Flower Pollination Algorithm

Submitted by : - Gitesh Katre (192CD009)

Ritesh Nikhoriya(192CD020)

1)Objective :-

To minimize the below problem using the flower pollinaton optimization algorithm.

$f(x1\cdots xn) = \sum X^{2}_{i}$

$-100.0 \leq X_i \leq 100.$

2)Introduction

Flower pollination algorithm is a metaheuristic algorithm that was developed by Xin-She Yang, based on the pollination process of flowering plants.

Assumptions

Biotic and cross-pollination is considered as a global pollination process with pollen carrying pollinators performing Levy flights.

Abiotic and self-pollination are considered as local pollination.

Flower constancy can be considered as the reproduction probability is proportional to the similarity of two flowers involved.

Local and global pollination are controlled by a switch probability in [0,1] . Due to the physical proximity and other factors such as wind, local pollination can have a significant fraction q in the overall pollination activities.

These rules can be translated into the following updating equations:

{\displaystyle x_{i}^{t+1}=x_{i}^{t}+L(x_{i}^{t}-g_{*})}{\displaystyle x_{i}^{t+1}=x_{i}^{t}+L(x_{i}^{t}-g_{*})}

{\displaystyle x_{i}^{t+1}=x_{i}^{t}+\epsilon (x_{i}^{t}-x_{k}^{t})}{\displaystyle x_{i}^{t+1}=x_{i}^{t}+\epsilon (x_{i}^{t}-x_{k}^{t})}

where {\displaystyle x_{i}^{t}}{\displaystyle x_{i}^{t}} is the solution vector and {\displaystyle g_{*}}{\displaystyle g_{*}} is the current best found so far during iteration. The switch probability between two equations during iterations is {\displaystyle p}p. In addition, {\displaystyle \epsilon }\epsilon is a random number drawn from a uniform distribution. {\displaystyle L}L is a step size drawn from a Lévy distribution.

Lévy flights using Lévy steps is a powerful random walk because both global and local search capabilities can be carried out at the same time. In contrast with standard Random walks, Lévy flights have occasional long jumps, which enable the algorithm to jump out any local valleys. Lévy steps obey the following approximation:

{\displaystyle L\sim {\frac {1}{s^{1+\beta }}},}{\displaystyle L\sim {\frac {1}{s^{1+\beta }}},}

where {\displaystyle \beta }\beta is the Lévy exponent.[48] It may be challenging to draw Lévy steps properly, and a simple way of generating Lévy flights {\displaystyle s}s is to use two normal distributions {\displaystyle u}u and {\displaystyle v}v by a transform[49]

{\displaystyle s={\frac {u}{|v|^{1+\beta }}},}{\displaystyle s={\frac {u}{|v|^{1+\beta }}},}

with

{\displaystyle u\sim N(0,\sigma ^{2}),\quad v\sim N(0,1),}{\displaystyle u\sim N(0,\sigma ^{2}),\quad v\sim N(0,1),}

where {\displaystyle \sigma }\sigma is a function of {\displaystyle \beta }\beta .

3) Flower pollination optimization algorithm

a)Standard algorithm

Flower Pollination Algorithm (or simply Flower Algorithm)

Objective min *or* max $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, ..., x_d)$
Initialize a population of n flowers/pollen gametes with random solutions
Find the best solution $\mathbf{g}_*$ *in the initial population*
Define a switch probability $p \in [0, 1]$
while *(t <MaxGeneration)*
  for $i = 1 : n$ *(all n flowers in the population)*
    if *rand* $< p$,
      Draw a (d-dimensional) step vector L which obeys a Lévy distribution
      Global pollination via $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + L(\mathbf{g}_* - \mathbf{x}_i^t)$
    else
      Draw ϵ *from a uniform distribution in [0,1]*
      Randomly choose j and k among all the solutions
      Do local pollination via $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \epsilon(\mathbf{x}_j^t - \mathbf{x}_k^t)$
    end if
    Evaluate new solutions
    If new solutions are better, update them in the population
  end for
    Find the current best solution $\mathbf{g}_*$
end while

· Pseudo code of the proposed Flower Pollination Algorithm (FPA).

b) Parallel algorithm

Flower pollination algorith for parallel execution with the help of CUDA

Flower Pollination Algorithm

Objective min or max *f (x),* **x =**

Initialize a population of n flowers/pollen gametes with random solutions

Find the best solution **g,,** *in the initial population*

Define a switch probability p E 10,1]

while *(t <MaxGeneration)*

  for i = 1 : n ***(all*** *n flowers in the population)*

    if **rand < p,**

      Draw *a (d-dimensional) step vector L which obeys a Levy distribution*

      Global pollination via $x_{id}^{t+1} = x^{t+1}_{id} + L(g^* - x^{t}_{id})$

    else

      Draw *c from* **a** *uniform distribution in 10,11*

      Randomly choose j and k among all the solutions

      Do local pollination via $x_{id}^{t+1} = x^{t+1}_{id} + e(x^{t}_{id} - x^{t}_{id})$

    end *if*

    syncthreads();

    Evaluate new solutions

    If new solutions are better, update them in the population

  end for

    Find the current best solution **g,**

end while

if f(~ Xtid) == best fitness then

  best solution = ~ Xtid;

  end

4) Experiment and results

1) GPU information

Table 1: GPU information

— General Information for device 0 —	
Name	GeForce GTX 680
Compute capability	3.0
Clock rate	1058500
Device copy overlap	Enabled
Kernel execution timeout	Disabled
— Memory Information for device 0 —	
Total global mem	2095382528
Total constant Mem	65536
Max mem pitch	2147483647
Texture Alignment	512
— MP Information for device 0 —	
Multiprocessor count	8
Shared mem per mp	49152
Registers per mp	65536
Threads in warp	32
Max threads per block	1024
Max thread dimensions	(1024, 1024, 64)
Max grid dimensions	(2147483647, 65535, 65535)

2) CPU information

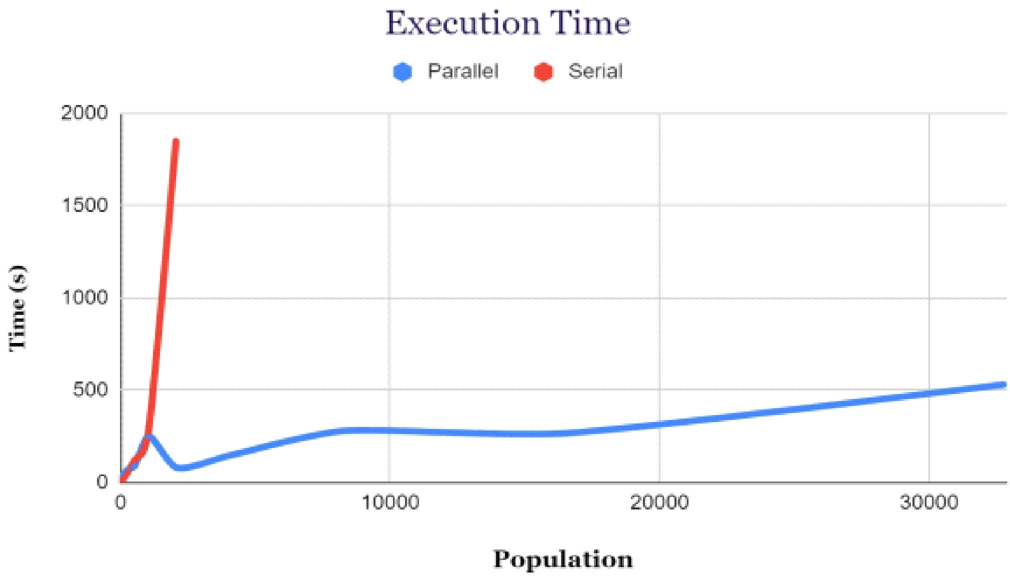
Table 2: CPU information

vendor_id	GenuineIntel
cpu family	6
model	63
model name	Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz
stepping	2
microcode	0x36
cpu MHz	3599.941
cache size	15360 KB

3) Scalability

Population	Parallel	Serial
64	29.025	13.581
128	42.97	27.348
256	70.419	56.59
512	89.634	120.905
1024	251.241	273.922
2048	82.411	1851
4096	149.026	NA
8192	279.417	NA
16384	266.923	NA
32768	531.6	NA

4) Speed up graph



5) Conclusions

In this project, we implement a Parallel Standard Flower pollination Optimization Algorithm. For this, we use various concepts of CUDA parallel programming, such as the use of shared memory, thread synchronization, and atomic operations. We start with simple tests to evaluate and validate theimplementations. After validate that both CPU and GPU implementations are equivalent, we evaluate the execution time using shared memory and without use it. The version using shared memory was faster than the previous one. After those initial experiments, we start exploiting the capabilities of the CUDA device and also the characteristics of the problem. Flower pollination algorithm is a metaheuristic algorithm , due to that it needs to be executed several times to avoid the effect of randomness to the result. With this in mind, instead of creating a code that implements one execution of FPA in parallel (and run it several times), we created an implementation capable of executing several FPA at once in parallel (using several independent blocks), each one also parallels inside their blocks.Also, it was analyzed the scalability in terms of population size. It was found that as higher the population size, better is the quality of the solutions. Also, using the proposed approach,it was possible to increase the population size 5 times. After population reaches 2048,so the serial code takes huge time ,therefore no further operations are there but fortunately given data is enough to draw conclusions.In a nutshell for very small population GPU has more overhead ,but yielding comparatively bad results. But as the population increases the problem shows speedup in case of GPU and retardation in case of CPU.so once it reaches to 2048 when the CPU can't solve it in an hour on the other hand GPU shows its potential of massively parallel architecture.And thus a sudden drop in execution time as memory is being efficiently handled after those iterations.

References

Xin-She Yang, Flower pollination algorithm for global optimization, in: Unconventional Computation and Natural Computation 2012, Lecture Notes in Computer Science, Vol. 7445, pp. 240-249 (2012).