23.252/H/05

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

STUDI PERBANDINGAN ANTARA ALGORITMA BIVARIATE MARGINAL DISTRIBUTION DENGAN ALGORITMA GENETIKA

TUGAS AKHIR

RSIf
005.1
Kus
s-1
2005

PERPUSTAKAAN
ITS

Tgl. Terima	11-7-2005
Terima Dari	H
No. Agenda Prp.	222403

Disusun Oleh :

Imam Artha Kusuma
5198 100 066

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

STUDI PERBANDINGAN ANTARA ALGORITMA BIVARIATE MARGINAL DISTRIBUTION DENGAN ALGORITMA GENETIKA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Yudhi Purwananto, S.Kom, M.Kom	Chastine Fatichah, S.Kom
NIP. 132 172 210	NIP. 132 298 829

SURABAYA
2005

ABSTRAK

ABSTRAK

Bivariate Marginal Distribution (BMDA) merupakan perkembangan lebih lanjut dari Estimation of Distribution Algorithm (EDA). Algoritma heuristik ini mengenalkan pendekatan baru dalam melakukan rekombinasi untuk membentuk individu baru, yaitu tidak menggunakan proses crossover dan mutasi seperti pada Algoritma Genetika (GA). BMDA menggunakan keterkaitan pasangan variabel dalam melakukan rekombinasi untuk membentuk individu baru. Keterkaitan antar variabel tersebut ditemukan selama proses optimasi berlangsung.

Aplikasi yang dibuat dalam Tugas Akhir ini ditujukan untuk membandingkan kinerja Genetic Algorithm sederhana persilangan satu titik dengan Bivariate Marginal Distribution Algorithm pada kasus Onemax, Fungsi De Jong F2, dan Traveling Salesman Problem. Onemax dipilih untuk menguji kedua algoritma karena Onemax sering digunakan dalam pengujian algoritma optimasi dan dianggap mewakili fungsi linear. Fungsi De Jong F2 dipilih untuk menguji kedua algoritma karena fungsi ini sering digunakan dalam uji coba algoritma optimasi pada masalah-masalah numerik. Dan Traveling Salesman Problem dipilih dalam pengujian kedua algoritma karena TSP merupakan masalah optimasi yang sudah sering digunakan oleh para peneliti karena kesederhanaannya, pentingnya, serta hubungannya dengan masalah kombinatorial.

Dari uji coba yang dilakukan, didapat hasil bahwa kinerja dari kedua algoritma tersebut dipengaruhi oleh parameter masing-masing dan juga besar ukuran populasi yang digunakan. Untuk kasus Onemax dengan ukuran masalah yang kecil, Genetic Algorithm lebih unggul dalam hal jumlah iterasi yang lebih sedikit dan waktu yang lebih cepat untuk mendapat hasil optimal. Namun, Bivariate Marginal Distribution Algorithm lebih unggul dalam hal hasil optimasi pada kasus Onemax dengan ukuran masalah yang lebih besar. Untuk Fungsi De Jong F2, Genetic Algorithm lebih unggul dari Bivariate Marginal Distribution Algorithm utamanya dalam hal jumlah iterasi dan waktu. Sedangkan untuk kasus Traveling Salesman Problem, Bivariate Marginal Distribution Algorithm dapat menunjukkan kinerja yang lebih baik dari Genetic Algorithm dalam hal hasil optimasi.

Kata kunci : heuristic algorithm, estimation of distribution algorithm, bivariate marginal distribution algorithm, genetic algorithm.

KATA PENGANTAR

KATA PENGANTAR

Puji dan syukur kehadirat Allah atas karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul :

"STUDI PERBANDINGAN ANTARA ALGORITMA BIVARIATE MARGINAL DISTRIBUTION DENGAN ALGORITMA GENETIKA"

Tugas Akhir ini dibuat untuk memenuhi persyaratan mendapatkan gelar sarjana pada jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya.

Pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dan mendorong penulis dalam menyelesaikan Tugas Akhir ini, khususnya kepada :

- Bpk. Yudhi Purwananto, S.Kom, M.Kom selaku Ketua Jurusan Teknik Informatika ITS, dosen wali, dan dosen pembimbing pertama dalam pengerjaan Tugas Akhir ini.
- Ibu Chastine Fatichah, S.Kom selaku koordinator Tugas Akhir dan dosen pembimbing kedua dalam pengerjaan Tugas Akhir ini.
- Seluruh dosen di lingkungan Teknik Informatika, FTIf, ITS yang tidak bisa penulis sebutkan satu per satu.
- Seluruh staff dan karyawan di lingkungan Teknik Informatika, FTIf, ITS yang tidak bisa penulis sebutkan satu per satu.
- Ibunda dan (alm) Ayahanda tercinta yang tiada henti memberikan curahan kasih sayang serta dukungan kepada penulis.

- Kakakku di rumah dan di Balikpapan yang senantiasa memberikan dorongan untuk segera menyelesaikan Tugas Akhir ini, juga Ninda, Lindri Getuk, Mbak Ainun, dan Yunan yang selalu memberikan dukungan semangat.
- Keluarga besar Wiyung, Pakdhe, Budhe, Mbak Ari, Mas Dodi, Ardi dan Ardo yang senantiasa memberikan dukungan kepada penulis.
- Keluarga besar di Bekasi.
- Seluruh keluarga besar Teratai Tunjung.
- Keluarga besar Mulyosari, Cphee, Deni Lele, Yuda Napakalas, Dimas Madmax, BuddyBoy si pembimbing bayangan, Kendy serta Wicax-Naruto.
- Semua teman-teman seangkatan, Hary kumis, Decky, Iqbal, Purna, Trisno, Iril, Ade, Caca, Ifah, Raras, Citra dan Wonderwoman serta crew c0e lainnya yang tidak bisa disebutkan satu per satu.
- Crew Lab Prog, AJK, Lab SI dan Lab RPL.
- Semua pihak dan rekan-rekan yang tidak bisa penulis sebutkan satu per satu.

Penulis menyadari bahwa masih banyak kekurangan-kekurangan pada Tugas Akhir ini. Untuk itu saran dan kritik membangun sangat penulis harapkan demi kesempurnaan Tugas Akhir ini. Akhir kata, penulis mengharapkan Tugas Akhir ini dapat memberikan manfaat bagi semua pihak.

Surabaya, Maret 2005

Penulis

DAFTAR ISI

DAFTAR ISI

ABSTRAK iii

KATA PENGANTAR iv

DAFTAR ISI vi

DAFTAR GAMBAR viii

DAFTAR TABEL x

BAB 1 PENDAHULUAN 1

1.1. Latar Belakang 1
1.2. Perumusan Masalah 2
1.3. Batasan Masalah 3
1.4. Tujuan Pembuatan Tugas Akhir 4
1.5. Metodologi Tugas Akhir 4
1.6. Sistematika Penulisan Tugas Akhir 5

BAB 2 GA DAN BMDA 7

2.1. Algoritma Genetika 7
2.1.1. Struktur Umum Algoritma Genetika 8
2.1.2. Komponen-Komponen Utama Algoritma Genetika 9
2.1.3. Seleksi 12
2.1.4. Rekombinasi 18
2.1.5. Mutasi 25
2.1.6. Algoritma Genetika Sederhana (Persilangan Satu Titik) 27
2.2. Algoritma Bivariate Marginal Distribution 30
2.2.1. Marginal Frequencies dan Statistik Pearson 32
2.2.2. Pembentukan Dependency Graph 34
2.2.3. Pembentukan Individu Baru 37
2.2.4. Deskripsi BMDA 38

BAB 3 PERANCANGAN APLIKASI OPTIMASI 41

3.1. Desain Aplikasi Optimasi 41
3.1.1. Desain Aplikasi Optimasi GA 41
3.1.2. Desain Aplikasi Optimasi BMDA 44
3.2. Keterangan Tentang Masalah Optimasi 47
3.2.1. Fungsi Onemax 47
3.2.2. Fungsi De Jong F2 47
3.2.3. Traveling Salesman Problem 49
3.3. Antarmuka Aplikasi Optimasi 50
3.3.1. Deskripsi Antarmuka untuk Masalah Optimasi Onemax 50
3.3.2. Deskripsi Antarmuka untuk Masalah Optimasi De Jong F2 53
3.3.3. Deskripsi Antarmuka untuk Masalah Optimasi TSP 54

BAB 4 UJI COBA DAN ANALISIS SISTEM 57

4.1. Lingkungan Uji Coba 57
4.2. Skenario Uji Coba 57
4.3. Pelaksanaan Uji Coba 58
4.3.1. Uji Coba untuk Kasus Onemax-20 58
4.3.2. Uji Coba untuk Kasus Onemax-75 61
4.3.3. Uji Coba untuk Kasus Onemax-100 65
4.3.4. Uji Coba untuk Kasus De Jong F2 68
4.3.5. Uji Coba untuk Kasus TSP-Rute 1 71
4.3.6. Uji Coba untuk Kasus TSP-Rute 2 74
4.4. Analisis Hasil Uji Coba 77
4.4.1. Analisis untuk Masalah Optimasi Onemax 77
4.4.2. Analisis untuk Masalah Optimasi Fungsi De Jong F2 86
4.4.3. Analisis untuk Kasus Traveling Salesman Problem 91

BAB 5 KESIMPULAN 98

DAFTAR PUSTAKA 101

LAMPIRAN 1 102

LAMPIRAN 2 103

DAFTAR GAMBAR

DAFTAR GAMBAR

Gambar 2.1 Seleksi roda roulett. 15
Gambar 2.2 Seleksi universal sampling. 15
Gambar 2.3 Rekombinasi diskret. 19
Gambar 2.4 Area induk dan anak pada rekombinasi menengah 19
Gambar 2.5 Posisi anak yang mungkin pada rekombinasi menengah 20
Gambar 2.6 Posisi anak yang mungkin pada rekombinasi garis 21
Gambar 2.7 One-point crossover 22
Gambar 2.8 Multi-point crossover 23
Gambar 3.1 Diagram alir seleksi dengan metode Seleksi Roda Roulette 41
Gambar 3.2 Diagram alir rekombinasi dengan metode Penyilangan Satu Titik 42
Gambar 3.3 Diagram alir mutasi 43
Gambar 3.4 Diagram alir algoritma genetika sederhana. 43
Gambar 3.5 Diagram alir pembentukan Dependency Graph. 44
Gambar 3.6 Diagram alir pembentukan individu baru pada BMDA. 45
Gambar 3.7 Diagram alir algoritma BMDA. 46
Gambar 3.8 Antarmuka untuk masalah optimasi Onemax. 50
Gambar 3.9 Antarmuka untuk masalah optimasi De Jong F2. 53
Gambar 3.10 Antarmuka untuk masalah optimasi TSP. 55
Gambar 4.1 Grafik perbandingan hasil optimasi kasus Onemax-20 antara GA dengan BMDA. 81
Gambar 4.2 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-20 antara GA dengan BMDA. 82
Gambar 4.3 Grafik perbandingan waktu penyelesaian kasus Onemax-20 antara GA dengan BMDA. 82
Gambar 4.4 Grafik perbandingan hasil optimasi kasus Onemax-75 antara GA dengan BMDA. 83
Gambar 4.5 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-75 antara GA dengan BMDA. 84
Gambar 4.6 Grafik perbandingan waktu penyelesaian kasus Onemax-75 antara GA dengan BMDA. 84
Gambar 4.7 Grafik perbandingan hasil optimasi kasus Onemax-100 antara GA dengan BMDA. 85
Gambar 4.8 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-100 antara GA dengan BMDA. 85
Gambar 4.9 Grafik perbandingan waktu penyelesaian kasus Onemax-100 antara GA dengan BMDA. 85
Gambar 4.10 Grafik perbandingan hasil optimasi Fungsi De Jong F2 antara GA dengan BMDA. 89

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Gambar 4.11 Grafik perbandingan jumlah iterasi penyelesaian Fungsi De jong F2 antara GA dengan BMDA. 90

Gambar 4.12 Grafik perbandingan waktu penyelesaian Fungsi De Jong F2 antara GA dengan BMDA. 90

Gambar 4.13 Grafik perbandingan hasil optimasi kasus TSP-Rute 1 antara GA dengan BMDA. 95

Gambar 4.14 Grafik perbandingan jumlah iterasi penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA. 96

Gambar 4.15 Grafik perbandingan waktu penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA. 96

Gambar 4.16 Grafik perbandingan hasil optimasi kasus TSP-Rute 2 antara GA dengan BMDA. 96

Gambar 4.17 Grafik perbandingan jumlah iterasi penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA. 97

Gambar 4.18 Grafik perbandingan waktu penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA. 97

DAFTAR TABEL

DAFTAR TABEL

Tabel 2.1 Probabilitas seleksi & nilai fitness. 14

Tabel 2.2 Jumlah tetangga (ukuran lingkungan) untuk seleksi total. 17

Tabel 4.1 Tabel hasil optimasi rata-rata penyelesaian Onemax-20 dengan GA. 59

Tabel 4.2 Tabel jumlah iterasi rata-rata generasi terbaik penyelesaian Onemax-20 dengan GA. 59

Tabel 4.3 Tabel waktu rata-rata untuk mencapai generasi terbaik pada penyelesaian Onemax-20 dengan GA. 59

Tabel 4.4 Tabel waktu total rata-rata untuk mencapai iterasi maksimal pada penyelesaian Onemax-20 dengan GA. 60

Tabel 4.5 Tabel pengaruh besar populasi pada penyelesaian Onemax-20 dengan GA. 60

Tabel 4.6 Tabel hasil optimasi rata-rata penyelesaian Onemax-20 dengan BMDA. 60

Tabel 4.7 Tabel jumlah iterasi rata-rata penyelesaian Onemax-20 dengan BMDA. 60

Tabel 4.8 Tabel waktu konvergensi rata-rata penyelesaian Onemax-20 dengan BMDA. 61

Tabel 4.9 Tabel pengaruh besar populasi pada penyelesaian Onemax-20 dengan BMDA. 61

Tabel 4.10 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-75 dengan GA. 62

Tabel 4.11 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-75 dengan GA. 62

Tabel 4.12 Tabel waktu iterasi rata-rata untuk mendapatkan generasi terbaik pada penyelesaian Onemax-75 dengan GA. 62

Tabel 4.13 Tabel waktu total rata-rata untuk mendapatkan iterasi maksimal pada penyelesaian Onemax-75 dengan GA. 63

Tabel 4.14 Tabel perbandingan besar populasi pada penyelesaian Onemax-75 dengan GA. 63

Tabel 4.15 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-75 dengan BMDA. 64

Tabel 4.16 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-75 dengan BMDA. 64

Tabel 4.17 Tabel waktu konvergensi rata-rata pada penyelesaian Onemax-75 dengan BMDA. 64

Tabel 4.18 Tabel pengaruh besar populasi pada penyelesaian Onemax-75 dengan BMDA. 64

Tabel 4.19 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-100 dengan GA. 65

Tabel 4.20 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-100 dengan GA. 65

Tabel 4.21 Tabel waktu iterasi rata-rata pada penyelesaian Onemax-100 dengan GA. 66

Tabel 4.22 Tabel waktu total rata-rata untuk mendapat generasi terbaik pada penyelesaian Onemax-100 dengan GA. 66

Tabel 4.23 Tabel pengaruh besar populasi pada penyelesaian Onemax-100 dengan GA. 66

Tabel 4.24 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-100 dengan BMDA. 67

Tabel 4.25 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-100 dengan BMDA. 67

Tabel 4.26 Tabel waktu konvergensi rata-rata pada penyelesaian Onemax-100 dengan BMDA. 67

Tabel 4.27 Tabel pengaruh besar populasi pada penyelesaian Onemax-100 dengan BMDA. 68

Tabel 4.28 Tabel hasil optimasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA. 68

Tabel 4.29 Tabel jumlah iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA. 69

Tabel 4.30 Tabel waktu iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA. 69

Tabel 4.31 Tabel waktu total rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA. . 69

Tabel 4.32 Tabel pengaruh besar populasi pada penyelesaian Fungsi DeJong F2 dengan GA. 70

Tabel 4.33 Tabel hasil optimasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA. 70

Tabel 4.34 Tabel jumlah iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA. 70

Tabel 4.35 Tabel waktu konvergensi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA. 70

Tabel 4.36 Tabel pengaruh besar populasi pada penyelesaian Fungsi DeJong F2 dengan BMDA. 71

Tabel 4.37 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA. 72

Tabel 4.38 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA. 72

Tabel 4.39Tabel waktu iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA. 72

Tabel 4.40 Tabel waktu total rata-rata pada penyelesaian TSP-Rute 1 dengan GA. 72

Tabel 4.41 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 1 dengan GA. 73

Tabel 4.42 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA. ... 73

Tabel 4.43 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA. ... 73

Tabel 4.44 Tabel waktu konvergensi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA. 74

Tabel 4.45 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 1 dengan BMDA. . 74

Tabel 4.46 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA. 75

Tabel 4.47 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA. 75

Tabel 4.48 Tabel waktu iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA. 75

Tabel 4.49 Tabel waktu total rata-rata pada penyelesaian TSP-Rute 2 dengan GA. 75

Tabel 4.50 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 2 dengan GA. 76

Tabel 4.51 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA. ... 76

Tabel 4.52 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA. ... 76

Tabel 4.53 Tabel waktu konvergensi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA. 76

Tabel 4.54 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 2 dengan BMDA. . 77

Tabel 4.55 Tabel perbandingan penyelesaian kasus Onemax-20 antara GA dengan BMDA. 82

Tabel 4.56 Tabel perbandingan penyelesaian kasus Onemax-75 antara GA dengan BMDA. 84

Tabel 4.57 Tabel perbandingan penyelesaian kasus Onemax-100 antara GA dengan BMDA. 86

Tabel 4.58 Tabel perbandingan penyelesaian Fungsi De Jong F2 antara GA dengan BMDA. 90

Tabel 4.59 Tabel perbandingan penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA. 96

Tabel 4.60 Tabel perbandingan penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA. 97

BAB 1

PENDAHULUAN

BAB 1

PENDAHULUAN

Pada bab ini akan dijelaskan beberapa hal dasar dalam Tugas Akhir yang meliputi latar belakang pembuatan, perumusan dan batasan masalah, tujuan dari Tugas Akhir ini serta sistematika dan metodologi yang digunakan dalam penulisan buku Tugas Akhir.

1.1. Latar Belakang

Algoritma Genetika (GA) bekerja dengan rangkaian populasi yang berukuran tetap. Dari populasi yang ada, kromosom yang lebih baik akan lebih dipilih dibandingkan dengan kromosom yang buruk. Rangkaian baru dihasilkan dengan menggunakan rekombinasi/penyilangan operator dan mutasi. Rekombinasi populasi menggabungkan informasi yang terkandung di dalam dua kromosom sedangkan mutasi melakukan gangguan pada rangkaian kromosom untuk menjaga keanekaragaman populasi dan mengenalkan informasi baru. Proses ini diketahui dapat menyebabkan gangguan pada kromosom yang menyebabkan algoritma ini berpenampilan buruk pada masalah yang membutuhkan hasil optimum [4]. Hal ini akhirnya menuntun pada pendekatan baru dalam melakukan rekombinasi yang dinamakan *Estimation of Distribution Algorithm* (EDA). EDA pertama kali diperkenalkan dalam lingkup *Evolutionary Computation* oleh Muehlenbein dan Paab (1996).

Perkembangan dari EDA lebih lanjut adalah *Univariate Marginal Distribution Algorithm* (UMDA) (Muehlenbeim, 1998), *Population Based Incremental Learning* (PBIL) (Baluja, 1994) dan *Compact Genetic Algorithm* (cGA) (Harik, 1998). Ketiga algoritma tersebut memberikan hasil yang baik pada kromosom dengan dependensi yang rendah di antara variabel-variabelnya, tetapi tidak memberikan hasil yang baik untuk tingkat ketergantungan yang lebih tinggi [1]. Untuk memecahkan masalah tersebut, diperkenalkanlah *Bivariate Marginal Distribution Algorithm* (BMDA) oleh Pelikan dan Muehlenbeim pada tahun 1999.

Hal yang menarik perhatian mengenai pendekatan baru dalam proses rekombinasi ini adalah apakah pendekatan baru tersebut benar-benar dapat menunjukkan nilai lebih dibanding GA. Untuk itu, perlu kiranya untuk membandingkan algoritma dengan pendekatan baru tersebut dengan GA.

Algoritma PBIL telah digunakan dan dijelaskan dalam Tugas Akhir berjudul "Rancang Bangun Aplikasi Optimasi pada Traveling Salesman Problem dan Fungsi De Jong F2 Menggunakan Algoritma Population Based Incremental Learning (PBIL)" yang disusun oleh Melissa Ariani Yahya pada tahun 2004 lalu[6]. Pada Tugas Akhir ini, penulis ingin mengetahui bagaimana kinerja dari BMDA yang menggunakan keterkaitan antar variable dalam melakukan proses rekombinasi dengan cara membandingkannya dengan kinerja dari GA sederhana.

1.2. Perumusan Masalah

1. Bagaimana menyelesaikan kasus Onemax, Fungsi De Jong F2, dan *Traveling Salesman Problem* dengan menggunakan BMDA.

2. Bagaimana menyelesaikan kasus Onemax, Fungsi De Jong F2, dan *Traveling Salesman Problem* dengan menggunakan GA sederhana.
3. Bagaimana perbandingan antara BMDA dan GA sederhana dalam penyelesaian kasus Onemax, Fungsi De Jong F2, dan *Traveling Salesman Problem* dalam hal hasil optimasi, jumlah iterasi dan waktu yang dibutuhkan untuk mendapatkan hasil optimal, serta pengaruh besar populasi terhadap optimalitas hasil.

1.3. Batasan Masalah

Dari permasalahan-permasalahan yang disebutkan di atas, maka batasan dalam Tugas Akhir ini adalah :

1. Algoritma yang digunakan dan dibahas dalam Tugas Akhir ini adalah Algoritma Bivariate Marginal Distribution (BMDA) dan Algoritma Genetika (GA) sederhana persilangan satu titik.
2. Algoritma yang digunakan dalam Tugas Akhir ini hanya dibuat untuk menyelesaikan kasus Onemax, Fungsi De Jong F2, dan *Traveling Salesman Problem* (TSP).
3. Hal-hal yang akan dibandingkan pada kedua algoritma dalam Tugas Akhir ini meliputi optimalitas hasil, jumlah iterasi dan waktu yang dibutuhkan untuk mendapatkan hasil optimal, serta pengaruh besar populasi terhadap hasil optimasi.

1.4. Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan Tugas Akhir ini adalah untuk mengetahui kinerja dari BMDA dengan perbandingannya terhadap GA sederhana persilangan satu titik dalam penyelesaian kasus Onemax, Fungsi De Jong F2, dan TSP dilihat dari segi hasil optimasi, jumlah iterasi dan waktu yang dibutuhkan untuk mendapatkan hasil optimal, serta pengaruh besar populasi terhadap hasil optimasi.

1.5. Metodologi Tugas Akhir

Pembuatan Tugas Akhir ini terbagi menjadi beberapa tahapan sebagai berikut :

1. Studi literatur dan pemahaman sistem.

   Mencari, mempelajari, dan merangkum berbagai literatur yang berkaitan dengan Algoritma Bivariate Marginal Distribution (BMDA) dan Algoritma Genetika (GA) sederhana persilangan satu titik.

2. Perancangan perangkat lunak dan desain sistem.

   Pada tahap ini dilakukan proses perancangan perangkat lunak, desain antar muka, desain struktur data dan variabel, serta algoritma yang akan digunakan.

3. Pembuatan perangkat lunak.

   Pada tahap ini dilakukan proses pembuatan aplikasi, sebagai implementasi dari tahap sebelumnya.

4. Uji coba, evaluasi dan revisi perangkat lunak.

   Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat dengan memberikan input berupa data-data yang relevan, menguji kebenaran

algoritma dan *output* yang dihasilkan. Revisi dilakukan apabila perangkat lunak yang dibangun menghasilkan *output* yang belum sesuai dengan tujuan. Kemudian dilakukan pembandingan hasil uji coba masing-masing algoritma dalam segi hasil optimasi, jumlah iterasi dan waktu yang dibutuhkan untuk mendapatkan hasil optimal.

5. Penyusunan buku Tugas Akhir.

   Pada tahap terakhir ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Dokumentasi ini juga dibuat agar orang yang ingin mengembangkan sistem dapat mempelajari dokumentasi tersebut.

1.6. Sistematika Penulisan Tugas Akhir

Sistematika penulisan Tugas Akhir ini disusun dalam beberapa bab, yang akan dijelaskan sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisikan pendahuluan yang memuat beberapa hal dasar dalam Tugas Akhir yang meliputi latar belakang pembuatan Tugas Akhir, perumusan dan batasan masalah, tujuan pembuatan Tugas Akhir ini serta sistematika dan metodologi yang digunakan dalam penulisan buku Tugas Akhir.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

BAB II GA DAN BMDA

Bab ini berisikan penjelasan mengenai konsep dan teori yang menjadi landasan pembuatan Tugas Akhir. Dasar teori di sini meliputi konsep kedua algoritma baik BMDA maupun GA khususnya GA dengan persilangan satu titik.

BAB III DESAIN DAN IMPLEMENTASI SISTEM

Bab ini berisikan penjelasan mengenai perancangan aplikasi optimasi menggunakan kedua algoritma untuk tiap-tiap kasus yang diselesaikan. Pada bagian implementasi akan dijelaskan bagaimana antarmuka aplikasi yang dikembangkan berdasarkan rancangan dan desain yang telah dibuat sebelumnya.

BAB IV UJI COBA DAN ANALISIS HASIL

Bab ini berisikan penjelasan bagaimana proses uji coba dan analisis terhadap aplikasi yang telah dibangun, yang dilakukan untuk mengetahui perbandingan kinerja dari masing-masing algoritma terhadap kasus-kasus yang diselesaikan.

BAB V PENUTUP

Bab ini berisikan kesimpulan dari keseluruhan proses uji coba dan analisis yang dilakukan.

BAB 2

GA DAN BMDA

BAB 2

GA DAN BMDA

Pada bab ini akan dijelaskan konsep dan teori yang menjadi landasan pembuatan Tugas Akhir. Dasar teori di sini meliputi konsep dari GA sederhana persilangan satu titik dan juga BMDA.

2.1. Algoritma Genetika

GA adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Pengertian "heuristik" itu sendiri adalah sebagai berikut[5]:

- Kata heuristik itu sendiri berasal dari akar yang sama dengan kata dalam bahasa Yunani yaitu "eureka" yang berarti "untuk menemukan".
- Heuristik selalu mungkin tidak mencapai hasil yang diinginkan, tetapi bisa teramat berharga sampai proses yang memecahkan masalah.
- Heuristik yang baik secara dramatis bisa mengurangi waktu yang diharuskan memecahkan masalah dengan menghapuskan keperluan untuk mempertimbangkan kemungkinan atau status tidak relevan yang tak mungkin.

Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup. Pada dasarnya ada 4 kondisi yang sangat mempengaruhi proses evaluasi, yaitu:

a. Kemampuan organisme untuk melakukan reproduksi.

b. Keberadaan populasi organisme yang bisa melakukan reproduksi.

c. Keberagaman organisme dalam suatu populasi.

d. Perbedaan kemampuan untuk *survive*.

Individu yang lebih kuat (fit) akan memiliki tingkat survival dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Pada kurun waktu tertentu (sering dikenal dengan istilah generasi), populasi secara keseluruhan akan lebih banyak memuat organisme yang fit.

GA pertama kali dikembangkan oleh John Holland dari Universitas Michigan (1975). John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami atau buatan) dapat diformulasikan dalam terminologi genetika. GA adalah simulasi dari proses evolusi Darwin dan operasi genetika atas kromosom.

2.1.1. Struktur Umum Algoritma Genetika

Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin yang dikenal dengan istilah **populasi**. Individu yang terdapat dalam satu populasi disebut dengan istilah **kromosom**. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibentuk secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan istilah **generasi**. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang

disebut dengan **fungsi fitness**. Nilai fitness dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah **anak (*offspring*)** yang terbentuk dari gabungan 2 kromosom generasi sekarang yang bertindak sebagai **induk (*parent*)** dengan menggunakan operator **penyilangan (*crossover*)**. Selain operator penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan operator **mutasi**. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai fitness dari kromosom induk (*parent*) dan nilai fitness dari kromosom anak (*offspring*), serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) adalah konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik.

2.1.2. Komponen-Komponen Utama Algoritma Genetika

Berikut ini merupakan komponen-komponen utama algoritma genetika:

1. **Teknik Penyandian**

   Teknik Penyandian merupakan penyandian gen dari kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk: string bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

2. **Prosedur Inisialisasi**

   Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi

ditentukan, dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.

3. **Fungsi Evaluasi**

Ada 2 hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu: evaluasi fungsi objektif (fungsi tujuan) dan konversi fungsi objektif ke dalam fungsi fitness. Secara umum, fungsi fitness diturunkan dari fungsi objektif dengan nilai yang tidak negatif. Apabila ternyata fungsi objektif memiliki nilai negatif, maka perlu ditambahkan suatu konstanta C agar nilai fitness yang terbentuk menjadi tidak negatif.

4. **Seleksi**

Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit.

Ada beberapa metode seleksi dari induk, antara lain:

a. Rank-based fitness assignment.
b. Roulette wheel selection.
c. Stochatic universal sampling.
d. Local selection.
e. Truncation selection.
f. Tournament selection.

5. **Operator Genetika**

Ada 2 operator genetika, yaitu:

a. operator untuk melakukan rekombinasi, yang terdiri dari:
   - Rekombinasi bernilai real
     - Rekombinasi diskret
     - Rekombinasi intermediate (menengah)
     - Rekombinasi garis
     - Rekombinasi garis yang diperluas.
   - Rekombinasi bernilai biner (crossover)
     - Crossover satu titik.
     - Crossover banyak titik.
     - Crossover seragam.
   - Crossover dengan permutasi

b. Mutasi.
   - Mutasi bernilai real
   - Mutasi bernilai biner

6. Penentuan Parameter

Yang disebut dengan parameter di sini adalah parameter kontrol algoritma genetika, yaitu ukuran populasi (*popsize*), peluang *crossover* (p_c), dan peluang mutasi (p_m). Nilai parameter ini ditentukan juga berdasarkan permasalahan yang akan dipecahkan. Ada beberapa rekombinasi yang bisa digunakan, antara lain:

- Untuk permasalahan yang memiliki kawasan solusi cukup besar, De Jong merekombinasikan untuk nilai parameter kontrol:

$$(\text{popsize}; p_c; p_m) = (50; 0{,}6; 0{,}001)$$

- Bila rata-rata fitness setiap generasi digunakan sebagai indikator, maka Grefenstette merekombinasikan:

$$(popsize;\ p_c;\ p_m) = (30;\ 0{,}95;\ 0{,}01)$$

- Bila fitness dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah:

$$(popsize;\ p_c;\ p_m) = (80;\ 0{,}45;\ 0{,}01)$$

Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan [3].

2.1.3. Seleksi

Seleksi akan menentukan individu-individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana *offspring* terbentuk dari individu-individu terpilih tersebut. Langkah pertama yang dilakukan dalam seleksi ini adalah pencarian nilai fitness. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai fitness inilah yang nantinya akan digunakan pada tahap-tahap seleksi berikutnya.

Ada beberapa definisi yang biasa digunakan untuk melakukan perbandingan terhadap beberapa metode yang akan digunakan, antara lain:

- *Selective pressure*: probabilitas dari individu terbaik yang akan diseleksi dibandingkan dengan rata-rata probabilitas dari semua individu yang diseleksi.

- *Bias*: perbedaan absolut antara fitness ternormalisasi dari suatu individu dan probabilitas reproduksi yang diharapkan.
- *Spread*: range nilai kemungkinan untuk sejumlah *offspring* dari suatu individu.
- *Loss of diversity*: proporsi dari individu-individu dalam suatu populasi yang tidak terseleksi selama fase seleksi.
- *Selection intensity*: nilai fitness rata-rata yang diharapkan dalam suatu populasi setelah dilakukan seleksi (menggunakan distribusi Gauss ternormalisasi).
- *Selection variance*: variansi yang diharapkan dari distribusi fitness dalam populasi setelah dilakukan seleksi (menggunakan distribusi Gauss ternormalisasi).

2.1.3.1. Rank-based Fitness

Pada *rank-based fitness*, populasi diurutkan menurut nilai objectifnya. Nilai fitness dari tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan, dan tidak dipengaruhi oleh nilai objektifnya.

Misalkan N adalah jumlah individu dalam suatu populasi. **Pos** adalah posisi individu dalam populasi tersebut (posisi terendah suatu individu adalah **Pos**=1, dan posisi tertingginya adalah **Pos**=N). Sedangkan **SP** adalah *selective pressure*. Nilai fitness dari suatu individu dapat dihitung sebagai:

- Linear ranking:

  $$\text{Fitness(Pos)} = 2 - SP + 2(SP-1)(Pos-1)/(N-1)$$

  Nilai SP $\in$ [1,2].

- Non-linear ranking:

  Fitness(Pos) = Nind * X^(Pos-1)/sum(X^(i-1)); i = 1..N.

  Sedangkan X dihitung sebagai akar polinomial:

  (SP-1)*X^(N-1) + SP*X^(N-2) + ... + SP*X + SP = 0

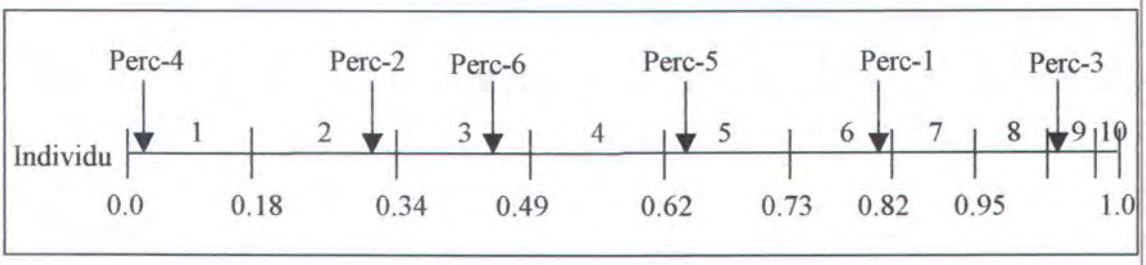
  Nilai SP $\in$ [1,N-2].

2.1.3.2. Seleksi Roda Roulette (Roulette Wheel Selection)

Metode seleksi roda roulette ini (Gambar 2.1) merupakan metode yang paling sederhana, dan sering juga dikenal dengan nama *stochastic sampling with replacement*. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran fitnessnya. Sebuah bilangan random dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan random tersebut akan terseleksi. Proses ini diulang hingga diperoleh sejumlah individu yang diharapkan. Tabel 2.1 menunjukkan probabilitas seleksi dari 11 individu. Individu pertama memiliki fitness terbesar, dengan demikian dia juga memiliki interval terbesar. Sedangkan individu ke-10 memiliki fitness terkecil kedua. Individu ke-11 memiliki fitness terkecil (=0), interval terkecil sehingga tidak memiliki kesempatan untuk melakukan reproduksi.

Tabel 2.1 Probabilitas seleksi & nilai fitness.

Individu ke-	1	2	3	4	5	6	7	8	9	10	11
Nilai Fitness	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
Probabilitas seleksi	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0



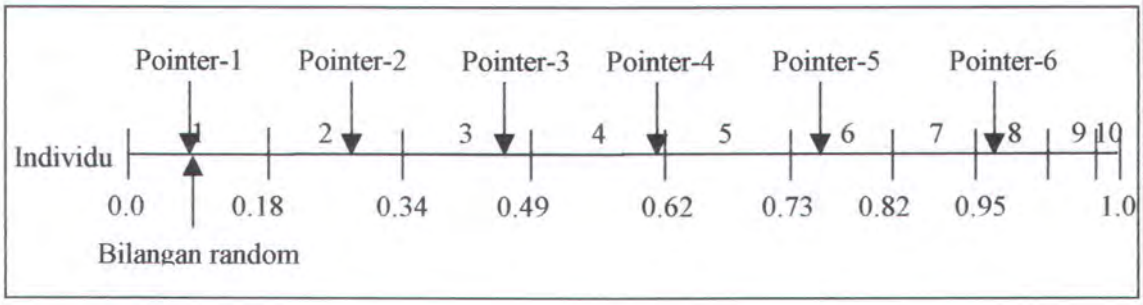
Gambar 2.1 Seleksi roda roulett.

Setelah dilakukan seleksi, maka individu-individu yang terpilih adalah:

1 2 3 5 6 9

2.1.3.3. Stocastic Universal Sampling

Stocastic universal sampling memiliki nilai bias nol dan penyebaran yang minimum. Pada metode ini, individu-individu dipetakan dalam suatu segmen garis secara berurutan sedemikian hingga tiap-tiap segmen individu memiliki ukuran yang sama dengan ukuran fitnessnya seperti halnya pada seleksi roda roulette. Kemudian diberikan sejumlah pointer sebanyak individu yang ingin diseleksi pada garis tersebut. Andaikan N adalah jumlah individu yang akan diseleksi, maka jarak antar pointer adalah 1/N, dan posisi pointer pertama diberikan secara acak pada range [1,1/N]. Apabila ada 6 individu yang akan diseleksi, maka jarak antar pointer adalah 1/6 = 0.167 (Gambar 2.2). Sehingga, misalkan bilangan random yang dibangkitkan pada [0,0.167] adalah 0,1 maka hasil yang diperoleh setelah seleksi adalah: 1 2 3 4 6 8



Gambar 2.2 Seleksi universal sampling.

2.1.3.4. Seleksi Lokal (Local Selection)

Pada seleksi lokal, setiap individu yang berada di dalam konstrain tertentu disebut dengan nama lingkungan lokal. Interaksi antar individu hanya dilakukan di dalam wilayah tersebut. Lingkungan tersebut ditetapkan sebagai struktur tempat populasi tersebut terdistribusi. Lingkungan tersebut juga dipandang sebagai kelompok pasangan-pasangan yang potensial.

Langkah pertama adalah menyeleksi separo pertama dari populasi yang berpasangan secara random (atau menggunakan salah satu metode yang sudah dijelaskan sebelumnya, seperti: *stochastic universal sampling*). Kemudian lingkungan baru tersebut diberikan pada setiap individu yang terseleksi. Pada lingkungan baru tersebut, kita bisa menyeleksi pasangan-pasangan yang cocok (pasangan yang terbaik, pasangan yang memiliki fitness proporsional, atau pasangan yang seragam). Struktur lingkungan pada seleksi lokal ini dapat berbentuk:

- Linear: *full ring* dan *half ring*
- Dimensi-2:
  - *full cross* dan *half cross*
  - *full star* dan *half star*
- Dimensi-3 dan struktur yang lebih kompleks yang merupakan kombinasi dari kedua struktur di atas.

Jarak antara individu dengan struktur tersebut akan sangat menentukan ukuran lingkungan (jumlah tetangga dari suatu individu). Hal ini dapat dilihat pada Tabel 2.2. Jarak antara satu individu dengan individu lainnya tentu saja akan terbatas

oleh ukuran lingkungan suatu individu. Individu yang terdapat dalam lingkungan yang lebih kecil tentu saja akan lebih terisolasi jika dibandingkan dengan individu yang terletak pada lingkungan dengan ukuran yang lebih besar.

Tabel 2.2 Jumlah tetangga (ukuran lingkungan) untuk seleksi total.

Struktur	Jarak	
	1	2
Full ring	2	4
Half ring	1	2
Full cross	4	8(12)
Half cross	2	4(5)
Ful star	8	24
Half star	3	8

2.1.3.5. Seleksi dengan Pemotongan (Truncation Selection)

Pada metode-metode seleksi yang telah dijelaskan terdahulu, seleksi dilakukan secara alami. Pada seleksi dengan pemotongan ini, lebih berkesan sebagai seleksi buatan. Seleksi ini biasanya digunakan oleh populasi yang jumlahnya sangat besar. Pada metode ini, individu-individu diurutkan berdasarkan nilai fitness-nya. Hanya individu-individu terbaik saja yang akan diseleksi sebagai induk. Parameter yang digunakan dalam metode ini adalah suatu nilai ambang *trunc* yang mengindikasikan ukuran populasi yang akan diseleksi sebagai induk yang berkisar antara 50%-10%. Individu-individu yang ada di bawah nilai ambang ini tidak akan menghasilkan keturunan.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

2.1.3.6. Seleksi dengan Turnamen (Tournament Selection)

Pada metode seleksi dengan turnamen ini, akan ditetapkan suatu nilai tour untuk individu-individu yang dipilih secara random dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang digunakan pada metode ini adalah ukuran tour yang bernilai antara 2 sampai N (jumlah individu dalam suatu populasi).

2.1.4. Rekombinasi

2.1.4.1. Rekombinasi Diskret

Rekombinasi diskret akan menukar nilai variabel antar kromosom induk. Misalkan ada 2 individu dengan 3 variabel, yaitu:

Induk 1:	12	25	5
Induk 2:	123	4	34

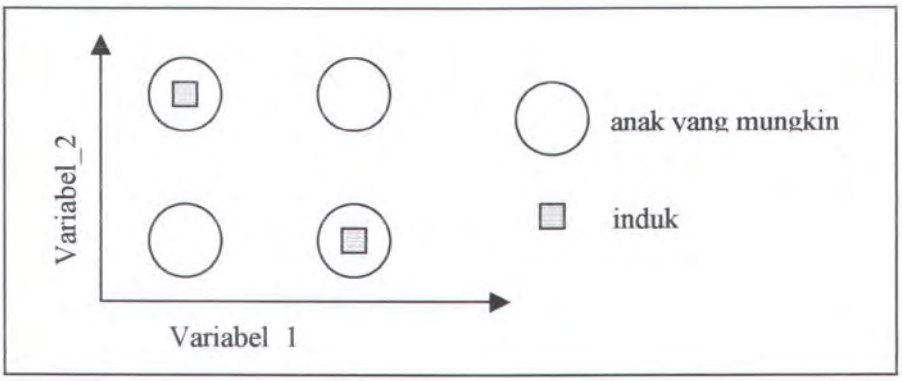
Untuk tiap-tiap variabel induk yang menyumbangkan variabelnya ke anak dipilih secara random dengan probabilitas yang sama.

Sampel 1:	2	2	1
Sampel 2:	1	2	1

Setelah rekombinasi, kromosom-kromosom baru yang terbentuk:

Anak 1:	123	4	5
Anak 2:	12	4	5

Gambar 2.3 menunjukkan posisi yang mungkin dari anak setelah rekombinasi diskret.



Gambar 2.3 Rekombinasi diskret.

Rekombinasi ini dapat digunakan untuk sembarang variabel (biner, real, atau simbol).

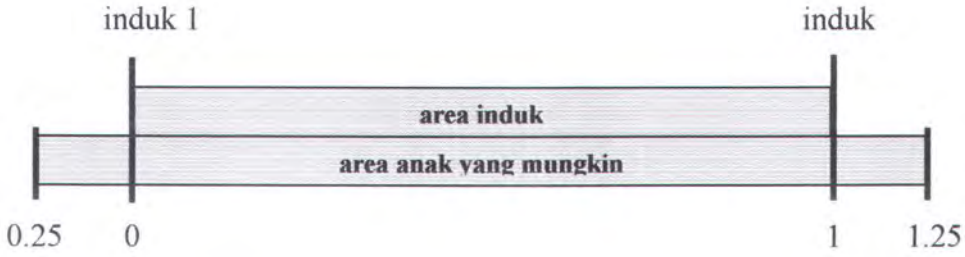
2.1.4.2. Rekombinasi Menengah

Rekombinasi menengah merupakan metode rekombinasi yang hanya dapat digunakan untuk variabel real (dan variabel yang bukan biner). Nilai variabel anak dipilih di sekitar dan antara nilai-nilai variabel induk.

Anak dihasilkan menurut aturan sebagai berikut:

Anak = induk 1 + alpha (induk 2 – induk 1)

Dengan alpha adalah factor skala yang dipilih secara random pada interval [-d, 1+d], biasanya d = 0.25. Tiap-tiap variabel pada anak merupakan hasil kombinasi variabel-variabel menurut aturan di atas dengan nilai alpha dipilih ulang untuk tiap variabel. Gambar 2.4 menunjukkan area induk dan anak yang mungkin.



Gambar 2.4 Area induk dan anak pada rekombinasi menengah

Misalkan ada 2 individu dengan 3 variabel, yaitu:

Induk 1:	12	25	5
Induk 2:	123	4	34

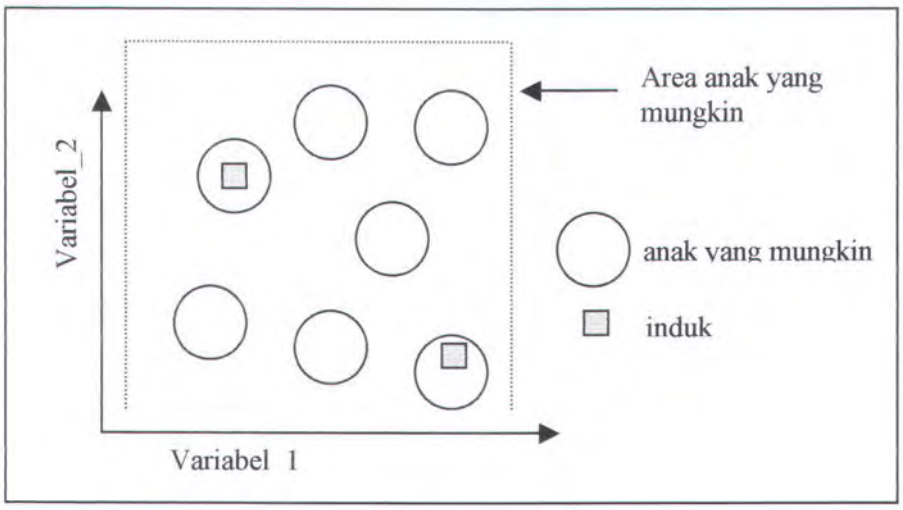
Misalkan nilai alpha yang terpilih adalah:

Sampel 1:	0,5	1,1	-0,1
Sampel 2:	0,1	0,8	0,5

Setelah rekombinasi, kromosom-kromosom baru yang terbentuk:

Anak 1:	67,5	1,9	2,1
Anak 2:	23,1	8,2	19,5

Gambar 2.5 menunjukkan posisi yang mungkin dari anak setelah rekombinasi menengah.



Gambar 2.5 Posisi anak yang mungkin pada rekombinasi menengah.

Rekombinasi ini dapat digunakan untuk sembarang variabel (biner, real, atau symbol).

2.1.4.3. Rekombinasi Garis

Pada dasarnya rekombinasi garis ini sama dengan rekombinasi menengah, hanya saja nilai alpha untuk semua variabel sama. Misalkan ada 2 kromosom dengan 3 variabel, yaitu:

Induk 1:	12	25	5
Induk 2:	123	4	34

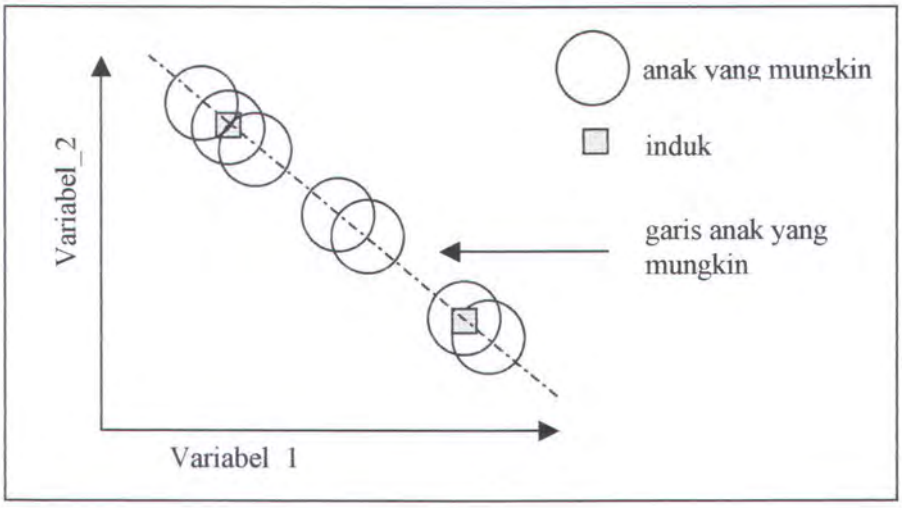
Untuk tiap-tiap variabel induk yang menyumbangkan variabelnya ke anak dipilih secara random dengan probabilitas yang sama.

Sampel 1:	0,5
Sampel 2:	0,1

Setelah rekombinasi, kromosom-kromosom baru yang terbentuk:

Anak 1:	67,5	14,5	19,5
Anak 2:	23,1	22,9	7,9

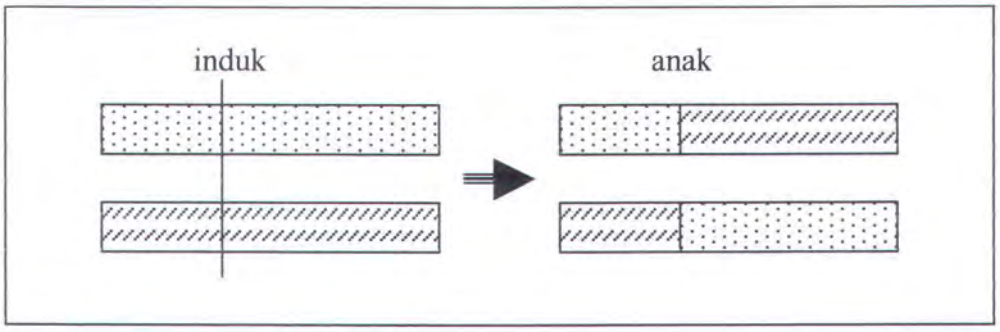
Gambar 2.6 menunjukkan posisi yang mungkin dari anak setelah rekombinasi garis.



Gambar 2.6 Posisi anak yang mungkin pada rekombinasi garis

2.1.4.4. Penyilangan Satu Titik (One-point Crossover)

Pada penyilangan satu titik, posisi penyilangan k (k – 1,2,...,N-1) dengan N=panjang kromosom diseleksi secara random. Variabel-variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak (Gambar 2.7).



Gambar 2.7 One-point crossover

Misalkan ada 2 kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 | 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 | 0 0 0 1 1 0 1

Posisi penyilangan yang terpilih: misalkan 5

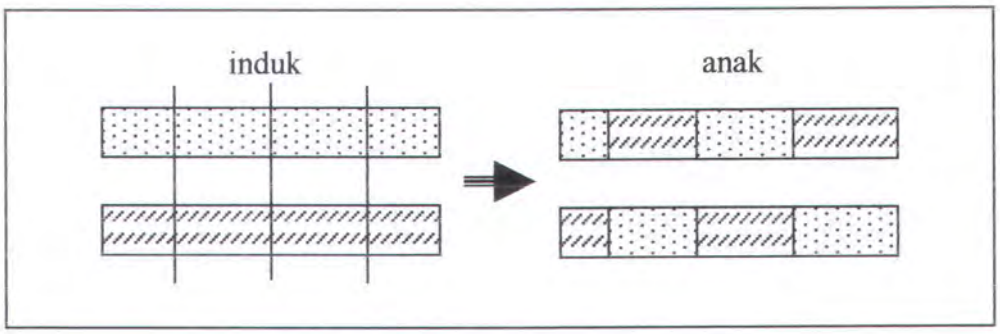
Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 1 1 0 | 0 0 0 1 1 0 1

Anak 2: 1 1 0 1 0 | 0 1 0 1 1 1 0

2.1.4.5. Penyilangan Banyak Titik (Multi-point Crossover)

Pada penyilangan banyak titik, m posisi penyilangan k_i (k=1,2,...,N-1; i=1,2,...,m) dengan N = panjang kromosom diseleksi secara random dan tidak diperbolehkan ada posisi yang sama, serta diurutkan naik. Variabel-variabel ditukar antar kromosom pada titik tersebut untuk menghasilkan anak (Gambar 2.8).



Gambar 2.8 Multi-point crossover

Misalkan ada 2 kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 0 0 0 1 1 0 1

Posisi penyilangan yang terpilih:

misalkan (m=3): 2 6 10

Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 | 0 1 0 0 | 1 0 1 1 | 0 1

Anak 2: 1 1 | 1 1 0 0 | 0 0 1 1 | 1 0

2.1.4.6. Penyilangan Seragam (Uniform Crossover)

Pada penyilangan seragam, setiap lokasi memiliki potensi sebagai tempat penyilangan. Sebuah *mask* penyilangan dibuat sepanjang panjang kromosom secara random yang menunjukkan bit-bit dalam *mask* yang mana induk akan menyediakan anak dengan bit-bit yang ada.

Induk mana yang akan menyumbangkan bit ke anak dipilih secara random dengan probabilitas yang sama. Di sini, anak_1 akan dihasilkan dari induk_1 jika bit *mask* bernilai 1, atau sebaliknya, anak_1 akan dihasilkan dari induk_2 jika bit *mask* bernilai 0. sedangkan anak_2 dihasilkan dari kebalikan *mask*.

Misalkan ada 2 kromosom dengan panjang 12:

Induk 1: 0 1 1 1 0 0 1 0 1 1 1 0

Induk 2: 1 1 0 1 0 0 0 0 1 1 0 1

Mask bit:

Sampel 1: 1 0 0 1 1 1 0 0 1 1 0 1

Sampel 2: 0 1 1 0 0 0 1 1 0 0 1 0

Setelah penyilangan, diperoleh kromosom-kromosom baru:

Anak 1: 0 1 0 1 0 0 0 0 1 1 0 0

Anak 2: 1 1 1 1 0 0 1 0 1 1 1 1

2.1.4.7. Penyilangan dengan Permutasi (Permutation Crossover)

Pada penyilangan dengan permutasi ini, kromosom-kromosom anak diperoleh dengan cara memilih sub-barisan suatu tour dari satu induk dengan tetap menjaga urutan dan posisi sejumlah kota yang mungkin terhadap induk yang lainnya (pada kasus TSP). Misal:

Induk 1: (1 2 3 | 4 5 6 7 | 8 9)

Induk 2: (4 5 3 | 1 8 7 6 | 9 2)

Anak 1: (x x x | 1 8 7 6 | x x)

Anak 2: (x x x | 4 5 6 7 | x x)

Dari sini akan didapatkan pemetaan:

1 – 4, 8 – 5, 7 – 6, 6 – 7 .

Kemudian sisa gen di induk 1 disalin ke anak 1 dengan menggunakan pemetaan yang ada.

Anak 1: (1-4 2 3 | 1 8 7 6 | 8-5 9)

Anak 1: (4 2 3 | 1 8 7 6 | 5 9)

Hal yang sama dilakukan untuk anak 2.

Anak 2: (4-1 5-8 3 | 4 5 6 7 | 9 2)

Anak 2: (1 8 3 | 4 5 6 7 | 9 2)

2.1.5. Mutasi

Setelah mengalami proses rekombinasi, pada offspring dapat dilakukan mutasi. Variabel offspring dimutasi dengan menambahkan nilai random yang sangat kecil (ukuran langkah mutasi), dengan probabilitas yang rendah. Peluang mutasi (p_m) didefinisikan sebagai persentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi bila peluang mutasi ini terlalu besar, maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya, dan juga algoritma akan kehilangan kemampuan untuk belajar dari histories pencarian [3]. Ada beberapa pendapat mengenai laju mutasi ini. Ada yang berpendapat bahwa, laju mutasi sebesar 1/n akan memberikan hasil yang cukup baik. Ada juga yang beranggapan bahwa laju mutasi tidak tergantung pada ukuran populasinya. Kromosom hasil mutasi harus diperiksa, apakah masih berada pada domain solusi, dan bila perlu bisa dilakukan perbaikan.

Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

2.1.5.1. Mutasi Bilangan Real

Pada mutasi bilangan real, ukuran langkah mutasi biasanya sangat sulit ditentukan. Ukuran yang kecil biasanya sering mengalami kesuksesan, namun ada kalanya ukuran yang lebih besar akan berjalan lebih cepat. Operator mutasi untuk bilangan real ini dapat ditetapkan sebagai:

- variabel yang dimutasi = variabel ± range*delta; (+ atau – memiliki probabilitas yang sama).
- Range = 0.5 * domain variabel; (interval pencarian).
- Delta = $\Sigma(a_i*2^{-i})$; a_i = 1 dengan probabilitas 1/m, selain itu a_i=0, dengan m=20.

2.1.5.2. Mutasi Biner

Cara sederhana untuk mendapatkan mutasi biner adalah dengan mengganti satu atau beberapa nilai gen dari kromosom. Langkah-langkah mutasi ini adalah:

- Hitung jumlah gen pada populasi (panjang kromosom dikalikan dengan ukuran populasi).
- Pilih secara acak gen yang akan dimutasi.
- Tentukan kromosom dari gen yang terpilih untuk dimutasi.
- Ganti nilai gen (0 ke 1, atau 1 ke 0) dari kromosom yang akan dimutasi tersebut.

2.1.6. Algoritma Genetika Sederhana (Persilangan Satu Titik)

Misalkan P(generasi) adalah populasi dari satu generasi, maka secara sederhana algoritma genetika terdiri dari langkah-langkah:

1. Generasi = 0 (generasi awal).
2. Inisialisasi populasi awal, P(generasi), secara acak.
3. Evaluasi nilai fitness pada setiap individu dalam P(generasi).
4. Kerjakan langkah-langkah berikut hingga generasi mencapai maksimum generasi:
   a. generasi = generasi + 1 (tambah generasi).
   b. Seleksi populasi tersebut untuk mendapatkan kandidat induk, P'(generasi).
   c. Lakukan *crossover* pada P'(generasi).
   d. Lakukan mutasi pada P'(generasi).
   e. Lakukan evaluasi fitness setiap individu pada P'(generasi).
   f. Bentuk populasi baru: P(generasi) = {P(generasi-1) yang survive, P'(generasi)}

2.1.6.1. Metode Seleksi dengan Roda Roulette

Seperti telah dijelaskan di depan bahwa seleksi induk dengan menggunakan roda roulette adalah metode yang paling sering digunakan. Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang memiliki fitness tinggi untuk melakukan reproduksi.

Algoritma seleksi roda roulette:

1. Hitung total fitness (F):
   - TotFitness = $\sum F_k$; k = 1, 2, ..., popsize.
2. Hitung fitness relatif tiap individu:
   - $p_k = F_k$ /TotFitness
3. Hitung fitness kumulatif:
   - $q_1 = p_1$
   - $q_k = q_{k-1} + p_k$; k = 2, 3, ..., popsize
4. Pilih induk yang akan menjadi kandidat untuk di-*crossover* dengan cara:
   - Bangkitkan bilangan random r.
   - Jika $q_k \leq r$ dan $q_{k+1} > r$, maka pilih kromosom ke (k+1) sebagai kandidat induk.

2.1.6.2. Crossover

Crossover (penyilangan) dilakukan atas 2 kromosom untuk menghasilkan kromosom anak (*offspring*). Kromosom anak yang terbentuk akan mewarisi sebagian sifat kromosom induknya. Metode *crossover* yang paling sering digunakan pada algoritma genetika sederhana dengan kromosom berbentuk string biner adalah *crossover* satu titik (*one-point crossover*) [3]. Seperti telah dijelaskan sebelumnya, pada metode ini, sembarang bilangan dipilih secara acak untuk menentukan posisi persilangan. Kemudian bagian kanan titik potong dari kedua induk kromosom tersebut ditukarkan untuk menghasilkan kromosom anak.

Misalkan:

v_1' =	1 0 1 0 0 1 1 0
v_2' =	0 1 0 1 0 1 0 0

Apabila posisi titik potong yang terpilih secara acak adalah 3, maka kromosom anak yang terbentuk adalah:

v_1'' =	1 0 1	0 0 1 1 0
v_2'' =	0 1 0	1 0 1 0 0

Pada *crossover* ada satu parameter yang sangat penting yaitu peluang *crossover* (p_c). Peluang *crossover* menunjukkan rasio dari anak yang dihasilkan dalam setiap generasi dengan ukuran populasi. Misalkan ukuran populasi (popsize=100), sedangkan peluang *crossover* (p_c=0,25), berarti bahwa diharapkan ada 25 kromosom dari 100 kromosom yang ada pada populasi tersebut akan mengalami *crossover*.

2.1.6.3. Mutasi

Mutasi yang digunakan pada algoritma genetika sederhana dengan kromosom biner seperti telah dijelaskan sebelumnya, pada dasarnya akan mengubah secara acak nilai suatu bit pada posisi tertentu. Kemudian bit 1 diganti dengan 0, atau bit 0 diganti dengan 1. pada mutasi ini sangat dimungkinkan muncul kromosom baru yang semula belum muncul dalam populasi awal.

Misalkan:

v_1' =	1 0 1 0	0	1 1 0

terkena mutasi pada gen ke-5, diperoleh:

v_1^* =	1 0 1 0	1	1 1 0

Pada mutasi ada satu parameter yang sangat penting yaitu peluang *crossover* (p_m). Peluang mutasi menunjukkan prosentasi jumlah total gen pada populasi yang akan mengalami mutasi. Untuk melakukan mutasi, terlebih dahulu harus dihitung

jumlah total gen pada populasi tersebut. Kemudian bangkitkan bilangan random yang akan menentukan posisi mana yang akan dimutasi (gen ke berapa pada kromosom ke berapa). Misalkan ukuran populasi (popsize=100), setiap kromosom memiliki panjang 20 gen, maka total gen adalah 100x20=2000 gen. Jika peluang mutasi (pm=0,01), berarti bahwa diharapkan ada (1/100)x2000 = 20 gen akan mengalami mutasi.

2.2. Algoritma Bivariate Marginal Distribution

BMDA merupakan pengembangan dari UMDA. Algoritma ini menggunakan keterkaitan pasangan gen untuk mengembangkan UMDA yang sederhana. BMDA merupakan contoh khusus dari FDA tapi tanpa adanya pengetahuan spesifik pada taraf awal. Pengetahuan tentang ketergantungan antar gen ditemukan selama proses optimasi itu sendiri.

Telah diketahui bahwa operator pada algoritma genetika sering mengakibatkan kekacauan pada skema rangkaian [4]. Hal tersebut dapat membuat kinerja algoritma genetika menjadi buruk pada masalah yang membutuhkan hasil optimum. Akhirnya hal ini menuntun pada pendekatan baru untuk melakukan rekombinasi.

Pendekatan pertama adalah menggunakan metode *reordering* yang berbeda dan metode lain yang mengurangi panjang skema. Pada metode ini, nilai bit pada posisi yang berbeda bukan satu-satunya yang dioptimasi sebagaimana algoritma genetika yang klasik. Urutan bit dan segi lainnya dioptimasi dengan baik. Metode ini bekerja cukup baik untuk merombak masalah meskipun membutuhkan

beberapa informasi terdahulu tentang masalah dan biasanya memakan banyak memori dan waktu. Ini mengarahkan pada GEMGA (*Gene Expression Messy Genetic Algorithm*).

Pendekatan kedua didasarkan pada estimasi distribusi kemungkinan. Cara termudah adalah dengan melakukan estimasi distribusi menggunakan *univariate marginal frequencies* dalam himpunan *parent* yang dipilih. Ini adalah dasar dari UMDA. Kinerja algoritma ini bagus pada masalah linear [4]. Algoritma ini juga bekerja sangat baik pada masalah yang tidak mengandung ketergantungan yang signifikan antar gen [4].

Untuk masalah dengan ketergantungan antara gen-gen yang berbeda, pendekatan ini tidak tepat [4]. Teori UMDA telah diperluas pada masalah dengan model probabilitas yang diketahui. Pada FDA, diasumsikan bahwa probabilitas ini dapat dituliskan sebagai beberapa produk *marginal frequencies*. Ini merupakan skema algoritma yang ideal, karena tidak ada skema rangkaian penting yang dapat dikacaukan. Cara pragmatis yang lebih lanjut untuk memperluas UMDA adalah menggunakan *bivariate marginal distribution*.

Pada algoritma MIMIC, distribusi diasumsikan sebagai rangkaian yang sederhana. Semua bit diarahkan pada rangkaian menggunakan algoritma greedy sederhana terlebih dahulu sebagai masukan *univariate* dan *bivariate marginal frequencies* di dalam himpunan *selected parents*. Untuk setiap individual baru, bit awal dihasilkan menggunakan probabilitas kondisional yang diberikan oleh bit terdahulu. Pendekatan lain mengggunakan struktur *tree* untuk model probabilitas. *Tree* dibuat untuk memaksimalkan jumlah informasi dari gen yang terhubung.

Nilai untuk bit yang berhubungan dengan *root* dari *tree* dihasilkan menggunakan *univariate frequencies*-nya. Bit-bit sisanya dihasilkan menggunakan probabilitas kondisional yang diberikan oleh nilai *parent* mereka pada *dependency tree*. Baik *univariate* maupun *bivariate frequencies* dihitung secara menaik. Dalam pendekatan ini, kedua metode tersebut digabungkan. Informasi berharga tentang ketergantungan antar gen digantikan oleh statistik chi-kuadrat Pearson, untuk memungkinkan mengidentifikasi pasangan yang independen dengan kemungkinan yang pasti. Pasangan ini kemudian dipetakan dalam *dependency graph*.

2.2.1. Marginal Frequencies dan Statistik Pearson

Diasumsikan panjang dari kromosom adalah n. Diasumsikan P sebagai himpunan dari rangkaian biner dengan panjang n (populasi). Ukuran P diasumsikan sebagai N. Untuk tiap posisi $i \in \{0,\ldots,n-1\}$ dan tiap nilai yang mungkin pada posisi ini $x_i \in \{0,1\}$, *univariate marginal frequency* $p_i(x_i)$ untuk himpunan P didefinisikan sebagai frekuensi rangkaian yang memiliki x_i pada posisi ke-i pada himpunan P. Secara hampir sama, untuk dua posisi $i \neq j \in \{0, \ldots, n-1\}$ dan setiap nilai yang mungkin pada posisi $x_i, x_j \in \{0,1\}$, *bivariate marginal frequencies* $p_{i,j}(x_i,x_j)$ untuk himpunan P didefinisikan sebagai frekuensi rangkaian yang memiliki x_i dan x_j pada posisi i dan j. Terkadang bentuk probabilitas digunakan sebagai frekuensi. Dengan kegunaan *univariate* dan *bivariate marginal frequencies*, probabilitas kondisional dari nilai x_i pada posisi ke-i dengan nilai x_j pada posisi ke-j dapat dikalkulasikan sebagai berikut:

$$p_{i,j}(x_i \mid x_j) = \frac{p_{i,j}(x_i, x_j)}{p_j(x_j)} \qquad \textbf{(Persamaan 2.1)}$$

di mana:

- $P_{i,j}(x_i|x_j)$ adalah probabilitas kondisional dari nilai x_i pada posisi ke-*i* dengan nilai x_j pada posisi ke-*j*
- x_i adalah nilai bit (0 atau 1) pada posisi ke-i dan x_j adalah nilai bit (0 atau 1) pada posisi ke-*j*
- $p_{i,j}(x_i,x_j)$ adalah nilai *bivariate marginal frequency* atau frekuensi rangkaian yang memiliki nilai x_i dan x_j pada posisi *i* dan *j*
- $p_j(x_j)$ adalah nilai *univariate marginal frequency* atau frekuensi rangkaian yang memiliki x_j pada posisi ke-*j*

Sedangkan statistik chi-kuadrat Pearson didefinisikan dengan:

$$X^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}} \qquad \textbf{(Persamaan 2.2)}$$

Di sini, untuk tiap pasangan posisi, nilai *observed* adalah nilai jumlah pasangan memungkinkan pada posisi ini. Jika dua posisi saling independen, jumlah tiap pasangan dapat secara mudah dikalkulasi menggunakan teori probabilitas dasar. Ini adalah nilai *expected*. Lalu, bentuk *univariate* dan *bivariate frequencies* serta jumlah total poin dibawa ke persamaan, untuk posisi i≠j, kita dapatkan persamaan:

$$X^2_{i,j} = \sum_{x_i, x_j} \frac{(Np_{i,j}(x_i, x_j) - Np_i(x_i)p_j(x_j))^2}{Np_i(x_i)p_j(x_j)} \qquad \textbf{(Persamaan 2.3)}$$

di mana:

- N adalah jumlah individu (dalam hal ini adalah jumlah parents)

- x_i adalah nilai bit (0 atau 1) pada posisi ke-i dan x_j adalah nilai bit (0 atau 1) pada posisi ke-j
- $p_{i,j}(x_i,x_j)$ adalah nilai *bivariate marginal frequency* atau frekuensi rangkaian yang memiliki nilai x_i dan x_j pada posisi i dan j
- $p_i(x_i)$ dan $p_j(x_j)$ adalah nilai *univariate marginal frequency*

jika posisi i dan j saling independen sebesar 95%, untuk statistik chi-kuadrat Pearson

$$X^2_{i,j} < 3.84 \qquad \textbf{(Persamaan 2.4)}$$

2.2.2. Pembentukan Dependency Graph

Dalam sesi ini dibahas mengenai pembentukan sebuah *dependency graph*. *Graph* tersebut didefinisikan oleh 3 himpunan V, E, dan R, yaitu $G = (V,E,R)$. V adalah himpunan dari *vertex-vertex*, $E \subset V \text{x} V$ adalah himpunan/set dari *edge-edge*, dan R adalah himpunan/set yang berisi satu *vertex* dari tiap komponen yang terkoneksi dari G.

Dalam *dependency graph*, setiap *node* menunjuk pada sebuah posisi pada rangkaian. Adalah hubungan langsung antara *vertex-vertex* dengan posisi pada rangkaian. Sehingga, kita dapat menggunakan himpunan dari *vertex-vertex* $V = \{0,\ldots,n\text{-}1\}$, dimana *vertex* i menunjukkan posisi ke-i. Dalam pembentukan *dependency graph*, *graph* tidak harus terhubung. Ini berarti bahwa *graph* tidak harus selalu berbentuk *tree*. *Dependency graph* selalu berbentuk *acyclic*. Generasi

dari rangkaian baru tidak tergantung pada jumlah dari komponen terhubung dari *dependency graph.*

2.2.2.1. Algoritma Pembentukan Dependency Graph

Anggap A sebagai himpunan dari *vertex-vertex* yang belum diproses. Pada awal algoritma, anggota A disamakan dengan anggota V. Kemudian, berangsur-angsur, saat *edge-edge* dimasukkan ke dalam E, anggota A menjadi semakin sedikit. Algoritma berakhir dengan A sebagai himpunan kosong yang berarti seluruh *vertex* telah diproses. Himpunan lain disebut dengan D. Ini adalah himpunan dari seluruh pasangan dari VxV yang tidak independen sebesar 95% (lihat persamaan 2.3 dan 2.4), yaitu:

$$D = \{(i, j) \mid i \neq j \in \{0,...,n-1\} \wedge X^2_{i,j} \geq 3.84\} \qquad \textbf{(Persamaan 2.5)}$$

di mana:

- n adalah panjang individu

Kemudian, algoritma untuk pembentukan *dependency tree* didefinisikan sebagai berikut:

Algoritma Pembentukan Dependency Graph

1. *set* $V \leftarrow \{0,\ldots,n\text{-}1\}$

   set $A \leftarrow V$

   set $E \leftarrow \varnothing$

2. $v \leftarrow$ *vertex* dari A

   tambahkan v ke R

3. hapus v dari A
4. jika tidak ada lagi *vertex* di A maka *finish*
5. jika di D tidak ada lagi dependensi antara v dan v' dimana $v \in A$ dan $v' \in V/A$, maka kembali ke langkah 2
6. *set* v sebagai *vertex* dari A yang paling besar nilai $X^2_{v,v'}$ dari semua $v \in A$ dan $v' \in V/A$
7. tambahkan *edge* (v,v') ke dalam himpunan *edge* E
8. lakukan langkah 3

Ide dasar dari algoritma tersebut sangat sederhana. Algoritma tersebut mirip dengan algoritma untuk mendapatkan *spanning tree* yang besar. Pertama, sembarang *vertex* ditambahkan pada *graph*. Lalu, *vertex* yang memiliki ketergantungan terbesar dengan *vertex* terdahulu tersebut dan *edge* yang berhubungan ditambahkan pada *graph*. Langkah terakhir adalah mengulang sampai tidak ada lagi dependensi antara *vertex-vertex* yang sudah ditambahkan ke *graph* dengan sisanya (yang belum dimasukkan ke *graph*). Lalu sembarang *vertex* ditambahkan lagi pada *graph* dan proses berulang. Seluruh proses terus berulang sampai semua *vertex* ditambahkan pada *graph*. Efeknya adalah terbentuknya sebuah *acyclic graph* dengan nilai statistik chi-kuadrat maksimal dari seluruh *vertex* yang terhubung. *Graph* yang dihasilkan tidak harus terhubung, sebagaimana telah disebutkan sebelumnya. Saat *vertex* pertama dari setiap komponen (yang telah berhasil dibuat dengan menambahkan *vertex-vertex* berdasarkan dependensinya) ditambahkan ke dalam *graph*, *vertex* itu ditambahkan

juga ke dalam himpunan R (yaitu himpunan *vertex-vertex* spesial, satu untuk setiap komponen dari *graph* yang dihasilkan).

Karena himpunan A telah diinisialisasikan sebagai himpunan terbatas dari *vertex-vertex* dan dalam setiap *cycle* setidaknya satu *vertex* dihapus darinya, algoritma selalu *finish* sehingga terdefinisi dengan baik. Kompleksitas waktu dari algoritma adalah $O(n^3)$.

2.2.3. Pembentukan Individu Baru

Untuk membentuk individu baru, *dependency graph* yang telah didefinisikan sebelumnya $G = (V,E,R)$ digunakan. Untuk setiap individu, nilai posisi yang terkandung di R dihasilkan oleh *univariate marginal frequency*. Lalu, jika terdapat posisi v yang belum dihasilkan dan terhubung pada beberapa posisi v' (berdasarkan pada himpunan *edge* E), nilai posisi akan dibentuk menggunakan probabilitas kondisional (lihat persamaan 2.1) untuk posisi v yang telah diberikan oleh nilai pada posisi v'. Langkah terakhir diulang sampai nilai untuk seluruh posisi dibentuk.

Dalam deskripsi berikut tentang algoritma penghasil individu baru, nampak satu himpunan penting di antara himpunan-himpunan yang mendeskripsikan *graph* G. Himpunan ini dianggap sebagai K yang merupakan himpunan dari semua posisi yang telah dibentuk. Individual ini adalah sebuah rangkaian dengan panjang n dan akan dinotasikan dengan x. Bit ke-i akan dinotasikan dengan x_i.

2.2.3.1. Algoritma untuk Pembentukan Individu Baru

1. *set* $K \leftarrow V$
2. bentuk x_r untuk semua $r \in R$ menggunakan *univariate frequencies*, yaitu *set* pada nilai α dengan probabilitas $p_r(\alpha)$

   set $K \leftarrow K \setminus R$
3. jika K telah kosong, maka *finish*
4. pilih k dari K yang memiliki koneksi dengan k' dari $V \setminus K$ dalam *graph* G
5. bentuk X_k menggunakan probabilitas kondisional yang diberi nilai x_k, yaitu *set* pada nilai α dengan probabilitas $p_{k,k'}(\alpha|x_{k'})$
6. hapus k dari himpunan K
7. kembali ke langkah 4

Lalu himpunan K diinisialisasikan sebagai sebuah himpunan terbatas dan di setiap *cycle* setiap *vertex* dihapus dari K. Untuk setiap komponen, setidaknya satu *vertex* dibentuk terlebih dahulu, sehingga algoritma terdefinisi dengan baik. Pembentukan satu individual dapat dilakukan dalam $O(n)$ langkah. Pembentukan individu yang berbeda adalah independen, karena itu algoritma pembentukan individu baru cocok untuk paralelisasi.

2.2.4. Deskripsi BMDA

Setelah algoritma untuk membentuk sebuah *dependency graph* dan pembentukan individu baru dideskripsikan, pada sesi ini BMDA akan dideskripsikan. Dalam BMDA, populasi dibentuk pertama kali secara acak. Dari populasi ini, individu yang lebih baik dipilih. *Univariate* dan *bivariate marginal*

frequency untuk individu-individu ini kemudian dikalkulasi. Menggunakan frekuensi ini, sebuah *dependency graph* dibentuk. Dengan *dependency graph* itu, individu-individu baru dihasilkan. Individu baru kemudian ditambahkan pada populasi lama dari tempat asal individu-individu tersebut dipilih sebelumnya. Mereka menggantikan beberapa yang lama, biasanya yang terburuk dari mereka, sehingga jumlah individu-individu dalam populasi tetap konstan. Dari populasi baru, individu-individu dipilih kembali. Proses ini, dimulai dengan pemilihan individu yang lebih baik dan berakhir dengan penambahan individu-individu baru pada populasi lama, diulang sampai ditemukan kondisi terakhir (*termination criteria*). *Termination criteria* itu menyebabkan algoritma berhenti jika sudah menemukan hasil optimum atau keragaman populasi terlalu rendah. Nilai optimum biasanya tidak diketahui oleh para keturunannya. Inilah alasan mengapa kondisi kedua menjadi hal penting. Saat keragaman menjadi terlalu rendah, hampir semua individu dalam populasi adalah sama. Ini berarti bahwa tidak ada cukup informasi dalam populasi untuk membuat individu-individu baru yang cocok dengan masalah yang lebih baik daripada yang telah ditemukan.

2.2.4.1. Algoritma Bivariate Marginal Distribution

1. *set* $t \leftarrow 0$

   buat populasi awal $P(0)$ secara *random*

2. pilih *parents* $S(t)$ dari $P(0)$

   kalkulasi *univariate frequencies* p_i dan *bivariate frequencies* $p_{i,j}$ untuk *selected set* $S(t)$

3. buat *dependency graph* $G = (V,E,R)$ menggunakan *frequencies* p_i dan $p_{i,j}$
4. buat himpunan dari individu-individu baru $O(t)$ menggunakan *dependency graph* G dan *frequencies* p_i dan $p_{i,j}$
5. gantikan beberapa individu dari $P(t)$ dengan individu-individu baru $O(t)$ *set* $t \leftarrow t + 1$
6. jika *termination criteria* tidak ditemukan, lakukan langkah 2

Termination criteria dari kurangnya keragaman didefinisikan sebagai berikut: jika semua *univariate frequencies* ($\varepsilon > 0$) lebih dekat ke 0 atau 1, algoritma dihentikan. Jika ini masalahnya, dapat dikatakan algoritma *ε-converged* (populasi telah konvergen). Untuk mengetahui keragaman populasi, digunakan persamaan sebagai berikut [5]:

$$x(p) = \frac{4}{n}\sum_{i=0}^{n-1}(p_i(1) - \frac{1}{2})^2 \qquad \textbf{(Persamaan 2.6)}$$

di mana:

- *x(p)* merupakan parameter keragaman populasi
- *p* merupakan vektor univariate marginal frequencies $p_i(1)$
- *n* merupakan panjang individu

Semakin dekat parameter tersebut ke angka 1, semakin kecil keragaman individu-individu dalam populasi.

BAB 3

PERANCANGAN APLIKASI OPTIMASI

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

BAB 3

PERANCANGAN APLIKASI OPTIMASI

Pada bab ini akan dijelaskan perancangan sistem yang meliputi perancangan aplikasi optimasi GA dan perancangan aplikasi optimasi BMDA. Pada bagian selanjutnya akan diuraikan tentang antarmuka yang disesuaikan dengan keperluan untuk melakukan uji coba.

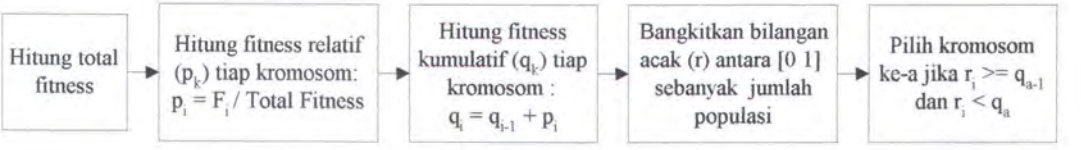
3.1. Desain Aplikasi Optimasi

Pada bagian ini akan dijelaskan desain sistem dalam bentuk diagram alir atau flowcart untuk masing-masing algoritma.

3.1.1. Desain Aplikasi Optimasi GA

Pada bagian ini akan dijelaskan diagram alir untuk algoritma genetika dan beberapa komponen algoritma tersebut.

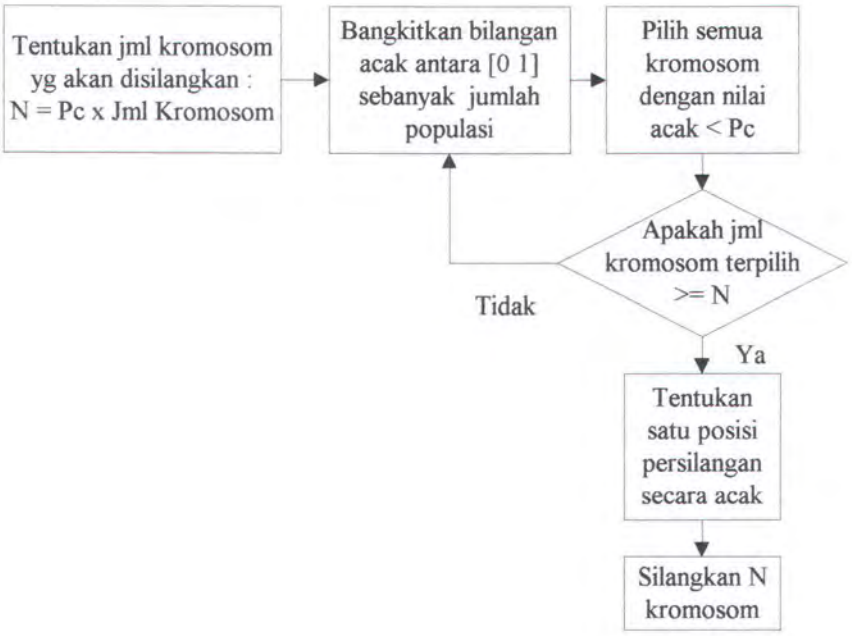
3.1.1.1. Diagram Alir Seleksi dengan Metode Seleksi Roda Roulette



Gambar 3.1 Diagram alir seleksi dengan metode Seleksi Roda Roulette

Gambar 3.1 menunjukkan alur seleksi dengan metode roda roulette. Seleksi ini bertujuan untuk memberikan kesempatan yang lebih besar bagi anggota populasi yang memiliki fitness yang lebih baik untuk melakukan reproduksi.

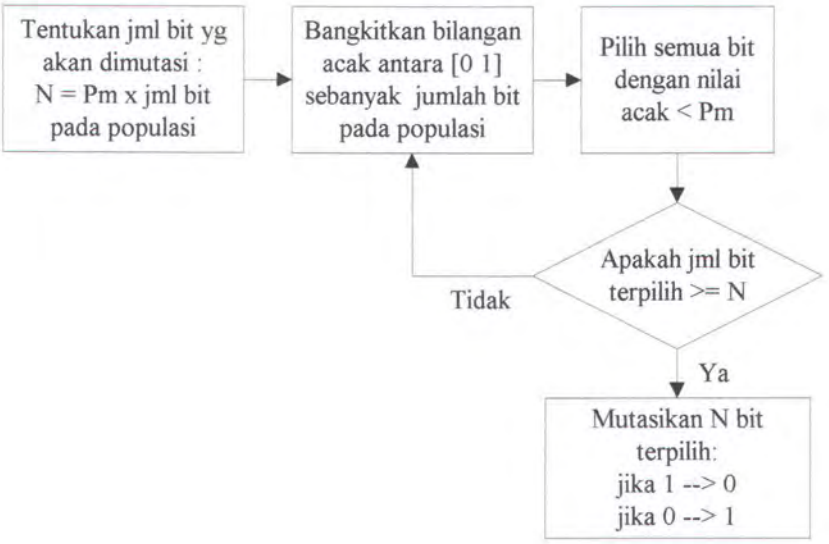
3.1.1.2. Diagram Alir Rekombinasi dengan Metode Penyilangan Satu Titik



Gambar 3.2 Diagram alir rekombinasi dengan metode Penyilangan Satu Titik

Seperti terlihat pada gambar 3.2, pemilihan individu atau kromosom yang akan mengalami penyilangan dilakukan secara acak. Jumlah kromosom yang akan disilangkan adalah sebanyak N yang didapat melalui perkalian peluang crossover (Pc) dengan jumlah kromosom dalam populasi. Penyilangan dilakukan terhadap sepasang kromosom atau 2 individu sehingga penyilangan dilakukan sebanyak N/2 kali.

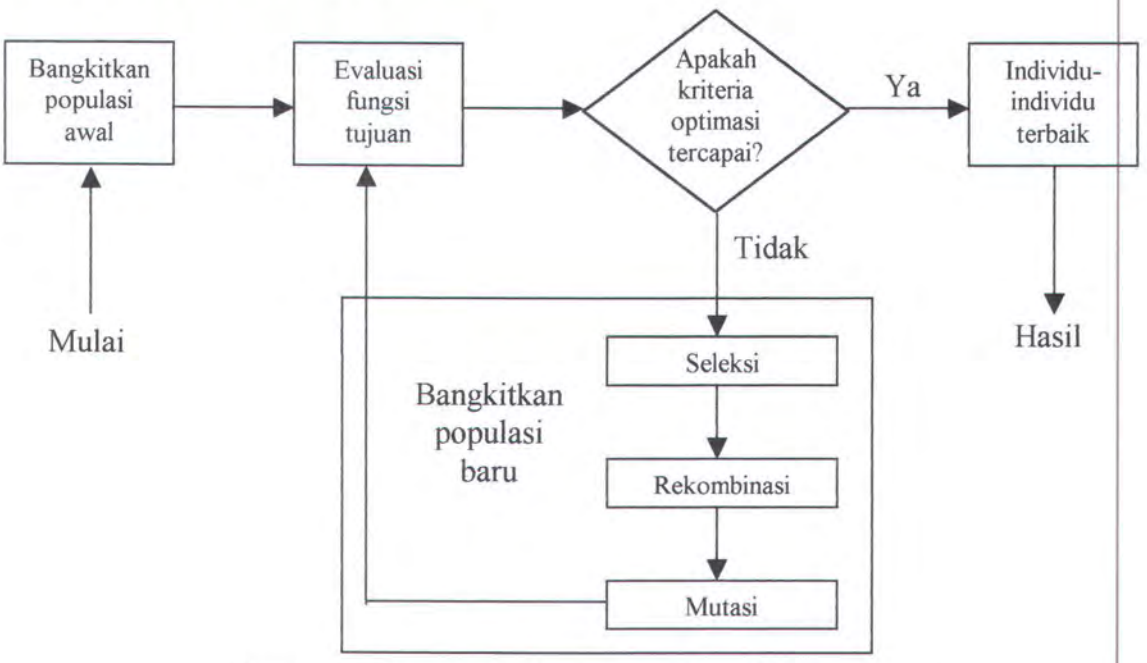
3.1.1.3. Diagram Alir Mutasi



Gambar 3.3 Diagram alir mutasi

Gambar 3.3 menunjukkan mutasi untuk bilangan biner. Pemilihan bit yang akan mengalami mutasi dilakukan secara acak.

3.1.1.4. Diagram Alir GA Sederhana



Gambar 3.4 Diagram alir algoritma genetika sederhana.

Gambar 3.4 menunjukkan diagram alir GA secara umum.

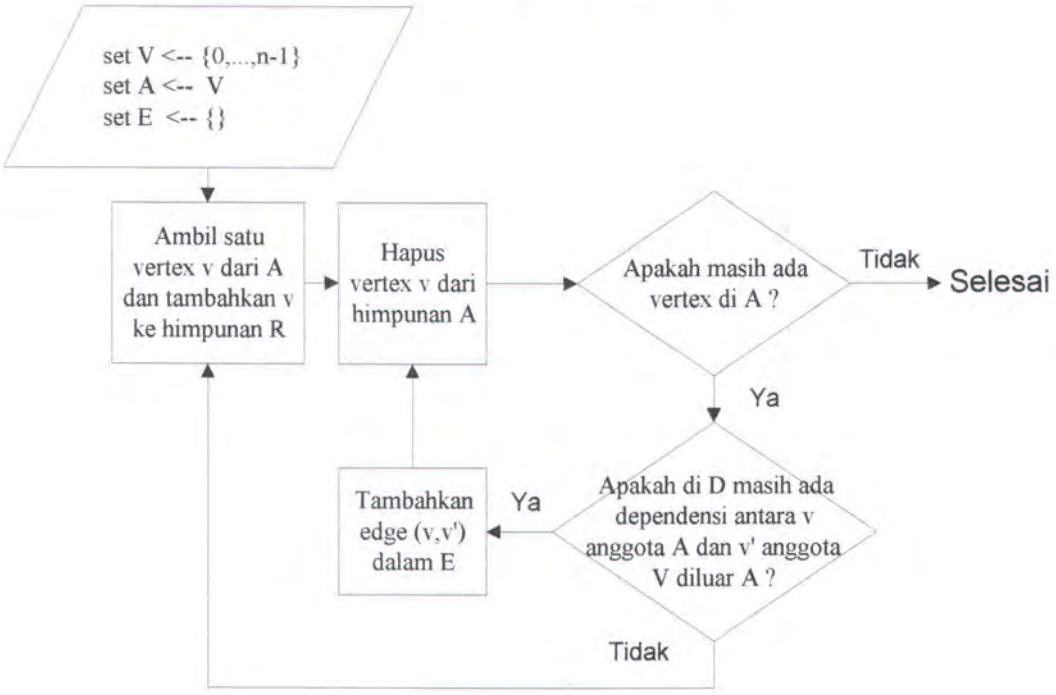
3.1.2. Desain Aplikasi Optimasi BMDA

Pada bagian ini akan dijelaskan diagram alir untuk BMDA dan beberapa komponen algoritma tersebut.

3.1.2.1. Diagram Alir Seleksi Parents

Untuk metode seleksi *parents* pada BMDA, penulis memilih menggunakan metode Seleksi Roda Roulette dengan tujuan memberikan kesempatan yang labih besar pada individu dengan nilai fitness yang lebih baik untuk bereproduksi. Diagram alir dapat dilihat pada gambar 3.1.

3.1.2.2. Diagram Alir Pembentukan Dependency Graph

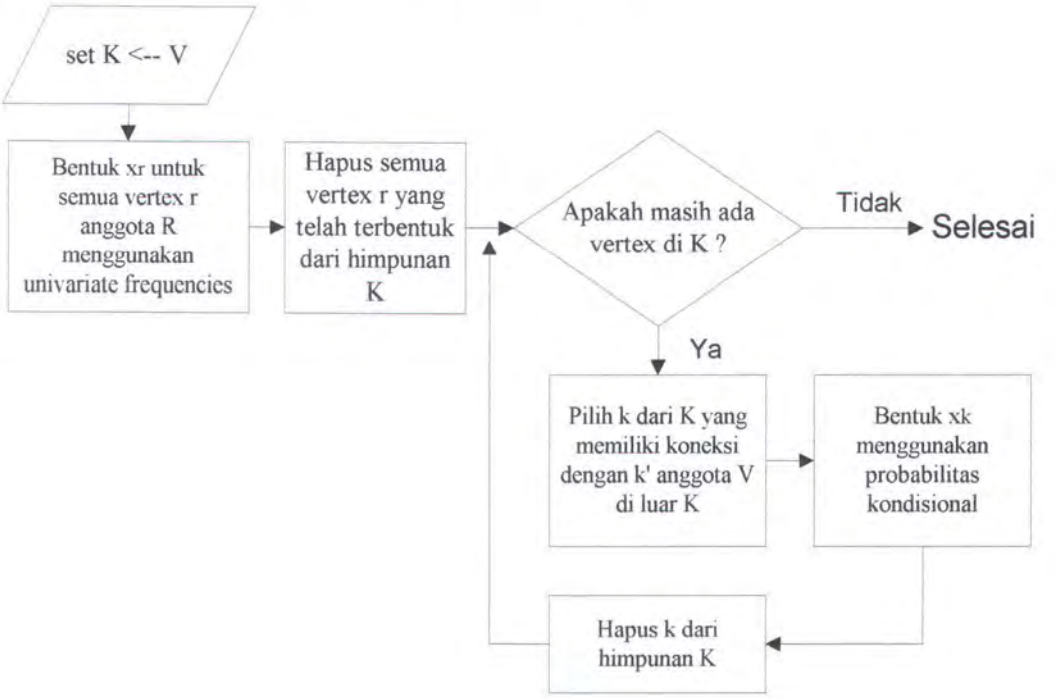


Gambar 3.5 Diagram alir pembentukan Dependency Graph.

Seperti telah dijelaskan dalam bab sebelumnya, *dependency graph* didefinisikan oleh 3 himpunan yaitu *V* yang merupakan himpunan vertex-vertex (dalam hal ini

adalah posisi-posisi bit dalam kromosom), E yang merupakan himpunan dari edge-edge, dan R yang merupakan himpunan vertex-vertex yang mengawali komponen-komponen pembentuk *dependency graph*. Sedangkan himpunan D merupakan himpunan dari seluruh pasangan vertex (posisi bit dalam kromosom) yang memiliki dependensi satu sama lain. Algoritma ini berakhir dengan A sebagai himpunan kosong yang berarti seluruh vertex telah diproses. Diagram alir pembentukan *dependency graph* dapat dilihat pada Gambar 3.5.

3.1.2.3. Diagram Alir Pembentukan Individu Baru

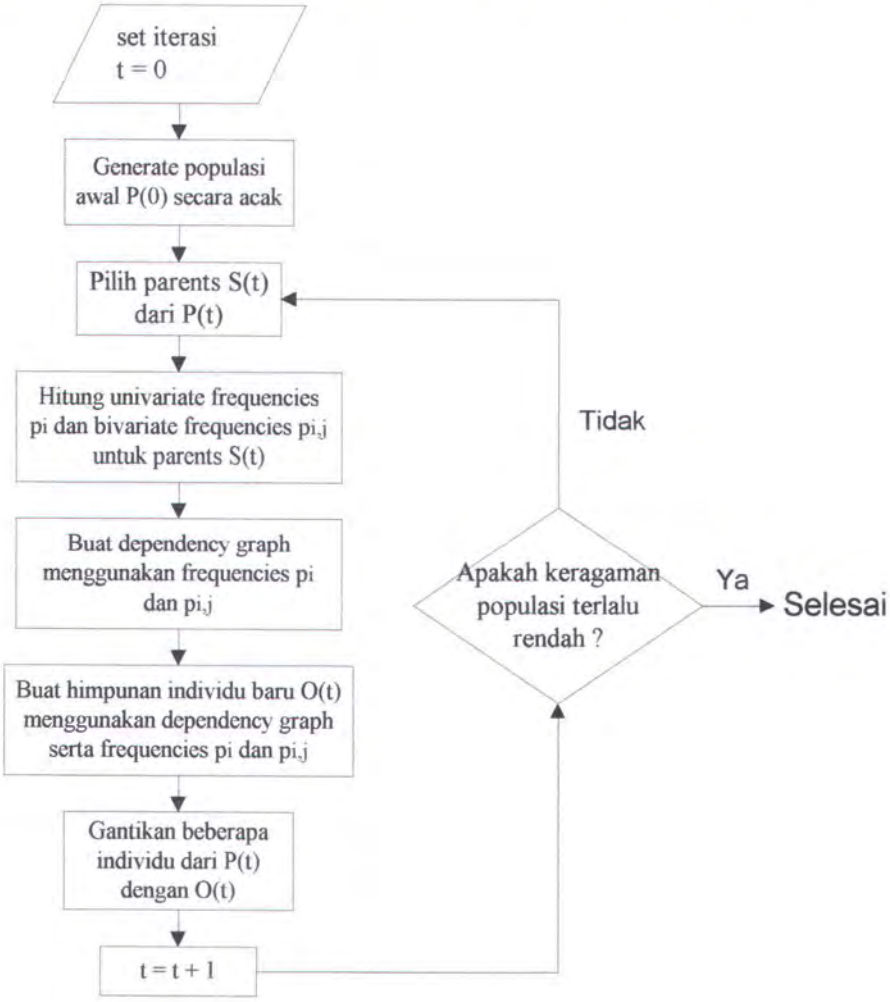


Gambar 3.6 Diagram alir pembentukan individu baru pada BMDA.

Individu baru dibentuk dengan menggunakan *dependency graph* yang telah dibentuk sebelumnya. Untuk setiap individu, nilai posisi yang terdapat pada R dihasilkan dengan menggunakan *univariate frequencies*, sedangkan posisi lainnya dihasilkan dengan menggunakan probabilitas kondisional. Pembentukan individu

yang berbeda adalah independen. Dari sebuah *dependency graph*, bisa dihasilkan lebih dari satu individu baru yang berbeda satu sama lain. Diagram alir pembentukan individu baru dapat dilihat pada Gambar 3.6.

3.1.2.4. Diagram Alir Algoritma Bivariate Marginal Distribution



Gambar 3.7 Diagram alir algoritma BMDA.

Seperti terlihat pada Gambar 3.7, penggantian individu pada BMDA dilakukan pada individu dari populasi lama yang memiliki nilai fitness terburuk. Algoritma ini berhenti apabila keragaman individu di dalam populasi terlalu rendah karena

hal ini menyebabkan kurangnya informasi dalam populasi untuk membuat individu-individu baru yang lebih baik.

3.2. Keterangan Tentang Masalah Optimasi

3.2.1. Fungsi Onemax

Onemax dipilih untuk menguji kedua algoritma karena Onemax sering digunakan dalam pengujian algoritma optimasi dan dianggap mewakili fungsi linear. Onemax merupakan fungsi linear sederhana dengan nilai semua bitnya adalah 1. Ini berarti nilai maksimal dari fungsi ini adalah jumlah dari seluruh bit.

Persamaan: $f_{onemax} = \sum_{i=0}^{n-1} x_i$ **(Persamaan 3.1)**

di mana x_i adalah nilai bit pada posisi ke-i pada rangkaian x dan n adalah jumlah bit.

3.2.2. Fungsi De Jong F2

Fungsi De Jong F2 dipilih untuk menguji kedua algoritma karena fungsi ini sering digunakan dalam uji coba algoritma optimasi pada masalah-masalah numerik. Nama fungsi ini diambil dari nama orang yang membuat fungsi tersebut yang juga seorang pengajar di University of Wisconsin USA.

Fungsi:

$$f(x_1,x_2) = 100\,(x_1^2 - x_2)^2 + (1 - x_1)^2$$ **(Persamaan 3.2)**

dimana: $-2{,}048 \leq X_1 \leq 2{,}048$

$-2{,}048 \leq X_2 \leq 2{,}048$

Fungsi ini mempunyai dua parameter yaitu X_1 dan X_2 yang akan mencapai nilai minimal yaitu 0 bila masing-masing parameter tersebut bernilai 1.

Berikut ini penjelasan mengenai bentuk representasi data untuk fungsi De Jong F2:

- nilai x_1 memiliki batas bawah –2,048 dan batas atas 2,048
- nilai x_2 memiliki batas bawah –2,048 dan batas atas 2,048

Dari data di atas dapat dihitung range nilai untuk x_1 dan x_2 dengan cara mengurangi nilai batas atas dengan nilai batas bawah sehingga didapatkan range untuk masing-masing nilai:

- range nilai x_1 = 4,096
- range nilai x_2 = 4,096

Kemudian range nilai tersebut harus direpresentasikan ke dalam bentuk biner.

- 4,096 = 4096 x 10^{-3} → untuk ketelitian 3 angka di belakang koma
- $(2^{11} = 2048) < 4096 \leq (2^{12} = 4096)$ sehingga panjang representasi biner untuk masing-masing nilai x_1 dan x_2 adalah 12.

Untuk mengetahui nilai x_1 dan x_2 dari bentuk biner, dilakukan penghitungan sebagai berikut:

$$x = a + desimal(x_{biner}).\frac{b-a}{2^m - 1}$$

di mana:

- x merupakan nilai x_1 atau x_2
- a merupakan nilai batas bawah x_1 atau x_2
- $desimal(x_{biner})$ merupakan fungsi untuk merubah biner ke desimal
- x_{biner} merupakan representasi nilai x_1 atau x_2 dalam bentuk biner
- b merupakan nilai batas atas x_1 atau x_2
- m merupakan panjang representasi biner

3.2.3. Traveling Salesman Problem

TSP dipilih dalam pengujian kedua algoritma karena TSP merupakan masalah optimasi yang sudah sering digunakan oleh para peneliti karena kesederhanaannya, pentingnya, serta hubungannya dengan masalah kombinatorial.

Konsep TSP sangat sederhana, yaitu seorang salesman harus mengunjungi setiap kota tujuan hanya sekali dan kembali lagi ke kota asal. Dengan adanya jarak/beban biaya antar kota, salesman tersebut harus menempuh rute dengan jarak/beban biaya terkecil.

Berikut ini penjelasan mengenai bentuk representasi data untuk masalah TSP:

- untuk masalah TSP dengan jumlah N kota, panjang kromosom adalah $N\log_2 N$
- panjang representasi biner (M) untuk masing-masing kota adalah $\log_2 N$.

Misalkan kita memiliki kromosom sebagai berikut:

1	2	3	4	5	6	7	8
110	010	001	111	101	011	001	000

maka rute urutan kota pada kromosom tersebut adalah:

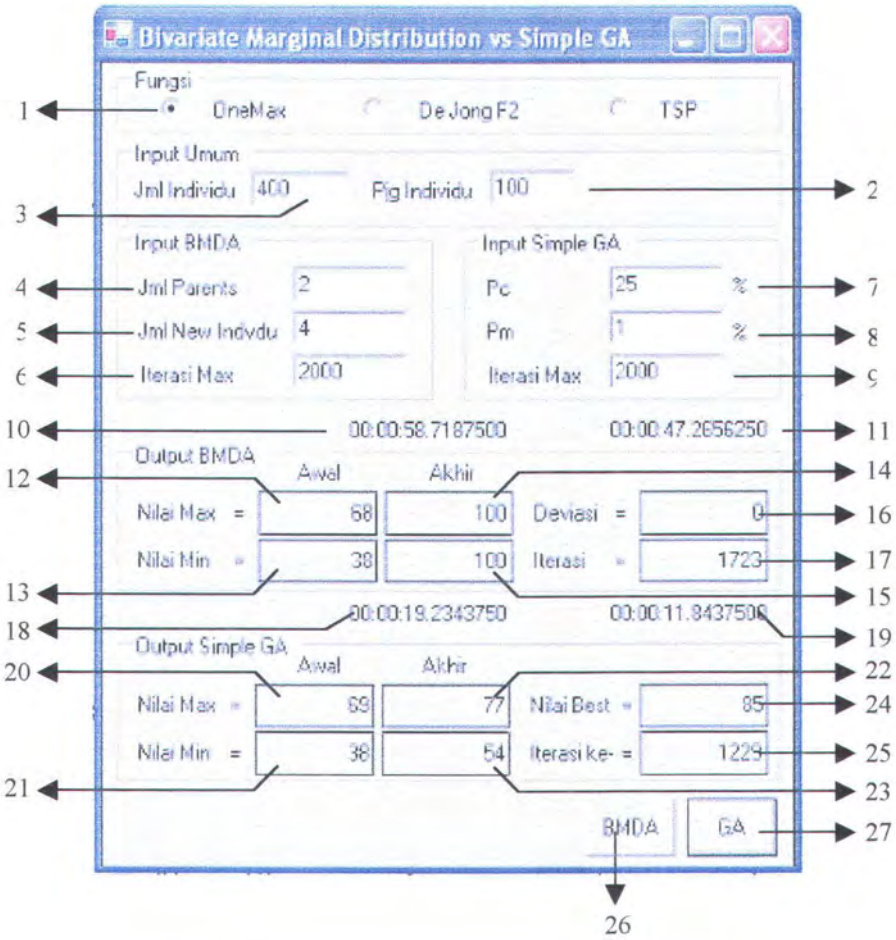
8	3	7	2	6	5	1	4
000	001	001	010	011	101	110	111

3.3. Antarmuka Aplikasi Optimasi

Pada bagian sebelumnya, perancangan aplikasi optimasi untuk GA dan BMDA telah dilakukan. Dalam bagian ini, rancangan antarmuka aplikasi akan dijelaskan.

3.3.1. Deskripsi Antarmuka untuk Masalah Optimasi Onemax

Antarmuka untuk masalah optimasi Onemax dapat dilihat pada Gambar 3.8 di bawah ini.



Gambar 3.8 Antarmuka untuk masalah optimasi Onemax.

Keterangan antarmuka:

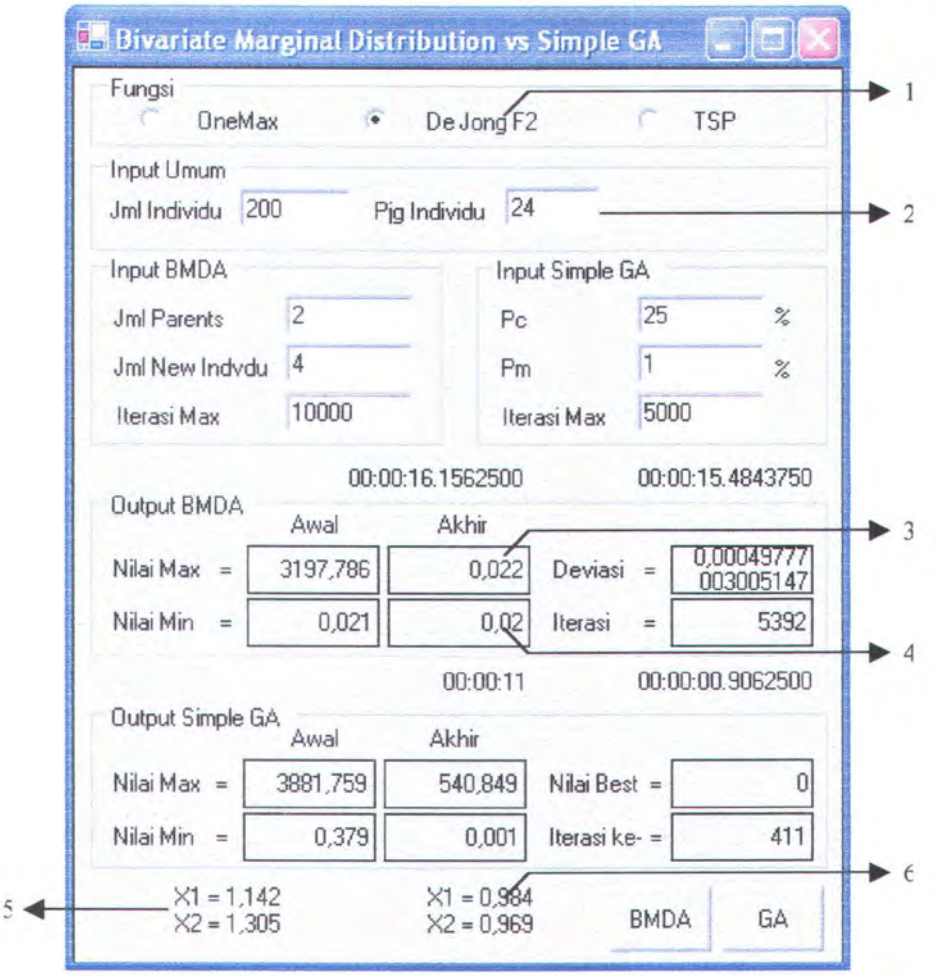
1. Merupakan tombol pilihan untuk memilih masalah optimasi yang akan dijalankan dalam hal ini adalah fungsi Onemax.
2. Kotak input untuk panjang kromosom/individu.
3. Kotak input untuk besar populasi atau jumlah individu dalam populasi.
4. Kotak input algoritma BMDA untuk jumlah *parents* yang akan bereproduksi dalam setiap iterasi.
5. Kotak input algoritma BMDA untuk jumlah individu baru yang akan dibentuk dalam setiap iterasi.
6. Kotak input untuk menentukan jumlah iterasi maksimal yang akan dijalankan oleh algoritma BMDA.
7. Kotak input GA untuk persentase besar peluang persilangan (Pc) yang diinginkan.
8. Kotak input GA untuk persentase besar peluang mutasi (Pm) yang diinginkan.
9. Kotak input untuk menentukan jumlah iterasi maksimal yang akan dijalankan oleh algoritma GA sederhana.
10. Output yang menunjukkan lama proses algoritma BMDA sampai iterasi yang terakhir.
11. Output yang menunjukkan lama proses algoritma BMDA sampai iterasi yang menghasilkan individu dengan fitness terbaik.
12. Output BMDA yang menunjukkan nilai fitness terbaik pada populasi awal untuk fungsi Onemax.

13. Output BMDA yang menunjukkan nilai fitness terburuk pada populasi awal untuk fungsi Onemax.
14. Output BMDA yang menunjukkan nilai fitness terbaik dalam populasi pada iterasi terakhir untuk fungsi Onemax.
15. Output BMDA yang menunjukkan nilai fitness terburuk dalam populasi pada iterasi terakhir untuk fungsi Onemax.
16. Output BMDA yang menunjukkan tingkat konvergensi populasi. Semakin rendah nilai output, menunjukkan semakin konvergen populasi.
17. Output BMDA yang menunjukkan jumlah iterasi yang dijalankan oleh algoritma BMDA.
18. Output yang menunjukkan lama proses algoritma GA sampai iterasi maksimal.
19. Output yang menunjukkan lama proses algoritma GA sampai iterasi yang menghasilkan individu dengan fitness terbaik dari semua generasi.
20. Output GA yang menunjukkan nilai fitness terbaik pada populasi awal untuk fungsi Onemax.
21. Output GA yang menunjukkan nilai fitness terburuk pada populasi awal untuk fungsi Onemax.
22. Output GA yang menunjukkan nilai fitness terbaik dalam populasi pada iterasi maksimal untuk fungsi Onemax.
23. Output GA yang menunjukkan nilai fitness terburuk dalam populasi pada iterasi maksimal untuk fungsi Onemax.
24. Output GA yang menunjukkan nilai fitness terbaik dari semua generasi.

25. Output GA yang menunjukkan generasi atau iterasi yang menghasilkan individu dengan nilai fitness terbaik dari semua generasi.
26. Tombol untuk menjalankan aplikasi optimasi BMDA.
27. Tombol untuk menjalankan aplikasi optimasi GA.

3.3.2. Deskripsi Antarmuka untuk Masalah Optimasi De Jong F2

Gambar 3.9 menunjukkan antarmuka untuk masalah optimasi De Jong F2.



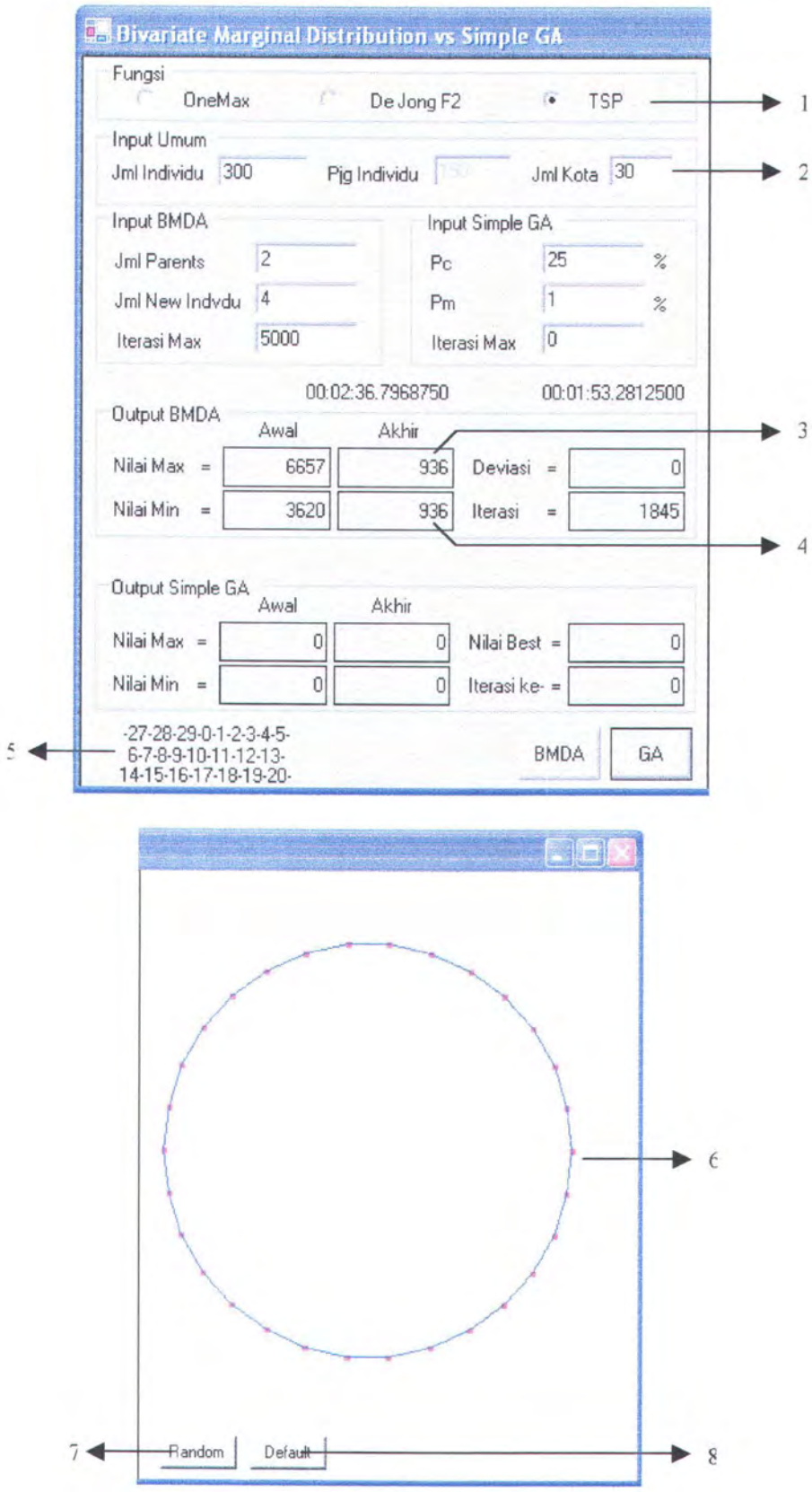
Gambar 3.9 Antarmuka untuk masalah optimasi De Jong F2.

Pada umumnya, antarmuka untuk fungsi OneMax dan De Jong F2 adalah sama. Untuk itu, berikut ini merupakan penjelasan antarmuka untuk fungsi De Jong F2 yang berbeda dengan antarmuka untuk fungsi Onemax:

1. Tombol pilihan untuk menjalankan aplikasi masalah optimasi Fungsi De Jong F2.
2. Panjang individu untuk masalah optimasi Fungsi De Jong F2 adalah 24. Penjelasan mengenai hal ini akan diberikan pada bab selanjutnya.
3. Output BMDA yang menunjukkan nilai fitness terburuk dalam populasi pada iterasi terakhir untuk fungsi De Jong F2.
4. Output BMDA yang menunjukkan nilai fitness terbaik dalam populasi pada iterasi terakhir untuk fungsi De Jong F2.
5. Output BMDA yang menunjukkan nilai x_1 dan x_2 yang menghasilkan nilai fitness terbaik dalam populasi untuk fungsi De Jong F2.
6. Output GA yang menunjukkan nilai x_1 dan x_2 yang menghasilkan nilai fitness terbaik dalam populasi untuk fungsi De Jong F2.

3.3.3. Deskripsi Antarmuka untuk Masalah Optimasi TSP

Pada bagian ini akan dijelaskan mengenai antarmuka untuk masalah optimasi TSP (Gambar 3.10). Untuk masalah optimasi TSP ini, antarmuka diberi tambahan bidang kanvas untuk memperlihatkan tampilan kota dan rute yang terbentuk dari hasil optimasi.



Gambar 3.10 Antarmuka untuk masalah optimasi TSP.

Berikut penjelasan antarmuka untuk masalah optimasi TSP:

1. Tombol pilihan untuk menjalankan aplikasi masalah optimasi *Traveling Salesman Problem* atau TSP.
2. Kotak input untuk menentukan jumlah kota. Bila pilihan tombol adalah Default, maka jumlah kota adalah 30.
3. Output BMDA yang menunjukkan nilai fitness terburuk dalam populasi pada iterasi terakhir untuk kasus TSP.
4. Output BMDA yang menunjukkan nilai fitness terbaik dalam populasi pada iterasi terakhir untuk kasus TSP.
5. Output BMDA yang menunjukkan urutan kota atau rute yang memiliki nilai fitness terbaik dalam populasi pada iterasi terakhir untuk kasus TSP.
6. Bidang kanvas yang menunjukkan posisi kota (kotak warna merah) dan juga rute hasil optimasi (garis warna biru).
7. Tombol untuk melakukan pengacakan posisi sejumlah kota pada kotak input jumlah kota.
8. Tombol untuk menampilkan 30 kota dengan posisi yang tetap yang membentuk lingkaran.

Setelah proses pembuatan perangkat lunak tersebut selesai, maka langkah berikutnya adalah melakukan uji coba dan analisis untuk ketiga masalah yang akan diselesaikan. Hal tersebut akan diuraikan pada bab berikutnya.

BAB 4

UJI COBA DAN ANALISIS SISTEM

BAB 4

UJI COBA DAN ANALISIS SISTEM

Pada bab ini akan dijelaskan proses uji coba dan analisis terhadap aplikasi yang telah dibangun untuk menyelesaikan masing-masing kasus, yang dilakukan untuk mengetahui hasil implementasi serta kinerja dari masing-masing algoritma.

4.1. Lingkungan Uji Coba

Lingkungan dari uji coba adalah sebagai berikut:

- sistem operasi Windows XP.
- spesifikasi hardware Intel Pentium IV Celeron – 1,8Ghz.
- bahasa pemrograman yang digunakan Microsoft Visual C#.Net.

4.2. Skenario Uji Coba

Kinerja algoritma GA dan BMDA dipengaruhi oleh parameter masing-masing. Untuk itu, sebelum membandingkan kinerja kedua algoritma tersebut, perlu juga diuji pengaruh parameter terhadap kinerja masing-masing algoritma. Algoritma GA dan BMDA memiliki parameter yang berbeda. Parameter untuk GA adalah peluang *crossover* (Pc) dan peluang mutasi (Pm). Sedangkan parameter untuk BMDA adalah jumlah *parents* dan jumlah individu baru. Pengaruh besar populasi juga akan diuji pada masing-masing algoritma. Uji coba

akan dilakukan untuk ketiga contoh kasus yang ada dengan menggunakan parameter-parameter terbaik dari uji coba tersebut.

4.3. Pelaksanaan Uji Coba

Pada bagian ini akan dijelaskan proses uji coba sesuai dengan skenario yang sudah dijelaskan sebelumnya.

4.3.1. Uji Coba untuk Kasus Onemax-20

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian kasus Onemax-20. Data hasil rata-rata uji coba disajikan dalam bentuk tabel.

4.3.1.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 20, jumlah individu dalam populasi sebesar 50. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 1500. Hasil uji coba dapat dilihat pada Tabel 4.1 sampai dengan Tabel 4.4.

4.3.1.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 20, peluang crossover (Pc) sebesar 50%, dan peluang mutasi (Pm) GA sebesar 10%.

Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 1500. Hasil uji coba dapat dilihat pada Tabel 4.5.

Tabel 4.1 Tabel hasil optimasi rata-rata penyelesaian Onemax-20 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	19,95	20,00	20,00	20,00	20,00
10	19,90	19,85	19,90	20,00	19,95
20	19,95	19,90	19,95	20,00	19,90
30	19,95	20,00	19,95	19,95	20,00
40	19,45	19,70	19,70	19,45	19,65
50	19,05	18,85	18,70	19,00	18,95

Tabel 4.2 Tabel jumlah iterasi rata-rata generasi terbaik penyelesaian Onemax-20 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	580	386	424	445	319
10	503	422	384	469	388
20	546	228	277	452	359
30	415	579	489	491	350
40	459	285	323	447	352
50	313	413	347	308	311

Tabel 4.3 Tabel waktu rata-rata untuk mencapai generasi terbaik pada penyelesaian Onemax-20 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	0,22418	0,14684	0,18040	0,17341	0,12496
10	0,23198	0,16949	0,17030	0,19059	0,18356
20	0,21557	0,09137	0,12340	0,17965	0,14293
30	0,16792	0,24060	0,21240	0,20230	0,14371
40	0,18668	0,11874	0,13900	0,18433	0,14215
50	0,12653	0,16638	0,16480	0,12965	0,12809

4.3.1.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 20 dan jumlah individu dalam populasi sebesar 400. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba dapat dilihat pada Tabel 4.6 sampai dengan Tabel 4.8.

Tabel 4.4 Tabel waktu total rata-rata untuk mencapai iterasi maksimal pada penyelesaian Onemax-20 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	0,57341	0,58435	0,65700	0,58357	0,58356
10	0,70777	0,62497	0,70152	0,62965	0,72416
20	0,60775	0,61480	0,71792	0,62966	0,61950
30	0,62183	0,63358	0,61794	0,63201	0,62654
40	0,62653	0,62498	0,65543	0,63514	0,62496
50	0,61479	0,62340	0,71636	0,63357	0,62106

Tabel 4.5 Tabel pengaruh besar populasi pada penyelesaian Onemax-20 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
30	19,90	529	0,45778	0,15622
40	19,90	210	0,60620	0,12028
50	19,95	388	0,18356	0,72416

Tabel 4.6 Tabel hasil optimasi rata-rata penyelesaian Onemax-20 dengan BMDA.

	Jumlah Parents		
New Idv	2	3	4
2	19,95	19,95	17,7
4	19,55	19,80	17,6

Tabel 4.7 Tabel jumlah iterasi rata-rata penyelesaian Onemax-20 dengan BMDA.

	Jumlah Parents		
New Idv	2	3	4
2	1500	1416	10170
4	763	736	6721

Tabel 4.8 Tabel waktu konvergensi rata-rata penyelesaian Onemax-20 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	2,8038	2,5483	14,7663
4	1,4937	1,3695	10,2031

4.3.1.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 20, jumlah parents 3, dan jumlah individu baru 2. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.9.

Tabel 4.9 Tabel pengaruh besar populasi pada penyelesaian Onemax-20 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
200	19,60	638	1,07496
300	19,95	1102	1,92965
400	19,95	1416	2,54830

4.3.2. Uji Coba untuk Kasus Onemax-75

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian kasus Onemax-75. Data hasil rata-rata uji coba disajikan dalam bentuk table nilai rata-rata.

4.3.2.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 75 dan jumlah individu dalam populasi sebesar 100. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 2000. Hasil uji coba ditunjukkan pada Tabel 4.10 sampai dengan Tabel 4.13.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Tabel 4.10 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-75 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	63,25	63,45	64,25	64,40	64,55
10	64,40	64,40	64,00	63,85	64,65
20	62,65	63,50	64,35	63,40	63,00
30	60,50	62,45	62,05	61,75	62,90
40	59,15	60,60	61,25	61,35	61,95
50	59,05	59,50	60,70	60,30	59,80

Tabel 4.11 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-75 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	759	1123	952	980	842
10	1245	952	1060	935	1150
20	1226	1252	1024	1275	1081
30	1144	833	1013	926	908
40	956	855	880	882	943
50	575	529	491	416	415

Tabel 4.12 Tabel waktu iterasi rata-rata untuk mendapatkan generasi terbaik pada penyelesaian Onemax-75 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	1,33199	1,97027	1,69137	1,76169	1,53512
10	2,24527	1,72262	1,94294	1,73903	2,14370
20	2,27809	2,32575	1,92652	2,40856	2,08511
30	2,16481	1,58200	1,94606	1,79137	1,78357
40	1,84607	1,66168	1,73825	1,73746	1,88277
50	1,16403	1,04528	0,97105	0,83355	0,84762

Tabel 4.13 Tabel waktu total rata-rata untuk mendapatkan iterasi maksimal pada penyelesaian Onemax-75 dengan GA.

Pm	Pc (%)				
	5	15	25	35	50
1	3,50465	3,51637	3,55386	3,58511	3,65621
10	3,60386	3,61636	3,67575	3,72496	3,73277
20	3,71167	3,71949	3,75857	3,77653	3,84839
30	3,77809	3,79215	3,83200	3,85778	3,92105
40	3,85153	3,92809	3,92809	3,93122	3,98434
50	3,93513	3,91637	3,95544	3,98045	4,07418

4.3.2.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 75, peluang crossover (Pc) sebesar 50%, dan peluang mutasi (Pm) GA sebesar 10%. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 1500. Hasil uji coba ditunjukkan pada Tabel 4.14.

Tabel 4.14 Tabel perbandingan besar populasi pada penyelesaian Onemax-75 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
100	64,65	1150	2,14370	3,73277
150	65,95	1107	3,04605	5,51246
200	66,95	1187	4,34294	7,33980

4.3.2.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 75 dan jumlah individu dalam populasi sebesar 200. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.15 sampai dengan Tabel 4.17.

4.3.2.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 75, jumlah parents 3, dan jumlah individu baru 2. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.18.

Tabel 4.15 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-75 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	74,9	74,95	53
4	74,4	74,7	52

Tabel 4.16 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-75 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	1457	1331	20000
4	803	713	20000

Tabel 4.17 Tabel waktu konvergensi rata-rata pada penyelesaian Onemax-75 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	29,3257	26,1778	229,2031
4	16,8765	14,1812	207,0312

Tabel 4.18 Tabel pengaruh besar populasi pada penyelesaian Onemax-75 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
100	74,85	713	13,43280
150	74,90	992	18,30307
200	74,95	1331	26,17780

4.3.3. Uji Coba untuk Kasus Onemax-100

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian kasus Onemax-100. Data hasil uji coba disajikan dalam bentuk tabel nilai rata-rata.

4.3.3.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 100, jumlah individu dalam populasi sebesar 100, dan iterasi maksimal 2000. Masing-masing uji coba dilakukan sebanyak 20 kali dengan. Hasil uji coba ditunjukkan pada Tabel 4.19 sampai dengan Tabel 4.22.

Tabel 4.19 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-100 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	79,55	80,70	80,50	81,05	81,50
10	81,05	81,35	81,75	81,85	82,00
20	77,75	78,95	79,75	80,10	81,55
30	76,20	78,05	78,60	79,25	79,35
40	74,10	76,70	77,50	78,65	79,00
50	74,05	75,30	77,15	77,50	78,65

Tabel 4.20 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-100 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	1037	981	797	979	911
10	1223	1300	1325	1263	1193
20	954	963	877	1092	1234
30	903	963	779	1055	973
40	72	687	977	734	874
50	721	807	605	814	969

Tabel 4.21 Tabel waktu iterasi rata-rata pada penyelesaian Onemax-100 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	2,36794	2,23277	1,83200	2,27339	2,15700
10	2,85152	3,03590	3,13122	3,03435	2,88901
20	2,29606	2,31794	2,13044	2,67417	3,06636
30	2,20231	2,37185	1,93668	2,69684	2,48434
40	1,78435	1,71638	2,47106	1,88668	2,27106
50	1,82497	2,04762	1,54918	2,10622	2,53746

Tabel 4.22 Tabel waktu total rata-rata untuk mendapat generasi terbaik pada penyelesaian Onemax-100 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	4,53200	4,55309	4,59372	4,63748	4,71715
10	4,66871	4,67575	4,72261	4,80074	4,83903
20	4,79919	4,81013	4,85308	4,90934	4,96246
30	4,87731	4,91402	4,96169	5,07497	5,08200
40	4,96560	4,99606	5,04840	5,13513	5,18512
50	5,03433	5,06403	5,10856	5,16638	5,21949

4.3.3.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 100, peluang crossover (Pc) sebesar 50%, dan peluang mutasi (Pm) GA sebesar 10%. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 2000. Hasil uji coba ditunjukkan pada Tabel 4.23.

Tabel 4.23 Tabel pengaruh besar populasi pada penyelesaian Onemax-100 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
100	82,00	1193	2,88901	4,83903
150	86,05	1116	4,05074	7,17418
200	87,95	1237	5,91559	9,43980

4.3.3.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 100, jumlah individu dalam populasi sebesar 200. Masing-masing uji coba dilakukan sebanyak 20 kali. Tabel 4.24 sampai dengan Tabel 4.26 menunjukkan hasil uji coba ini.

Tabel 4.24 Tabel hasil optimasi rata-rata pada penyelesaian Onemax-100 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	99,95	99,95	71
4	98,5	99,5	67

Tabel 4.25 Tabel jumlah iterasi rata-rata pada penyelesaian Onemax-100 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	1760	1552	20000
4	954	808	20000

Tabel 4.26 Tabel waktu konvergensi rata-rata pada penyelesaian Onemax-100 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	58,5257	48,7757	368,9062
4	34,2625	25,9632	372,8750

4.3.3.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada kasus Onemax dengan panjang individu 100, jumlah *parents* sebanyak 3, dan jumlah individu baru tiap iterasi sebanyak 2. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.27.

Tabel 4.27 Tabel pengaruh besar populasi pada penyelesaian Onemax-100 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
100	99,40	836	26,79918
150	99,90	1190	37,45075
200	99,95	1552	48,77570

4.3.4. Uji Coba untuk Kasus De Jong F2

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian fungsi De Jong F2. Data hasil uji coba disajikan dalam bentuk tabel nilai rata-rata.

4.3.4.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada fungsi De Jong F2 dengan jumlah individu dalam populasi sebesar 50. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 2000. Hasil uji coba dapat dilihat pada Tabel 4.28 sampai dengan Tabel 4.31.

Tabel 4.28 Tabel hasil optimasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	0,02820	0,01070	0,00755	0,00520	0,01365
10	0,00020	0,00005	0,00015	0,00010	0,00025
20	0,00010	0,00005	0,00010	0,00020	0,00000
30	0,00925	0,00270	0,00810	0,00450	0,00470
40	0,06350	0,07520	0,07295	0,05370	0,06190
50	0,09950	0,25200	0,20985	0,15160	0,36025

Tabel 4.29 Tabel jumlah iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	878	752	1041	680	708
10	695	640	678	626	849
20	669	621	619	555	675
30	583	374	328	763	753
40	452	510	332	388	404
50	270	204	166	282	255

Tabel 4.30 Tabel waktu iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	0,55310	0,47028	0,76480	0,43669	0,44136
10	0,45635	0,42418	0,44841	0,41247	0,59294
20	0,43043	0,40074	0,41559	0,36090	0,44059
30	0,37419	0,25183	0,22730	0,49841	0,49060
40	0,29606	0,32889	0,22419	0,25857	0,26559
50	0,17655	0,13670	0,11403	0,18592	0,16874

Tabel 4.31 Tabel waktu total rata-rata pada penyelesaian Fungsi DeJong F2 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	1,25699	1,25935	1,41377	1,27497	1,26091
10	1,32496	1,30933	1,34058	1,30699	1,38277
20	1,29215	1,29918	1,31325	1,31325	1,31482
30	1,29760	1,32341	1,34293	1,32420	1,32263
40	1,30699	1,31403	1,34137	1,32731	1,33122
50	1,31716	1,32810	1,32809	1,34193	1,34293

4.3.4.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada fungsi De Jong F2 dengan peluang crossover (Pc) sebesar 50% dan peluang mutasi (Pm) GA sebesar 20%. Masing-masing uji coba dilakukan sebanyak 20 kali dengan iterasi maksimal 2000. Hasil uji coba dapat dilihat pada Tabel 4.32.

Tabel 4.32 Tabel pengaruh besar populasi pada penyelesaian Fungsi DeJong F2 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
30	0,00035	934	0,41637	0,90465
40	0,00025	574	0,31716	1,11324
50	0,00000	675	0,44059	1,31482

4.3.4.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada fungsi De Jong F2 dengan jumlah individu dalam populasi sebesar 400. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.33 sampai dengan 4.35.

Tabel 4.33 Tabel hasil optimasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	0,00275	0,00250	0,23300
4	0,00360	0,00300	0,01400

Tabel 4.34 Tabel jumlah iterasi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	6280	7561	40000
4	2432	5934	40000

Tabel 4.35 Tabel waktu konvergensi rata-rata pada penyelesaian Fungsi DeJong F2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	23,8468	26,9484	142,0156
4	9,3616	21,9437	141,0468

4.3.4.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada fungsi De Jong F2 dengan jumlah *parents* sebanyak 3 dan jumlah individu baru tiap iterasi sebanyak 2. Masing-masing uji coba dilakukan sebanyak 20 kali. Hasil uji coba ditunjukkan pada Tabel 4.36.

Tabel 4.36 Tabel pengaruh besar populasi pada penyelesaian Fungsi DeJong F2 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
50	0,51190	611	1,37030
100	0,07550	1781	4,42490
400	0,00250	7561	26,94840

4.3.5. Uji Coba untuk Kasus TSP-Rute 1

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian TSP-Rute 1 yang berbentuk lingkaran (data koordinat ada di Lampiran 1). Data hasil uji coba disajikan dalam bentuk tabel nilai rata-rata.

4.3.5.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada kasus TSP dengan jumlah individu dalam populasi sebesar 100. Masing-masing uji coba dilakukan sebanyak 10 kali dengan iterasi maksimal 2000. Hasil uji coba dapat dilihat pada Tabel 4.37 sampai dengan Tabel 4.40.

Tabel 4.37 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	1958,00	1822,40	1636,60	1594,70	1469,00
10	1968,50	1883,20	1800,10	1813,20	1695,40
20	2537,70	2425,90	2264,50	2213,20	2127,20
30	3007,90	2454,30	2688,30	2269,70	2390,20
40	3010,90	2710,00	2458,90	2524,70	2457,50
50	2923,20	2712,70	2682,80	2594,80	2578,90

Tabel 4.38 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	978	1142	1392	1207	1365
10	1526	1359	1531	1470	1465
20	1268	1261	1003	761	1303
30	1232	628	788	1378	571
40	410	464	753	464	396
50	586	311	302	289	399

Tabel 4.39 Tabel waktu iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	5,37340	6,27967	7,68122	6,70776	7,68747
10	8,53590	7,65308	8,62497	8,39684	8,45621
20	7,26246	7,27496	5,83276	4,46089	7,74529
30	7,18119	3,72495	4,68745	8,19996	3,52652
40	2,48745	2,81559	4,50461	2,85778	2,57496
50	3,51557	1,89371	1,86246	1,79060	2,44058

Tabel 4.40 Tabel waktu total rata-rata pada penyelesaian TSP-Rute 1 dengan GA.

	Pc (%)				
Pm	5	15	25	35	50
1	10,80934	10,84215	10,90465	11,18747	11,11089
10	11,12498	11,14840	11,21560	11,33590	11,45934
20	11,32809	11,39527	11,46091	11,63590	11,74526
30	11,51559	11,63747	11,68121	11,84370	12,07497
40	11,79527	11,80153	11,72965	11,91869	12,24995
50	11,83277	11,87498	11,97184	12,00150	12,07653

4.3.5.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada pada kasus TSP dengan peluang crossover (Pc) sebesar 50% dan peluang mutasi (Pm) GA sebesar 1%. Masing-masing uji coba dilakukan sebanyak 10 kali dengan iterasi maksimal 2000. Hasil uji coba ini dapat dilihat pada Tabel 4.41.

Tabel 4.41 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 1 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
100	1469,00	1365	7,68747	11,11089
200	1317,90	1419	15,52340	21,74682
300	1180,20	1212	19,81870	32,47184

4.3.5.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada pada kasus TSP dengan jumlah individu dalam populasi sebesar 300. Masing-masing uji coba dilakukan sebanyak 10 kali. Hasil uji coba dapat dilihat pada Tabel 4.42 sampai dengan Tabel 4.44.

Tabel 4.42 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	995,6	936,0	936,0
4	1511,6	1126,2	936,0

Tabel 4.43 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	1906	668	277
4	1017	949	143

Tabel 4.44 Tabel waktu konvergensi rata-rata pada penyelesaian TSP-Rute 1 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	161,5156	136,8765	20,2734
4	87,7406	79,5500	10,9406

4.3.5.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada pada kasus TSP dengan jumlah *parents* sebanyak 3 dan jumlah individu baru tiap iterasi sebanyak 2. Masing-masing uji coba dilakukan sebanyak 10 kali. Hasil uji coba ditunjukkan pada Tabel 4.45.

Tabel 4.45 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 1 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
100	1528,50	669	51,78278
200	1020,00	1228	97,57475
300	936,00	1683	136,87650

4.3.6. Uji Coba untuk Kasus TSP-Rute 2

Pada bagian ini akan dijelaskan hasil uji coba untuk pengaruh parameter terhadap kinerja masing-masing algoritma serta perbandingan kinerja masing-masing algoritma dengan parameter tertentu pada penyelesaian TSP-Rute 2 (data koordinat rute 2 dapat dilihat pada Lampiran 2). Data hasil uji coba disajikan dalam bentuk tabel nilai rata-rata.

4.3.6.1. Pengaruh Parameter Pada Kinerja GA

Uji coba dilakukan pada kasus TSP dengan jumlah individu dalam populasi sebesar 300. Masing-masing uji coba dilakukan sebanyak 10 kali dengan iterasi maksimal 2000. Hasil uji coba ditunjukkan pada Tabel 4.46 sampai dengan Tabel 4.49.

Tabel 4.46 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA.

Pm	Pc (%)				
	5	15	25	35	50
1	1730,60	1732,80	1700,50	1685,00	1611,00
10	1999,10	2010,90	1876,40	1804,80	1700,30
20	2348,40	2185,10	2156,60	2126,70	2078,70
30	2461,60	2342,30	2269,30	2284,00	2183,60
40	2374,40	2485,10	2287,20	2413,00	2316,10
50	2618,30	2613,90	2483,90	2422,70	2382,30

Tabel 4.47 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA.

Pm	Pc (%)				
	5	15	25	35	50
1	1534	808	1560	1236	1255
10	1520	1552	1550	1456	1219
20	1389	1260	1338	1147	1398
30	1173	1413	952	1001	996
40	1041	1133	740	218	705
50	654	485	348	457	230

Tabel 4.48 Tabel waktu iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan GA.

Pm	Pc (%)				
	5	15	25	35	50
1	8,18276	4,39058	8,42497	6,74059	6,92184
10	8,44215	8,62964	8,64685	8,15621	6,99996
20	7,82028	7,20779	7,61561	6,57957	8,05464
30	6,78434	8,04060	5,55779	5,89058	5,89685
40	6,18277	6,61715	4,37809	1,32650	4,24372
50	3,87497	2,83903	2,09214	2,79370	1,43120

Tabel 4.49 Tabel waktu total rata-rata pada penyelesaian TSP-Rute 2 dengan GA.

Pm	Pc (%)				
	5	15	25	35	50
1	10,59684	10,64842	10,72964	10,77810	10,87497
10	10,97341	11,03495	11,07964	11,09996	11,33432
20	11,17495	11,29839	11,28902	11,33580	11,44370
30	11,38902	11,30777	11,47653	11,63590	11,62340
40	11,73120	11,57340	11,71090	11,69997	11,76713
50	11,57496	11,64526	11,71559	11,72966	11,81872

4.3.6.2. Pengaruh Besar Populasi Pada Kinerja GA

Uji coba dilakukan pada pada kasus TSP dengan peluang crossover (Pc) sebesar 50% dan peluang mutasi (Pm) GA sebesar 1%. Masing-masing uji coba dilakukan sebanyak 10 kali dengan iterasi maksimal 2000. Hasil uji coba ditunjukkan pada Tabel 4.50.

Tabel 4.50 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 2 dengan GA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
100	1611,00	1255	6,92184	10,87497
200	1512,70	1047	11,47809	21,42185
300	1500,80	1363	21,56090	31,45465

4.3.6.3. Pengaruh Parameter Pada Kinerja BMDA

Uji coba dilakukan pada pada kasus TSP dengan jumlah individu dalam populasi sebesar 300. Masing-masing uji coba dilakukan sebanyak 10 kali. Hasil uji coba ditunjukkan pada Tabel 4.51 sampai dengan 4.53.

Tabel 4.51 Tabel hasil optimasi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	1540,1	1427,6	2143,4
4	2068,8	1615,0	2193,4

Tabel 4.52 Tabel jumlah iterasi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	3823	4954	2645
4	1316	2118	705

Tabel 4.53 Tabel waktu konvergensi rata-rata pada penyelesaian TSP-Rute 2 dengan BMDA.

New Idv	Jumlah Parents		
	2	3	4
2	317,3219	365,5953	215,7640
4	109,1562	160,8750	58,4750

4.3.6.4. Pengaruh Besar Populasi Pada Kinerja BMDA

Uji coba dilakukan pada pada kasus TSP dengan jumlah *parents* sebanyak 3 dan jumlah individu baru tiap iterasi sebanyak 2. Masing-masing uji coba dilakukan sebanyak 10 kali. Hasil uji coba ditunjukkan pada Tabel 4.54.

Tabel 4.54 Tabel pengaruh besar populasi pada penyelesaian TSP-Rute 2 dengan BMDA.

Pop	Hasil optimasi	Jumlah Iterasi	Waktu Total
300	1427,60	4954	365,59530
400	1386,00	7577	606,45780

4.4. Analisis Hasil Uji Coba

Pada bagian ini akan dijelaskan analisis terhadap hasil uji coba yang telah dilakukan.

4.4.1. Analisis untuk Masalah Optimasi Onemax

4.4.1.1. Analisis untuk Algoritma Genetika

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter peluang crossover (Pc), peluang mutasi (Pm), dan besar populasi terhadap kinerja Algoritma Genetika dalam menyelesaikan kasus Onemax:

✓ Pengaruh peluang crossover (Pc) dan peluang mutasi (Pm)

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-20 dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal. Nilai Pm 1%, 10%, 20%, dan 30% memberikan hasil optimasi rata-rata yang bagus yaitu berkisar antara 19,85 sampai 20.

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-75 dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal.
- Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-100 dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal.
- Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-20, Onemax-75, dan Onemax-100 didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran besar akan memberikan hasil optimasi yang lebih baik, namun waktu yang dibutuhkan menjadi semakin lama.

4.4.1.2. Analisis untuk Algoritma Bivariate Marginal Distribution

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter jumlah parents, jumlah individu baru tiap iterasi, dan besar populasi terhadap kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan kasus Onemax:

✓ Pengaruh jumlah parents

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-20, Onemax-75, dan Onemax-100 dengan tiga nilai jumlah parents yang berbeda (2, 3, dan 4), didapatkan kecenderungan sebagai berikut:

- Percobaan dengan menggunakan jumlah parents 3 memberikan rata-rata hasil optimasi yang terbaik dan waktu yang lebih cepat dibandingkan dengan percobaan yang menggunakan jumlah parents lebih besar
- Percobaan dengan menggunakan jumlah parents 4 memberikan rata-rata hasil optimasi dan waktu yang buruk dibandingkan dengan percobaan menggunakan jumlah parents 2 dan 3.

✓ Pengaruh jumlah individu baru

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-20, Onemax-75, dan Onemax-100 dengan menggunakan dua nilai jumlah individu baru (2 dan 4), didapatkan kecenderungan sebagai berikut:

- Pada percobaan dengan menggunakan jumlah individu baru yang lebih besar, konvergensi populasi terbentuk lebih cepat. Bila jumlah individu baru terlalu banyak, hasil optimasi menjadi buruk. Dalam uji coba terhadap kasus Onemax ini, hasil optimasi terbaik didapat dengan menggunakan 2 individu baru.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan untuk menyelesaikan Onemax-20, Onemax-75, dan Onemax-100 dengan beberapa besar populasi yang berbeda, didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran besar akan memberikan hasil optimasi yang baik, namun jumlah iterasi dan waktu yang dibutuhkan agar populasi konvergen menjadi semakin lama.

4.4.1.3. Analisis Perbandingan Antara GA dengan BMDA

Berikut ini analisis perbandingan antara kinerja Algoritma Genetika dan kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan masalah optimasi Onemax (Onemax-20, Onemax-75, dan Onemax-100). Parameter yang digunakan untuk masing-masing algoritma adalah parameter terbaik yang didapat

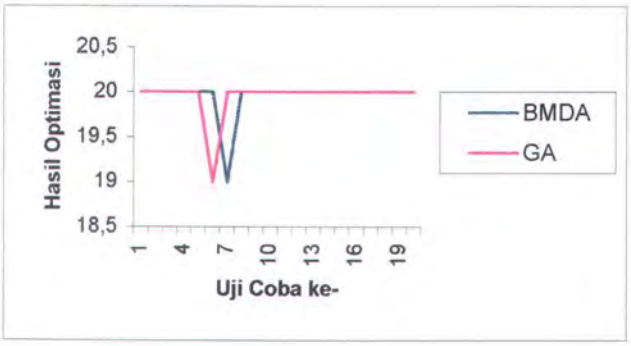
dari hasil uji coba yang telah dilakukan untuk masing-masing algoritma pada masalah optimasi Onemax. Perbandingan dilakukan tiga kali, yaitu pada penyelesaian kasus Onemax-20, Onemax-75, dan kasus Onemax-100.

✓ Perbandingan Kinerja GA dan BMDA dalam Menyelesaikan Onemax-20

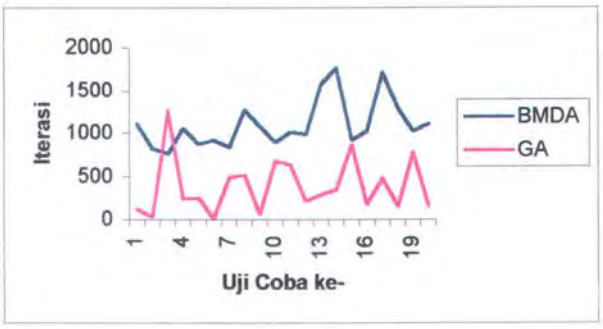
Dalam menyelesaikan kasus Onemax-20 ini, parameter yang digunakan oleh masing-masing algoritma adalah:

- GA → besar populasi 50, Pc = 50%, Pm = 10%, iterasi maksimal 1500
- BMDA → besar populasi 300, jumlah parent = 3, dan jumlah individu baru = 2.

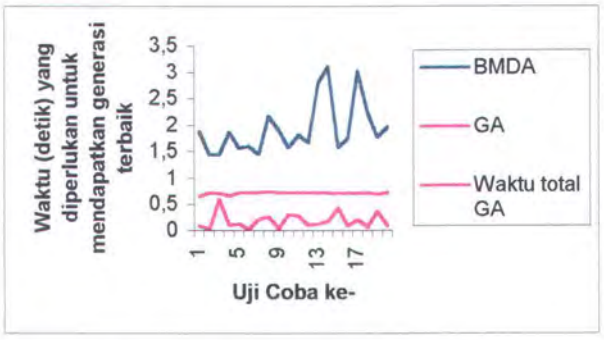
Perbandingan kinerja GA dan BMDA dalam menyelesaikan kasus Onemax-20 ditunjukkan pada Gambar 4.1 sampai dengan Gambar 4.3 serta Tabel 4.55.



Gambar 4.1 Grafik perbandingan hasil optimasi kasus Onemax-20 antara GA dengan BMDA.



Gambar 4.2 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-20 antara GA dengan BMDA.



Gambar 4.3 Grafik perbandingan waktu penyelesaian kasus Onemax-20 antara GA dengan BMDA.

Tabel 4.55 Tabel perbandingan penyelesaian kasus Onemax-20 antara GA dengan BMDA.

Algoritma	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	19,95	388	0,18356	0,72416
BMDA	19,95	1102	1,92965	

✓ Perbandingan Kinerja GA dan BMDA dalam Menyelesaikan Onemax-75

Dalam menyelesaikan kasus Onemax-75 ini, parameter yang digunakan oleh masing-masing algoritma adalah:

- GA → besar populasi 200, Pc = 50%, Pm = 10%, iterasi maksimal 8000.
- BMDA → besar populasi 200, jumlah parent = 3, dan jumlah individu baru = 2.

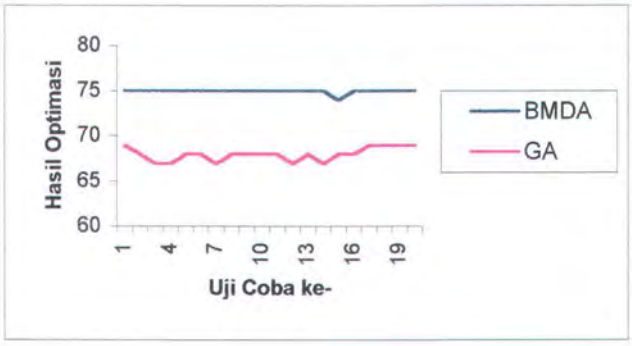
Perbandingan kinerja GA dan BMDA dalam menyelesaikan kasus Onemax-75 ditunjukkan pada Gambar 4.4 sampai dengan Gambar 4.6 serta Tabel 4.56.

✓ Perbandingan Kinerja GA dan BMDA dalam Menyelesaikan Onemax-100

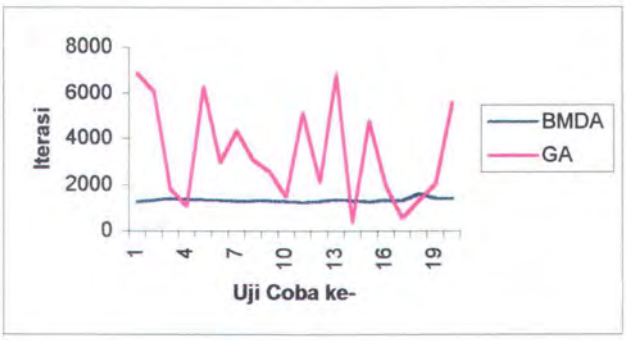
Dalam menyelesaikan kasus Onemax-100 ini, parameter yang digunakan oleh masing-masing algoritma adalah:

- GA → besar populasi 200, Pc = 50%, Pm = 10%, iterasi maksimal 11000.
- BMDA → besar populasi 200, jumlah parent = 3, jumlah individu baru = 2.

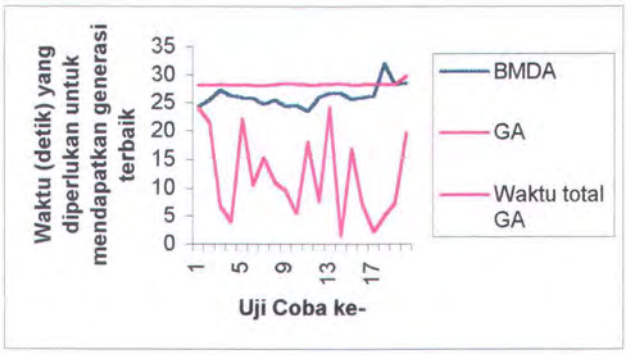
Perbandingan kinerja GA dan BMDA dalam menyelesaikan kasus Onemax-100 ditunjukkan pada Gambar 4.7 sampai dengan 4.9 serta Tabel 4.57.



Gambar 4.4 Grafik perbandingan hasil optimasi kasus Onemax-75 antara GA dengan BMDA.



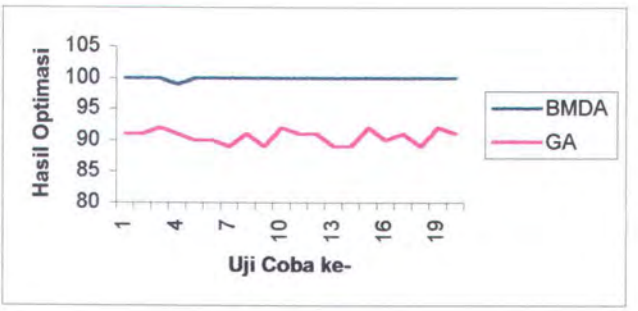
Gambar 4.5 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-75 antara GA dengan BMDA.



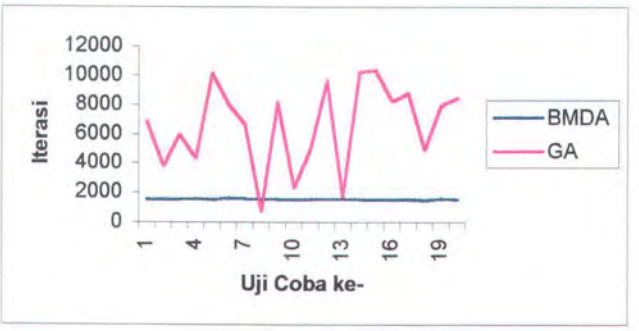
Gambar 4.6 Grafik perbandingan waktu penyelesaian kasus Onemax-75 antara GA dengan BMDA.

Tabel 4.56 Tabel perbandingan penyelesaian kasus Onemax-75 antara GA dengan BMDA.

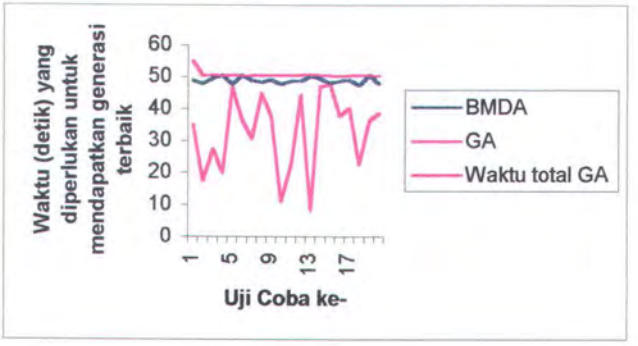
Algoritma	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	68,00	3362	12,01559	28,28278
BMDA	74,95	1331	26,17780	



Gambar 4.7 Grafik perbandingan hasil optimasi kasus Onemax-100 antara GA dengan BMDA.



Gambar 4.8 Grafik perbandingan jumlah iterasi penyelesaian kasus Onemax-100 antara GA dengan BMDA.



Gambar 4.9 Grafik perbandingan waktu penyelesaian kasus Onemax-100 antara GA dengan BMDA.

Tabel 4.57 Tabel perbandingan penyelesaian kasus Onemax-100 antara GA dengan BMDA.

	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	90,55	7075	32,65856	50,53747
BMDA	99,95	1552	48,77570	

Dari perbandingan pada ketiga kasus tersebut didapatkan kecenderungan sebagai berikut:

- Untuk penyelesaian kasus Onemax dengan ukuran yang tidak terlalu besar (dalam hal ini Onemax-20), kedua algoritma mampu mencapai hasil yang baik, tetapi GA lebih unggul dalam hal jumlah iterasi dan waktu.
- Untuk penyelesaian kasus Onemax dengan ukuran yang besar (dalam hal ini ditunjukkan oleh Onemax-75 dan Onemax-100), kinerja BMDA lebih baik daripada GA.

4.4.2. Analisis untuk Masalah Optimasi Fungsi De Jong F2

4.4.2.1. Analisis untuk Algoritma Genetika

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter peluang crossover (Pc), peluang mutasi (Pm), dan besar populasi terhadap kinerja Algoritma Genetika dalam menyelesaikan kasus Fungsi De Jong F2:

✓ Pengaruh peluang crossover (Pc) dan peluang mutasi (Pm)

Dari uji coba yang telah dilakukan dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal. Pada percobaan dengan nilai Pm yang terlalu besar, generasi terbaik sering diperoleh pada iterasi awal namun dengan nilai yang tidak optimal. Ini disebabkan gangguan pada populasi yang terlalu besar.
- Penggunaan nilai Pm 10% dan 20% memberikan hasil yang lebih baik dibanding penggunaan nilai Pm lain.
- Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.
- Penggunaan nilai parameter Pc 50% dan Pm 20% memberikan hasil optimasi terbaik.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan dengan menggunakan nilai Pc 50%, Pm 20%, dan tiga nilai besar populasi yang berbeda (30, 40, dan 50), didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran besar akan memberikan hasil optimasi yang lebih baik, namun waktu yang dibutuhkan untuk mendapatkan generasi terbaik menjadi semakin lama.

4.4.2.2. Analisis untuk Algoritma Bivariate Marginal Distribution

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter jumlah parents, jumlah individu baru tiap iterasi, dan besar populasi terhadap

kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan kasus Fungsi De Jong F2:

✓ Pengaruh jumlah parents

Dari uji coba yang telah dilakukan dengan tiga nilai jumlah parents yang berbeda (2, 3, dan 4), didapatkan kecenderungan sebagai berikut:

- Percobaan dengan menggunakan jumlah parents 3 memberikan hasil optimasi yang lebih baik daripada percobaan dengan jumlah parents 2. Namun, penggunaan jumlah parents 2 memberikan waktu yang lebih cepat dibandingkan dengan percobaan yang menggunakan jumlah parents lebih besar. Penggunaan jumlah parents 4 menunjukkan kinerja yang buruk, yaitu populasi sulit untuk konvergen dan hasil optimal susah didapat.
- Penggunaan nilai jumlah parents 3 dan jumlah individu baru sebanyak 2 memberikan hasil optimasi terbaik.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan dengan menggunakan jumlah parents 3, jumlah individu baru 2, dan tiga nilai besar populasi yang berbeda (50, 100, dan 400), didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran lebih besar akan memberikan hasil optimasi yang lebih baik, namun waktu yang dibutuhkan agar populasi konvergen menjadi semakin lama.

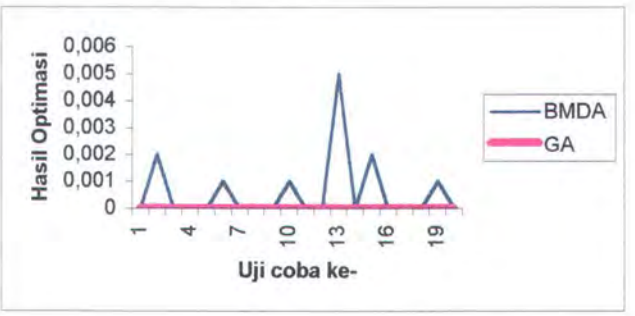
4.4.2.3. Analisis Perbandingan Antara GA dengan BMDA

Berikut ini analisis perbandingan antara kinerja Algoritma Genetika dan kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan masalah optimasi Fungsi De Jong F2. Parameter yang digunakan untuk masing-masing algoritma adalah parameter terbaik yang didapat dari hasil uji coba yang telah dilakukan untuk masing-masing algoritma pada masalah optimasi Fungsi De jong F2.

Dalam menyelesaikan Fungsi De Jong F2 ini, parameter yang digunakan oleh masing-masing algoritma adalah:

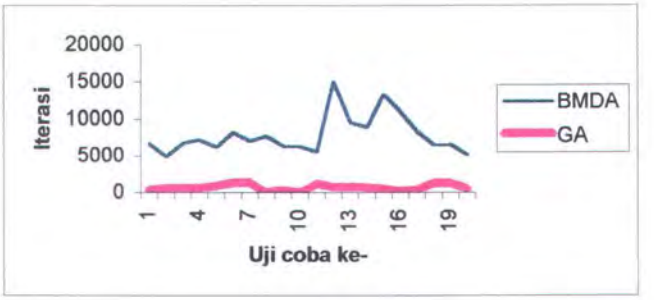
- GA → besar populasi 50, Pc = 50%, Pm = 20%, iterasi maksimal 2000.
- BMDA → besar populasi 400, jumlah parent = 3, dan jumlah individu baru = 2.

Perbandingan antara GA dan BMDA dalam menyelesaikan masalah optimasi Fungsi De Jong F2 ditunjukkan pada Gambar 4.10 sampai dengan Gambar 4.12 serta Tabel 4.58.

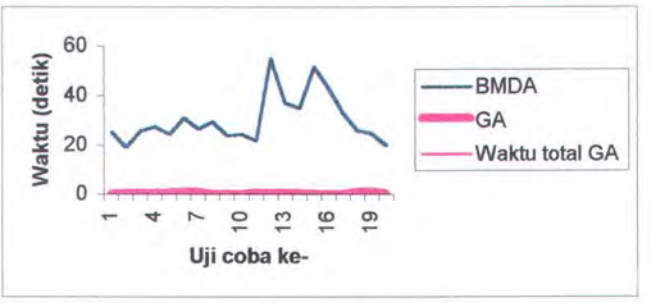


Gambar 4.10 Grafik perbandingan hasil optimasi Fungsi De Jong F2 antara GA dengan BMDA.

PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER



Gambar 4.11 Grafik perbandingan jumlah iterasi penyelesaian Fungsi De Jong F2 antara GA dengan BMDA.



Gambar 4.12 Grafik perbandingan waktu penyelesaian Fungsi De Jong F2 antara GA dengan BMDA.

Tabel 4.58 Tabel perbandingan penyelesaian Fungsi De Jong F2 antara GA dengan BMDA.

	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	0,00000	675	0,44059	1,31482
BMDA	0,00250	7561	26,94840	

Dari perbandingan tersebut didapatkan kecenderungan sebagai berikut:

- Untuk penyelesaian Fungsi De Jong F2, dengan parameter-parameter yang digunakan di atas, GA lebih mampu untuk selalu mencapai hasil yang optimal dan jauh lebih unggul dalam hal jumlah iterasi dan waktu.

4.4.3. Analisis untuk Kasus Traveling Salesman Problem

4.4.3.1. Analisis untuk Algoritma Genetika

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter peluang crossover (Pc), peluang mutasi (Pm), dan besar populasi terhadap kinerja Algoritma Genetika dalam menyelesaikan kasus TSP-Rute 1 dan TSP-Rute 2:

✓ Pengaruh peluang crossover (Pc) dan peluang mutasi (Pm)

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 1 dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal. Penggunaan nilai Pm 1% dan 10% memberikan hasil yang lebih baik dibanding penggunaan nilai Pm lain.
- Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.
- Penggunaan nilai Pc 50% memberikan hasil optimasi yang lebih baik dibanding penggunaan nilai Pc lain.

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 2 dengan menggunakan enam nilai Pm (1%, 10%, 20%, 30%, 40%, dan 50%) dan lima nilai Pc yang berbeda (5%, 15%, 25%, 35%, dan 50%), didapatkan kecenderungan sebagai berikut:

- Nilai Pm yang terlalu besar akan membuat hasil optimasi menjadi kurang optimal. Penggunaan nilai Pm 1% memberikan hasil yang lebih baik dibanding penggunaan nilai Pm lain.
- Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.
- Penggunaan nilai Pc 50% memberikan hasil optimasi yang cenderung lebih baik dibanding penggunaan nilai Pc lain.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 1 dan TSP-Rute 2 dengan menggunakan nilai Pc 50%, Pm 1%, dan tiga nilai besar populasi yang berbeda (100, 200, dan 300), didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran lebih besar akan cenderung memberikan hasil optimasi yang lebih baik, namun waktu yang dibutuhkan untuk mendapatkan generasi terbaik menjadi semakin lama.

4.4.3.2. Analisis untuk Algoritma Bivariate Marginal Distribution

Berikut ini analisis terhadap hasil uji coba tentang pengaruh parameter jumlah parents, jumlah individu baru tiap iterasi, dan besar populasi terhadap kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan kasus TSP:

✓ Pengaruh jumlah parents

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 1 dengan menggunakan tiga nilai jumlah parents yang berbeda (2, 3, dan 4), didapatkan kecenderungan sebagai berikut:

- Percobaan dengan menggunakan jumlah parents 3 dan 4 memberikan hasil optimasi yang lebih baik dan waktu yang jauh lebih cepat dibandingkan dengan percobaan yang menggunakan jumlah parents 2. Dalam hal jumlah iterasi dan waktu, percobaan dengan menggunakan jumlah parents 4 lebih baik daripada percobaan dengan menggunakan jumlah parents 3.

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 2 dengan menggunakan tiga nilai jumlah parents yang berbeda (2, 3, dan 4), didapatkan kecenderungan sebagai berikut:

- Percobaan dengan menggunakan jumlah parents 3 memberikan rata-rata hasil optimasi yang lebih baik daripada percobaan dengan menggunakan jumlah parents 2 dan 4.

✓ Pengaruh jumlah individu baru

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 1 dan TSP-Rute 2 dengan menggunakan dua nilai jumlah individu baru yang berbeda (2 dan 4), didapatkan kecenderungan sebagai berikut:

- Pada percobaan dengan menggunakan jumlah individu baru yang lebih besar, konvergensi populasi terbentuk lebih cepat. Namun, hasil optimasi

menjadi buruk bila jumlah individu baru terlalu banyak. Dalam hal ini, percobaan dengan menggunakan individu baru sebanyak 2 memberikan hasil yang baik untuk kedua rute.

✓ Pengaruh besar populasi

Dari uji coba yang telah dilakukan untuk menyelesaikan TSP-Rute 1 dan TSP-Rute 2 dengan menggunakan beberapa nilai besar populasi yang berbeda, didapatkan kecenderungan sebagai berikut:

- Populasi dengan ukuran lebih besar akan memberikan hasil optimasi yang lebih baik, namun waktu yang dibutuhkan agar populasi konvergen menjadi semakin lama.

4.4.3.3. Analisis Perbandingan Antara GA dengan BMDA

Berikut ini analisis perbandingan antara kinerja Algoritma Genetika dan kinerja Algoritma Bivariate Marginal Distribution dalam menyelesaikan kasus TSP-Rute 1 dan TSP-Rute 2. Parameter yang digunakan untuk masing-masing algoritma adalah parameter terbaik yang didapat dari hasil uji coba yang telah dilakukan untuk masing-masing algoritma pada kasus TSP.

Dalam menyelesaikan kasus TSP-Rute 1, parameter yang digunakan oleh masing-masing algoritma adalah:

- GA → besar populasi 300, Pc = 50%, Pm = 1%, iterasi maksimal 9000

- BMDA → besar populasi 300, jumlah parent = 3, dan jumlah individu baru = 2.

Perbandingan kinerja GA dan BMDA dalam menyelesaikan masalah TSP-Rute 1 ditunjukkan pada Gambar 4.13 sampai dengan Gambar 4.15 serta Tabel 4.59.

Dalam menyelesaikan kasus TSP-Rute 2, parameter yang digunakan oleh masing-masing algoritma adalah:

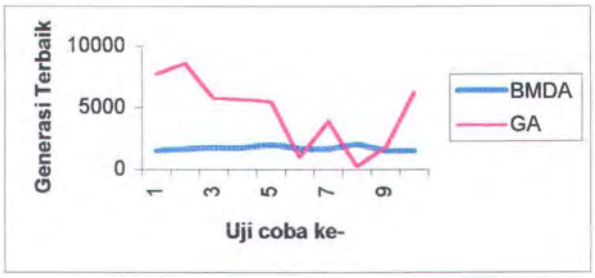
- GA → besar populasi 400, Pc = 50%, Pm = 1%, iterasi maksimal 30000
- BMDA → besar populasi 400, jumlah parent = 3, dan jumlah individu baru = 2.

Perbandingan GA dan BMDA dalam menyelesaikan masalah TSP-Rute 2 ditunjukkan pada Gambar 4.16 sampai dengan Gambar 4.18 serta Tabel 4.60.

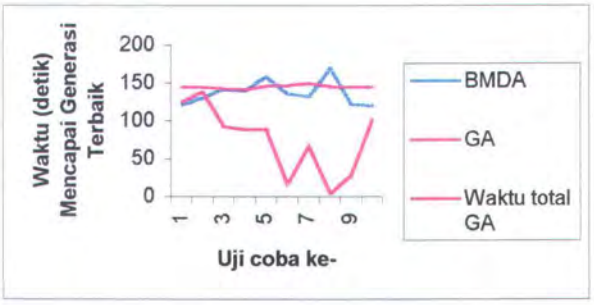
Dari perbandingan tersebut didapatkan kecenderungan sebagai berikut:

- Untuk penyelesaian kasus TSP (baik TSP-Rute 1 maupun TSP-Rute 2), dengan parameter-parameter yang digunakan di atas, BMDA lebih mampu untuk menunjukkan kinerja yang lebih baik dibanding GA.

Gambar 4.13 Grafik perbandingan hasil optimasi kasus TSP-Rute 1 antara GA dengan BMDA.



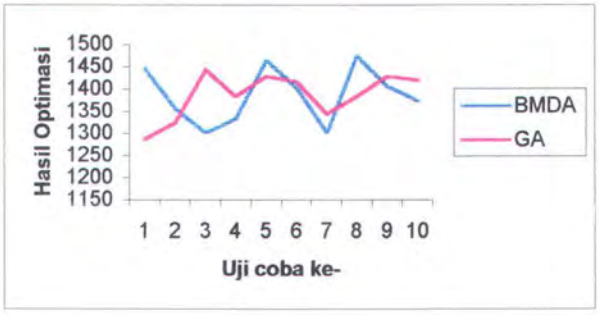
Gambar 4.14 Grafik perbandingan jumlah iterasi penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA.



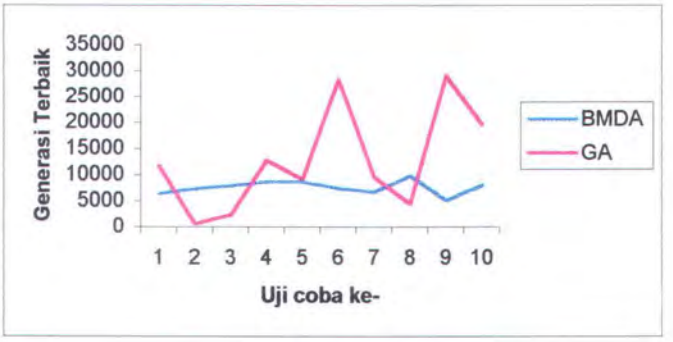
Gambar 4.15 Grafik perbandingan waktu penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA.

Tabel 4.59 Tabel perbandingan penyelesaian kasus TSP-Rute 1 antara GA dengan BMDA.

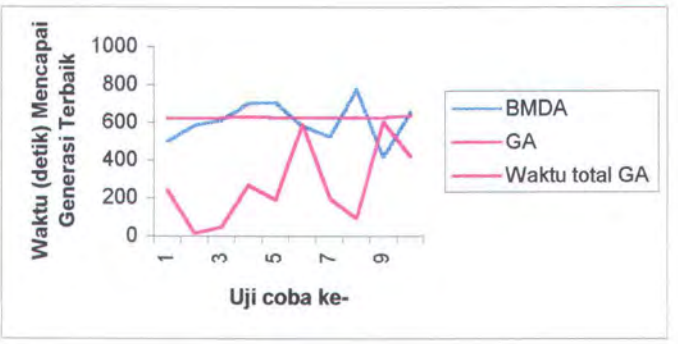
	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	1155,40	4648	74,97650	144,92340
BMDA	936,00	1683	136,87650	



Gambar 4.16 Grafik perbandingan hasil optimasi kasus TSP-Rute 2 antara GA dengan BMDA.



Gambar 4.17 Grafik perbandingan jumlah iterasi penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA.



Gambar 4.18 Grafik perbandingan waktu penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA.

Tabel 4.60 Tabel perbandingan penyelesaian kasus TSP-Rute 2 antara GA dengan BMDA.

Algoritma	Hasil optimasi	Jumlah Iterasi	Waktu Iterasi	Waktu Total
GA	1386,40	12796	268,14059	628,09215
BMDA	1386,00	7577	606,45780	

BAB 5

KESIMPULAN

BAB 5

KESIMPULAN

Bab ini akan menjelaskan kesimpulan dari keseluruhan proses dan hasil uji coba yang telah dilakukan. Beberapa kesimpulan yang dapat ditarik dari hasil pengamatan selama proses perancangan, pembuatan sampai dengan uji coba dan analisis pada Tugas Akhir ini adalah sebagai berikut :

1. Parameter pada Algoritma Genetika (GA) adalah peluang *crossover* (Pc) dan peluang mutasi (Pm), sedangkan parameter Algoritma Bivariate Marginal Distribution (BMDA) adalah jumlah *parents* dan banyak individu baru. Berikut ini pengaruh parameter dan jumlah populasi pada masing-masing algoritma:
    - Secara umum, baik pada GA maupun BMDA, semakin besar populasi yang digunakan, hasil optimasi akan cenderung bertambah baik, namun waktu proses akan bertambah lama. BMDA cenderung membutuhkan populasi yang lebih besar dari GA (khususnya pada masalah dengan ukuran kecil) karena membutuhkan keragaman individu yang lebih besar untuk membentuk individu baru yang lebih baik sebelum populasi menjadi konvergen.
    - Untuk Algoritma Genetika:
        - Makin besar nilai Pc dan nilai Pm, makin besar pula waktu total untuk mencapai iterasi maksimal.

- Nilai peluang mutasi (Pm) yang terlalu besar akan memberikan hasil yang buruk karena gangguan terhadap populasi menjadi terlalu besar. Nilai Pm yang cenderung memberikan hasil optimasi terbaik adalah sebesar 1% dan 10%.

- Untuk Algoritma Bivariate Marginal Distribution:
  - Jumlah parents yang terlalu banyak akan membuat algoritma sulit untuk membuat individu baru yang baik. Jumlah parents yang memiliki kecenderungan untuk memberikan hasil yang baik adalah sebanyak 3.
  - Jumlah individu baru yang dapat memberikan hasil yang baik adalah sebanyak 2. Jumlah individu baru yang banyak akan membuat populasi menjadi lebih cepat menjadi konvergen yang menyebabkan algoritma berhenti saat hasil yang didapatkan mungkin belum terlalu baik.

2. Untuk penyelesaian kasus Onemax dengan ukuran masalah yang tidak terlalu besar (ditunjukkan oleh percobaan Onemax-20), GA lebih unggul dalam hal jumlah iterasi yang lebih kecil dan waktu yang lebih cepat dibanding BMDA. Namun, untuk penyelesaian kasus Onemax dengan ukuran masalah yang cukup besar (ditunjukkan oleh percobaan Onemax-100), BMDA lebih mampu memberikan hasil optimasi dan jumlah iterasi yang lebih baik dibanding GA dalam kisaran waktu proses yang sama. Parameter GA yang cocok untuk permasalahan Onemax adalah peluang *crossover* (Pc) 50% dan peluang mutasi

(Pm) 10%, sedangkan parameter BMDA adalah jumlah *parents* 3 dan jumlah individu baru 2.

3. Untuk kasus Fungsi De Jong F2, GA menunjukkan kinerja yang lebih baik dibanding BMDA. Dalam hal hasil optimasi, GA cenderung selalu memberikan hasil yang lebih baik. Dalam hal jumlah iterasi dan waktu yang dibutuhkan, GA juga lebih unggul. Parameter GA yang cocok untuk permasalahan Onemax adalah peluang *crossover* (Pc) 50% dan peluang mutasi (Pm) 20%, sedangkan parameter BMDA adalah jumlah *parents* 3 dan jumlah individu baru 2.
4. Untuk kasus *Traveling Salesman Problem*, BMDA menunjukkan kinerja yang lebih baik dalam hal hasil optimasi dan jumlah iterasi dalam kisaran waktu proses yang sama. Parameter GA yang cocok untuk permasalahan Onemax adalah peluang *crossover* (Pc) 50% dan peluang mutasi (Pm) 1%, sedangkan parameter BMDA adalah jumlah *parents* 3 dan jumlah individu baru 2.
5. Secara umum, untuk masalah dengan ukuran panjang individu yang kecil (pada penyelesaian kasus Onemax-20 dan Fungsi De Jong F2), GA lebih baik dari BMDA dalam hal jumlah iterasi dan waktu proses yang lebih cepat. Sedangkan untuk masalah dengan ukuran panjang individu yang besar (pada penyelesaian kasus Onemax-100 dan TSP), BMDA lebih unggul dalam hal hasil optimasi dan jumlah iterasi dalam kisaran waktu proses yang sama.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[01] Kumar, Paul Topon & Hitoshi, Iba. **Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms**. 9th MPS symposium on Evolutionary Computation, IPSJ, Japan (2002).

[02] Kusumadewi, Sri. **Artificial Intelligence (Teknik dan Aplikasinya)**. Penerbit Graha Ilmu. Yogyakarta (2003).

[03] Michalewicz, Zbigniew. **Genetic Algorithms + Data Structures = Evolution Programs, Third, Revised and Extended Edition**. Springer-Verlag. Berlin (1999).

[04] Pelikan, Martin & Muehlenbein, Heinz. **The Bivariate Marginal Distribution Algorithm**. Dept. of Mathematics, Slovak Technical University. Bratislava, Slovakia (1999).

[05] **www.wikipedia.co.id**

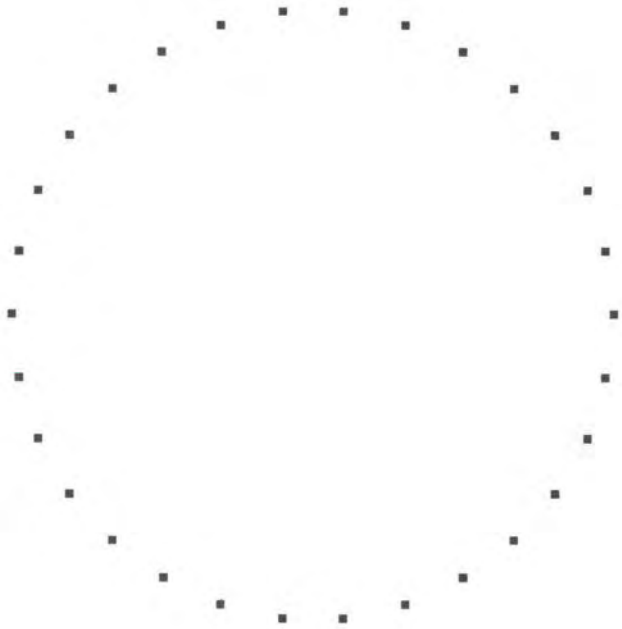
[06] Yahya, Melissa Ariani. **Rancang Bangun Aplikasi Optimasi Pada Traveling Salesman Problem dan Fungsi De jong F2 Menggunakan Algoritma Population Based Incremental learning (PBIL)**. Institut Teknologi Sepuluh Nopember (ITS). Surabaya (2004).

LAMPIRAN

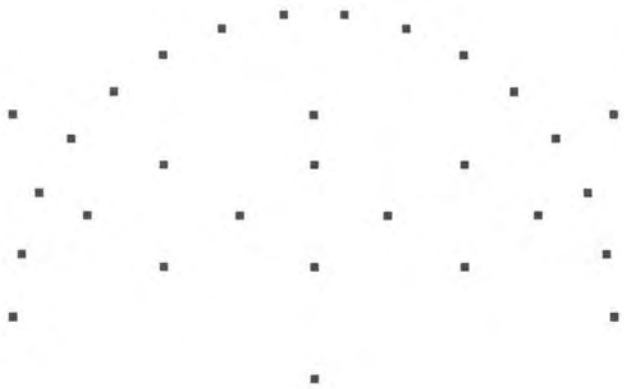
LAMPIRAN 1

Koordinat 30 Kota pada TSP Rute 1

Kota	X	Y	Kota	X	Y	Kota	X	Y
0	700	200	10	476	329	20	475	71
1	696	231	11	450	311	21	504	58
2	687	261	12	429	288	22	535	51
3	671	288	13	413	261	23	565	51
4	650	311	14	404	231	24	596	58
5	625	329	15	400	200	25	625	71
6	596	342	16	404	169	26	650	89
7	565	349	17	413	139	27	671	112
8	535	349	18	429	112	28	687	139
9	504	342	19	450	89	29	696	169



LAMPIRAN 2

Koordinat 30 Kota pada TSP Rute 2

Kota	X	Y	Kota	X	Y	Kota	X	Y
0	400	100	10	475	175	20	475	71
1	550	100	11	550	175	21	504	58
2	700	100	12	625	175	22	535	51
3	475	125	13	550	230	23	565	51
4	550	125	14	700	200	24	596	58
5	625	125	15	400	200	25	625	71
6	437	150	16	404	169	26	650	89
7	513	150	17	413	139	27	671	112
8	587	150	18	429	112	28	687	139
9	662	150	19	450	89	29	696	169