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*—The abstract goes here.**

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.

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

$$
\begin{array}{r}
\text{Parameters:} n, \alpha\, \lambda \\
\text{Initialize Bats} \\
\text{Evaluate fitness bats} \\
\text{Selects best} \\
\textbf{while} \text{ stop criteria false } \textbf{do} \\
\textbf{for } \text{i=1 to n } \textbf{do} \\
f_i = f_{min} + (f_{max} - f_{min})\beta, \in \beta[0,1] \\
\vec{v}_i^{t+1} = \vec{v}_i^t + (\vec{x}_i^t + \vec{x}_*^t) f_i \\
\vec{x}_{temp} = \vec{x}_i^t + \vec{v}_i^{t+1} \\
\textbf{if } rand < r_i, rand \in [0,1] \textbf{then} \text{ local search} \\
\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1,1] \\
\textbf{end if} \\
\text{Single dimension perturbation in} x_{temp} \\
\textbf{if} a < A_i^t, a \in [0,1] \\
\vec{x}_i^t = \vec{x}_{temp} \\
r_i = exp(\lambda * i) \\
a_i = a_0 * \alpha^i \\
\textbf{end if} \\
\text{Selects best} \\
\textbf{end for} \\
\textbf{end while}
\end{array}
\tag{1}
$$

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. [**?**] used a similar method for a GPU implementation for the PSO algorithm.

A. Pseudo-code GPU

$$
\begin{aligned}
&\text{Parameters:}n, \alpha\, \lambda \\
&\text{Initialize bats assicrously} \\
&\text{syncronize threads} \\
&\text{Evaluate fitness bats} \\
&\text{Selects best} \\
&\textbf{while} \text{ stop criteria false } \textbf{do} \\
&\text{for each thread } i \\
&f_i = f_{min} + (f_{max} - f_{min})\beta, \in \beta[0,1] \\
&\vec{v}_i^{t+1} = \vec{v}_i^t + (\vec{x}_i^t + \vec{x}_*^t)f_i \\
&\vec{x}_{temp} = \vec{x}_i^t + \vec{v}_i^{t+1} \\
&\textbf{if } rand < r_i, rand \in [0,1] \textbf{then} \text{ local search} \\
&\vec{x}_{temp} = \vec{x}_* + \epsilon A_m, \epsilon \in [-1,1] \\
&\textbf{end if} \\
&\text{Single dimension perturbation in} x_{temp} \\
&\textbf{if} a < A_i^t, a \in [0,1] \\
&\vec{x}_i^t = \vec{x}_{temp} \\
&r_i = exp(\lambda * i) \\
&a_i = a_0 * \alpha^i \\
&\textbf{end if} \\
&\text{syncronize threads} \\
&\text{Selects best} \\
&\textbf{end for} \\
&\textbf{end while}
\end{aligned} \tag{2}
$$

For the GPU version the approach used was the split of each individual in one thread.

The benchmark functions used were the following:

ROSENBROOK, SPHERE, SCHWEFEL, ACKLEY, RASTRINGIN, GRIEWANK, SHUBER

IV. EXPERIMENT

A. Experiment 1

- **Function:** Griewank
- **Bats:** 256
- **Iterations:** 10000

B. Experiment 2

- **Function:** Griewank
- **Bats:** 768
- **Iterations:** 10000

V. RESULTS

A. Device details

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

B. Experiment 1

- **CPU:** 1m4.888s
- **GPU:** 0m55.439s

Speedup: 1.16x

C. Experiment 2

- **CPU:** 2m27.902s
- **GPU:** 0m21.976s

Speedup: 7x

mds
August 26, 2015

D. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

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.

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*.