PERBANDINGAN ALGORITMA BRUTE FORCE DENGAN ALGORITMA GENETIK MENGGUNAKAN PENDEKATAN KOMPUTASI KINERJA TINGGI

Erqy Sara Quartian[1], Fhira Nhita[2], Fitriyani[3]

[1,2,3]Prodi Ilmu Komputasi, Fakultas Informatika, Universitas Telkom
Jalan Telekomunikasi 1, Dayeuh Kolot Bandung 40257 Indonesia
erqyquartian@gmail.com[1], fhiranhita@telkomuniversity.ac.id[2], fitriyani@telkomuniversity.ac.id[3]

ABSTRAK

Kemajuan yang terjadi pada bidang komputasi semakin terlihat dengan dikembangkannya berbagai penemuan dalam bidang teknologi, terutama untuk permasalahan yang menggunakan data yang besar sebagai perhitungan, sehingga apabila dilakukan perhitungan secara serial akan cukup memakan waktu. Kerja komputer secara parallel sangatlah mempengaruhi kinerja dari komputasi itu sendiri. Dengan *coding* yang efektif, maka pekerjaan komputer secara parallel akan memangkas waktu perhitungan algoritma secara signifikan. Penelitian tugas akhir ini akan membahas bagaimana perbandingan yang dihasilkan dari dua algoritma, yaitu Algoritma Genetika (AG) baik secara serial maupun parallel dan Algoritma Brute Force (BF) dengan menggunakan bahasa C yang terdapat pada arsitektur CUDA yang diterapkan pada studi kasus TSP untuk 101 kota. Keunggulan dari penggunaan AG adalah kemampuannya untuk menyimpan individu terbaik dari sekian perulangan yang telah ditentukan sehingga akan didapat individu terbaik pada akhir perhitungan. AG Serial menghasilkan performansi nilai jarak terpendek sebesar 4801.91 dan nilai fitness 0.000208 untuk penggunaan 100 generasi, ukuran populasi sebanyak 100, probabilitas *crossover* 0.9, dan probabilitas mutasi 0.1 dengan perhitungan waktu 1.19 detik, sedangkan AG Parallel menghasilkan performansi nilai jarak terpendek sebesar 4739,34 dan fitness 0,000211 untuk penggunaan 100 generasi, ukuran populasi sebanyak 150, probabilitas crossover 0.9, probabilitas mutasi 0.5 dengan perhitungan waktu 1.04 detik.

Kata kunci : Algoritma Genetika, Algoritma *Bruteforce, Trevelling Salesman Problem, High Performance Computing, CUDA Programming.*

ABSTRACT

The improvements in computational phenomenons could be seen from many developments of new technology, especially for some problems that require to compute big datas, that if done by serial computation, it will be consuming great amount of time wasted. How computer does the parallel programming will determined their performance. With some effective codes, computating with parallel environment would have decrease usage time from well-developed-algorithm efficiently. This assignment will somehow examine how different it is when two algorithms are used to compute an example problem, which is Genetic Algorithm and Brute Force Algorithm determined between parallel or serial that wrote in C code compared one another, within CUDA from NVIDIA architecture. Travelling for 101 cities will be used for the main case of both algorithms. The result of this research has the shortest path of Serial Genetic Algorithm at 4801.91 and 0.000208 for fitness using 100 generations, 100 populations, 90% chance of crossover, 10% chance of mutation, with 1,19 seconds execution time, on the other hand, Parallel Genetic Algorithm has the shortest path at 4739,34 and 0,000211 fitness using 100 generation, 150 populations, 90% chance of crossover, 50% chance of mutation with 1,04 seconds of execution time.

***Keywords* :** *Genetic Algorithm , Bruteforce Algorithm, Trevelling Salesman Problem, High Performance Computing, CUDA programming.*

1. PENDAHULUAN

1.1. Latar Belakang

Banyaknya permasalahan menggunakan data yang besar menjadi pertimbangan untuk pencarian solusi yang optimal dengan segala efisiensi yang bisa dibuat. Efisiensi yang dimaksud antara lain efisiensi waktu, pengerjaan, hasil, dan cara hitung. Pemanfaatan beberapa algoritma juga telah digunakan dalam penyelesaian masalah Travelling Salesman Problem (TSP) yang membutuhkan komputasi tinggi dikarenakan data yang besar.

Penggunaan algoritma yang sesuai akan sangat membantu penyelesaian permasalahan. Beberapa algoritma yang dapat digunakan antara lain Algoritma Brute Force dan Algoritma Genetika.

Permasalahan TSP menggunakan AG dengan tools CUDA telah dibahas dalam Tugas Akhir mahasiswa Teknik Informatika yang menggunakan dataset pengujian ulysses22, att48, dan gr137 dengan data yang didapat dari TSPLIB [10]. Penelitian yang dilakukan oleh beberapa mahasiswa dan dosen di Universitas Parma juga telah membahas tentang AG yang diimplementasikan pada Compute Unified Device Architecture (CUDA) dan didapat bahwa penggunaan GAlib sangat membantu dalam mempercepat waktu eksekusi [7]. Untuk penggunaan Brute Force dalam AG dijelaskan bahwa algoritma harus menguji semua posisi dan jarak untuk perhitungan yang akan dilakukan selanjutnya.

Tugas Akhir ini akan membahas bagaimana perbedaan yang dihasilkan dengan menggunakan metode serial dan parallel dalam perhitungan jarak optimal TSP dengan data yang besar dan dengan menggunakan dua algoritma yang berbeda, yaitu Algoritma Brute Force dan Algoritma Genetika. Perbedaan yang sangat mendasar antara kedua algoritma ini adalah waktu, susunan kode, dan hasil yang akan didapat. Oleh karena itu, maka akan dilakukan pengujian untuk menjabarkan perbedaan hasil tersebut.

Mengingat perbandingan dalam penyelesaian masalah TSP antara kedua algoritma ini sangatlah signifikan, maka diperlukan sejumlah data latih dan data uji yang memiliki jumlah lebih dari 100 data agar pemrosesan secara parallel dapat diimplementasikan, karena apabila data terlalu sedikit dalam penggunaan algoritma parallel, akan sia-sia dan kurang optimal.

Maka untuk observasi pada tugas akhir ini, diharapkan dapat menjelaskan dangan baik perbedaan antara pengunaan kedua algoritma tersebut dengan penggunaan parallel computing.

1.2. Perumusan Masalah

Berdasarkan latar belakang, maka perumusan masalah yang dicari adalah sebagai berikut :

a. Bagaimana perbedaan hasil/solusi yang didapat dari penelitian penggunaan Algoritma Brute Force dan Algoritma Genetika pada persoalan TSP dalam komputasi kinerja tinggi.
b. Apa saja yang mempengaruhi perbedaan yang didapat dari penggunaan kedua algoritma tersebut.
c. Apa saja perbedaan yang dihasilkan TSP dengan algoritma genetika, namun dengan parameter yang berbeda.

1.3. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

a. Melakukan penelitian terhadap algoritma Brute Force dan algoritma genetika pada persoalan TSP dalam komputasi kinerja tinggi.
b. Mengetahui perbedaan dari penggunaan kedua algoritma tersebut.
c. Mendapatkan perbedaan hasil dalam pengimplementasian TSP terhadap algoritma genetika, namun dengan parameter yang berbeda.

2. DASAR TEORI

2.1. Travelling Salesman Problem (TSP)

Travelling Salesman Problem didasari oleh algoritma graf yang memuat tentang bagaimana mencari jalur yang terpendek untuk mengunjungi satu tempat ke tempat lain dengan melewati beberapa titik persinggahan sehingga jalur yang akan digunakan merupakan jalur terbaik. TSP ini menggunakan metode perhitungan yang telah ditentukan sebelumnya. Setiap tempat atau node hanya dapat dilalui sekali dan kembali ke tempat awal perjalanan. Jarak antar node dapat dihitung dengan normalisasi antar dua titik menggunakan rumus Euclidean Distance, berikut penjelasan rumus tersebut [12]:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2} \quad (1)$$

Keterangan :

d(x,y) = jarak dari kota x ke kota y
x_n = koordinat x kota ke-n
y_n = koordinat y kota ke-n

Hasil yang didapat dari rumus diatas adalah total bobot dari keseluruhan jarak dari kota pertama ke kota yang dituju. Untuk kasus TSP dimana dibutuhkan jarak terpendek dari setiap iterasi untuk selanjutnya digunakan sebagai individu terbaik, maka fungsi yang digunakan adalah fungsi pemaksimalan terhadap nilai fitness. Fungsi ini didasari oleh sifat AG sendiri, dimana individu dengan fitness terbesar memiliki kemampuan bertahan lebih dibandingkan dengan yang fitnessnya lebih kecil.

Untuk penggunaan TSP adalah bagaimana melakukan pencarian jarak terpendek antara node-node dengan total cost terendah. Jenis TSP ynag digunakan adalah asymmetric TSP, dimana kedua jarak pulang – pergi dari satu kota ke kota lain bisa berbeda. [5]

Penggunaan TSP dalam jumlah kecil dapat digambarkan dalam graf, namun untuk jumlah data yang besar, disarankan untuk menggunakan matriks dalam penggambaran tiap node, dan penggunaan algoritma yang lebih rumit dari sekedar algoritma sederhana seperti Brute Force.

2.2. Algoritma Brute Force

Algoritma Brute Force adalah algoritma dalam pencarian solusi dengan cara langsung, namun dengan jumlah langkah yang banyak. Metode pencarian ini dikatakan sebagai metode paling sederhana dan *to-the-point*.

Tahapan pada algoritma Brute Force untuk TSP [8]:

a. Enumerasi solusi dari node-node dengan cara yang sistematis
b. Evaluasi tiap kemungkinan solusi yang ditemukan satu per satu, dan menyimpan solusi terbaik yang didapat sampai akhir pencarian
c. Bila sudah sampai iterasi terakhir, maka dilakukan pemanggilan terhadap solusi optimal (jarak terpendek) yang didapat sebelumnya

Kekurangan yang dapat terjadi karena penggunaan metode ini adalah pembangkitan dari pencarian solusi yang banyak, yaitu sebanyak (n-1)!/2, sehingga untuk jumlah data yang banyak, akan menjadi tidak efisien dalam hal waktu pengerjaan, sehingga penggunaan BF untuk penelitian ini hanya sebagai *baseline* pembanding dengan AG.

2.3. Algoritma Genetika

Algoritma Genetika awalnya didasari oleh teori Darwin yang menyebutkan bahwa orang tua akan mewarisi sifat-sifat tertentu pada keturunannya, namun dengan kombinasi karakter antar kedua orang tua tersebut. Lalu AG dikembangkan oleh John Holland pada tahun 1975 di Universitas Michigan. Komponen yang digunakan dalam AG , yaitu skema pengkodean, nilai *fitness*, seleksi orang tua, kawin silang, mutasi, penggantian populasi dan kriteria penghentian[2]. Keunggulan AG adalah algoritma ini mampu mempertahankan solusi terbaik dan membuang solusi yang tidak baik dengan kawin silang antara orang tua yang terpilih.

1. Representasi Individu
   Data dikonversikan kedalam bentuk individu dengan menggunkan kode tertentu, Tugas Akhir ini menggunakan representasi permutasi yang digunakan apabila urutan menjadi faktor yang penting.

Tabel 1 Contoh Representasi Permutasi

9	6	3	7	1	8	2	4	10	5

2. Nilai *Fitness*
   Nilai *fitness* dapat didefinisikan sebagai nilai terbaik yang didapat dari setiap hasil pemrosesan, nilai *fitness* lama yang terbaik akan bertahan dan digunakan dalam perhitungan berikutnya. Pada persoalan TSP digunakan *invers fitness*.

$$f = \frac{1}{h+a} \quad (2)$$

   Keterangan :
   f = nilai *fitness* untuk permutasi
   h = total bobot yang didapat dari penjumlahan jarak dari kota 1 ke kota 2, dari kota 2 ke kota 3, dan seterusnya hingga kota terakhir.
   a = suatu nilai kecil untuk mencegah pembagian dengan nol
3. Linear Fitness Ranking
   Proses ini merupakan metode perankingan untuk keperluan pemilihan orang tua dengan megurutkan individu secara *ascending* berdasarkan nilai *fitness* yang dihasilkan dengan menggunakan rumus berikut [3]:

$$f(i) = f\ max - (f\ \max - f\ min)(\frac{R(i)-1}{N-1}) \quad (3)$$

   Keterangan:
   fmax = Nilai *fitness* tertinggi
   fmin = Nilai *fitness* terendah
   R(i) = *Ranking* individu ke-i. *R(i)*=1 untuk individu *fitness* tertinggi dan *R(i)* = N untuk individu *fitness* terendah.
   i = Indeks individu dalam populasi
   N =Jumlah individu dalam populasi.
4. Seleksi *Parents*
   Merupakan pemilihan nilai *fitness* dari perhitungan sebelumnya yang dapat bertahan dan digunakan dalam perhitungan selanjutnya. Rekombinasi dilakukan dengan menggunakan metode *Roulette Wheel*[11].
5. Rekombinasi
   Setiap orang tua akan menghasilkan 2 anak hasil rekombinasi. Peluang terjadinya rekombinasi ditentukan oleh Probabilitas Rekombinasi (Pc). Interval Pc dengan interval 0.5 sampai 0.9.
6. *Crossover* (Kawin Silang)
   Kawin Silang adalah proses untuk mencari *fitness* terbaik untuk bakal *parents*. Untuk permutasi pada AG TSP, dilakukan *order crossover* untuk mencegah kota yang sama dilewati lebih dari satu kali.
7. Mutasi
   Mutasi ditentukan oleh Probabilitas Mutasi (Pm) dengan interval [0,1]. Pada kasus TSP, mutasi yang dilakukan adalah swapping mutation, sehingga mencegah kota yang sama terlewati lebih dari satu kali, yaitu dengan memilih dua gen secara acak, gen pertama yang terpilih akan ditukar dengan gen terpilih kedua.
8. Seleksi *Survivor*
   Seleksi survivor merupakan langkah yang digunakan untuk menentukan nilai terbaik yang pantas untuk dijadikan *parent* pada proses berikutnya.

2.4. Parallel Algoritma Genetika

Perbedaan antara Parallel AG dengan Serial AG adalah perhitungan komputasi relatif *fitness* nya memungkinkan untuk menjadikan data *layout* menjadi penting. Akan digunakan parallelisasi pada fungsi inisialisasi populasi dan evaluasi individu, dikarenakan kebutuhan untuk membangkitkan populasi sebanyak ukuran populasi dan kebutuhan untuk mengevaluasi individu untuk seratus satu kota akan menghabiskan banyak waktu apabila dilakukan secara serial. Data pada *host* akan di *copy* ke *device*, dan diproses pada GPU, lalu hasil akhir akan di *copy* kembali ke *host*.

2.5. Evaluasi Performansi

Beberapa hal yang dijadikan bahan perhitungan dari perbandingan antara algoritma serial dan parallel antara lain [4]:

$$Speedup = \frac{Sequential\ (time)}{Parallel\ (time)} \quad (4)$$

$$Performance\ Improvement = \frac{Sequential\ (time) - Parallel\ (time)}{Sequential\ (time)} \quad (5)$$

2.6. Compute Unified Device Architecture (CUDA)

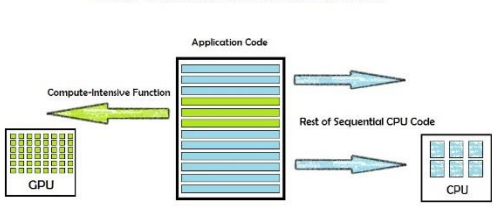
Compute Unified Device Architecture adalah teknologi yang pengembangannya dilakukan oleh NVIDIA untuk memudahkan penggunaan GPU secara non-grafis. Dimana pada generasi sebelumnya, penggunaan komputasi yang dapat terbagi akan membagi *resource* nya dalam *vertex* dan *pixel*, arsitektur CUDA memasukkan metode *pipeline* yang menyebabkan *Arithmetic Logic Unit* (ALU) dalam chip dapat bekerja secara parallel. *Syntax* yang digunakan pada NVIDIA CUDA ini mirip dengan *syntax* bahasa C, C++, dan Fortran sehingga lebih mudah mempelajarinya.

Dalam proses pemrogramannya, syntax C akan diproses oleh compiler C, namun apabila dalam proses ditemukan syntax CUDA, maka program akan otomatis mengeksekusi perintah yang diawali syntax tersebut kedalam GPU CUDA. CUDA telah diimplementasikan dalam berbagai bidang, diantaranya : *Molecular Dynamics*, *Seismic Simulation*, Simulasi saham dan finansial, perhitungan dinamika fluida, sains lingkungan, dan *Image* & *Video Processing*[9]. Umumnya langkah yang dilakukan CUDA untuk melakukan operasi matematik adalah :

1. Deklarasi Kernel
   Didefinisikan dengan *syntax* __global__, __device__, __shared__, dan semacamnya pada permulaan dibuatnya fungsi. Pengoperasian di dalam *kernel* dilakukan dengan menjumlahkan elemen matriks satu demi satu, tujuannya adalah untuk menghitung *array* secara bersamaan.
2. Penyalinan data dari host ke GPU
   Berbagai proses pada CUDA yang berhubungan tentang pemindahan data dari *host* (CPU) ke *device* (GPU) maupun sebaliknya ditangani oleh CUDA dengan *syntax* cudaMemcpy.
3. Parameter Kernel
   Kernel akan dieksekusi oleh *block* yang berisi *thread*, jumlah *thread* yang ada sebanding dengan jumlah elemen pada matriks yang akan dihitung.
4. Eksekusi Kernel
   Dilakukan inisialisasi *kernel* dengan beberapa parameter yang telah ditentukan lalu *passing* data ke kernel.
5. Mengambil hasil
   Untuk pemindahan hasil eksekusi dari *device* ke *host*, *syntax* yang digunakan adalah cudaMemcpyDeviceToHost. *Syntax* tersebut digunakan untuk memindahkan atau menyalin memori dari *device* ke *host*. Apabila dilanjutkan dengan penggunaan parameter dalam proses pemindahan hasil, maka syntax yang digunakan menjadi cudaMemcpyHostToDevice.
6. Membebaskan *pointer*
   Untuk membebaskan *pointer* dari memori GPU, *syntax* yang digunakan adalah cudaFree(*pointer*).

2.7. Graphic Processing Unit (GPU)

Graphic Processing Unit pertama kali dibuat sebagai akselerasi grafis dan hanya memiliki fungsi untuk mengolah tampilan grafis. Penggunaan GPU mengandalkan bahasa tingkat tinggi, aplikasi-aplikasi yang mendukung memiliki tugas sendiri pada pemrosesan data, yang optimal untuk digunakan dalam *single thread* saat proses lain dijalankan pada GPU. Perhitungan yang ada pada aplikasi yang sedang berjalan di CPU menawarkan beberapa *intensive-portion* dari tugas kepada GPU untuk diproses saat beberapa diantaranya tetap dijalankan pada CPU, sehingga aplikasi yang dijalankan terasa semakin cepat dari sisi pengguna.



Gambar 1 Cara Kerja GPU

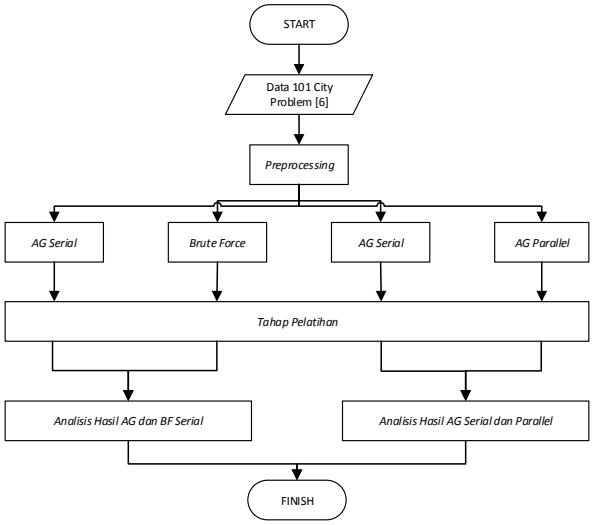
Perbedaan antara *Central Processing Unit* (CPU) dan GPU adalah perbedaan cara kerja antara keduanya. CPU terdiri hanya dari beberapa *core* yang dibuat untuk mengerjakan proses serial, sedangkan GPU memiliki ribuan *core-core* kecil dan digunakan untuk mengerjakan proses secara parallel, yang dikondisikan untuk melakukan perhitungan yang besar[1].

3. Deskripsi Sistem

Pada tugas akhir ini, akan dibangun sistem perbandingan antara AG yang diolah secara serial maupun parallel, dan perbandingan hasil antara AG dengan BF. Hasil terbaik merupakan hasil yang memiliki tingkat perhitungan waktu yang lebih cepat dan jarak tempuh antar kota yang paling rendah.

```
{{41,49},{35,17},{55,45},{55,20},{15,30},{25,30},{20,50},{10,43},{55,60},{30,60},
 {20,65},{50,35},{30,25},{15,10},{30,5},{10,20},{5,30},{20,40},{15,60},{45,65},
 {45,20},{45,10},{55,5},{65,35},{65,20},{45,30},{35,40},{41,37},{64,42},{40,60},
 {31,52},{35,69},{53,52},{65,55},{63,65},{2,60},{20,20},{5,5},{60,12},{40,25},
 {42,7},{24,12},{23,3},{11,14},{6,38},{2,48},{8,56},{13,52},{6,68},{47,47},
 {49,58},{27,43},{37,31},{57,29},{63,23},{53,12},{32,12},{36,26},{21,24},{17,34},
 {12,24},{24,58},{27,69},{15,77},{62,77},{49,73},{67,5},{56,39},{37,47},{37,56},
 {57,68},{47,16},{44,17},{46,13},{49,11},{49,42},{53,43},{61,52},{57,48},{56,37},
 {55,54},{15,47},{14,37},{11,31},{16,22},{4,18},{28,18},{26,52},{26,35},{31,67},
 {15,19},{22,22},{18,24},{26,27},{25,24},{22,27},{25,21},{19,21},{20,26},{18,18},{35,35}};
```

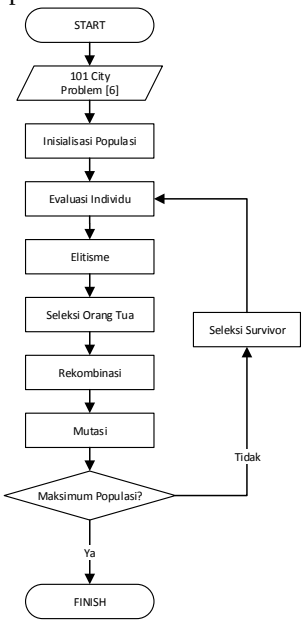
Gambar 2 Data 101 Kota [6]



Gambar 3 Diagram Sistem

3.1. Algoritma Genetika Serial

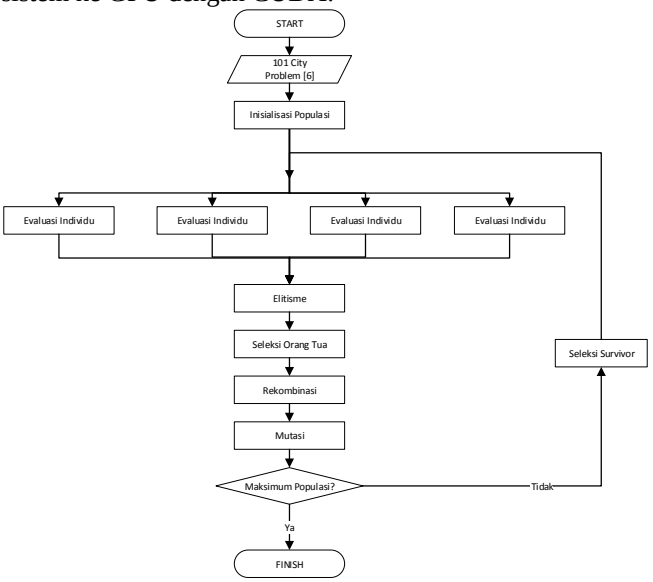
Tahap selanjutnya adalah implementasi koding Algoritma Genetika serial dalam bahasa C, dengan menggunakan data yang ada dan studi kasus TSP, implementasi koding dilakukan dengan membagi tiap tahapan pada AG.



Gambar 4 Diagram Algoritma Genetika Serial

3.2. Algoritma Genetika Parallel

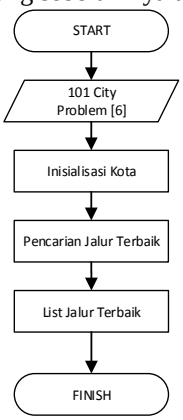
Pada AG parallel dilakukan penambahan beberapa algoritma pada inisialisasi populasi dan evaluasi individu untuk mengimplementasikan sistem ke GPU dengan CUDA.



Gambar 5 Diagram Algoritma Genetika Parallel

3.3. Algoritma Brute Force

Proses TSP dengan algoritma Brute Force hanya terjadi satu kali iterasi, hasil terbaik akan disimpan di memori sampai proses selesai, dan akan dipilih jalur dengan nilai bobot terendah dari semua jalur/kromoson yang sebelumnya telah tersimpan.



Gambar 6 Diagram Algoritma Brute Force

4. HASIL DAN ANALISIS

Berdasarkan skenario yang telah dijelaskan sebelumnya, telah dilakukan observasi untuk pengujian kombinasi parameter berikut diatas secara serial maupun paralel. Berikut ditampilkan hasil observasi untuk masing-masing skenario :

4.1. Algoritma Genetika Serial dan Parallel

Tabel 2 Hasil Observasi AG Serial

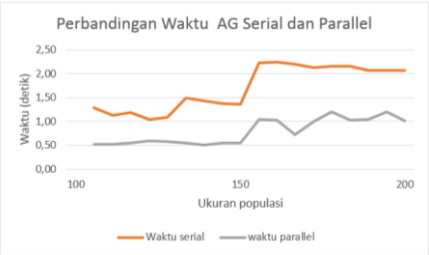
Ukuran Populasi	Probabilitas Crossover	Probabilitas Mutasi	Panjang lintasan	Fitness Terbaik	Waktu (detik)
100	0,9	0,5	4900,89	0,000204	1,29
		0,3	4950,32	0,000202	1,13
		0,1	4801,91	0,000208	1,19
	0,7	0,5	4844,92	0,000206	1,05
		0,3	4942,37	0,000202	1,09
		0,1	4878,14	0,000205	1,48
	0,5	0,5	4834,72	0,000207	1,43
		0,3	4898,96	0,000204	1,37
		0,1	4935,61	0,000203	1,36

Penelitian yang dilakukan membuktikan bahwa semakin besar ukuran populasi, maka semakin lama pula waktu proses, dimana jumlah populasi yang digunakan berbanding lurus dengan waktu proses. Penentuan ukuran populasi, Pc, dan Pm juga dapat mempengaruhi performansi. Tabel observasi menunjukkan generasi terbaik terdapat pada ukuran populasi 100, Pc 0,9 dan Pm 0,1. Jarak terpendek yang diperoleh sebesar 4801,91 dengan *fitness* terbaik 0,000208 dan waktu komputasi 1,19 detik.

Tabel 3 Hasil Observasi AG Parallel

Ukuran Populasi	Probabilitas Crossover	Probabilitas Mutasi	Panjang lintasan	Fitness Terbaik	Waktu Parallel (detik)
150	0,9	0,5	4739,34	0,000211	1,04
		0,3	4797,78	0,000199	1,03
		0,1	4917,11	0,000205	0,72
	0,7	0,5	4766,87	0,000210	1,00
		0,3	4822,92	0,000204	1,19
		0,1	4852,58	0,000208	1,03
	0,5	0,5	4795,98	0,000210	1,04
		0,3	4873,57	0,000207	1,20
		0,1	4864,20	0,000207	1,02

Berdasarkan hasil penelitian dari parameter AG parallel yang disamakan dengan AG serial, maka didapat jarak terpendek untuk 101 kota pada ukuran populasi 150, Pc 0,9 dan Pm 0,5. Jarak terpendek yang diperoleh sebesar 4739.34 dengan *fitness* terbaik 0,000211 dan waktu komputasi 1,04 detik. Dari hasil tersebut dapat dihitung nilai *speedup* antara serial dengan parallel sebesar 1,14 kali dan peningkatan performansi sebesar 13%.



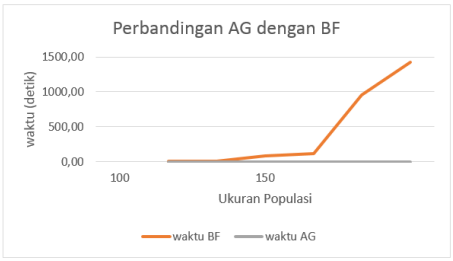
Gambar 7 Grafik Perbandingan Waktu AG

4.2. Brute Force dengan Algoritma Genetika

Tabel 4 Algoritma BF Serial dengan AG Serial

Ukuran Populasi	Jumlah Kota	Panjang Jalur BF	Panjang Jalur AG	Waktu Eksekusi BF Serial	Waktu Eksekusi AG Serial
100	9	208,34	207,88	7,43	1,77
	10	220,98	220,59	81,21	1,98
	11	222,19	221,62	950,52	2,05
150	9	208,34	207,72	11,23	2,59
	10	220,98	220,24	122,37	2,77
	11	222,19	221,66	1420,07	2,83

Penelitian terhadap AG dan BF membuktikan bahwa perbedaan waktu yang dihasilkan sangatlah besar, hal ini tentu disebabkan oleh perbedaan cara kerja dan kompleksitas yang digunakan oleh kedua algoritma. Observasi yang dilakukan menghasilkan nilai *speedup* 185,46 kali dan peningkatan performansi sebesar 99,46%



Gambar 8 Grafik Perbandingan Waktu AG dengan BF

4.3. Brute Force Serial dan Parallel

Tabel 5 Hasil Observasi Brute Force

Ukuran Populasi	Jumlah Kota	Panjang Jalur	Waktu Eksekusi Serial	Waktu Eksekusi Parallel
100	9	208,34	7,43	9,52
	10	220,98	81,21	77,38
	11	222,19	950,52	882,26
150	9	208,34	11,23	19,62
	10	220,98	122,37	135,28
	11	222,19	1420,07	1251,50

Penggunaan parallel pada BF diterapkan pada fungsi *generate*, dan menghasilkan sedikit perbedaan pada segi waktu eksekusi, dimana perbedaan waktu tersebut berbanding lurus dengan jumlah kota yang digunakan. *Speedup* dari observasi BF memiliki nilai 1,091 kali lebih besar parallel dengan peningkatan performansi sebesar 22,26%.

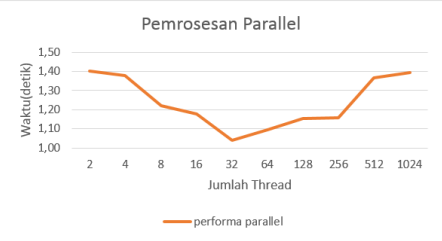
4.4. Penggunaan Komputasi Kinerja Tinggi

Berdasarkan skenario terbaik dari AG, dapat dilakukan observasi untuk membuktikan apakah skenario terbaik untuk penggunaan *thread* dari KKT sudah optimal. Observasi dilakukan dengan mengubah penggunaan jumlah *thread* dengan masing-masing 3 kali percobaan.

Tabel 6 Observasi Jumlah Thread pada GPU

Jumlah Thread	Performa Parallel (detik)
2	1,40
4	1,38
8	1,22
16	1,18
32	1,04
64	1,09
128	1,15
256	1,16
512	1,37
1024	1,39

Berdasarkan hasil tabel diatas, dapat digambarkan grafik performa perubahan jumlah *thread* yang dipakai pada parallelisme AG terhadap waktu yang membuktikan bahwa penggunaan 32 *thread* merupakan skenario optimal untuk digunakan dengan parameter percobaan.



Gambar 9 Grafik Proses Parallel pada KKT

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan observasi yang telah dilakukan, penelitian tugas akhir ini dapat disimpulkan seperti berikut:

1. Banyaknya masing-masing parameter pada tiap-tiap percobaan berpengaruh terhadap hasil, terutama pada variabel ukuran populasi AG.
2. Algoritma Genetika cocok untuk dilakukan parallelisasi pada spesifikasi diatas, karena berdasarkan hasil yang didapat, terjadi perubahan jarak terpendek dari 4801,91 menjadi 4739,34 dan waktu komputasi dari 1,19 detik pada AG serial menjadi 1,04 detik untuk AG parallel.
3. Nilai *speedup* sebesar 1,14 kali dan peningkatan performansi sebesar 13% juga dapat dijadikan acuan untuk melihat bahwa penggunaan algoritma parallel AG cukup efektif untuk diimplementasikan.
4. Perbedaan waktu yang sangat besar antara AG dengan BF membuktikan bahwa BF memang tidak efisien untuk digunakan pada TSP.
5. Dari observasi yang dilakukan, dapat disimpulkan bahwa hasil yang didapat sudah sesuai dengan hipotesis yang dibuat sebelumnya, maka dapat dilakukan penelitian lebih lanjut dari data dan hasil uji.
6. Nilai *random* Pc maupun Pm yang merupakan algoritma dasar pada AG dan mempengaruhi nilai *fitness* yang didapat, karena ketika individu memenuhi kondisi untuk masuk fungsi *crossover* maupun mutasi, individu tersebut memiliki peluang lebih untuk menghasilkan individu baru yang lebih baik dari *parent* nya.

5.2. Saran

1. Penggunaan lebih dari 150 populasi dimungkinkan bagi perangkat yang memiliki spesifikasi yang lebih baik dari spesifikasi perangkat yang digunakan penulis.
2. Dapat dilakukan parallelisme kepada seluruh fungsi algoritma yang digunakan, namun baiknya waktu *copy* antar *device* tidak perlu dihitung karena mengurangi efisiensi waktu.
3. Diperlukan ukuran populasi ataupun maxgen yang lebih besar untuk mendapat perbandingan yang lebih terlihat antara serial dengan parallel.

DAFTAR PUSTAKA

[1] Dr. O. John, UC Davis, Dr. L. David Luebke, NVIDIA, "NVIDIA CUDA," *Parallel Programming and Computing Platform*. [Online] . Available: http://www.nvidia.com/object/what-is-gpu-computing.html [Diakses : 1 Maret 2014]

[2] A. Brigida (2013). *Algoritma Genetik*. Informatika, Bandung : IF.

[3] H.John (1992). *Genetic Algorithm, Easy Introduction by Father of the Field*. America : Scientific American.

[4] S. Jason (2011). K. Edward (2011). *CUDA By Example*. US : NVIDIA.

[5] G. Federico (2008). *Travelling Salesman Problem*. Australia : In-teh.

[6] Eilon, Cristopher, "101 City Problem," TSP. [Online]. Available : http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/eil101.tsp [Diakses : 10 Februari 2014]

[7] D.Stefano (2008). M.Nicola (2008). M.Lucca (2008). C.Stefano (2008). *Implementation Of Simple Genetic Algorithm Within the CUDA Architecture* . Italy : Universita degli Studi di Parma.

[8] A.K.Wendy (2005). F.Tora (2005). H.Arinto (2005). *Pemakaian Algoritma Brute Force pada Permasalahan TSP*. Teknik Informatika, Bandung : STT.

[9] M.Taufiq (2010). *Segala Tentang Nvidia CUDA*. Bandung : EPC, 2010.

[10] M.D.Duano (2006). Suyanto,ST.,MSc (2006). F.A.Yulianto (2006). *Analisis Penerapan Algoritma Genetika Parallel Pada Graphic Processing Unit*. Informatika Bandung.

[11] Suyanto. (2008). *Evolutionary Computation Komputasi Berbasis Evolusi dan Genetika*, Informatika, Bandung.

[12] Deza, Elena; Deza, Michel Marie (2009). *Encyclopedia of Distances*. Springer. p. 94.