Bat Optimization on GPU

Jean Carlo Machado
Univerisdade do Estado de Santa Catarina
Computer Engineering
UDESC
Santa Catarina, Joinville,
Email: contato@jeancarlomachado.com.br

***Abstract*—This works presents the Bat in GPU. The results show that...**

I. Introduction

The bat algorithm was introduced by [1]. It uses the inspiration of microbas to look at their preys.

Many populational optimization algorithms can benefit from paralization.

This work attemps to investigate the applicability of GPU parallization on the bat algorithm. Previously some demonstrations of the bat algorithm paralelized on CPU were presented (reference referece), however, til the day of this publication no implementation of the bat algorithm was found on GPU.

It was developed two versions of the algorithm. One that runs on CPU and the other which uses CPU.

mds December 21, 2016

II. Bat Design on CPU

In this work the bath algorithm used was the one proposed by [2], since it represents a concrete demonstration of how the bat metha-heuristic.

The CPU version was single threaded.

The random algorightm used was the mersenne twister.

A. Pseudo-code

```
Parameters:$n, \alpha\ \lambda$
Initialize Bats
Evaluate fitness bats
Selects best
while stop criteria false do
    for i=1 to n do
        $f_i = f_{min} + (f_{max} - f_{min})\beta, \in \beta[0,1]$
        $\vec{v}_i^{t+1} = \vec{v}_i^t + (\vec{x}_i^t + \vec{x}_*^t) f_i$
        $\vec{x}_{temp} = \vec{x}_i^t + \vec{v}_i^{t+1}$
        if $rand < r_i, rand \in [0,1]$then local search
            $\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1,1]$        (1)
        end if
        Single dimension perturbation in$x_{temp}$
        if$a < A_i^t, a \in [0,1]$
            $\vec{x}_i^t = \vec{x}_{temp}$
            $r_i = exp(\lambda * i)$
            $A_i = A_0 * \alpha^i$
        end if
        Selects best
    end for
end while
```

III. Bat Design on GPU

Since the BAT agorithm uses a population of bats, the most intuitive parallization method to apply on it is to use each bat on a GPU core. [3] used a similar method for a GPU implementation for the PSO algorithm. For the GPU version the approach used was the split of each individual in one thread.

A. Pseudo-code GPU

```
Parameters:$n, \alpha\ \lambda$
Initialize bats assicrously
syncronize threads
Evaluate fitness bats
Selects best
**while** stop criteria false **do**
    for each thread $i$
        $f_i = f_{min} + (f_{max} - f_{min})\beta, \in \beta[0, 1]$
        $\vec{v}_i^{t+1} = \vec{v}_i^t + (\vec{x}_i^t + \vec{x}_*^t)f_i$
        $\vec{x}_{temp} = \vec{x}_i^t + \vec{v}_i^{t+1}$
        **if** $rand < r_i, rand \in [0, 1]$**then** local search
            $\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1, 1]$        (2)
        **end if**
        Single dimension perturbation in$x_{temp}$
        **if**$a < A_i^t, a \in [0, 1]$
            $\vec{x}_i^t = \vec{x}_{temp}$
            $r_i = exp(\lambda * i)$
            $A_i = A_0 * \alpha^i$
        **end if**
        syncronize threads
        Selects best
    **end for**
**end while**
```

IV. Experiments

The benchmark functions used were the following:

- Ackley
- Griewank
- Rastringin
- Rosenbrook

The experiments were executed on a machine with the following configuration:

Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz GK208 GeForce GT 720 1024 MB of vram

Each experiment runned a total of 1x10^4 $times$.

V. Results

VI. Conclusion

It was observed speedups with big populations. The original BAT was proposed with 40 individuals and the speedups was seen with 250 individuals. The advantages of the algorithm may be tested against a CPU implementation to be fair. With this work it's clear that is possible to speedup the bat methaueristic using GPU but the best results are only achieved by a great population size.

VII. Further works

It may be explored the usage of blocks as representation for the dimensions in which each bat deals.

TABLE I
Experiments

Name	Function	Dimensions	Agents
E1	Ackley	100	256
E2	Ackley	100	768
E3	Griewank	100	256
E4	Griewank	100	768
E5	Rastringin	100	256
E6	Rastringin	100	768
E7	Rosenbrook	100	256
E8	Rosenbrook	100	768

TABLE II
Results

Name	Total CPU	Total GPU	Speedup
E3	1m4.888s	0m55.439s	1.16x
E4	2m27.902s	0m21.976s	7x

Acknowledgment

The authors would like to thank...

References

[1] Xin-She Yang *A New Metaheuristics Bat-Inspired Algorithm.* Department of Engineering, Cambridge, 2010.

[2] Jelson A. Cordeiro, Rafael Stubs Parpinelli Heitor Silvrio Lopes *Anlise de Sensibilidade dos Parmetros do Bat Algorithm e Comparao de Desempenho.*

[3] PSO-GPU: Accelerating Particle Swarm Optimization in CUDA-Based Graphics Processing Unit