimensional
- ▶ Ambiente: Espaço N dimensional
- ▶ Sensores: Som
- ▶ Atuadores: Frequência, amplitude de onda
- ▶ Observável: Parcial
- ▶ Determinístico/estocástico: Estocástico
- ▶ Estático ou dinâmico: dinâmico
- ▶ Episódico/sequencial: Sequencial
- ▶ Discreto ou contínuo: Contínuo
- ▶ Multiagente: Sim

Parâmetros

- ▶ Alfa: amplitude (probabilidade de valores ruins entrarem), 0.9, 0.5
- ▶ Beta: taxa de diminuição da amplitude
- ▶ Lambda: taxa de pulsos (probabilidade de busca local) 0.9, 0.1
- ▶ População: 40

Variáveis

- ▶ Frequência: aleatória (Varia as chances de intensificação e diversificação)
- ▶ Taxa de pulso: Aumenta com o tempo (Aumenta as chances de busca local)
- ▶ Amplitude: Diminui conforme vai se aproximando de um valor ótimo

Regras

Conforme o tempo vai passando somente as melhores soluções globais vão permanecendo e a probabilidade de busca local acontecer vai aumentando.

Quão mais longe do GBEST maior a velocidade de movimentação.

Pseudo Código

Os morcegos iniciam em posições aleatórias.

Enquanto o critério de parada não for atingido itera sobre todos os morcegos.

Para cada morcego

- ▶ Gera-se uma frequência aleatória
- ▶ Gera-se uma velocidade inversamente proporcional a distancia do gbest utilizando a frequência gerada
- ▶ Gera-se a nova posição utilizando a posição atual mais velocidade.
- ▶ Conforme a taxa de emissão aumenta a probabilidade de fazer busca local é maior
- ▶ Na busca local pode-se efetuar um random walk considerando a média de amplitude dos agentes
- ▶ Troca a posição do morcego se a temporária é melhor ou um fator estocástico é obedecido
- ▶ Atualiza o melhor morcego

Pseudo Código Original

Bat Algorithm

Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, \dots, x_d)^T$

Initialize the bat population \mathbf{x}_i ($i = 1, 2, \dots, n$) and \mathbf{v}_i

Define pulse frequency f_i at \mathbf{x}_i

Initialize pulse rates r_i and the loudness A_i

while ($t < \text{Max number of iterations}$)

Generate new solutions by adjusting frequency,

and updating velocities and locations/solutions [equations (2) to (4)]

if ($\text{rand} > r_i$)

Select a solution among the best solutions

Generate a local solution around the selected best solution

end if

Generate a new solution by flying randomly

if ($\text{rand} < A_i$ & $f(\mathbf{x}_i) < f(\mathbf{x}_*)$)

Accept the new solutions

Increase r_i and reduce A_i

end if

*Rank the bats and find the current best \mathbf{x}_**

end while

Postprocess results and visualization

Figure 1:pseudo-code.png

Pseudo Código Jelson's

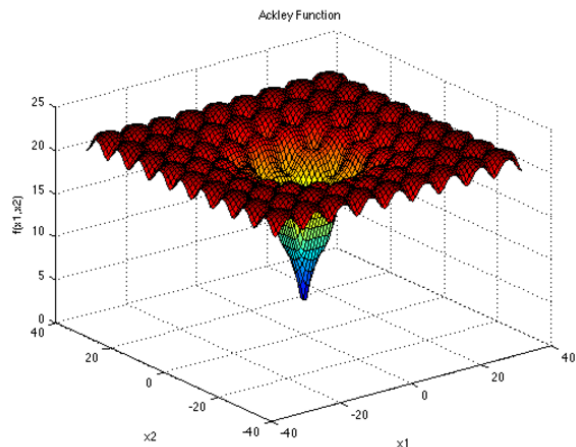
```
1: Parâmetros:  $n, \alpha, \lambda$ 
2: Inicializa morcegos  $\vec{x}_i$ 
3: Avalia  $f(\vec{x}_i)$  para todos os morcegos
4: Atualiza melhor morcego  $\vec{x}_*$ 
5: while critério de parada não atingido do
6:   for  $i = 1$  to  $n$  do
7:      $f_i = f_{min} + (f_{max} - f_{min})\beta, \beta \in [0, 1]$ 
8:      $\vec{v}_i^{t+1} = \vec{v}_i^t + (\vec{x}_i^t - \vec{x}_*^t)f_i$ 
9:      $\vec{x}_{temp} = \vec{x}_i^t + \vec{v}_i^{t+1}$ 
10:    if  $rand < r_i, rand \in [0, 1]$  then {Faz busca local}
11:       $\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1, 1]$ 
12:    end if
13:    Realiza perturbação em uma dimensão de  $\vec{x}_{temp}$ 
14:    if  $rand < A_i$  or  $f(\vec{x}_{temp}) \leq f(\vec{x}_i), rand \in [0, 1]$  then {Aceita solução temporária}
15:       $\vec{x}_i = \vec{x}_{temp}$ 
16:       $r_i^{t+1} = 1 - exp(-\lambda t)$ 
17:       $A_i^{t+1} = \alpha A_i^t$ 
18:    end if
19:    Atualiza melhor morcego  $\vec{x}_*$ 
20:  end for
21: end while
22: Pós-processamento
```

Figure 2:pseudo-code-v2.png

Benchmarks

Funções de Benchmarks

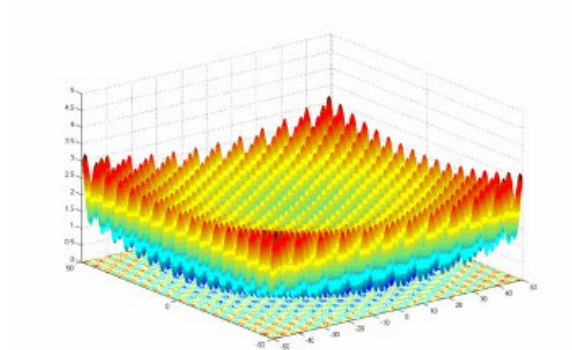
- ▶ Ackley
- ▶ Griewank
- ▶ Rastrigin
- ▶ Sphere



$$f(\mathbf{x}) = -a \exp \left(-b \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2} \right) - \exp \left(\frac{1}{d} \sum_{i=1}^d \cos(cx_i) \right) + a + \exp(1)$$

Figure 3:ackley.png

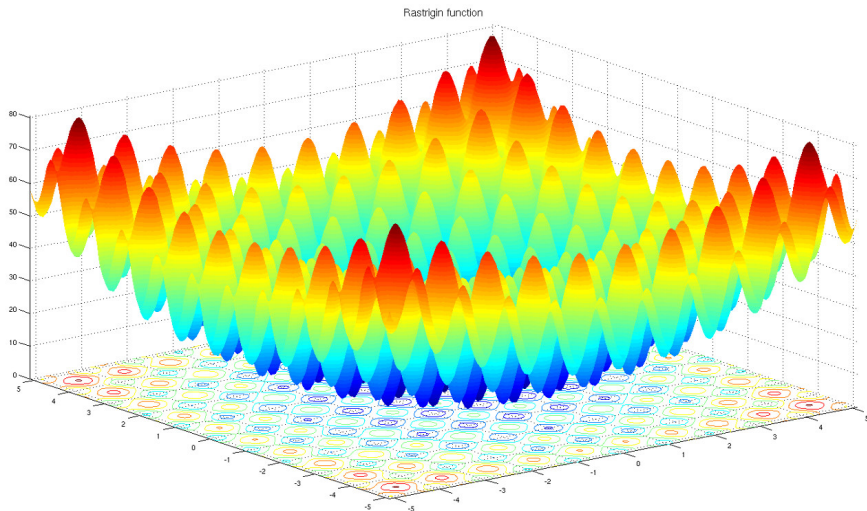
Griewank



$$f_n(x_1, \dots, x_n) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Figure 4:griewank.jpg

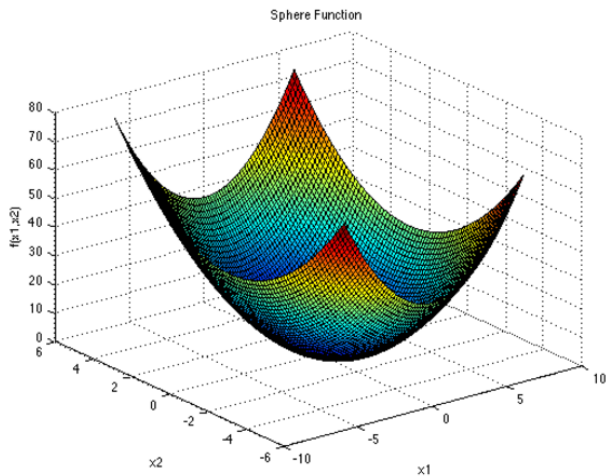
Rastrigin



$$f(\mathbf{x}) = An + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)]$$

Figure 5:rastringin.jpg

Sphere



$$f(\mathbf{x}) = \sum_{i=1}^d x_i^2$$

Figure 6:sphere.png

Benchmarks Yang's 2010

- ▶ Ackley
- ▶ De Jong
- ▶ Easoms
- ▶ Eggcrate
- ▶ Griewank
- ▶ Michalewicz
- ▶ Rastrigin
- ▶ Rosenbrocks
- ▶ Schewefels
- ▶ Schuberts

Benchmarks Jelson's

- ▶ Ackley
- ▶ Griewank
- ▶ Rastrigin
- ▶ Sphere

Benchmarks Adis

- ▶ Ackley
- ▶ Griewank
- ▶ Rastrigin
- ▶ Rosenbrok
- ▶ Sphere

Benchmarks Li

- ▶ Ackley
- ▶ Eliptic
- ▶ Rastrigin
- ▶ Rosenbrocks
- ▶ Schwefel
- ▶ Sphere

Referências

- ▶ Bat Algorithm: An Overview and it's Applicaions, S Induja, International Journal of Advanced Research in Computer and Communication Engineering, 2016
- ▶ A New Metaheuristics Bat-Inspired Algorithm, Xin-She Yang, Department of Engineering, Cambridge, 2010
- ▶ Análise de Sensibilidade dos Parâmetros do Bat Algorithm e Comparação de Desempenho, Jelson A. Cordeiro, Rafael Stubs Parpinelli, Heitor Silvério Lopes
- ▶ Parallel bat algorithm for optimizing makespan in job scheduling problems, Thi-Kien Dao, Tien-Szu Pan, Trong-The Nguyen, Jeng-Shyang Pan, 2015, Springer Science Review
- ▶ Improved Hibridized Bat algorithm for Global Numerical Optimization, Adis Alihodzic, Milan Tuba, 2014, International Conference on Computer Modeling and simulation
- ▶ Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization, Xiaodong Li, Ke Tang, Mohammad N. Omidvar, Zhenyu Yang, Kai Qin, 2013