Optimasi Pemetaan Tugas Mengajar Dosen Menggunakan *Memetic Algorithm*

Skripsi

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Okvio Akbar Karuniawan
NIM: 125150218113002

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

OPTIMASI PEMETAAN TUGAS MENGAJAR DOSEN MENGGUNAKAN MEMETIC ALGORITHM

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Okvio Akbar Karuniawan
NIM: 125150218113002

Skripsi ini telah diuji dan dinyatakan lulus pada
02 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Nurul Hidayat, S.Pd, M.Sc	Drs. Marji, M.T
NIP: 19680430 200212 1 001	NIP: 19670801 199203 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

IDENTITAS PENGUJI

PENGUJI

Majelis penguji ujian skripsi

Ir. Sutrisno, M.T (ke I) * ketua majelis
NIP. 195703251987011001

Tri Afirianto, S.T, M.T (ke II)
NIK. 201309 851213 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 02 Januari 2019

Okvio Akbar Karuniawan

NIM: 125150218113002

DAFTAR RIWAYAT HIDUP

Bahwa yang bertanda tangan dibawah ini :

Nama	: Okvio Akbar Karuniawan
Tempat / Tanggal Lahir	: Jakarta, 14 Oktober 1994
Jenis Kelamin	: Laki-laki
Agama	: Islam
Alamat	: Perum Demangan Regency no.15
No. Telp / HP	: 087753000514
E-mail	: viozocoolz@gmail.com

Menerangkan dengan sesungguhnya :

PENDIDIKAN FORMAL

- Tahun 1999 : TK ISLAM PUSPA INDAH PAMULANG
- Tahun 2005 : SD PELITA BANGSA PAMULANG
- Tahun 2009 : SMP NEGERI 3 LAMONGAN
- Tahun 2012 : R-SMA-BI NEGERI 2 LAMONGAN

Saya yang bersangkutan,

OKVIO AKBAR KARUNIAWAN

UCAPAN TERIMA KASIH

Dalam penyusunan laporan penelitian ini, penulis menyampaikan terima kasih kepada semua pihak yang sudah memberikan dukungan baik secara moril yang berupa bimbingan, kritik, saran, motivasi, serta doa. Ucapan terima kasih penulis sampaikan kepadaDalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan, baik bantuan moral maupun materiil dari berbagai pihak. Oleh karena itu, penulis mengucapkan banyak terimakasih kepada:

1. Nurul Hidayat, S.Pd., M.Sc selaku dosen pembimbing 1 dan Drs. Marji, M.T selaku dosen pembimbing 2, yang telah bersedia untuk membimbing dan memberikan arahan-arahan selama proses penelitian ini.
2. Agus Wahyu Widodo, S.Kom., M.Cs., selaku Ketua Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.
3. Bapak/Ibu dosen dan karyawan Fakultas Ilmu Komputer Universitas Brawijaya, atas ilmu dan arahannya sehingga penulis dapat menyelesaikan penelitian ini.
4. Kedua Orang Tua yaitu Ibu saya Kristi Wardani dan Ibu saya Sri Utami S.E dan adikku yaitu Juvico Akbar Karuniawan beserta keluarga besar yang telah mendukung penulis dengan segala usahanya, mulai dari doa, materi, dukungan moral, semangat hidup, dan tauladan yang semata-mata untuk keberhasilan penulis.
5. dr. Monica Rizky W selaku *hey love you* terimakasih sudah menemani dalam *relativity times* penulisan tugas akhir ini dan menyemangati untuk menyelesaikan *goals* ini dan lainnya.
6. Rekan seperjuangan di Fakultas Ilmu Komputer yang telah memberikan dukungan, motivasi dan masukan-masukan bagi penulis dalam menyelesaikan penelitian ini.

Penulis mengucapkan terima kasih atas segala bantuan yang telah diberikan. Semoga penulisan laporan skripsi ini bermanfaat bagi pembaca dan untuk pengembangan selanjutnya. Penulis menyadari bahwa skripsi ini masih jauh dari sempurna serta banyak kekurangan disebabkan oleh keterbatasan kemampuan dan pengalaman, dengan kerendahan hati penulis mengharapkan kritik dan saran yang bersifat membangun dari pembaca.

Malang, 02 Januari 2019

Penulis

viozocoolz@gmail.com

ABSTRAK

Penyampaian informasi yang berkaitan dengan kegiatan belajar-mengajar merupakan hal yang sangat penting, salah satu hal pertama yang perlu diperhatikan adalah penjadwalan. Pada Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang Setiap pergantian semester akan ada perubahan kompetensi pada fakultas artinya akan ada pembaharuan setiap semesternya dosen atau pengajar di FILKOM akan diplot atau dipetakan untuk mengajar apa tiap semesternya, karna setiap semester ada matakuliah yang dibuka dan tidak, dibuka disemester genap dan ganjil, matakuliah dihapus diganti baru atau adanya penambahan matakuliah baru semua sangat fleksibel dan kompleks. Sampai saat ini dosen atau pengajar proses pemetaan masih dilakukan dengan cara manual dimana pengerjaannya membutuhkan waktu yang tidak sedikit, sehingga dibutuhkanlah suatu metode optimasi yang tepat dalam menangani hal ini. Permasalahan penugasan ini dapat diselesaikan dengan metode *heuristic* berbasis populasi, *Memetic Algorithm* (MA) yang telah diterapkan pada berbagai bidang seperti penjadwalan dan penugasan. Data yang digunakan pada penelitian ini merupakan representasi dari angket pemetaan mengajar dosen mata kuliah yang berupa prioritas keminatan mengajar dosen terhadap suatu mata kuliah. Dari data yang didapat, ditentukanlah batasan-batasan seperti prioritas keminatan mengajar dosen, minimum dan maksimum mata kuliah yang dapat diambil, kelas yang dibuka pada semester baru, dan jumlah maksimum dan minimum SKS untuk dibuat representasi kromosomnya lalu dicari nilai *fitness* masing-masing populasi. Berdasarkan hasil perhitungan yang didapat, dilakukanlah pengujian parameter untuk mengetahui pengaruh parameter uji terhadap nilai *fitness* yang dihasilkan. Berdasarkan hasil perhitungan yang didapat, dilakukanlah pengujian parameter untuk mengetahui pengaruh parameter uji terhadap nilai *fitness* yang dihasilkan. Dari hasil pengujian parameter MA, diperoleh jumlah populasi terbaik sebesar 100, jumlah iterasi terbaik 100, dan kombinasi parameter kecepatan *cr* dan *mr* sebesar 0.8 dan 0.2 dengan nilai *fitness* yang didapat sebesar 87830. Nilai fitness tersebut merupakan solusi optimal dari generasi 100 karena *stop conditioning memetic algorithm* berupa iterasi maksimal. Hasil dari parameter Cr dan Mr tidak menjamin jika nilai Cr semakin Kecil dan Nilai Mr semakin besar atau nilai Mr lebih besar dari Cr akan menghasilkan fitness yang semakin baik tetapi tergantung pada penyesuaian parameter terhadap representasi kromosomnya.

Kata kunci: Optimasi, Pemetaan, *Memetic Algorithm*, Dosen, Tugas Mengajar, Keminatan Mengajar, Mata Kuliah, Nilai *Fitness*.

ABSTRACT

Submission of information relating to teaching and learning activities is very important, one of the first things to consider is scheduling. At Faculty of Computer Science (FILKOM) Brawijaya University, the assignment process is still manually designed where it requires some substantial time, therefore it needed a right optimization methods in dealing with this case. This assignment problem can be solved by a population-based heuristic methods, Memetic Algorithm (MA) which has been applied in various fields such as scheduling and assignments. The data used in this study is the data division of lecturers teaching tasks as a priority of lecturer's teaching interests to a course. From the obtained data, it determined constraints such a lecturer's teaching priority, the maximum and minimum amount of credits, and the number of course that can be taken to calculate fitness value for each particles. Through the obtained results, it had parameter tested to find the effect of tested parameters on the resulted fitness values. From MA parameters test results, it obtained the best particle number as 100, best iteration number as 100, and combination of velocity parameter cr and mr as 0.8 and 0.2 with resulted fitness value as 87830*. From the results of system, But the results do not guarantee if the value of Cr is getting smaller and the greater the value of Mr will produce better fitness.*

Keywords: Optimization, Assignments, Memetic Algorithm, Lecturer, Teaching Interests, Courses, Fitness Values.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas anugerah serta limpahan rahmat-Nya, sehingga penulis dapat menyelesaikan skripsi ini. Skripsi ini disusun sebagai syarat memperoleh gelar sarjana pada Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan, baik bantuan moral maupun materiil dari berbagai pihak. Oleh karena itu, penulis mengucapkan banyak terimakasih kepada:

7. Nurul Hidayat, S.Pd, M.Sc, selaku Pembimbing I yang telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini.
8. Drs. Marji, M.T, selaku Pembimbing II yang juga telah memberikan bimbingan, arahan, ilmu, dan masukan dalam penyelesaian skripsi ini.
9. Bapak dan Ibu dosen yang telah mendidik dan memberikan ilmu selama Penulis menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
10. Segenap karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang membantu Penulis dalam pelaksanaan skripsi ini.
11. Kedua Orang Tua yaitu Ibu saya Kristi Wardani dan Ibu saya Sri Utami S.E dan adikku yaitu Juvico Akbar Karuniawan beserta keluarga besar yang telah mendukung penulis dengan segala usahanya, mulai dari doa, materi, dukungan moral, semangat hidup, dan tauladan yang semata-mata untuk keberhasilan penulis.
12. Monica Rizky W, dr. selaku *hey love you* terimakasih sudah menemani dalam *relativity times* penulisan tugas akhir ini dan menyemangati untuk menyelesaikan *goals* ini dan lainnya.
13. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah membantu dan terlibat baik secara langsung maupun tidak langsung dalam penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini tidak lepas dari kesalahan dan kekurangan. Oleh karena itu, Penulis bersedia menerima kritik dan saran yang membangun untuk memperbaiki diri. Penulis berharap semoga skripsi ini dapat memberi manfaat.

Malang, 02 Januari 2019

Penulis

viozocoolz@gmail.com

DAFTAR ISI

PENGESAHAN ..... **Error! Bookmark not defined.**
PERNYATAAN ORISINALITAS ii
KATA PENGANTAR v
ABSTRAK vii
ABSTRACT viii
DAFTAR ISI x
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
DAFTAR LAMPIRAN xv
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 2
1.4 Manfaat 3
1.5 Batasan Masalah 3
1.6 Sistematika pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 *Assignment problem* 7
2.3 Optimasi 9
2.4 *Memetic Algorithm* 9
BAB 3 METODOLOGI 11
3.1 Tipe Penelitian 11
3.2 Strategi Penelitian 11
3.3 Partisipan Penelitian 11
3.4 Lokasi Penelitian 11
3.5 Teknik Pengumpulan Data 12
3.6 Teknik Analisis Data 12
3.7 Implementasi Algoritme 12
3.8 Pengujian Algoritme 13
3.8.1 Pengujian Jumlah Populasi 13

3.8.2 Pengujian Jumlah Iterasi 14
3.8.3 Pengujian Parameter Reproduksi *Cr* dan *Mr* 14
3.9 Kesimpulan dan Saran 15
BAB 4 PERANCANGAN.....16
4.1 Formulasi Permasalahan 16
4.2 Perancangan Sistem Algoritma *Memetic* 16
4.2.1 Inisialisasi Populasi.....20
4.2.2 Membangkitkan Matriks Biaya 22
4.2.3 Hitung Nilai *Fitness*.....26
4.2.4 Seleksi.....30
4.2.5 Crossover.....32
4.2.6 Mutasi 34
4.2.7 Local Search 36
4.2.8 Kondisi Terpenuhi 38
4.3 Siklus Penyelesaian Masalah Menggunakan *Memetic Algorithm*.....38
4.3.1 Inisialisasi Populasi.....38
4.3.2 Membangkitan Matriks Biaya 39
4.3.3 Menghitung Nilai *Fitness* 40
4.3.4 Crossover.....41
4.3.5 Mutasi 42
4.3.6 Menghitung Nilai Kecepatan.....43
4.3.7 Menghitung Nilai Posisi.....43
4.3.8 Evaluasi Nilai *Fitness* Pada Posisi Partikel Baru.....43
BAB 5 IMPLEMENTASI.....44
5.1 Implementasi sistem.....44
5.1.1 Spesifikasi Perangkat Keras.....44
5.1.2 Spesifikasi Perangkat Lunak.....45
5.2 Batasan Implementasi 45
5.3 Implementasi Algoritma 45
5.3.1 Implementasi Database Connection.....45
5.3.2 Implementasi Inisialisasi Variable dan Representasi Kromosom 48
5.3.3 Implementasi Hitung Fitness Parent.....52

5.3.4 Implementasi Reproduksi 53
5.3.5 Implementasi Local Search dan Hitung Fitness 54
5.3.6 Proses Seleksi 57
5.4 Implementasi Antarmuka 58
5.4.1 Implementasi Antarmuka Data Kompetensi.....58
5.4.2 Implementasi Antarmuka Proses Algoritma Memetic 59
5.4.3 Implementasi Antarmuka Hasil Pemetaan Mengajar.....60
BAB 6 PENGUJIAN DAN ANALISIS 61
6.1 Sistematika Pengujian.....61
6.2 Analisis dan Pembahasan 61
6.3 Pengujian dan Analisis Ukuran Populasi.....61
6.4 Pengujian dan Analisis Jumlah Iterasi.....63
6.5 Pengujian dan Analisis Cr/Mr 64
6.6 Analisis Hasil Pengujian 65
BAB 7 KESIMPULAN DAN SARAN 67
7.1 Kesimpulan 67
7.2 Saran 68
DAFTAR PUSTAKA.....69
LAMPIRAN A ANGKET PEMETAAN TUGAS MENGAJAR DOSEN PENGAMPU.....70
.....70

DAFTAR TABEL

Tabel 2.1 Kajian pustaka 5
Tabel 2.2 Kajian pustaka (lanjutan)..... 6
Tabel 3.1 Tabel rancangan pengujian jumlah populasi 13
Tabel 3.2 Tabel rancangan pengujian jumlah iterasi 14
Tabel 3.3 Tabel rancangan pengujian parameter kecepatan *c1* dan *c2* 15
Tabel 4.1 Daftar mata kuliah yang dibuka pada semester genap 2015/2016..... 16
Tabel 4.2 Daftar mata kuliah yang dibuka pada semester genap 2015/2016 (lanjutan)..... 17
Tabel 4.3 Data pemetaan mengajar dosen..... 18
Tabel 4.4 Tabel perancangan data mata kuliah..... 20
Tabel 4.5 Tabel perancangan data dosen 20
Tabel 4.6 Tabel perancangan inisialisasi partikel..... 21
Tabel 4.7 Tabel pembobotan *cost* keminatan 22
Tabel 4.8 Tabel perancangan matriks biaya 23
Tabel 4.9 Tabel perancangan matriks biaya (lanjutan)..... 24
Tabel 4.10 Tabel perancangan nilai total *cost* keminatan 27
Tabel 4.11 Tabel perancangan nilai *fitness*..... 27
Tabel 5.1 Spesifikasi Perangkat Keras..... 44
Tabel 5.2 Spesifikasi Perangkat Lunak 45
Tabel 6.1 Hasil Uji Coba Ukuran populasi 62
Tabel 6.2 Hasil Uji Jumlah Iterasi 63
Tabel 6.3 Hasil Uji Coba nilai Cr/Mr 64

DAFTAR GAMBAR

Gambar 2.1 Proses pembentukan nilai penugasan 8
Gambar 5.1 Bagan Implementasi..... 44
Gambar 5.2 Antarmuka Data Kompetensi..... 59
Gambar 5.3 Antarmuka proses algoritma memetic 59
Gambar 5.4 Antarmuka hasil pemetaan mengajar..... 60
Gambar 6.1 Hasil Uji Coba Ukuran populasi 62
Gambar 6.2 Hasil Uji Jumlah Iterasi 63
Gambar 6.3 Hasil Uji Coba nilai Cr/Mr 65

DAFTAR LAMPIRAN

LAMPIRAN A ANGKET PEMETAAN TUGAS MENGAJAR DOSEN PENGAMPU.....70

BAB 1 PENDAHULUAN

1.1 Latar belakang

Penyampaian informasi yang berkaitan dengan kegiatan belajar-mengajar merupakan hal yang sangat penting, salah satu hal pertama yang perlu diperhatikan adalah penjadwalan. Dalam sebuah aktifitas mengajar masalah penjadwalan kuliah merupakan permasalahan yang sangat kompleks karena adanya berbagai komponen yang terdiri dari dosen, mahasiswa, ruang kelas, dan kegiatan perkuliahan dengan memperhatikan batasan dan syarat tertentu. Jadwal dibuat harus memenuhi aturan akademik yang ada pada suatu lembaga pendidikan. Dalam upaya meningkatkan efisiensi dan relevansi pendidikan, seharusnya hasil pemetaan tugas mengajar dosen sesuai dengan bidang kompetensi dan keminatan pengajar. karena proses penjadwalan yang efektif dapat menentukan keberhasilan kegiatan belajar-mengajar yang optimal. Sehingga aktifitas pemetaan tugas mengajar dosen adalah tugas yang kompleks dan selalu dihadapi oleh pihak akademik tiap semesternya. Salah satu lembaga pendidikan yang menemui kendala pada objek ini adalah Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang.

Setiap pergantian semester akan ada perubahan kompetensi pada fakultas artinya akan ada pembaharuan setiap semesternya dimana dosen atau pengajar di FILKOM akan diplot atau dipetakan untuk mengajar apa tiap semesternya, karna setiap semester ada matakuliah yang dibuka ada yang tidak, dibuka disemester genap dan ganjil, matakuliah dihapus diganti baru atau ada penambahan matakuliah baru semua sangat fleksibel mengikuti perkembangan dan *trend* zaman dunia pendidikan. Pemetaan mengajar yang telah diisi oleh dosen terdiri dari mata kuliah mayor dan minor dengan memperhatikan urutan prioritas pilihan mata kuliah yang akan diajarkan. Adapun aturan remunerasi dosen serta batasan yang harus diperhatikan menurut kriterianya seperti jumlah minimal dan maksimal mata kuliah, beban total SKS, kelas yang dibuka dan beban prioritas minat mata kuliah yang dipilih.

Sebelumnya, pihak akademik masih menggunakan perangkat lunak berupa Microsoft Excel dalam menyelesaikan susunan pemetaan tugas mengajar. Akibatnya, jika terjadi ketidaksesuaian dalam perhitungannya, maka pengoreksiannya pun dilakukan dengan cara manual. Proses seperti ini memakan waktu lebih lama dan terhitung tidak efisien. Dari latar belakang masalah diatas perlunya pembuatan sistem untuk optimasi pemetaan tugas mengajar dosen akan sangat membantu pihak akademik dalam mengoptimasi data yang ada.

Pada penelitian sebelumnya yang dilakukan oleh Albar (2013), membandingkan metode *Memetic Algorithm* dan Algoritma Genetik Tabu Search dalam sistem penjadwalan kuliah yang menghasilkan solusi jadwal yang optimal. Variabel pengujiannya adalah Soft Constraint, Hard Constraint, Total Soft Constraint, Total Hard Constraint dan Waktu Eksekusi. Berdasarkan hasil penelitian *Memetic Algorithm* lebih unggul daripada Algoritma Genetik Tabu

Search. Sedangkan pada penelitian oleh Hetari et al. (2016), algoritma *hybrid Genetic Algorithm-Simulated Annealing* (GA-SA) dapat digunakan untuk menyelesaikan masalah pembagian tugas yang sama, tetapi nilai *cost* yang didapat masih tergolong tinggi dikarenakan jumlah data yang digunakan tergolong sedikit. Berdasarkan uraian diatas Pada penelitian tugas akhir ini, permasalahan pemetaan tugas mengajar diselesaikan dengan menerapkan metode *Memetic Algorithm*, kandidat solusi merepresentasikan kromosom sebuah mata kuliah dan dosen pada suatu tabel jadwal dimana setiap kandidat solusi memiliki nilai *cost. Gen parent* pada kromosom dibangkitkan di awal iterasi pertama, lalu *parent-parent* tersebut dikawinkan, sehingga menghasilkan *protochild* yang memiliki gen mendekati optimum, setelah *protochild-protochild* ini lahir dilakukan *compete* antar protochild untuk menentukan siapa *child* yang dapat dikawinkan dengan *parent* sebelumnya, sehingga lahirlah generasi-generasi penerus yang membawa gen mendekati optimum sebuah mata kuliah pada tabel jadwal sampai konvergen kromosom. Proses ini dilakukan untuk setiap mata kuliah, sehingga diperoleh sebuah jadwal yang utuh. Dengan metode algoritme memetik ini, diharapkan didapatlah proses penjadwalan yang efisien dengan nilai akurasi yang memuaskan dan nilai *penalty* seminimal mungkin.

Berdasarkan uraian di atas, penulis mengajukan penelitian yang berjudul Optimasi Pemetaan Mengajar Dosen Menggunakan *Memetic Algorithm*. Dengan penelitian ini, diharapkan dapat menyelesaikan permasalahan pemetaan mengajar dosen di FILKOM Universitas Brawijaya Malang.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan di atas, dapat dirumuskan beberapa masalah sebagai berikut :

1. Bagaimana menerapkan metode *Memetic Algorithm* untuk mengoptimasi pemetaan pengajar mata kuliah?
2. Bagaimana representasi kromosom yang dapat digunakan untuk mengoptimasi pemetaan pengajar mata kuliah?
3. Bagaimana pengaruh parameter *Memetic Algorithm* terhadap hasil optimasi pemetaan pengajar matakuliah?

1.3 Tujuan

Tujuan yang ingin dicapai dari pembuatan skripsi ini yaitu :

1. Menerapkan metode *Memetic Algorithm* untuk mengoptimasi pemetaan pengajar mata kuliah.
2. Menentukan representasi kromosom yang dapat digunakan untuk mengoptimasi pemetaan pengajar mata kuliah
3. Menganalisa dan menguji setiap pengaruh parameter pada penerapan metode *Memetic Algorithm* untuk pemetaan pengajar mata kuliah.

1.4 Manfaat

Manfaat yang didapatkan oleh penelitian ini adalah sebagai berikut.

1. Bagi Penulis
   - Menerapkan ilmu-ilmu yang diperoleh selama di bangku perkuliahan.
   - Mengoptimasi metode *Memetic Algorithm* untuk penjadwalan dosen FILKOM di Universitas Brawijaya Malang.
2. Bagi Universitas
   - Memberikan gambaran tentang persiapan kepada mahasiswa dalam menghadapi dunia luar.
   - Untuk memberikan masukan bagi para mahasiswa informatika yang ingin mengembangkan kembali penelitian ini sehingga hasil yang didapatkan bisa lebih baik.
   - Untuk mengetahui sejauh mana penguasaan materi yang diberikan, sehingga dapat dijadikan acuan untuk angkatan-angkatan mendatang.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi dalam hal:

1. Data yang digunakan adalah data mata kuliah dan data *dummy* untuk menentukan prioritas mata kuliah yang diambil oleh dosen FILKOM Universitas Brawijaya Malang yang jumlahnya lebih sedikit dibandingkan dengan data aslinya. Apabila nantinya akan digunakan data asli yang jumlahnya lebih banyak, maka perlu dilakukan perubahan pada *database*.
2. Hasil optimasi dari sistem hanya mencangkup penempatan dosen terhadap kelas mata kuliah yang telah ditentukan saja, tidak mencangkup penjadwalan waktu, baik hari maupun jam mengajar.

1.6 Sistematika pembahasan

Sistematika pembahasan penelitian ini disusun untuk memberikan gambaran umum dan uraian dari penyususnan tugas akhir secara garis besar. Sustematika pembahasan ini adalah sebagai berikut :

BAB 1 Pendahuluan

Bab ini menjelaskan tentang latar belakang penelitian, rumusan masalah, batasan masalah, tujuan dan manfaat penulisan.

BAB 2 Landasan Kepustakaan

Pada bab ini berisi tentang kajian pustaka dari penelitian sebelumnya, dan membahas mengenai dasar teori yang berkaitan dan menunjang dalam penyelesaian tugas akhir ini.

BAB 3 Metodologi

Bab ini membahas metode-metode yang digunakan perancangan, pengujian dan analisis dalam sistem optimasi untuk penentuan pengajar matakuliah yang tepat.

BAB 4 Perancangan

Pada bab ini berisi tentang perancangan perhitungan, diagram alir dari aplikasi yang akan dibuat, skema perhitungan manual, dan rancangan antrarmuka aplikasi.

BAB 5 Implementasi

Bab ini membahas implementasi dari Algoritme *Memetic Algorithm* untuk optimasi penentuan pengajar matakuliah yang tepat.

BAB 6 Pengujian dan Analisis

Pada bab ini berisi hasil dari pengujian terhadap aplikasi yang telah dibuat berdasarkan rancangan yang telah ditentukan dilanjutkan dengan analisis terhadap hasil penelitian.

BAB 7 Penutup

Pada bab ini berisi ringkasan dan pencapaian hasil dan menjawab pertanyaan dari rumusan masalah serta saran yang dapat diberikan untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dijelaskan kajian pustaka dan dasar teori yang digunakan dalam skripsi ini. Kajian pustaka membahas secara umum mengenai penelitian-penelitian sebelumnya yang berhubungan dengan topik skripsi dan membandingkan penelitian-penelitian tersebut dengan penelitian skripsi yang sedang dilakukan. Penelitian sebelumnya yang menjadi acuan dalam penelitian ini adalah "Analisa dan Penerapan Metode *Memetic Algorithm* pada Optimasi Penjadwalan Kuliah" oleh Rachman et al. (2012). Dasar teori membahas mengenai teori yang mendukung penelitian ini dan diapatkan dari berbagai sumber pustaka. Pada penelitian ini, dasar teori memuat penjelasan mengenai konsep algoritma *Memetic Algorithm,* serta penugasan.

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahas dan membandingkan penelitian sebelumnya yang berkaitan dengan skripsi ini. Penelitian-penelitian yang digunakan adalah "Algoritma Genetik Tabu Search Dan Memetika Pada Permasalahan Penjadwalan Kuliah" (Albar et al., 2013), dan "Implementasi *Hybrid Genetic Algorithm dan Simulated Annealing* untuk Pembagian Tugas Mengajar Dosen PTIIK" (Hetari et al., 2016). Perbandingan ke-2 penelitian ini ditunjukkan pada Tabel 2.1.

Tabel 2.1 Kajian pustaka

No.	Judul Penelitian	Objek	Metode yang digunakan	Hasil dan pengujian
		Input dan Parameter		
1.	Algoritma Genetik Tabu Search dan Memetika pada Permasalahan Penjadwalan Kuliah	-Data penjadwalan mata kuliah Fakultas Pertanian Universitas Mataram -Parameter *hard constraint*, *soft constraint,* waktu	Algoritma memetik, Algoritma genetik tabu search.	Pembagian jadwal mata kuliah yang optimal. Hasil dari penelitian ini adalah perbandingan waktu eksekusi kedua algoritma, Memetika lebih cepat dengan waktu 135 detik.
2.	Implementasi *Hybrid Genetic Algorithm* dan *Simulated Annealing* untuk Pembagian Tugas Mengajar Dosen PTIIK	-Data penugasan mengajar dosen semester genap di FILKOM UB -Parameter urutan minat mengajar dosen, jumlah kelas, jumlah generasi, jumlah populasi, *crossover rate, mutation rate, t, β*	Hybrid Algortima Genetika dan *Simulated Annealing*	Pembagian tugas mengajar dosen berdasarkan nilai *fitness* tertinggi. Hasil dari penelitian ini adalah parameter optimum cr = 0,8, mr = 0,2, *t* = 1, *β* = 0,9, gen = 1000 dan pop = 100.

Tabel 2.2 Kajian pustaka (lanjutan)

No.	Judul Penelitian	Objek	Metode yang digunakan	Hasil dan pengujian
		Input dan Parameter		
3.	*A Comparison of Memetic & Tabu Search for The Cryptanalysis of Simplified Data Encryption Standard Algorithm*	- Simplified data encryption standard problem - Chiper text	*Memetic and Tabu Search*	Hasil dari penelitian ini adalah perbandingan kedua algoritma, *Memetic* lebih cepat dengan waktu 2,14 menit dan akurasi kecocokan bit 9,17 dari 1000 karakter *chiper text*.
4.	Analisa dan Penerapan Metode *Particle Swarm Optimization* pada Optimasi Penjadwalan Kuliah	-Data penjadwalan mata kuliah Politeknik Caltex Riau -Parameter partikel, *w* (inertia weight), posisi, kecepatan, nilai *fitness,*	*Particle Swarm Optimization*	Pembagian jadwal mata kuliah yang optimal. Hasil dari penelitian ini adalah parameter partikel lebih besar mempengaruhi hasil solusi lebih baik.
5.	*Penerapan Hybrid Algoritma Genetika dan Particle Swarm Optimization (PSO) untuk menyelesaikan Multi Trip Vehicle Routing Problem.*	-Rute pendistribusian barang depot - Parameter jumlah populasi, jumlah populasi, *crossover rate, mutation rate,* partikel,pbest, gbest,*max* iterasi, kecepatan, c1,c2.	*Hybrid Algoritma Genetika dan Particle Swarm Optimization*	Pembagian Rute pendistribusian barang depot yang optimal. Hasil dari penelitian ini adalah parameter c2 tidak begitu mempengaruhi hasil solusi.

Penelitian ini bertujuan untuk membandingkan ketiga Algoritma tersebut dalam sistem penjadwalan kuliah yang menghasilkan solusi jadwal yang optimal. Variabel pengujiannya adalah Soft Constraint, Hard Constraint, Total Soft Constraint, Total Hard Constraint dan Waktu Eksekusi

Penelitian pertama menyelesaikan permasalahan penjadwalan dengan metode membandingkan kedua Algoritma tersebut dalam sistem penjadwalan kuliah yang menghasilkan solusi jadwal yang optimal dimana nilai *fitness* didapat dari pelanggaran-pelanggaran terhadap adalah soft constraint, hard constraint, total soft constraint, total hard constraint dan waktu eksekusi dari penjadwalan kuliah. Dari penelitian yang dilakukan didapat kesimpulan yaitu, Algoritma

Memetika lebih unggul daripada Algoritma Genetik Tabu Search pada hal waktu (Albar et al., 2012).

Penelitian kedua menyelesaikan permasalahan pembagian tugas mengajar dosen pengampu di FILKOM dengan menggunakan metode *hybrid genetic algorithm* dan *simulated annealing*. Kedua metode ini dilakukan secara sekuensial dengan input berupa berupa data mata kuliah yang diambil oleh dosen pengampu, jumlah kelas, serta batasan-batasan yang ditentukan seperti jumlah minimum dan maksimum dari SKS dan mata kuliah yang dapat diambil.

Penelitian ketiga menyelesaikan permasalahan pengenkripsian dari 1000 karakter *chiper text* dengan melakukan perbandingan *memetic algorithm* dan *tabu search.* Dari penelitian yang dilakukan didapat hasil bahwa performa *memetic algorithm* lebih unggul, hasil akhir yang diperoleh 2 menit 14 detik dan akurasi kecocokan bit adalah 9,17. sedangkan untuk *Tabu search* 8 menit 4 detik dan akurasi kecocokan bit adalah 8. (Ghaziabad, 2008).

Penelitian ketiga menyelesaikan permasalahan penjadwalan dengan metode PSO dimana nilai *fitness* didapat dari pelanggaran-pelanggaran terhadap *constraint* dari penjadwalan kuliah. Dari penelitian yang dilakukan didapat kesimpulan yaitu, jumlah partikel yang lebih besar dapat mempengaruhi hasil solusi yang lebih baik walaupun membutuhkan waktu yang lebih lama dalam pemrosesannya (Rachman et al., 2012).

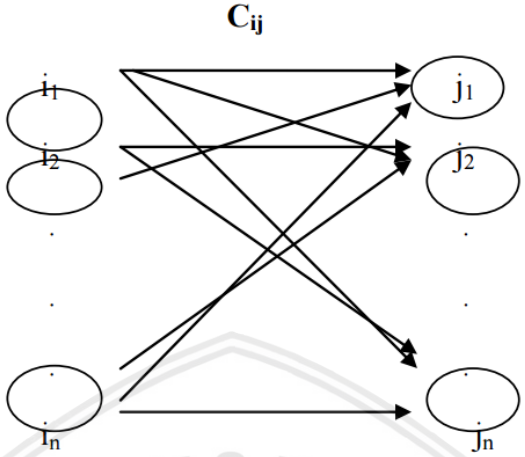
Penelitian keempat menyelesaikan permasalahan Rute pendistribusian barang depot dengan metode *hybrid* GA-PSO dimana penggabungan dari algoritma ini dilakukan secara seri. Proses algoritma dijalankan terlebih dahulu lalu dilanjutkan proses PSO hingga iterasi yg ditentukan tercapai. Dari penelitian yang dilakukan didapat kesimpulan yaitu, semakin besar nilau popsize, max iterasi, dan probabilitas crossover yg diberikan maka nilai fusngsi tujuan yang dihasilkan cenderung lebih baik. Sedangkabn parameter c2 tidak terlalu berpengaruh terhadapa nilai hasil. (Nurrochma et al., 2018).

Berdasarkan penelitian-penelitian yang telah dibahas diatas, penulis menyelesaikan Pemetaan Mengajar Dosen Menggunakan *Memetic Algorithm*. Dengan input berupa data mata kuliah yang diambil oleh dosen pengampu, jumlah kelas, serta batasan-batasan yang ditentukan, sistem ini menghasilkan output berupa pembagian tugas mengajar dosen pengampu dengan total nilai *cost* terkecil. Penelitian ini diharapkan dapat menyelesaikan permasalahan pembagian tugas mengajar dosen pengampu di FILKOM.

2.2 *Assignment problem*

Masalah penugasan (*assignment problem*) adalah suatu masalah mengenai pengaturan objek untuk melaksanakan tugas, dengan tujuan meminimalkan biaya, waktu, jarak, dan sebagainya ataupun memaksimalkan keuntungan (Soemartojo, 1997). Sedangkan menurut Taha (2007), deskripsi yang tepat mengenai model penugasan (*assignment model*) ialah orang terbaik untuk melakukan sebuah pekerjaan. Hal ini dilihat bahwa pekerja yang memiliki kemampuan yang sesuai

dengan pekerjaannya akan memerlukan *cost* yang lebih sedikit dibandingkan dengan yang tidak sesuai, sehingga tujuan dari model penugasan ini adalah untuk menentukan *cost* minimum penugasan pekerja untuk sebuah pekerjaan.



Gambar 2.1 Proses pembentukan nilai penugasan

Sumber: Paendong & Prang (2011)

Pada Gambar 2.1, Paendong & Prang (2011) menyebutkan secara matematis masalah penugasan dapat dinyatakan dalam bentuk variabel keputusan X_{ij} yaitu:

X_{ij} = 1, apabila objek *i* ditugaskan untuk tugas *j*

X_{ij} = 0, apabila objek *i* tidak ditugaskan untuk tugas *j*

Lebih detailnya, model untuk masalah penugasan dapat ditulis sebagai berikut:

Meminimumkan:

$$Z = \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij} X_{ij} \qquad (2.1)$$

$$Z = C_{11}X_{11} + C_{22}X_{22} + ... + C_{mn}X_{mn}$$

Dengan kendala:

$$\sum_{i=1}^{m} X_{ij} = 1 \text{ untuk j = 1, 2, 3, …, n} \qquad (2.2)$$

$$\sum_{j=1}^{n} X_{ij} = 1 \text{ untuk i = 1, 2, 3, …, m} \qquad (2.3)$$

$X_{ij} \geq 0$ untuk semua *i* dan *j*

Dimana:

Z = Fungsi tujuan

X_{ij} = Variabel keputusan

C_{ij} = Nilai konstribusi objek *i* terhadap tugas *j*

m = Jumlah objek (individu atau sumber daya)

n = Jumlah tugas/pekerjaan yang akan diselesaikan

i = Karyawan

j = Tugas/pekerjaan

Dalam Agustini & Rahmadi (2004), dijelaskan bahwa masalah penugasan dalam keadaan tertentu akan mengalami cacah penugasan, dimana jumlah $m \neq n$. Namun, menurut Paendong dan Prang (2011) masalah ini dapat diatasi dengan cara menambahkan *dummy worker* atau *dummy job.*

Pemetaan tugas pengajaran mata kuliah merupakan proses menyusun sejumlah komponen yang terdiri atas dosen, mata kuliah, dan jumlah kelas. Batasan yang diberikan oleh tiap-tiap dosen di FILKOM Universitas Brawijaya Malang adalah sebagai berikut:

1. Beban total SKS yang diambil adalah minimal 12 SKS dan maksimal 25 SKS
2. Jumlah total mata kuliah yang dapat diambil adalah minimal 1 mata kuliah dan maksimal 4 mata kuliah.

2.3 Optimasi

Penyelesaian masalah akan mudah dilakukan jika ukuran data relatif kecil. Masalah akan menjadi kompleks jika data berukuran besar atau melibatkan sejumlah besar entitas. Pada masalah kompleks dibutuhkan juga formulasi matematika yang kompleks yang bisa jadi sangat sulit dibangun atau membutuhkan waktu yang lama. Berdasarkan model matematis yang dibangun bisa dilakukan analisis untuk mencari solusi yang terbaik. Solusi terbaik mungkin dapat diperoleh tetapi memerlukan proses perhitungan yang panjang. Oleh karena itu optimasi dapat didefinisikan sebagai proses pemilihan sebuah solusi dari sejumlah alternatif solusi dengan memenuhi sejumlah batasan *contstraints.*

2.4 *Memetic Algorithm*

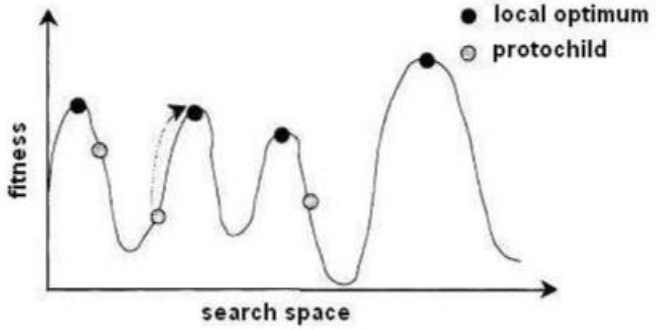
Menurut Yamada et al. (1997), *Memetic Algorithm* adalah perluasan dari Algoritma Genetik. Kelebihan dari *Genetic Algorithm* adalah pada cara kerjanya yang paralel. *Genetic Algorithm* bekerja dalam ruang pencarian yang menggunakan banyak individu sekaligus, sehingga kemungkinan *Genetic Algorithm* untuk terjebak pada ekstrim lokal lebih kecil dibandingkan metode lain. Kekurangan dari *Genetic Algorithm* adalah dalam hal waktu komputasi karena harus melakukan evaluasi fitness pada semua solusi di setiap iterasinya, sehingga *Genetic Algorithm* bisa lebih lambat dibandingkan dengan metode lain. Namun telah diketahui bahwa *Genetic Algorithm* tidak cukup baik untuk mencari solusi yang sangat dekat dengan solusi optimal. Kekurangann ini dapat diakomodasi dengan cara menambahkan metode *Local Search* pada Genetic Algorithm. Hasil penggabungan ini dikenal dengan sebutan *Memetic Algorithm*. *Memetic*

Algorithm adalah suatu metode pencarian heuristic yang memiliki karakteristik yang sama dengan *Genetic Algorithm* dikombinasikan dengan metode Local Search yang secara bersama-sama dapat meningkatkan kualitas pencarian solusi (Moscato, 1989). Pada *Memetic Algorithm*, Local Search bertujuan untuk melakukan perbaikan lokal yang dapat diterapkan sebelum dan atau sesudah proses seleksi, crossover, dan mutasi. Local Search juga dapat berguna untuk mengontrol besarnya ruang pencarian solusi. *Memetic Algorithm* dapat memberikan hasil yang lebih baik daripada Genetic Algorithm, namun memerlukan waktu komputasi yang lebih lama. Dengan metode *Memetic Algorithm* ini, didapatlah proses dengan nilai akurasi yang memuaskan dan nilai *error* seminimal mungkin.

Maka struktur Memetic Algorithm (MA) bisa disusun dengan menambahkan perbaikan lokal sebagai berikut:

Procedure MemeticAlgorithm
Begin
t = 0
Inisialisasi P(t)
While (bukan kondisi berhenti) *do*
Reproduksi C(t) dari P(t)
Evaluasi P(t) dan C(t)
Perbaiki C(t)
Seleksi P(t+1) dari P(t) dan C(t)
t = t + 1
end while
End

Mekanisme perbaikan offspring tersibut diilustrasikan pada Gambar 2.2. Anak yang baru terbentuk (protochild) akan didorong menuju optimum lokal.



Gambar 2.2 Ilustrasi pembentukan *local optimum*

BAB 3 METODOLOGI

Bab ini menjelaskan metodododologi yang digunakan serta langkah - langkah yang dilakukan dalam proses penelitian sistem optimasi pembagian tugas dosen pengampu dengan metode *Memetic Algorithm*. Metodologi penelitian yang dilakukan dalam penelitian ini terdiri dari beberapa tahap diantaranya adalah tipe penelitian, startegi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumuplan data, dan implementasi algoritme.

3.1 Tipe Penelitian

Tipe penelitian yang digunakan oleh peneliti adalah tipe non-implementatif deskripstif, karena penelitian ini lebih menitikberatkan pada karakteristik objek penelitian dari fenomena/situasi tertentu yang sedang diteliti yang kemudian menghasilkan Produk/artefak berupa hasil analisis melalui proses studi kasus.

3.2 Strategi Penelitian

Dalam membangun penelitian ini diperlukan strategi penelitian berupa studi kasus dengan metode algoritme memetik. Metode algoritma memetic yang digunakan ditujukan untuk mengkaji suatu pengaruh *variable* parameter terhadap optimasi solusi. Selain itu strategi penilitian dengan memilih studi kasus pada tempat tertentu ditujukan untuk menggali permasalahan yang penyelesaiannya dapat diselesaikan oleh metode pembelajaran.

3.3 Partisipan Penelitian

Partisipan penelitian yang terlibat dalam penelitian adalah para dosen di fakultas pada tahun 2016. sejumlah data keminatan dosen terhadap mata kuliah yang akan diambil pada suatu semester. Data keminatan dosen tersebut didalamnya mencakup mata kuliah apa saja yang akan diambil oleh dosen untuk diajar pada semester selanjutnya. Jumlah mata kuliah yang dapat diambil, seperti yang dicantumkan pada Lampiran A, bisa berjumlah lima sesuai dengan kolom yang disediakan. Data lainnya yang diperlukan adalah jumlah sks dari setiap mata kuliah yang dapat diambil, Lalu jumlah maksimal dan minimal dari sks dan mata kuliah yang dapat diambil oleh masing-masing dosen, dan prioritas matakuliah yang ingin diambil dosen.

3.4 Lokasi Penelitian

Penelitian ini dilakukan di Fakultas Ilmu Komputer Universitas Brawijaya. Penilitan ditempat tersebut ditujukan untuk mengambil data pemetaan yang digunakan dalam proses yang bersangkutan dengan cara melakukan wawancara kepada pihak yang bertanggung jawab atas penyusunan data tersebut.

3.5 Teknik Pengumpulan Data

Pada tahap pengumpulan data peneliti melakukan pengumpulan data dari narasumber. Dalam penelitian ini data yang digunakan berasal dari angket serta *softcopy* dalam format *.xlsx* pemetaan mengajar dosen periode semester genap 2015/2016 Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang. Data-data ini didapat dengan cara melakukan wawancara kepada pihak akademik yang bertanggung jawab dalam melakukan pembagian tugas mengajar para dosen FILKOM tiap semesternya. Wawancara ini dilakukan untuk mengetahui bagaimana cara kerja tiap parameter pada angket tersebut, seperti apa saja yang menentukan seorang dosen memenuhi kriteria dalam mengajar suatu mata kuliah, atau apa saja yang perlu diperhatikan tiap dosen dalam mengisi angket tersebut, dan sebagainya. Namun karena ketidak cukupan jumlah data yang didapat, maka digunakanlah data *dummy* untuk menentukan prioritas mata kuliah dosen yang dibuat berdasarkan format data asli. Data *dummy* yang digunakan dapat digunakan sebagai tolak ukur dalam pembuatan sistem, sehingga apabila dikemudian hari data asli diterapkan, sistem dapat lebih mudah beradaptasi. Data yang digunakan akan disimpan ke dalam *database* untuk memudahkan sistem dalam mengambil (*fetch*) data dan menyimpan *output* sistem nantinya.

3.6 Teknik Analisis Data

Teknik analisis data dilakukan untuk mengetahui kinerja metode terhadap objek yang digunakan dengan cara mengamati hubungan nilai *fitness* dengan parameter yang diujikan. Disebut parameter yang baik apabila hasil nilai *fitness* dari parameter yang diuji mencapai nilai yang tinggi difase puncak konvergen. Pengujian dilakukan dengan menguji coba beberapa Memetic Algorithm yang diterapkan dalam sistem.

3.7 Implementasi Algoritme

Tahap awal dalam proses implemetasi algoritme adalah menuntukan *constraint* dan memberi bobot pada objek lalu melakukan perhitungan manualisasi dengan tujuan untuk mempermudah peneliti memahami setiap langkah perhitungan dalam metode algoritme memetik dan juga sebagai pembanding hasil perhitungan manual dengan perhitungan sistem. Tahap kedua membuat bagan *flow chart* untuk menentukan alur perhitungan metode algoritme memetic didalam kode program.

3.8 Pengujian Algoritme

Pengujian terhadap sistem dilakukan untuk mengetahui kinerja metode terhadap objek yang digunakan dengan cara mengamati hubungan nilai *fitness* dengan parameter yang diujikan. Disebut parameter yang baik apabila hasil nilai *fitness* dari parameter yang diuji mencapai nilai yang tinggi. Pengujian dilakukan dengan menguji coba beberapa parameter *Memetic Algorithm* yang diterapkan dalam sistem, yaitu:

3.8.1 Pengujian Jumlah Populasi

Pengujian dalam menentukan jumlah populasi ini bertujuan untuk menentukan ukuran populasi yang tepat agar diperoleh solusi yang terbaik. Pengujian ini juga dilakukan untuk mengetahui pengaruh jumlah populasi terhadap nilai *fitness* yang dihasilkan. Pada pengujian ini, jumlah populasi yang digunakan terdapat 10 jenis jumlah yang berbeda dengan masing-masing jenis diuji sebanyak 10 kali untuk nantinya dirata-rata dan mendapatkan nilai acuan pada jenis jumlah populasi tersebut. Jenis jumlah pengujian populasi dan parameter yang digunakan adalah sebagai berikut:

1. Jumlah populasi : 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
2. Jumlah iterasi : 50
3. *Cr* : 0,8
4. *Mr* : 0,2

Tabel rancangan pengujian jumlah populasi dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tabel rancangan pengujian jumlah populasi

Ukuran Populasi	**Nilai *Fitness* pada percobaan ke-**										**Rata – Rata Nilai *Fitness***
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

3.8.2 Pengujian Jumlah Iterasi

Pengujian dalam menentukan jumlah iterasi ini bertujuan untuk menentukan ukuran iterasi yang tepat agar diperoleh iterasi yang terbaik serta mengetahui pengaruh jumlah iterasi terhadap nilai *fitness* yang didapat. Pada pengujian ini, jumlah iterasi yang digunakan terdapat 10 jenis jumlah iterasi yang berbeda dengan masing-masing jenis diuji sebanyak 10 kali untuk nantinya dirata-rata dan mendapatkan nilai acuan pada jenis iterasi populasi tersebut. Jenis jumlah iterasi dan parameter yang digunakan adalah sebagai berikut:

1. Jumlah populasi : Hasil jumlah populasi terbaik dari uji coba jumlah populasi.
2. Jumlah iterasi : 10, 20 , 30, 40, 50, 60, 70, 80, 90, 100
3. *cr* : 0,8
4. *mr* : 0,2

Tabel rancangan pengujian jumlah iterasi populasi dapat dilihat pada Tabel 3.2.

Tabel 3.2 Tabel rancangan pengujian jumlah iterasi

Jumlah Iterasi	Nilai *Fitness* pada percobaan ke-										Rata – Rata Nilai *Fitness*
	1	2	3	4	5	6	7	8	9	10	
10											
20											
30											
40											
50											
60											
70											
80											
90											
100											

3.8.3 Pengujian Parameter Reproduksi *Cr* dan *Mr*

Pengujian dalam menentukan nilai parameter kecepatan *c1* dan *c2* (*cognitive* dan *social components*) bertujuan untuk menentukan nilai parameter yang tepat guna mencapai nilai solusi yang terbaik dan mengetahui pengaruh parameter terhadap nilai *fitness* yang dihasilkan. Terdapat 3 jenis nilai berbeda untuk kombinasi nilai parameter *Cr* dan *Mr*, yaitu 0.1 – 0.9 begitu pula sebaliknya 0.9 – 0.1 dengan catatan nilai kedua parameter harus 1. Untuk parameter lain yang akan digunakan adalah sebagai berikut:

1. Ukuran populas : Hasil jumlah populas terbaik dari uji coba jumlah populasi.

2. Ukuran iterasi : Hasil jumlah iterasi terbaik dari uji coba jumlah iterasi.

Tabel rancangan pengujian parameter *cr* dan *mr* dapat dilihat pada Tabel 3.3.

Tabel 3.3 Tabel rancangan pengujian parameter kecepatan *cr* dan *mr*

Parameter		**Nilai *Fitness* pada percobaan ke-**										**Rata – Rata Nilai *Fitness***
Cr	***Mr***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
0.1	0.9											
0.2	0.8											
0.3	0.7											
0.4	0.6											
0.5	0.5											
0.6	0.4											
0.7	0.3											
0.8	0.2											
0.9	0.1											

3.9 Kesimpulan dan Saran

Berdasarkan hasil pada langkah-langkah penelitian yang telah dilakukan, maka ditariklah kesimpulan dan saran. Kesimpulan akan menjadi jawaban dari rumusan masalah yang telah dibuat pada Bab 1. Sedangkan Saran merupakan harapan penulis yang dimaksudkan untuk memperbaiki kesalahan yang terjadi serta memberikan pertimbangan untuk penelitan dan pengembangan kedepannya.

BAB 4 PERANCANGAN

Bab ini menjelaskan formulasi permasalahan, siklus algoritma Memetic, siklus penyelesaian optimasi pemetaan tugas mengajar dosen dengan menggunanakan *Memetic Algorithm*, perhitungan manual, perancangan antar muka, serta skenario pengujian yang akan digunakan.

4.1 Formulasi Permasalahan

Permasalahan yang dihadapi adalah pembagian tugas mengajar dosen terhadap mata kuliah yang dibuka yang dilakukan oleh pihak akademik dengan melihat data angket yang telah diisi oleh masing-masing dosen seperti pada Lampiran A. Pada data angket tersebut terdapat beberapa parameter seperti prioritas mata kuliah yang akan diambil mulai dari prioritas satu sampai prioritas lima dengan keterangan prioritas satu merupakan prioritas yang paling diutamakan. Parameter selanjutnya adalah mata kuliah mayor dan minor yang menentukan mata kuliah mana yang lebih diprioritaskan untuk diajar oleh dosen. Sifat mata kuliah mayor dan minor ini mirip dengan parameter prioritas mata kuliah, dimana prioritas mata kuliah yang termasuk jenis mayor lebih diutamakan. Parameter selanjutnya adalah pendukung kompetensi yang terdiri dari delapan kompetensi. Kegunaan pendukung kompetensi ini adalah sebagai parameter pendukung yang dapat dijadikan referensi bagi akademik dalam menentukan dosen yang sesuai dengan mata kuliahnya. Setiap dosen dapat mengisi angket tersebut sesuai dengan mata kuliah yang ingin diajarkan dengan memperhatikan batasan-batasan yang ada seperti setiap dosen hanya boleh mengambil mata kuliah yang ingin diajarkan dengan ketentuan untuk jumlah SKS, minimal adalah 12 dan maksimal adalah 25; dan jumlah mata kuliah, minimal adalah 1 dan maksimal adalah 4. Berdasarkan data yang telah diisi para dosen, maka pihak akademik akan menentukan mata kuliah apa saja yang sebaiknya diambil oleh setiap dosen. Namun untuk saat ini proses pembagian tugas mengajar ini masih dikerjakan dengan menggunakan *Microsoft Excel*, sehingga apabila terjadi ketidaksesuaian diperlukan proses yang cukup menyita waktu untuk membenahinya dikarenakan proses kerjanya yang masih *semi-manual*.

4.2 Perancangan Sistem Algoritma *Memetic*

Data yang digunakan pada penelitian ini adalah data *dummy* berupa pemetaan mengajar dosen semester genap 2015/2016 prodi S1 Informatika/Ilmu Komputer di FILKOM Universitas Brawijaya Malang. Berikut contoh data mata kuliah yang dibuka dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar mata kuliah yang dibuka pada semester genap 2015/2016

No.	Mata Kuliah	SKS	Kelas
1	Analisis dan Perancangan Sistem	3	5
2	Arsitektur dan Organisasi Komputer	3	10

Tabel 4.2 Daftar mata kuliah yang dibuka pada semester genap 2015/2016 (lanjutan)

No.	Mata Kuliah	SKS	Kelas
3	Arsitektur Jaringan Terkini	3	7
4	Basis Data Terdistribusi	3	8
5	Desain dan Analisis Algoritma	3	6
6	Game Artificial Intelligence	3	6
7	Grafika Komputer	3	2
8	Jaringan Komputer	4	11
9	Jaringan Multimedia	3	6
10	Jaringan Syaraf Tiruan	3	5
11	Manajemen Industri Teknologi Informasi	3	6
12	Matematika dan Komputasi Lanjut	4	9
13	Metodologi Penelitian Teknologi Informasi	3	7
14	Mixed Reality	3	4
15	Pemrograman Jaringan	3	1
16	Pemrograman Lanjut	5	6
17	Pemrograman Multi Player Game	3	8
18	Pemrograman Web	4	9
19	Pemrosesan Teks	3	5
20	Pengenalan Pola	3	5
21	Perancangan Game	3	3
22	Perencanaan Sumber Daya Perusahaan	3	7
23	Pola-Pola Perancangan	3	5
24	Probabilitas Statistika	4	12
25	Sistem Pakar	3	7
26	Sistem Terdistribusi	3	5

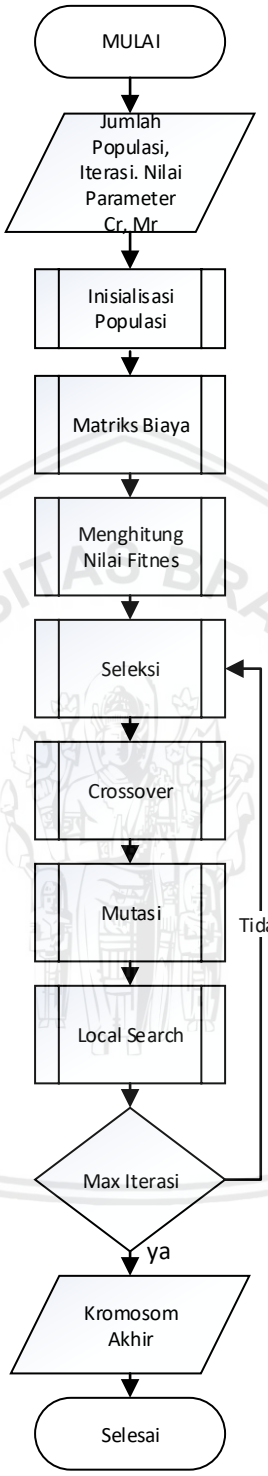
Adapun data dosen FILKOM beserta prioritas mengajar mata kuliah beserta mata kuliah yang masuk kedalam tipe mata kuliah mayor, yaitu mata kuliah yang lebih diprioritaskan untuk diambil oleh dosen tersebut dan tipe mata kuliah minor, yaitu mata kuliah yang kurang diprioritaskan untuk diambil pada Tabel 4.2.

Tabel 4.3 Data pemetaan mengajar dosen

No.	Dosen	Urutan Prioritas Mengajar	Mata Kuliah Mayor	Mata Kuliah Minor
1	Dosen A	1, 3, 5, 14, 21	1, 3, 5	14, 21
2	Dosen B	3, 7, 13, 18, 20	3, 7, 13, 18	20
3	Dosen C	10, 2, 14, 18, 23	10, 2, 14	18, 23
4	Dosen D	8, 9, 10, 14, 21	8, 9, 10	14, 21
5	Dosen E	4, 7, 18, 19, 22	4, 7, 18, 19	22
6	Dosen F	14, 18, 7, 4	14, 18, 7	4
7	Dosen G	1, 3, 20, 19, 18	1, 3, 20	19, 18
8	Dosen H	2, 4, 6, 5, 8	2	4, 6, 5, 8
9	Dosen I	1, 3, 5, 7, 9	1, 3, 5, 7	9
10	Dosen J	11, 13, 15, 17, 19	11, 13	15, 17, 19
11	Dosen K	12, 14, 16, 18, 20	12, 14, 16	18, 20
12	Dosen L	21, 23, 25, 26	21, 23, 25	26
13	Dosen M	22, 24, 7, 19, 3	22, 24, 7	19, 3
14	Dosen N	5, 7, 18, 21, 2	5, 7, 18, 21	2
15	Dosen O	21, 3, 5, 24, 23	21, 3, 5	24, 23
16	Dosen P	4, 6, 7, 19, 3	4, 6, 7	19, 3
17	Dosen Q	10, 8, 7, 9	10, 8, 7	9
18	Dosen R	7, 11, 2, 3, 15	7, 11, 2	3, 15
19	Dosen S	2, 16, 6, 8, 3	2, 16, 6	8, 3
20	Dosen T	6, 3, 23, 1	6, 3	23, 1

Pada memetic, langkah awal yang dilakukan adalah menentukan nilai parameter *populasi, cr, mr,* jumlah iterasi maksimal, dan inisialisasi kromosom secara *random* yang didapat dari data yang telah dikumpulkan. Nilai *fitness* didapat dengan cara menghitung *cost* atau pelanggaran yang didapat dari inisialisasi populasi dengan cara melihat matriks biaya yang telah dibuat. Selanjutnya, Inisialisasi populasi awal, perhitungan fitness masing-masing dari individu, seleksi untuk memilih hasil individu terbaik, tahap crossover, tahap mutasi, dan tahap *local search* untuk mengoptimasi populasi yang dihasilkan dari proses reproduksi. Hal ini diulang kembali pada iterasi selanjutnya hingga terjadi *stopping condition*.

Diagram alir proses Memetic akan dijelaskan pada Gambar 4.1.



Gambar 4. 1 Diagram alir Memetic

4.2.1 Inisialisasi Populasi

Inisialisasi populasi merupakan langkah awal untuk representasi kromosom pada perhitungan Memetic. Pada inisialisasi populasi, kromosom disusun dalam bentuk matriks yang sesuai dengan jumlah data yang akan digunakan. Pada perhitungan manual ini, hanya diambil sebagian data saja yaitu dengan menggunakan 9 mata kuliah dan 7 orang dosen. Untuk tabel perancangan data mata kuliah dapat dilihat pada Tabel 4.3, sedangkan untuk tabel perancangan data dosen dapat dilihat pada Tabel 4.4.

Tabel 4.4 Tabel perancangan data mata kuliah

No	Nama Mata Kuliah	Kode	Kelas
1	Analisis dan Perancangan Sistem	APS	3
2	Jaringan Komputer	JARKOM	4
3	Sistem Terdistribusi	SISTER	3
4	Basis Data Terdistribusi	BDT	2
5	Desain dan Analisis Algoritma	DAA	5
6	Mixed Reality	MIXER	4
7	Pemrograman Web	PWEB	3
8	Pengenalan Pola	PPOL	2
9	Sistem Pakar	SISPAK	3

Tabel 4.5 Tabel perancangan data dosen

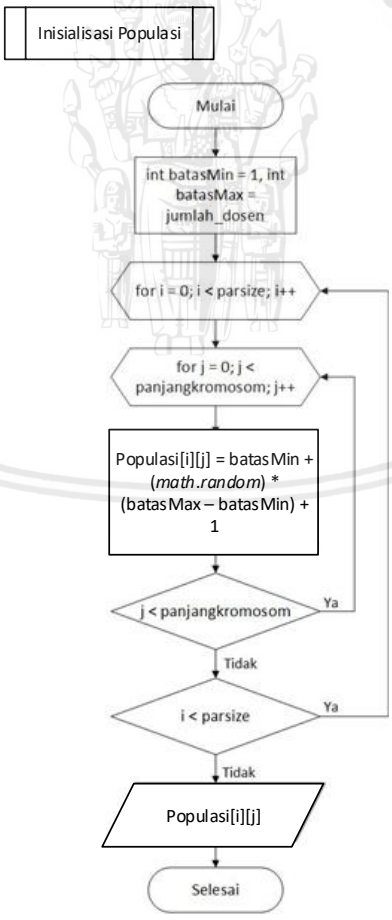
No.	Nama Dosen	Kode Dosen	Prioritas Mengajar Dosen	Mata Kuliah Mayor	Mata Kuliah Minor	Total SKS	Total MatKul
1.	Dosen A	Dosen1	1, 6 , 5	1, 6	5	9	3
2.	Dosen B	Dosen2	1, 7, 6, 4	1,7,6	4	12	4
3.	Dosen C	Dosen3	7, 1, 5, 6	7, 1	5, 6	12	4
4.	Dosen D	Dosen4	2, 8, 1	2, 8	1	10	3
5.	Dosen E	Dosen5	5, 9, 7	5,9	7	9	3
6.	Dosen F	Dosen6	1, 9, 6, 3	1, 9 ,6	3	12	4
7.	Dosen G	Dosen7	3, 4, 8, 2	3, 4	8, 2	13	4

Dari data yang digunakan maka dibuatlah rancangan bentuk dari proses inisialisasi populasi dalam bentuk sebuah tabel berupa Tabel 4.5 sebagai berikut:

Tabel 4.6 Tabel perancangan inisialisasi populasi

P	APS			JARKOM				...	PPOL		SISPAK		
P1	x	x	x	x	x	X	x	....	x	x	x	x	x
P2	x	x	x	x	x	X	x	....	x	x	x	x	x

Pada Tabel 4.6 diatas, jumlah kolom yang merupakan panjang kromosom menunjukkan jumlah kelas mata kuliah yang dibuka, sebagai contoh mata kuliah APS yang dibuka sebanyak 3 kelas, mata kuliah JARKOM yang dibuka sebanyak 4 kelas, dan seterusnya. Baris tabel menunjukkan jumlah partikel yang akan digunakan, sedangkan isi masing-masing kolom tabel diisi oleh kromosom yang menunjukkan kode dosen yang telah ditentukan (disini diberi tanda "x"). Kode dosen yang digunakan sesuai dengan jumlah yang ditentukan, sehingga nilai *range* untuk me-*random* kode dosen dilihat dari jumlah minimum dan maksimum jumlah dosen yang ditentukan. Diagram alir inisialisasi partikel dapat dilihat pada Gambar 4.2.



Gambar 4.2 Diagram alir inisialisasi populasi

Berdasarkan diagram alir pada Gambar 4.2. langkah-langkah inisialisasi populasi dapat dijelaskan sebagai berikut:

1. Mulai.
2. Menginisialisasikan variabel batasMin sebagai nilai minimum populasi dan batasMax sebagai nilai maksimum populasi.
3. Perulangan *i* sebanyak jumlah populasi.
4. Perulangan *j* sebanyak jumlah kelas.
5. Nilai kromosom *random* dengan nilai antara 1 sampai dengan jumlah dosen (variabel batasMax).
6. Keluaran berupa matriks inisialisasi populasi.
7. Selesai.

4.2.2 Membangkitkan Matriks Biaya

Pembangkitan matriks biaya digunakan untuk mengetahui *cost* keminatan untuk setiap pelanggaran yang dibuat. *Cost* keminatan merupakan salah satu dari ketiga *cost* yang digunakan untuk menghitung nilai *fitness.* Proses ini melibatkan inputan user berupa urutan prioritas mengajar dosen mulai dari yang disukai dan jumlah kelas mata kuliah yang dibuka. Langkah awal yang dilakukan adalah memetakan data mata kuliah dan dosen seperti pada Tabel 4.3 dan Tabel 4.4. Setelah memetakan kedua data tersebut, maka ditentukanlah nilai *cost* untuk masing-masing pelanggaran. Nilai *cost* yang digunakan dapat dilihat pada Tabel 4.6.

Tabel 4.7 Tabel pembobotan *cost* keminatan

Prioritas Mata Kuliah yang Diambil	**Jenis Mata Kuliah**	
	Mayor	**Minor**
Prioritas ke-1	1	2
Prioritas ke-2	3	4
Prioritas ke-3	5	6
Prioritas ke-4	7	8
Prioritas ke-5	9	10
Bukan merupakan prioritas	100	

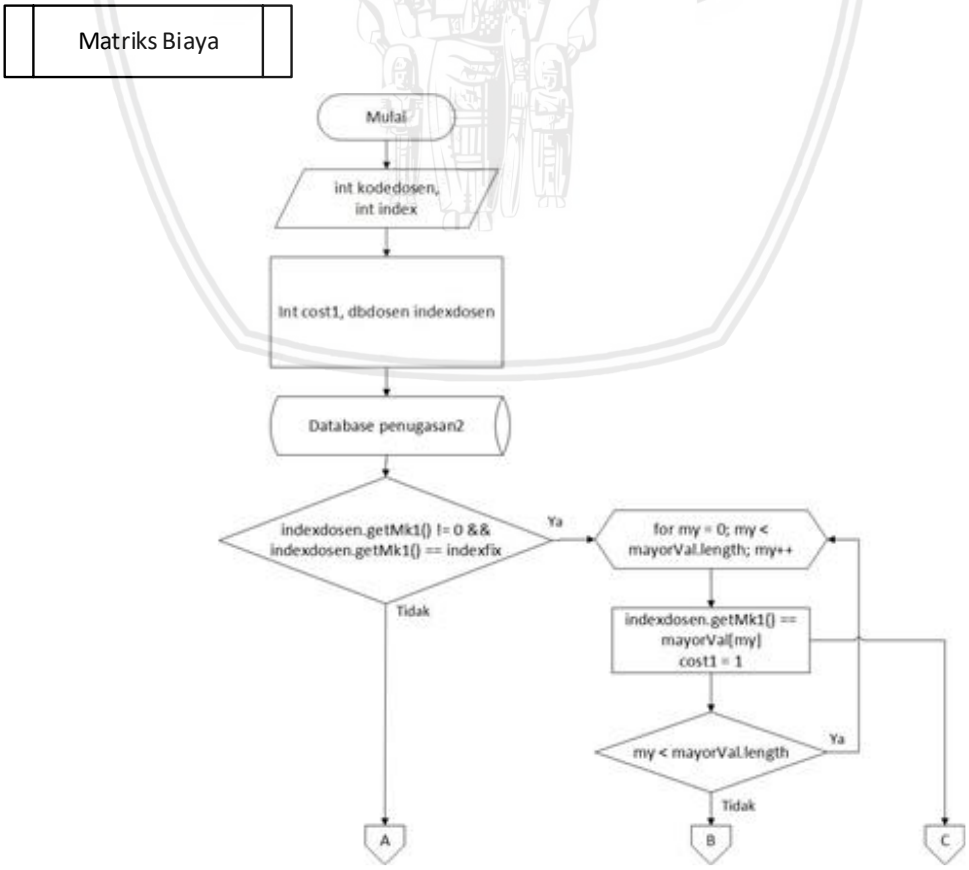
Dengan melihat data mata kuliah, data prioritas dosen tersebut dibuatlah matriks biaya seperti pada Tabel 4.7.

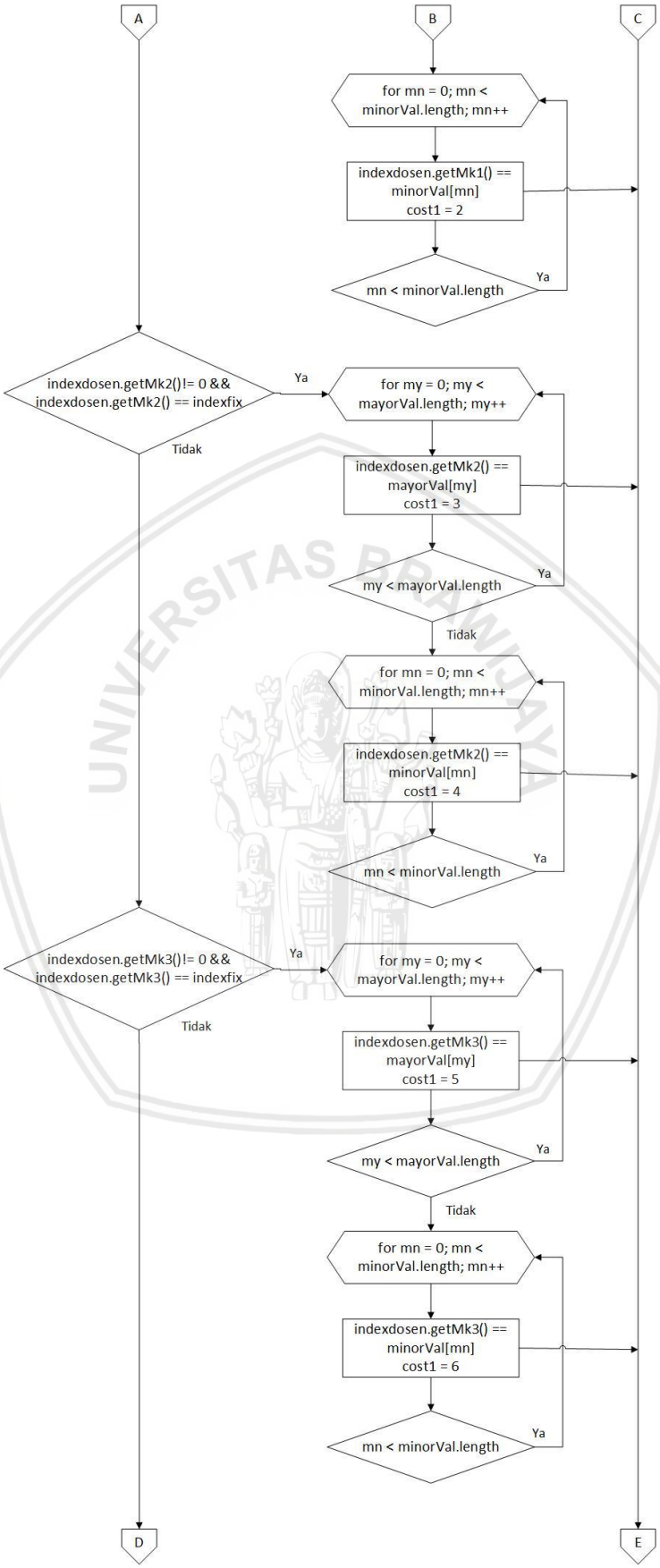
Tabel 4.8 Tabel perancangan matriks biaya

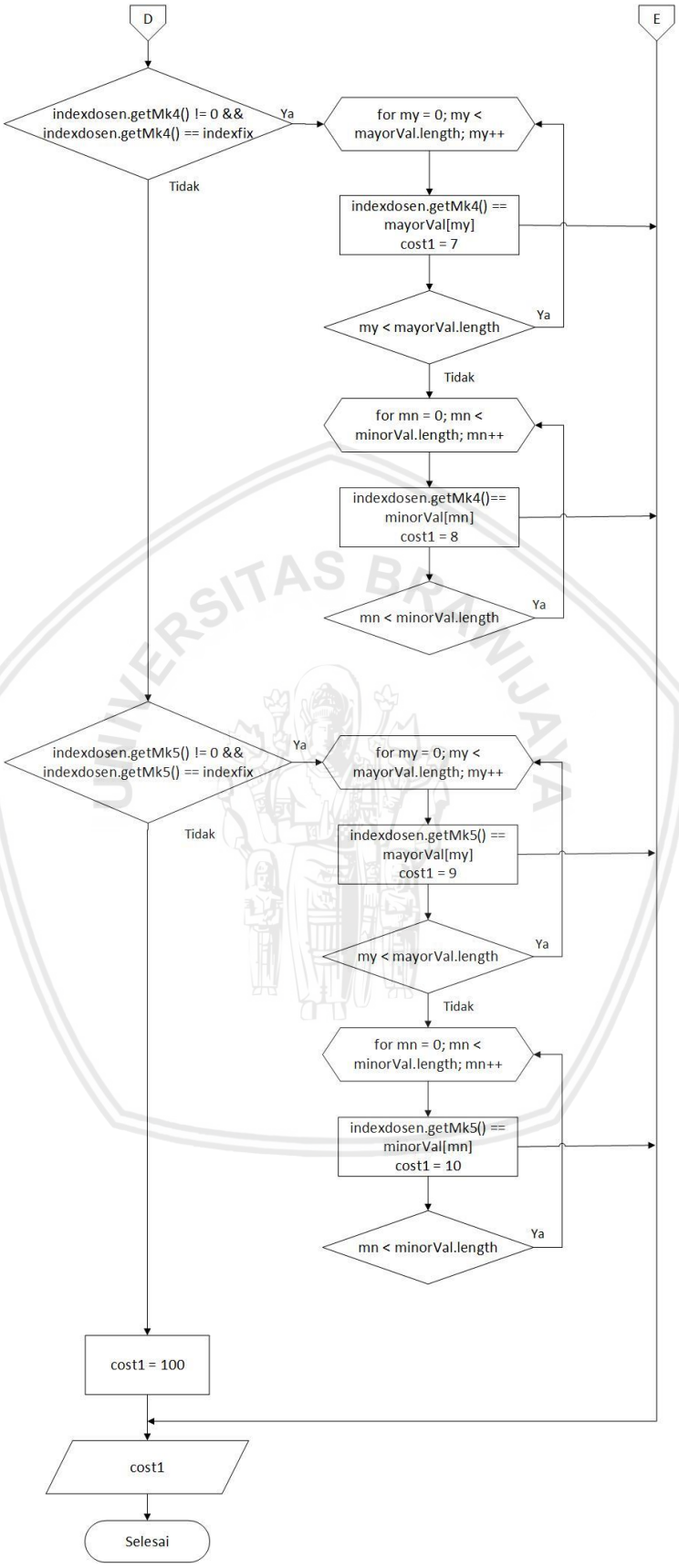
No.	Kode Matkul	Dosen A	Dosen B	Dosen C	Dosen D	Dosen E
1	MK1_KELAS1	x	X	x	x	x
2	MK1_KELAS2	x	X	x	x	x
3	MK2_KELAS1	y	Y	y	y	y
4	MK2_KELAS2	y	Y	y	y	y
5	MK2_KELAS3	y	Y	y	y	y
6	MK3_KELAS1	z	Z	z	z	z

Pada Tabel 4.7, terdapat kolom Kode Matkul dan kolom Dosen. Kolom Kode Matkul merupakan kolom untuk memuat seluruh kelas mata kuliah, sedangkan kolom Dosen untuk memuat data semua dosen. Pada baris dan kolom yang diberi tanda "x, y, z", adalah isi matriks biaya. Untuk menghitung isi matriks biaya ini, dilihat prioritas masing-masing mata kuliah pada dosen yang bersangkutan. Semisal untuk mata kuliah dengan kode MK1 berada pada prioritas keberapa bagi Dosen A. Nilai *cost* keminatan untuk MK1 bagi Dosen A dilihat berdasarkan Tabel 4.6. Proses ini diulang secara berulang-ulang sampai didapatlah matriks biaya untuk seluruh mata kuliah dan dosen.

Diagram alir pembangkitan matriks biaya dapat dilihat pada Gambar 4.3.




A
B
C
for mn = 0; mn < minorVal.length; mn++
indexdosen.getMk1() == minorVal[mn]
cost1 = 2
mn < minorVal.length
Ya
indexdosen.getMk2()!= 0 && indexdosen.getMk2() == indexfix
Ya
Tidak
for my = 0; my < mayorVal.length; my++
indexdosen.getMk2() == mayorVal[my]
cost1 = 3
my < mayorVal.length
Ya
Tidak
for mn = 0; mn < minorVal.length; mn++
indexdosen.getMk2() == minorVal[mn]
cost1 = 4
mn < minorVal.length
Ya
indexdosen.getMk3()!= 0 && indexdosen.getMk3() == indexfix
Ya
Tidak
for my = 0; my < mayorVal.length; my++
indexdosen.getMk3() == mayorVal[my]
cost1 = 5
my < mayorVal.length
Ya
Tidak
for mn = 0; mn < minorVal.length; mn++
indexdosen.getMk3() == minorVal[mn]
cost1 = 6
mn < minorVal.length
Ya
D
E



Gambar 4. 3 Diagram alir matriks biaya

Berikut adalah penjelasan operasi membangkitkan matriks biaya pada Gambar 4.3:

1. Mulai.
2. Input berupa variable *kodedosen* yang merupakan nilai kode dosen yang didapat dari matriks populasi dan variable *index* yang digunakan untuk menentukan kelas matakuliah pada matriks populasi.
3. Melakukan pengambilan data pada database *penugasan2* dan menyimpannya ke dalam kelas array list.
4. Inisialisasi variable *cost1* untuk menampung nilai *cost*, lalu variable *indexdosen* sebagai tempat untuk menampung nilai matakuliah suatu dosen.
5. Melakukan pengecekan kondisi dengan menggunakan *if-else*. Apabila matakuliah yang didapat dari matriks populasi sama dengan nilai mata kuliah dosen pada prioritas ke 1, 2, 3, 4, 5, maka akan dilakukan pengecekan lagi dengan data mata kuliah mayor dan minor dari dosen tersebut. Nilai mata kuliah mayor dan minor dimasukkan kedalam array, sehingga untuk pengecekan mata kuliah mayor dan minor dilakukan per-indeks dalam array tersebut. Setelah dilakukan pengecekan ini, maka nilai *cost* yang telah ditentukan akan didapat.
6. Apabila nilai matakuliah yang didapat dari matriks populasi tidak termasuk dalam prioritas ke 1, 2, 3, 4, 5 dari dosen tersebut, maka nilai *cost* adalah 100.
7. Output berupa nilai *cost* yang dimasukkan dalam variable *cost1*.
8. Selesai.

4.2.3 Hitung Nilai *Fitness*

Nilai *fitness* merupakan nilai yang menunjukkan baik tidaknya suatu kromosom dalam solusi awal populasi. Setiap populasi yang terdiri dari barisan kromosom memiliki nilai *fitness* yang didapat dengan cara menghitung *cost* pelanggaran pada ketiga batasan yang telah ditetapkan, lalu menjumlahkan seluruh *cost* tersebut. Lalu sebuah konstanta digunakan untuk kemudian dikurang dengan total *cost* tersebut. Untuk lebih jelasnya dapat dilihat pada persamaan berikut:

$$total\_cost = cost\_keminatan + cost\_sks + cost\_kelas \quad (4.1)$$

$$Fitness = C - total\_cost \quad (4.2)$$

Untuk nilai konstanta yang digunakan disini adalah 100000, karena bilangan tersebut mudah dikurangkan dengan nilai total *cost* yang didapat dari tiap populasi yang berkisar sampai puluhan ribu. Selain itu hasil pengurangan dari konstanta dapat digunakan sebagai nilai *fitness* yang umumnya memandang nilai yang lebih besar sebagai nilai *fitness* yang lebih baik.

Langkah awal yang perlu dilakukan adalah dengan mencari total nilai *cost* keminatan untuk masing-masing populasi. Nilai *cost* keminatan untuk masing-

masing kromosom yang ada di dalam populasi dicari dengan menggunakan matriks biaya, setelah itu nilai *cost* keminatan seluruh kromosom pada suatu populasi dijumlahkan sehingga didapatlah nilai total *cost* keminatan. Tabel perancangan untuk mencari nilai total *cost* keminatan digambarkan pada Tabel 4.8.

Tabel 4.9 Tabel perancangan nilai total *cost* keminatan

P	APS			JARKOM				…	SISPAK			Total *Cost* Keminatan
P1	x	x	x	x	x	x	x	…	x	x	x	∑P1(x)
P2	x	x	x	x	x	x	x	…	x	x	x	∑P2(x)

Langkah kedua adalah mencari nilai *cost* terhadap SKS. Disini kita mencari pelanggaran terhadap batas minimum dan maksimum terhadap SKS yang boleh diambil lalu menjumlahkannya. Ketentuan batas minimum dan maksimum sks dan kelas yang digunakan ini berdasarkan dengan meninjau hasil pengumpulan data terhadap pihak akademik. Batas minimum SKS yang diberikan adalah 12, sedangkan untuk batas maksimum adalah 30. Sebagai contoh apabila terdapat seorang dosen memiliki total SKS dari semua mata kuliah yang ia ambil sebanyak 34 SKS, maka nilai *cost* SKS-nya adalah 4 yang didapat dari 34 dikurangi 30. Hal yang sama berlaku apabila terdapat seorang dosen yang memiliki total SKS sebanyak 10, maka nilai *cost* SKS-nya adalah 2 yang didapat dari 12 dikurangi 10.

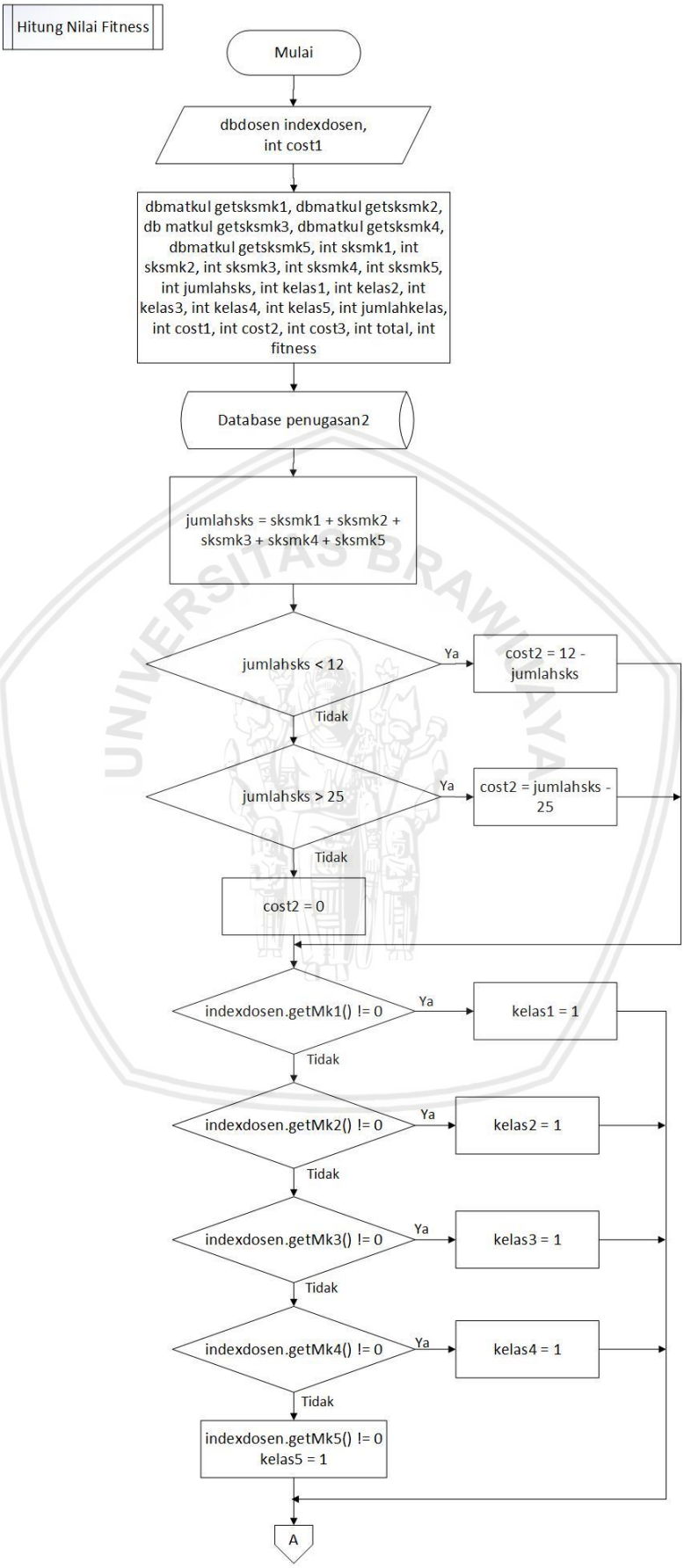
Langkah ketiga adalah mencari nilai *cost* terhadap kelas mata kuliah. Disini kita mencari pelanggaran terhadap batas minimum dan maksimum terhadap kelas mata kuliah yang boleh diambil lalu menjumlahkannya. Batas minimum kelas mata kuliah yang diberikan adalah 1 dan maksimum adalah 4. Cara yang digunakan dalam mencari *cost* kelas ini mirip dengan cara untuk mencari *cost* SKS.

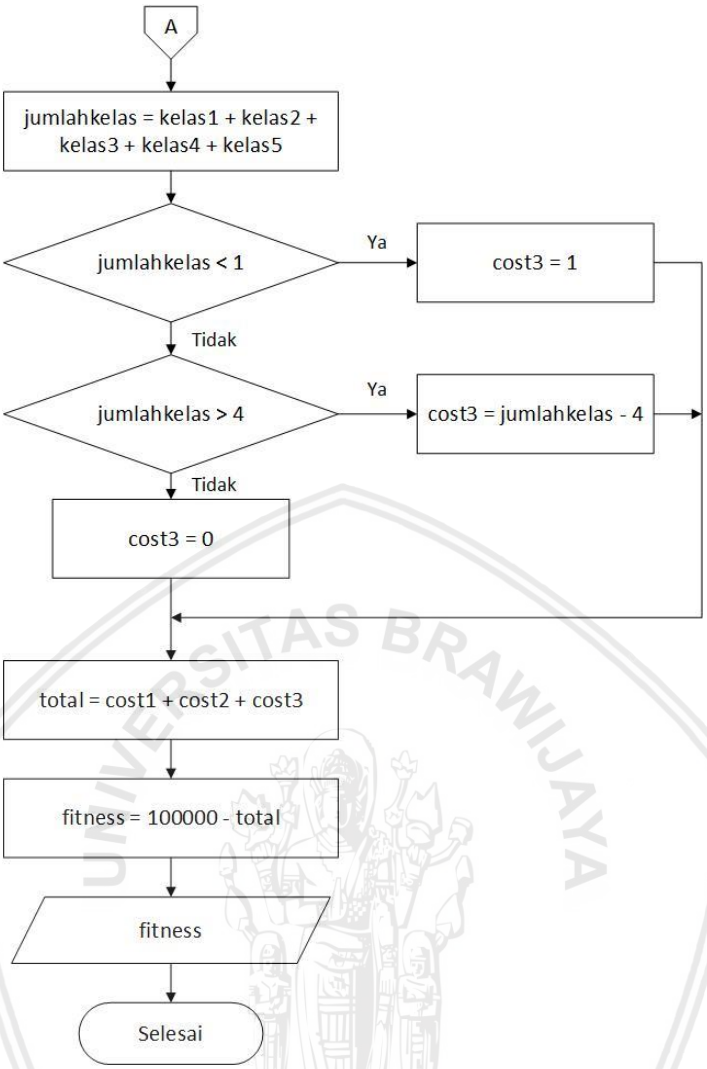
Dari ketiga *cost* yang telah didapat, maka dijumlahkan ketiga *cost* tersebut sehingga didapatlah total *cost*. Setelah total *cost* didapat, maka digunakanlah persamaan (4.2) untuk mencari nilai *fitness* masing-masing populasi. Tabel perancangan untuk mencari nilai *fitness* dapat dilihat pada Tabel 4.9.

Tabel 4.10 Tabel perancangan nilai *fitness*

P	Total *Cost* Keminatan	Total *Cost* SKS	Total *Cost* Kelas	Total *Cost*	Nilai *Fitness*
P1	a	b	c	a + b + c	100000 - (a + b + c)
P2	x	y	z	x + y + z	100000 - (x + y + z)

Diagram alir proses hitung nilai *fitness* dapat dilihat pada Gambar 4.4.


Hitung Nilai Fitness
Mulai
dbdosen indexdosen, int cost1
dbmatkul getsksmk1, dbmatkul getsksmk2, db matkul getsksmk3, dbmatkul getsksmk4, dbmatkul getsksmk5, int sksmk1, int sksmk2, int sksmk3, int sksmk4, int sksmk5, int jumlahsks, int kelas1, int kelas2, int kelas3, int kelas4, int kelas5, int jumlahkelas, int cost1, int cost2, int cost3, int total, int fitness
Database penugasan2
jumlahsks = sksmk1 + sksmk2 + sksmk3 + sksmk4 + sksmk5
jumlahsks < 12
Ya
cost2 = 12 - jumlahsks
Tidak
jumlahsks > 25
Ya
cost2 = jumlahsks - 25
Tidak
cost2 = 0
indexdosen.getMk1() != 0
Ya
kelas1 = 1
Tidak
indexdosen.getMk2() != 0
Ya
kelas2 = 1
Tidak
indexdosen.getMk3() != 0
Ya
kelas3 = 1
Tidak
indexdosen.getMk4() != 0
Ya
kelas4 = 1
Tidak
indexdosen.getMk5() != 0
kelas5 = 1
A



Gambar 4. 4 Diagram alir menghitung nilai *fitness*

Berdasarkan diagram alir pada Gambar 4.4, langkah-langkah menghitung nilai *fitness* dapat dijelaskan sebagai berikut:

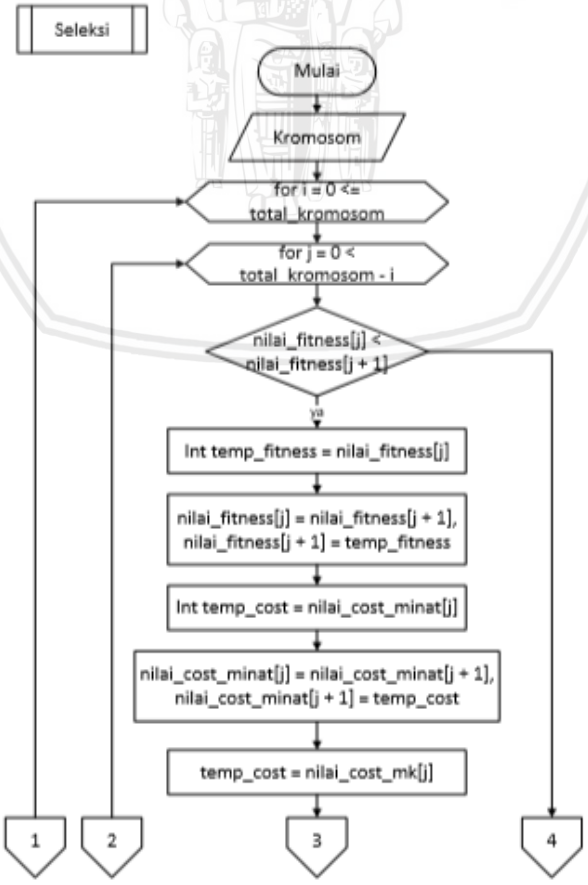
1. Mulai.
2. Input berupa variable *indexdosen* yang didapat dari kelas array list untuk menyimpan data dosen. Nilai cost1 didapat dari hasil proses matriks biaya.
3. Inisialisasi variable yang akan digunakan.
4. Melakukan pengecekan pada array list untuk menyimpan data mata kuliah untuk menentukan nilai *sksmk1*, *sksmk2*, *sksmk3*, *sksmk4*, *sksmk5*.
5. Menentukan jumlah sks dari masing-masing dosen.
6. Membuat kondisi pengecekan dengan *if-else*. Apabila jumlah sks dari dosen tersebut kurang dari 12, maka nilai *cost2* adalah 12 - *jumlahsks*. Sedangkan apabila jumlah sks dari dosen tersebut lebih dari 25, maka nilai *cost2*-nya adalah *jumlahsks* – 25.

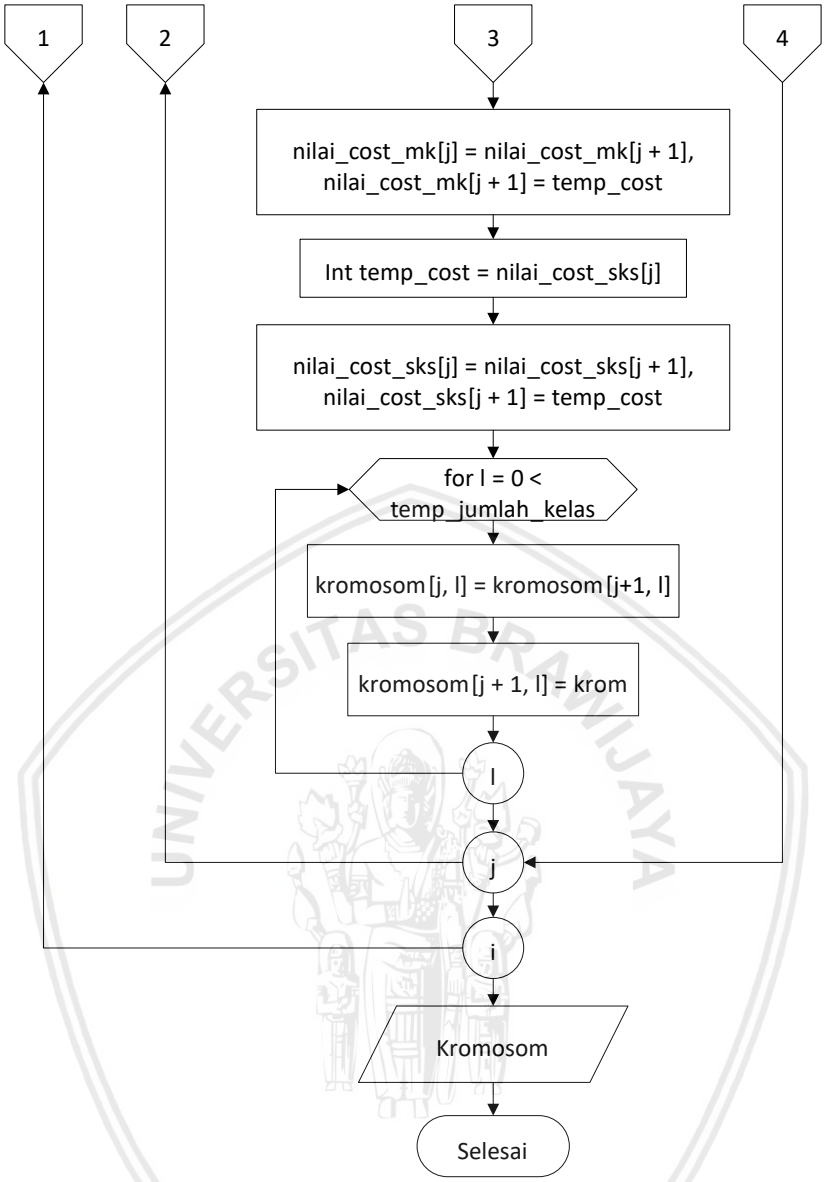
7. Apabila jumlah sks dosen yang didapat masuk dalam range 12 sampai 25, maka nilai *cost2*-nya adalah 0.
8. Membuat kondisi pengecekan dengan *if-else*, apabila nilai mata kuliah yang terdapat dalam *indexdosen* tidak sama dengan 0, maka nilai variable kelas adalah 1.
9. Menjumlahkan nilai variable-variable kelas yang telah didapat.
10. Membuat kondisi pengecekan dengan *if-else*. Apabila jumlah kelas yang didapat kurang dari 1, maka nilai *cost3* adalah 1. Sedangkan apabila jumlah kelas yang didapat lebih dari 4, maka nilai *cost3* adalah *jumlahkelas* - 4.
11. Apabila jumlah kelas yang didapat masuk dalam range 1 sampai 4, maka nilai *cost3* adalah 0.
12. Menjumlahkan ketiga nilai *cost* yaitu *cost1* + *cost2* + *cost3*.
13. Mencari nilai *fitness* dengan rumus 100000 – total cost.
14. Output berupa nilai *fitness*.

Selesai.

4.2.4 Seleksi

Proses seleksi menggunakan *elitism selection*. Masing-masing individu dihitung nilai *fitness*-nya, kemudian dirangking yang terbaik sesuai jumlah *popSize*.





Gambar 4. 5 Diagram alir seleksi

Berdasarkan diagram alir pada Gambar 4.5, langkah-langkah menentukan seleksi dapat dijelaskan sebagai berikut:

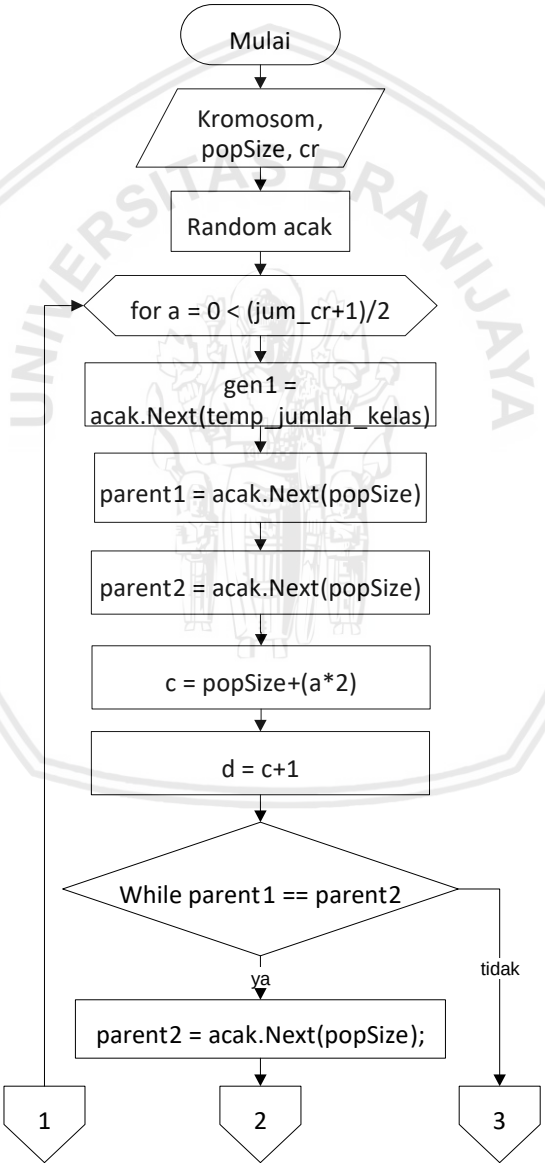
1. Mulai
2. Masukan berupa kromosom.
3. Pengulangan i sebanyak total_kromosom.
4. Pengulangan j sebanyak total_kromosom – i
5. Kondisi jika nilai_fitness[j] < nilai_fitness[j+1], maka nilai_fitness[j] digantikan dengan nilai_fitness[j+1].
6. Nilai_cost_minat[j] digantikan dengan nilai_cost_minat [j+1].
7. Nilai_cost_mk[j] digantikan dengan nilai_cost_mk [j+1].
8. Nilai_cost_sks[j] digantikan dengan nilai_cost_sks[j+1].

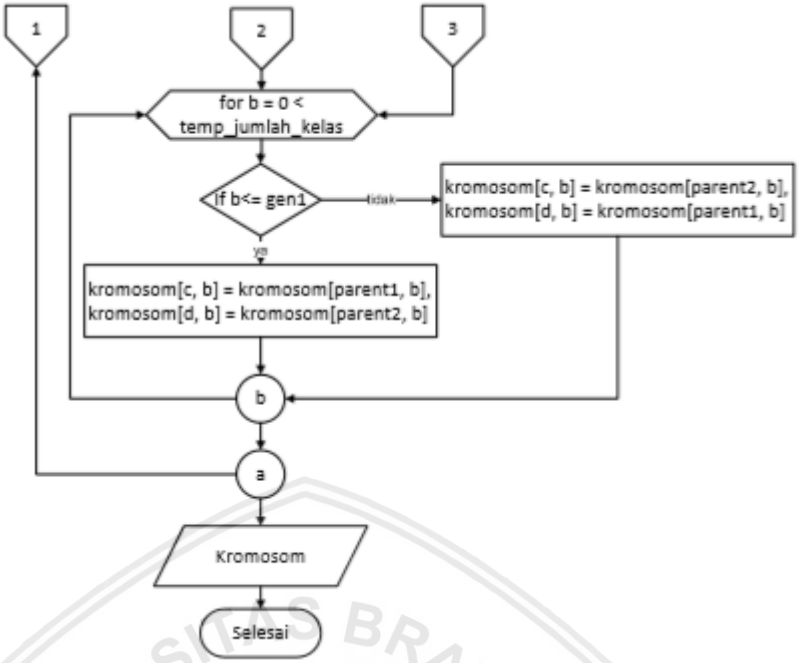
9. Pengulangan l sebanyak temp_jumlah_kelas.
10. Kromosom[j,l] digantikan dengan kromosom[j+1,l].
11. Keluaran berupa populasi kromosom.
12. Selesai.

4.2.5 Crossover

Proses *crossover* menggunakan *one-cut-point*. Melibatkan dua individu yang dipilih secara acak, kemudian menukar nilai gen induk tersebut untuk mendapat hasil *offspring*.

Diagram alir proses crossover dapat dilihat pada Gambar 4.6.





Gambar 4. 6 Diagram alir *crossover*

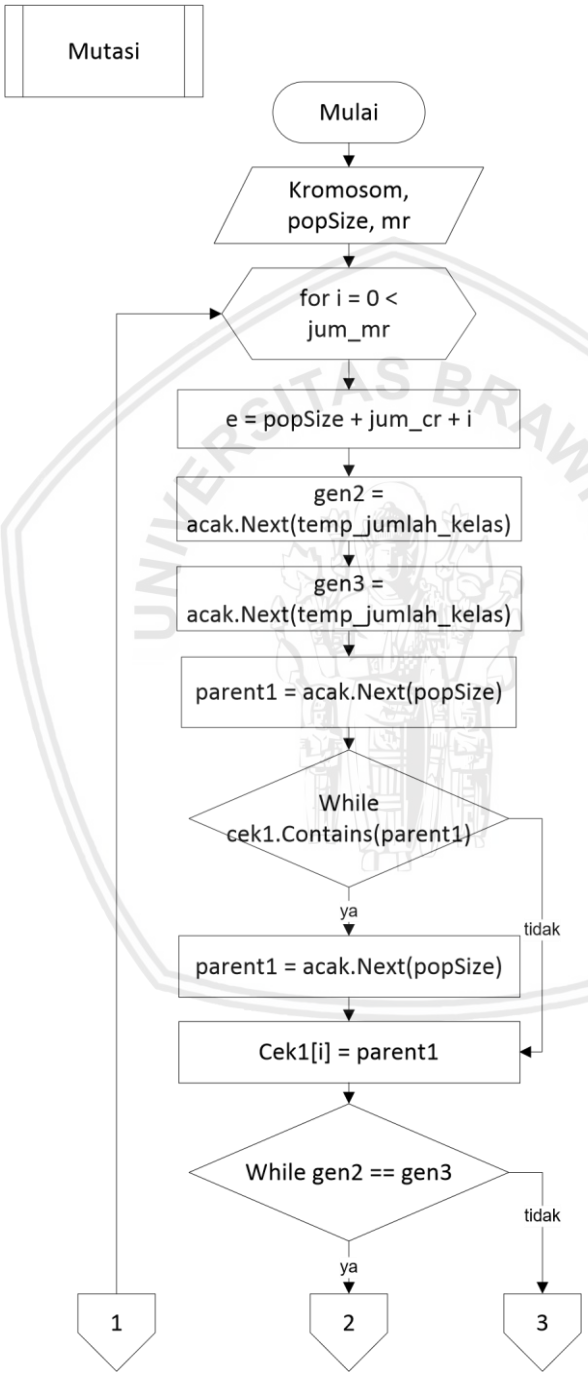
Berdasarkan diagram alir pada Gambar 4.6, langkah-langkah menentukan *crossover* dapat dijelaskan sebagai berikut:

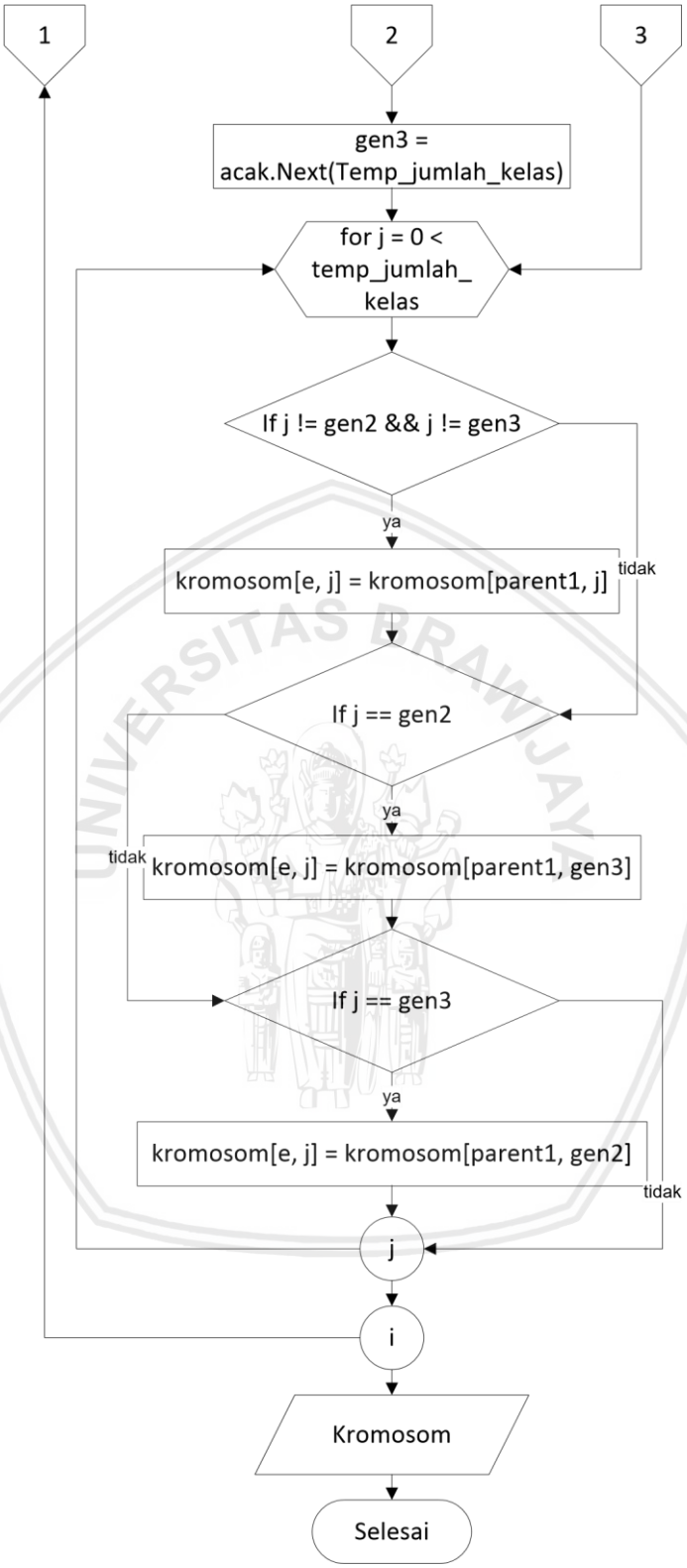
1. Mulai.
2. Masukan berupa kromosom, popSize dan *cr*.
3. Inisialisasi fungsi random acak.
4. Pengulangan a sebanyak jumlah *offspring crossover*+1 dibagi 2.
5. Menentukan titik *crossover* gen1 secara acak.
6. Menentukan *parent*1 dan *parent*2 secara acak.
7. c adalah popSize+(a*2), sedangkan d adalah c+1
8. Kondisi jika *parent*1 bernilai sama dengan *parent*2, maka *parent*2 dipilih ulang secara acak.
9. Pengulangan b sebanyak temp_jumlah_kelas.
10. Kondisi jika nilai b <= nilai gen1, maka kromosom[c,b] digantikan dengan kromosom[parent1, b], dan kromosom[d,b] digantikan dengan kromosom[parent2,b].
11. Jika kondisi b > nilai gen1, maka kromosom[c,b] digantikan dengan kromosom[parent2, b], dan kromosom[d,b] digantikan dengan kromosom[parent1,b].
12. Keluaran berupa kromosom.
13. Selesai

4.2.6 Mutasi

Proses mutasi menggunakan *insertion mutation*. Melibatkan satu individu yang dipilih secara acak, kemudian mengambil dan menyisipkan nilai gen yang dipilih secara acak.

Diagram alir hitung nilai posisi populasi dapat dilihat pada Gambar 4.7.





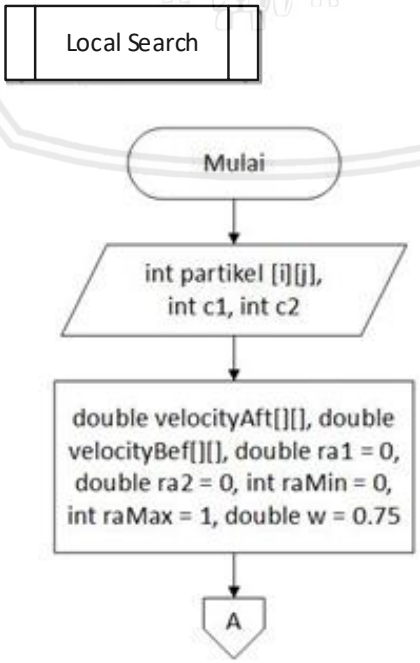
Gambar 4. 7 Diagram mutasi

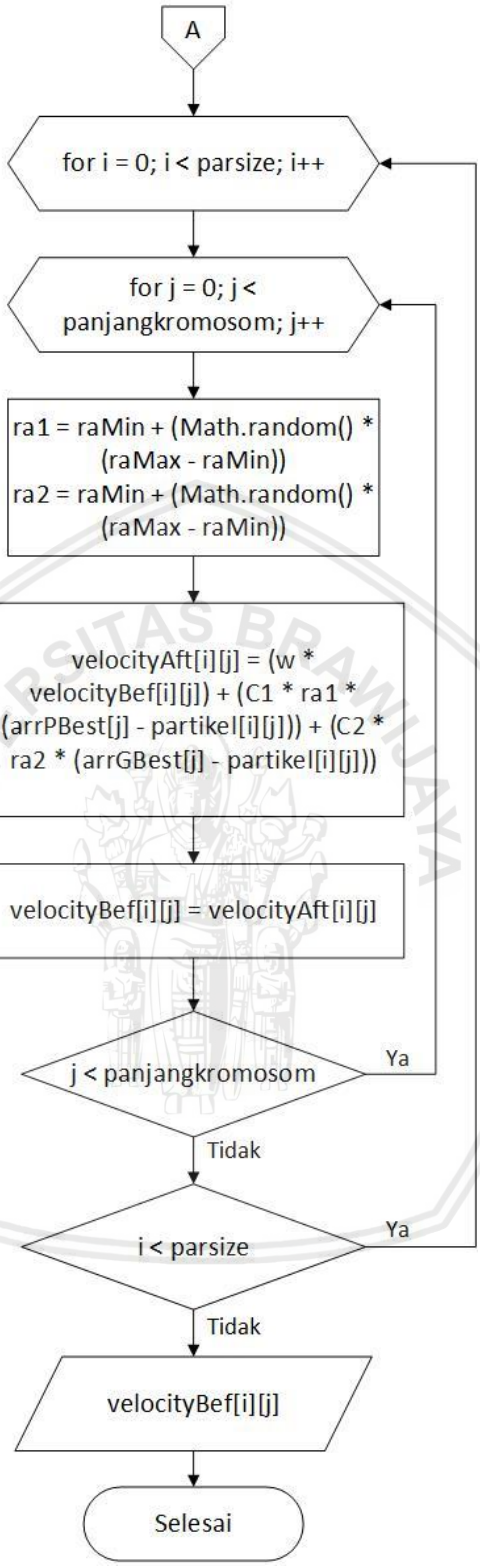
Berdasarkan diagram alir pada Gambar 4.7, langkah-langkah menentukan nilai posisi baru populasi dapat dijelaskan sebagai berikut:

1. Mulai.
2. Masukan berupa kromosom, popSize dan *mr*.
3. Pengulangan i sebanyak jumlah *offspring* mutasi.
4. Inisialisasi e adalah popSize + jum_mr + i.
5. Memilih secara acak titik mutasi gen2 dan gen3.
6. Memilih secara acak *parent* proses mutasi.
7. Kondisi jika nilai *parent* mutasi sama dengan *parent crossover*, maka *parent* mutasi dipilih ulang secara acak.
8. Kondisi jika nilai gen2 sama dengan nilai gen3, maka gen3 dipilih ulang secara acak.
9. Pengulangan j sebanyak temp_jumlah_kelas.
10. Kondisi jika j tidak sama dengan gen2 dan j tidak sama dengan gen3, maka kromosom[e,j] digantikan dengan kromosom[parent1,j].
11. Kondisi jika j bernilai sama dengan gen2, maka kromosom[e,j] digantikan dengan kromosom[parent1,gen3].
12. Kondisi jika j bernilai sama dengan gen3, maka kromosom[e,j] digantikan dengan kromosom[parent1,gen2].
13. Keluaran berupa kromosom.
14. Selesai.

4.2.7 Local Search

Selanjutnya memasuki proses *Local Search* dengan melibatkan hasil *offspring* dari proses reproduksi crossover dan mutasi. Populasi offspring tersebut dijadikan partikel lalu dihitung nilai kecepatannya Diagram alir proses *Local Search* dapat dilihat pada Gambar 4.8.





Gambar 4. 8 Diagram alir menghitung *Local Search*

Berdasarkan diagram alir pada Gambar 4.8, langkah-langkah menentukan nilai kecepatan partikel pada *local search* dapat dijelaskan sebagai berikut:

1. Mulai.

2. Input berupa partikel, dan parameter c1, c2 yang sudah konstan.
3. Deklasrasi variable velocityAft[][] sebagai array untuk menyimpan nilai kecepatan pada iterasi saat ini, variable velocityBef[][] sebagai array untuk menyimpan nilai kecepatan pada iterasi sebelumnya. Variable raMin dan raMax sebagai batas nilai minimum dan maksimum untuk me-*random* parameter ra1 dan ra2, lalu variable w sebagai nilai dari bobot *inertia*.
4. Membuat fungsi perulangan untuk menghitung nilai kecepatan dari tiap-tiap kromosom pada masing-masing partikel. Hasil akan dimasukkan ke dalam variable velocityAft[][].
5. Variable velocityAft[][] dirubah menjadi variable velocityBef[][] untuk digunakan dalam mencari nilai kecepatan pada iterasi selanjutnya.
6. Output berupa matriks kecepatan partikel.
7. Selesai.

4.2.8 Kondisi Terpenuhi

Pada Memetic kondisi terpenuhi yang digunakan, jika jumlah iterasi yang ditentukan telah terpenuhi.

4.3 Siklus Penyelesaian Masalah Menggunakan *Memetic Algorithm*

Siklus penyelesaian masalah menggunakan *Memetic Algorithm* menjelaskan penerapan algoritma MA dalam menyelesaikan masalah optimasi pembagian tugas mengajar dosen pengampu. Pada pembahasan ini akan difokuskan pada perhitungan manual yang sederhana yaitu menggunakan data optimasi untuk 1 iterasi saja, dengan menggunakan sebagian data dosen dan data *dummy* matakuliah prodi Informatika. Pada siklus penyelesaian masalah terdapat pembangkitan matriks biaya, inisialisasi populasi, mencari nilai *fitness*, crossover, mutase, kecepatan, posisi.

4.3.1 Inisialisasi Populasi

Pada proses inisialisasi populasi, nilai dalam bentuk bilangan bulat dibangkitkan secara *random* sesuai dengan jumlah populas dan jumlah kromosom yang digunakan. Untuk populas, pada perhitungan ini dibangkitkan sebanyak 5 buah populasi. Untuk proses inisialisasi populasi dapat dilihat pada Tabel 4.8.

Tabel 4. 1 Tabel inisialisasi populasi

P	APS			JARKOM				…	PPOL		SISPAK		
P1	4	5	3	5	7	7	2	….	7	3	4	6	4
P2	6	4	5	1	5	2	7	….	5	7	3	1	1
P3	4	2	5	6	7	4	1	….	5	4	1	3	5
P4	1	5	3	3	6	5	5	….	4	6	1	1	2

P5	3	2	4	3	1	4	4	....	1	2	5	3	6

Pada Tabel 4.10, baris merepresentasikan jumlah populasi, sedangkan kolom merepresentasikan panjang kromosom.

4.3.2 Membangkitan Matriks Biaya

Pada perhitungan manual, pemetaan data mata kuliah dapat dilihat pada Tabel 4.3 dan untuk pemetaan data dosen dapat dilihat pada Tabel 4.4. Dengan melihat data tersebut, maka dihitunglah nilai *cost* keminatannya berdasarkan nilai pembobotan untuk masing-masing *cost* yang terdapat pada Tabel 4.6. Semisal pada kode Dosen1, *cost* keminatan terhadap mata kuliah Analisis dan Perancangan Sistem (APS) merupakan 1, karena mata kuliah APS merupakan mata kuliah prioritas pertama yang ingin diajar sekaligus termasuk mata kuliah mayor, lalu untuk *cost* terhadap mata kuliah Mixed Reality (MIXER) merupakan 3, karena MIXER merupakan mata kuliah prioritas kedua dan termasuk jenis mata kuliah mayor dosen tersebut, begitu pula seterusnya. Sedangkan untuk mata kuliah yang tidak diambil oleh dosen, maka akan diberi nilai 100 untuk memberikan *cost* yang cukup besar apabila nilai kromosom yang terdapat dalam populasi memberikan nilai yang kurang cocok dengan data yang ada. Data matriks biaya dapat dilihat pada Tabel 4.11.

Tabel 4. 2 Tabel matriks biaya

No.	Kode Matkul	Dosen1	Dosen2	Dosen3	Dosen4	Dosen5	Dosen6	Dosen7
1	APS_1	1	1	3	5	100	1	100
2	APS_2	1	1	3	5	100	1	100
3	APS_3	1	1	3	5	100	1	100
4	JARKOM_1	100	100	100	1	100	100	8
5	JARKOM_2	100	100	100	1	100	100	8
6	JARKOM_3	100	100	100	1	100	100	8
7	JARKOM_4	100	100	100	1	100	100	8
8	SISTER_1	100	100	100	100	100	8	1
9	SISTER_2	100	100	100	100	100	8	1
10	SISTER_3	100	100	100	100	100	8	1
11	BDT_1	100	8	100	100	100	100	3
12	BDT_2	100	8	100	100	100	100	3
13	DAA_1	6	100	6	100	1	100	100
14	DAA_2	6	100	6	100	1	100	100
15	DAA_3	6	100	6	100	1	100	100

Tabel 4. 11 Tabel matriks biaya (lanjutan)

16	DAA_4	6	100	6	100	1	100	100
17	DAA_5	6	100	6	100	1	100	100
18	MIXER_1	3	5	8	100	100	5	100
19	MIXER_2	3	5	8	100	100	5	100
20	MIXER_3	3	5	8	100	100	5	100

Pada Tabel 4.11 diatas, untuk baris merepresentasikan kode mata kuliah dan kolom merepresentasikan kode dosen.

4.3.3 Menghitung Nilai *Fitness*

Dengan melihat data inisialisasi populasi pada Tabel 4.10, serta data matriks biaya pada Tabel 4.11, dihitunglah nilai *cost* pertama, yaitu *cost* keminatan. Untuk mencari *cost* Keminatan perlu diperhatikan pada rangkaian kromosom-kromosom populasi tersebut. Pada populasi 1 (P1) baris pertama dengan kolom APS pertama (indeks 1,1) terdapat nilai 4. Artinya adalah nilai *cost* dosen dengan kode nomor 4 (Dosen4) terhadap mata kuliah APS. Untuk indeks 1,2, adalah nilai *cost* dosen dengan kode nomor 5 (Dosen5) terhadap mata kuliah APS, begitu pula seterusnya. Nilai *cost* ini akan dijumlahkan tiap populasinya sehingga disini didapat 5 buah total *cost* keminatan. Untuk total *cost* keminatan dapat dilihat pada Tabel 4.12.

Tabel 4. 3 Tabel total *cost* keminatan

P	APS			JARKOM				…	PPOL		SISPAK			Total Cost
P1	5	100	3	100	8	8	100	….	6	100	100	3	100	1443
P2	1	5	100	100	100	100	8	….	100	100	100	100	100	1539
P3	5	1	100	100	8	1	100	….	100	3	100	100	3	1727
P4	1	100	3	100	100	100	100	….	3	100	100	100	100	1924
P5	3	1	5	100	100	1	1	….	100	100	3	100	3	1626

Nilai *cost* kedua adalah *cost* terhadap SKS. Disini kita mencari pelanggaran terhadap batas minimum dan maksimum terhadap SKS yang boleh diambil lalu menjumlahkannya. Pada Tabel 4.4, dapat dilihat bahwa terdapat pelanggaran terhadap batas minimum SKS. Batas minimum SKS yang diberikan adalah 12, sedangkan pada Dosen1 jumlah SKS yang diambil adalah 9. Artinya Dosen1 mendapat pelanggaran sebanyak 3 berdasarkan selisih SKS yang diambil dan batas minimum SKS. Pelanggaran yang sama juga terjadi pada Dosen4, dan Dosen5, sehingga jumlah *cost* untuk SKS ini adalah 5.

Nilai *cost* ketiga adalah *cost* terhadap jumlah mata kuliah yang diambil. Disini kita mencari pelanggaran terhadap batas minimum dan maksimum terhadap mata kuliah yang boleh diambil lalu menjumlahkannya. Pada Tabel 4.4, dapat dilihat

bahwa seluruh dosen mengambil 3-4 mata kuliah, yang artinya tidak terdapat pelanggaran terhadap mata kuliah sehingga untuk *cost* mata kuliah disini nilainya adalah 0.

Nilai untuk ketiga *cost* diatas dapat dilihat pada Tabel 4.13.

Tabel 4. 4 Tabel total ketiga *cost*

P	*Cost* Keminatan	*Cost* SKS	*Cost* Kelas	Total *Cost*
P1	1443	6	1	1450
P2	1539	20	3	1562
P3	1727	15	3	1290
P4	1924	4	2	1930
P5	1626	14	2	1642

Setelah didapat total *cost*, maka dicarilah nilai *fitness* dengan rumus pada persamaan (4.2). Hasil dari nilai *fitness* dapat dilihat pada Tabel 4.14.

Tabel 4. 5 Tabel nilai *fitness*

P	*Cost* Keminatan	*Cost* SKS	*Cost* Kelas	Total *Cost*	Nilai *Fitness*
P1	1443	6	1	1450	98550
P2	1539	20	3	1562	98438
P3	1727	15	3	1290	98256
P4	1924	4	2	1930	98070
P5	1626	14	2	1642	98358

4.3.4 Crossover

Proses *crossover* diawali dengan memasukan nilai *crossover rate* (Cr). Jumlah *popsize* telah diketahui yaitu 5. Nantinya akan terbentuk jumlah anak (*offspring*) dari nilai popsize dikalikan dengan nilai Cr. Misal nilai Cr adalah 0.4, maka *offspring* hasil dari crossover adalah 2 *offspring*. Pemilihan induk dilakukan secara *random* Proses *crossover* induk 1 dan induk 2 ditunjukkan pada Gambar 4.15.

Tabel 4. 6 Tabel nilai Crossover

P	APS			JARKOM				...	PPOL		SISPAK			FITNESS
P1	4	5	3	5	7	7	2	....	7	3	4	6	4	98069
P2	6	4	5	1	5	2	7	....	5	7	3	1	1	98456
O1	4	5	3	5	7	7	2	....	5	7	3	1	1	98443
O2	6	4	5	1	5	2	7	....	7	3	4	6	4	98537

Dari Hasil crossover menghasilkan 2 *offspring* yaitu *offpring1*, *offspring2*. Dengan menggunakan one-cut-point crossover pada gambar diatas titik potong persilangan antara Parent1 dan Parent2 antara gen ke-8 dan ke-8

4.3.5 Mutasi

Proses mutasi dilakukan dengan cara memilih satu induk secara random dari populasi kemudian mengubah nilai gen titik tersebut. Mutasi ini dilakukan dengan metode *reciprocal exchange mutation*, yaitu dengan memilih dua posisi gen secara acak kemudian nilai pada gen tersebut ditukarkan. Jumlah anak (*offspring*) terbentuk ditentukan oleh nilai popsize yang dikalikan dengan nilai

mutation rate (Mr). Misal nilai Mr yaitu 0.2 maka hasil dari *offspring* adalah 1 offspring dari 5 *popsize*

Tabel 4. 16 Tabel nilai Crossover

P	APS			JARKOM				...	PPOL		SISPAK			FITNESS
P4	4	2	5	5	7	4	1	....	5	4	1	3	5	98070
O3	4	2	6	6	7	4	1	....	5	4	1	3	5	98454

Hasil proses mutasi tersebut menghasilkan 1 *offspring* yaitu *offspring3*. Pada offspring3, nilai gen berada pada gen ke-3 dengan nilai 5 sedangkan nilai gen berada pada gen ke-8 dengan nilai 6, nilai tersebut nantinya saling ditukarkan.

Setelah melakukan proses *crossover* dan mutasi, didapatkan kromosom *offspring* berjumlah 3 kromosom. Kemudian hitung nilai fitness masing masing kromosom. Hasil penggabungan kromosom *offspring* beserta nilai fitness-nya ditunjukkan pada tabel 4.17. Ketiga offspring akan dilakukan compete dari pupulasi dijadikan populasi di *local search* dengan dua tahap menghitung nilai kecepatan dan menghitung nilai posisi dan hanya diambil satu populasi offspring terbaik.

Tabel 4. 17 Tabel Offspring

P	APS			JARKOM				...	PPOL		SISPAK			FITNESS
O1	4	5	3	5	7	7	2	....	5	7	3	1	1	98443
O2	6	4	5	1	5	2	7	....	7	3	4	6	4	98537
O3	4	2	6	6	7	4	1	....	5	4	1	3	5	98454

Ketiga offspring akan dilakukan compete dari populasi menjadi partikel di *local search* dengan dua tahap menghitung nilai kecepatan dan menghitung nilai posisi dan hanya diambil satu partikel *offspring* terbaik. nilai *pbest* dan *gbest* didapat dari inisialisasi partikel pada setiap iterasi. Pada iterasi 0 ini, partikel O2 yang menjadi *pbest.*

4.3.6 Menghitung Nilai Kecepatan

Untuk mencari nilai kecepatan pada PSO, tiap kromosom pada masing-masing partikel akan dihitung dengan rumus kecepatan pada persamaan (2.7) dengan parameter-parameter pendukung seperti *w*, *c1*, *c2*, *r1*, dan *r2*. Disini, nilai *w* = 0.75, c1 = 1, *c2* = 1.5, *r1* = 0.625, *r2* = 0.375. Untuk hasil perhitungan kecepatan dapat dilihat pada Tabel 4.16.

Tabel 4. 18 Tabel nilai kecepatan

P	APS			JARKOM			…	SISPAK
O1	0.75	-0.375	0.75	-1.5	-0.75	-1.875	…	1.125
O2	0	0	0	0	0	0	…	0
O3	0.75	0.75	-0.375	-1.875	-0.75	-0.75	…	-0.375

4.3.7 Menghitung Nilai Posisi

Setelah menentukan nilai kecepatan, selanjutnya kita menentukan nilai posisi. Posisi ini merupakan posisi baru dari posisi sebelumnya yang nantinya akan dihitung nilai *fitness*-nya untuk mendapatkan posisi baru yang lebih baik lagi. Rumus untuk mencari posisi partikel terdapat pada persamaan (2.6). Untuk hasil dari perhitungan posisi dapat dilihat pada Tabel 4.17.

Tabel 4.18 Tabel nilai posisi partikel

P	APS			JARKOM				…	PPOL		SISPAK		
O1	5	5	4	4	6	5	4	….	6	6	3	3	2
O2	6	4	5	1	5	2	7	….	7	3	4	6	4
O3	5	3	6	4	6	3	5	….	6	4	2	4	5

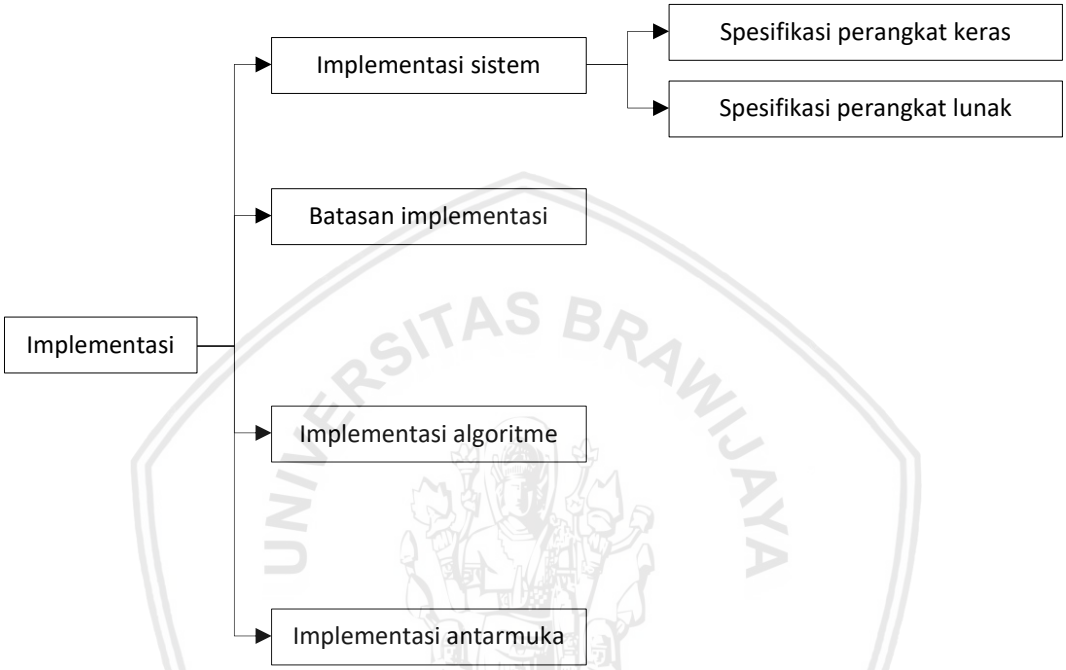
Nilai partikel posisi diatas adalah hasil setelah dilakukan pembulatan nilai supaya sama dengan data yang digunakan.

4.3.8 Evaluasi Nilai *Fitness* Pada Posisi Partikel Baru

Setelah didapat posisi partikel, maka yang dilakukan adalah mencari nilai *fitness* untuk posisi partikel ini. Rumus yang digunakan untuk mencari nilai *fitness* posisi partikel ini masih sama dengan rumus untuk mencari nilai *fitness* pada inisialisasi partikel, yaitu dengan menggunakan *cost* pelanggaran dengan melihat kromosom baru pada posisi partikel. Setelah didapat nilai *fitness*-nya maka akan dicari partikel *pbest* dan *gbest* yang baru. Apabila nilai *fitness* partikel *gbest* pada posisi baru lebih rendah dari nilai *fitness* partikel *gbest* pada inisialisasi partikel, maka partikel *gbest* tidak akan di-*update* dan tetap dipegang oleh partikel *pbest* pada inisialisasi partikel. Selanjutnya proses akan diulangi dengan kembali ke proses reproduksi untuk dikawinkan dengan parent2 sebelumnya dengan partikel gbest yang merupakn *offspring optimum.*

BAB 5 IMPLEMENTASI

Berdasarkan analisis kebutuhan dan perancangan yang telah dilakukan pada sebelumnya, bagian ini akan membahas tentang implementasi algoritme MK-NN untuk deteksi kerusakan mesin sepeda motor. Bahasan-bahasan dalam bab ini terdiri dari spesifikasi sistem, batasan implementasi, implementasi algoritme, dan implementasi antarmuka, yang ditunjukan pada Gambar 5.1.



Gambar 5.1 Bagan Implementasi

5.1 Implementasi sistem

Berdasarkan Gambar 5.1, implementasi sistem memerlukan dua aspek penting agar sistem dapat berjalan dengan baik sesuai fungsinya. Dua aspek tersebut adalah perangkat keras dan perangkat lunak yang digunakan untuk membangun sistem.

5.1.1 Spesifikasi Perangkat Keras

Tabel 5.1 menunjukan daftar spesifikasi perangkat keras yang digunakan untuk membangun sistem.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i5-3217U CPU @ 1,80GHz
Memori	4 GB
Kartu grafis	NVDIA GEFORCE 630M
Hardisk	500 GB

5.1.2 Spesifikasi Perangkat Lunak

Tabel 5.2 menunjukan daftar spesifikasi perangkat lunak yang digunakan untuk membangun sistem.

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem operasi	Windows 10
Bahasa pemrograman	Java
Tool pemrograman	Netbeans 8.0.1, JDK

5.2 Batasan Implementasi

Adapun batasan-batasan dalam implementasi algoritma *memetic* untuk memetakan tugas mengajar dosen yaitu:

1. Sistem yang dibangun menggunakan bahasa pemrograman java.
2. Data yang digunakan dalam sistem disimpan dalam localhost database.
3. Data yang digunakan berupa data dosen data mata kuliah.
4. Keluaran dari sistem adalah rekomendasi pemetaan tugas mengajar setiap mata kuliah pada setiap dosen.
5. Metode yang digunakan adalah algoritma *memetic*.

5.3 Implementasi Algoritma

Implementasi algoritma *memetic* untuk memetakan tugas mengajar dosen terdiri dari inisialisasi, pembangkitan matrik biaya dan hitung nilai fitness.

5.3.1 Implementasi Database Connection

Sebelum melakukan perhitungan algoritma memetic, karena data disimpan dalam database, maka harus dilakukan connect database terlebih dahulu seperti yang ditunjukan pada Source Code 5.1.

```
1  String db = "jdbc:mysql://localhost/penugasan2";
2     String user = "root";
3     String pw = "";
4
5     Connection conn;
6     Statement st;
7     public ResultSet rs;
      NumberFormat satuAngka = new DecimalFormat("#.0"), duaAngka = new
8  DecimalFormat("#.00");
9     Hitungan() {
10       popsize = 10;
11       cr = 0.2;
12       mr = 0.1;
```

```
13        jmlMatkulDipilih = 8;
14        matriksMayorMinor = new int[][]{{1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}};
15
16        try {
17           Class.forName("com.mysql.jdbc.Driver");
18           conn = DriverManager.getConnection(db, user, pw);
19           st = conn.createStatement();
20           rs = st.executeQuery("SELECT * from matakuliah");
21           rs.last();
22           int size = rs.getRow();
23           rs.beforeFirst();
24           int no = 0;
25           mataKuliah = new Object[size][6];
26           while (rs.next()) {
27              mataKuliah[no][0] = rs.getString(1);//nomer matkul
28              mataKuliah[no][1] = rs.getString(2);//nama
29              mataKuliah[no][2] = rs.getString(3);//kelas
30              mataKuliah[no][3] = rs.getString(4);//sks
31              mataKuliah[no][4] = rs.getString(5);//sks
32              no++;
33           }
34           rs = st.executeQuery("SELECT * from datadosen");
35           rs.last();
36           size = rs.getRow();
37           rs.beforeFirst();
38           no = 0;
39           dataDosen = new Object[size][9];
40           while (rs.next()) {
41              dataDosen[no][0] = rs.getString(1);//nomer dataDosen
42              dataDosen[no][1] = rs.getString(2);//nama
43              dataDosen[no][2] = rs.getString(8);//mayor
44              dataDosen[no][3] = rs.getString(9);//minor
45              dataDosen[no][4] = rs.getString(3);//urutan 1
46              dataDosen[no][5] = rs.getString(4);//2
47              dataDosen[no][6] = rs.getString(5);//3
48              dataDosen[no][7] = rs.getString(6);//4
49              dataDosen[no][8] = rs.getString(7);//5
50              no++;
51           }
52           rs = st.executeQuery("SELECT * from matriksbiaya");
53           rs.last();
54           size = rs.getRow();
55           rs.beforeFirst();
56           no = 0;
```

```
57          matriksBiaya = new int[size][1 + dataDosen.length];
58          while (rs.next()) {
59             for (int i = 0; i < dataDosen.length; i++) {
60
61                matriksBiaya[no][i] = rs.getInt(i + 1);//nomer matkul
62             }
63             no++;
64          }
            rs = st.executeQuery("SELECT * FROM matakuliah MK INNER JOIN
65  matriksbiaya MB ON MK.NO=MB.NO;");
66          rs.last();
67          size = rs.getRow();
68          rs.beforeFirst();
69          no = 0;
70          dataJoin = new Object[size][4 + dataDosen.length];
71          while (rs.next()) {
72             dataJoin[no][0] = rs.getString(1);
73             dataJoin[no][1] = rs.getString(2);
74             dataJoin[no][2] = rs.getString(3);
75             dataJoin[no][3] = rs.getString(4);
76             for (int i = 0; i < dataDosen.length; i++) {
77                dataJoin[no][i + 4] = rs.getString(i + 6);//nomer matkul
78             }
79             no++;
80          }
81          conn.close();
82       } catch (ClassNotFoundException ex) {
83          Logger.getLogger(Hitungan.class.getName()).log(Level.SEVERE, null, ex);
84       } catch (SQLException ex) {
85          Logger.getLogger(Hitungan.class.getName()).log(Level.SEVERE, null, ex);
86       }
87       int[] randomMatkul;//= new int[]{0, 7, 25, 3, 4, 13, 17, 6, 24};// =
88       randomMatkul = new int[getMataKuliah().length];
89       for (int i = 0; i < randomMatkul.length; i++) {
90          randomMatkul[i] = i;
91       }
92       setRandomMatkul(randomMatkul);
93    }
```

Source Code 5.1 Database Connection

Pada penelitian ini, data disimpan dalam database localhost, sehingga untuk mengambil data yang akan diproses maka harus melakukan connect agar system dapat mengakses database, sedangkan untuk jumlah populasi, iterasi, crossoverrate, mutationrate dan jumlah mata kuliah dapat ditentukan oleh pengguna.

5.3.2 Implementasi Inisialisasi Variable dan Representasi Kromosom

Tahapan pertama sebelum masuk ke proses perhitungan algoritma memetic adalah melakukan inisialisasi variable variable yang dibutuhkan untuk melakukan perhitungan algoritma memetic, kemudian dari inisialisasi variable itu sendiri nanti akan langsung berlanjut ke tahapan pertama yaitu representasi kromosom, inisialisasi variable dan representasi kromosom untuk perhitungan algoritma memetic ditunjukkan oleh source code 5.2.

```
1   int getTotalKelas(int rnd[]) {
2     int totalSKS = 0;
3     for (int i : rnd) {
4       totalSKS += Integer.valueOf(mataKuliah[i][2].toString());
5     }
6     return totalSKS;
7   }
8
9   Object[] generateRandomChromosome(int width) {
10    Object chromosome[] = new Object[width];
11    //perulangan sebanyak jumalh kelas, sekitar 165 an
12    for (int i = 0; i < width; i++) {
13      //hasilkan nilai random
14      int rnd = ThreadLocalRandom.current().nextInt(0, dosenDipilih.length);
15      //masukan nilai random ke array kromosom
16      chromosome[i] = dosenDipilih[rnd];
17    }
18    return chromosome;
19  }
20
21  Object[][] generateRandParent(int width) {
22    Object chromosome[];
23    int jumlahKelas;
24    //hitung jumlah semua kelas
25    jumlahKelas = getTotalKelas(randomMatkul);
26    //deklarasi ukuran parent
      parent = new Object[popsize][jumlahKelas + 2];//set ukuran parent sesuai
popsize dengan panjang kromosom sesuai jmlMatkulDipilih, +2 --> nyimpen
27  nama parent dan fitnes
28    for (int i = 0; i < popsize; i++) {
29      //generate random kromosom
```

```
        chromosome = generateRandomChromosome(jumlahKelas);//hasilkan
30  cormosom random
31          //penamaan individu parent
            parent[i][0] = "P" + (i + 1);//simpan pada array parent pada indeks 0
32  simpan nama
33          //masukan hasil generate random kromosom ke array paretn
            System.arraycopy(chromosome, 0, parent[i], 1, jumlahKelas);//pada
34  array ke 1 hingga jmlSKStotal simpan cromosom
35          //hitung fitnes parent tersebut
            parent[i][jumlahKelas + 1] = fitness(parent[i]);//costMinat(parent[i]) +
    costMataKuliah(parent[i]) + costSKS(parent[i]) +
36  costKompetensi(parent[i]);//kolom fitnes
37      }
38
39      return parent;
40  }
41
42  Double costMataKuliah(Object[] crom) {
        double penalty = 0, count[] = new double[dosenDipilih.length], plus[] = new
43  double[dosenDipilih.length];
44  //      vardump(crom);
45      int in = 1;
46      //perhitungan cost matakuliah
47      for (int i = 0; i < randomMatkul.length; i++) {
48          for (int j = 0; j < plus.length; j++) {
49              plus[j] = 1;
50          }
51  //          System.out.println("sd" + mataKuliah[randomMatkul[i]][4]);
52          for (int j = 0; j < keInt(mataKuliah[randomMatkul[i]][2]); j++) {
53              count[keInt(crom[in]) - 1] += plus[keInt(crom[in]) - 1];
54              plus[keInt(crom[in]) - 1] = 0;
55              in++;
56          }
57      }
58      //perhitungan pinalti sesuai batas
59      for (int i = 0; i < count.length; i++) {
60          System.out.println("dosen " + (i + 1) + " -> " + count[i]);
61          if (count[i] < batasMatkul[0]) {
62              penalty += (batasMatkul[0] - count[i]);
63          } else if (count[i] > batasMatkul[1]) {
64              penalty += (count[i] - batasMatkul[1]);
65          }
66      }
67      return penalty;
68  }
```

```
69
70    Double costSKS(Object[] crom) {
71        double penalty = 0, count[] = new double[dosenDipilih.length];
72  //        vardump(crom);
73        int in = 1;
74        //pehitugan  cost sks
75        for (int i = 0; i < randomMatkul.length; i++) {
76            for (int j = 0; j < keInt(mataKuliah[randomMatkul[i]][2]); j++) {
77                count[keInt(crom[in]) - 1] += keInt(mataKuliah[(randomMatkul[i])][3]);
78                in++;
79            }
80        }
81        //perhitungan pinalti sesuai batas
82        for (int i = 0; i < count.length; i++) {
83            System.out.println("dosen " + (i + 1) + " " + count[i]);
84            if (count[i] < batasSKS[0]) {
85                penalty += (batasSKS[0] - count[i]);
86            } else if (count[i] > batasSKS[1]) {
87                penalty += (count[i] - batasSKS[1]);
88            }
89        }
90        return penalty;
91    }
92
93    Double costKompetensi(Object[] crom) {
94        double hasil = 0, tempo = 0;
95        int in = 1;
96        System.out.println("");
97  //        vardump(crom);
98        for (int i = 0; i < randomMatkul.length; i++) {
99  //            System.out.print(hasil + " ");
100           for (int j = 0; j < keInt(mataKuliah[randomMatkul[i]][2]); j++) {
                  if (("," + dataDosen[keInt(crom[in]) - 1][2] + ",").contains("," +
101 (randomMatkul[i] + 1) + ","))//mayor
102               {
103                   tempo = getCostMinat(randomMatkul[i], keInt(crom[in]), 0);
                  } else if (("," + dataDosen[keInt(crom[in]) - 1][3] + ",").contains("," +
104 (randomMatkul[i] + 1) + ","))//minor
105               {
106                   tempo = getCostMinat(randomMatkul[i], keInt(crom[in]), 1);
107               } else//gak masuk
108               {
109                   tempo = 100;
110               }
111
```

```
            System.out.println(tempo + " " + (randomMatkul[i]+1) + "  " +
keInt(crom[in]) + "  " + dataDosen[keInt(crom[in]) - 1][2] + "  " +
dataDosen[keInt(crom[in]) - 1][3]);
            hasil += tempo;
            in++;
         }
      }
//       System.out.println("");
//       System.out.println(hasil);
      return hasil;
   }

   int getCostMinat(int MK, int crom, int MayMin) {
      int hasil = 0;
      for (int i = 4; i < 9; i++) {
         if (MK + 1 == keInt(dataDosen[crom - 1][i])) {
            break;
         }
         hasil++;
      }
      hasil = hasil >= 5 ? 100 : matriksMayorMinor[hasil][MayMin];
      return hasil;
   }

   public int[][] getMatriksBiaya() {
//       System.out.println("*****");
      matriksBiaya = new int[(randomMatkul.length)][dataDosen.length + 1];
      for (int i = 0; i < matriksBiaya.length; i++) {
         matriksBiaya[i][0] = randomMatkul[i];
         for (int j = 0; j < matriksBiaya[0].length - 1; j++) {
            if ((","  + dataDosen[j][2] + ",").contains("," + (randomMatkul[i] + 1) +
","))//mayor
            {
               matriksBiaya[i][j + 1] = getCostMinat((randomMatkul[i] + 1), j + 1, 0);
            } else if ((","  + dataDosen[j][3] + ",").contains("," + (randomMatkul[i] +
1) + ","))//minor
            {
               matriksBiaya[i][j + 1] = getCostMinat((randomMatkul[i] + 1), j + 1, 1);
            } else//gak masuk
            {
               matriksBiaya[i][j + 1] = 100;
            }
         }
         vardump(matriksBiaya[i]);
      }
```

```
153         return matriksBiaya;
154     }
155 
156     public Object[][] getMatriksBiayaObj() {
157 //      System.out.println("*****");
            Object matriksBiayaObj[][] = new
158 Object[(randomMatkul.length)][dataDosen.length + 1];
159         for (int i = 0; i < matriksBiayaObj.length; i++) {
160             matriksBiayaObj[i][0] = randomMatkul[i];
161             for (int j = 0; j < matriksBiayaObj[0].length - 1; j++) {
                    if ((",," + dataDosen[j][2] + ",").contains(",," + (randomMatkul[i] + 1) +
162 ",,"))//mayor
163                 {
                        matriksBiayaObj[i][j + 1] = getCostMinat((randomMatkul[i] + 1), j + 1,
164 0);
                    } else if ((",," + dataDosen[j][3] + ",").contains(",," + (randomMatkul[i] +
165 1) + ",,"))//minor
166                 {
                        matriksBiayaObj[i][j + 1] = getCostMinat((randomMatkul[i] + 1), j + 1,
167 1);
168                 } else//gak masuk
169                 {
170                     matriksBiayaObj[i][j + 1] = 100;
171                 }
172             }
173 //          vardump(matriksBiaya[i]);
174         }
175         return matriksBiayaObj;
176     }
```

Source Code 5.2 Inisialisasi Variabel dan Representasi Kromosom

Inisialisasi variable yang dimaksudkan disini adalah inisialisasi variable untuk mendapatkan nilai cost mata kuliah, cost SKS, cost kompetensi dan representasi chromosom, sedangkan untuk jumlah populasi, iterasi, crossoverrate, mutationrate dan jumlah mata kuliah telah ditentukan oleh pengguna.

5.3.3 Implementasi Hitung Fitness Parent

Setelah kromosom parent dari proses inisialisasi sebelumnya, selanjutnya akan dihitung nilai fitness seperti yang ditunjukkan oleh source code 5.3.

```
1  double fitness(Object chromos[]) {
2      double fitness;
3      //rumus hitungan fitnes
       fitness = C - (costMataKuliah(chromos) + costSKS(chromos) +
4  costKompetensi(chromos));
5      return fitness;
6    }
```

Source Code 5.3 Hitung Fitness Parent

Setelah sebelumnya didapatkan nilai cost mata kuliah, cost SKS dan cost kompetensi pada setiap gen dari setiap kromosom parent kemudian dihitung fitness dari setiap kromosom dengan cara 100000 dikurangi dengan total keseluruhan cost.

5.3.4 Implementasi Reproduksi

Setelah didapatkan hasil berupa representasi kromosom dan fitnesnya, selanjutnya adalah masuk ke proses reproduksi yang ditunjukan pada Source Code 5.4.

```
1  Object[][] crossover(Object[][] parent) {
       int offspring = (int) Math.round(cr * popsize), widthP = parent[0].length,
2  rand = 0;
3      randParent = shuffledArray(0, popsize - 1);
4      int jumlahSKS;
       jumlahSKS = getTotalKelas(randomMatkul);//costMinat(chromosome) +
   costMataKuliah(chromosome) + costSKS(chromosome) +
5  costKompetensi(chromosome);
6      for (int i = 0; i < offspring; i = i + 2) {
7        int cutPoint = randomRange2(1, jumlahSKS - 2);
8  //       System.out.println(cutPoint);
9        child[i][0] = /*parent[randParent[i]][0] +*/ "C" + (i + 1);
10 //       child[i][0] = parent[randParent[i]][0] + " C" + (i + 1);
11       System.arraycopy(parent[randParent[i]], 1, child[i], 1, cutPoint);
         System.arraycopy(parent[randParent[i + 1]], cutPoint + 1, child[i], cutPoint
12 + 1, widthP - cutPoint - 1);
13
         child[i][jumlahSKS + 1] = fitness(child[i]);//costMinat(child[i]) +
14 costMataKuliah(child[i]) + costSKS(child[i]) + costKompetensi(child[i]);
15       //child 2
16       child[i + 1][0] = /*parent[randParent[i + 1]][0] +*/ "C" + (i + 2);
17 //       child[i + 1][0] = parent[randParent[i + 1]][0] + " C" + (i + 2);
18       System.arraycopy(parent[randParent[i + 1]], 1, child[i + 1], 1, cutPoint);
         System.arraycopy(parent[randParent[i]], cutPoint + 1, child[i + 1], cutPoint
19 + 1, widthP - cutPoint - 1);
20
         child[i + 1][jumlahSKS + 1] = fitness(child[i + 1]);//costMinat(child[i + 1]) +
21 costMataKuliah(child[i + 1]) + costSKS(child[i + 1]) + costKompetensi(child[i + 1]);
```

```
22
23          }
24
25          return child;
26      }
27
28      Object[][] mutasi(Object[][] parent) {
            int jml = getTotalKelas(randomMatkul), offspring = (int) Math.round(mr *
        popsize), offspringCR = (int) Math.round(cr * popsize), point[];//set ukuran
29      offspring sepanjang mr kali popsize
            for (int i = offspringCR; i < offspring + offspringCR; i++) {//perulngan dimulai
30      dari banyak offsping sebelumnya pada crosover
31              point = shuffledArray(1, jml);//hasilkan point random unutk mutasi
32              System.arraycopy(parent[randParent[i]], 1, child[i], 1, jml);
33              child[i][0] = /*parent[randParent[i]][0] + */ "C" + (i + 1);//nama
34              child[i][0] = /*parent[randParent[i]][0] + */ "C" + (i + 1);//nama
                child[i][point[0] + 1] = parent[randParent[i]][point[1] + 1];//swap point
35      mutasi
                child[i][point[1] + 1] = parent[randParent[i]][point[0] + 1];//swap point
36      mutasi
                child[i][jml + 1] = fitness(child[i]);//child[i][jmlMatkulDipilih + 1]--> indeks
37      terakhir diisi fitnes
38          }
39
40          return child;
41      }
```

Source Code 5.4 Reproduksi

Pada proses ini terdapat dua tahapan untuk mendapatkan child yaitu dengan cara melakukan crossover dan mutase, crossover rate dan mutation rate digunakan untuk menentukan jumlah child yang akan dihasilkan pada proses reproduksi, dimana nilai crossoverrate dikalikan dengan jumlah populasi awal untuk mendapatkan jumlah child yang akan dihasilkan pada proses cross over dan nilai mutationrate dikalikan dengan jumlah populasi awal untuk mendapatkan jumlah child yang akan dihasilkan pada proses mutase.

Jenis crossover yang digunakan pada penelitian ini adalah one point cut cross over, sedangkan jenis mutasi yang digunakan adalah insertion mutation.

5.3.5 Implementasi Local Search dan Hitung Fitness

Setelah kromosom child dari proses reproduksi sebelumnya, selanjutnya akan di proses local search dan dihitung nilai fitness seperti yang ditunjukkan oleh source code 5.5.

```
1   Object[][] PSO(Object[][] child, double c1, double c2, int iterMax, double W) {
2       Object temp[][] = new Object[child.length][child[0].length],
3               Vid[][] = new Object[child.length][child[0].length],
4               Xid[][] = new Object[child.length][child[0].length],
5               pbest[] = new Object[child[0].length],
6               gbest[] = new Object[child[0].length];
7       int max = 0, jumlahSKS = getTotalKelas(randomMatkul), tmp;
8       double r1 = Math.random(), r2 = Math.random();
9       //iterasi awal ke-0
10      //cek pbest dan gbest
11      for (Object[] child1 : child) {   //****PROSES 1*****
12          if (max < keInt(child1[jumlahSKS + 1])) {
13              System.arraycopy(child1, 0, gbest, 0, child1.length);
14              System.arraycopy(child1, 0, pbest, 0, child1.length);
15              max = keInt(child1[jumlahSKS + 1]);
16          }
17      }
18
19      for (int i = 0; i < child.length; i++){   //****PROSES 2******
20          System.arraycopy(child[i], 0, temp[i], 0, child[i].length);
21          Vid[i][0] = child[i][0];
22          for (int j = 1; j < child[0].length - 1; j++) {
23              Vid[i][j] = (W * 0)
24                      + (c1 * r1 * (keDbl(pbest[j]) - keDbl(child[i][j])))
25                      + (c2 * r2 * (keDbl(pbest[j]) - keDbl(child[i][j])));
26          }
27      }
28
29  //      System.out.println("vid");
30  //      vardump(Vid);
31      for (int i = 0; i < child.length; i++) {
32          Xid[i][0] = child[i][0];
33          //pembentukan kromosom baru
34          for (int j = 1; j < child[0].length - 1; j++) {
35              //penjumlahan PROSES 1 dan 2
36              Xid[i][j] = (int) (keDbl(temp[i][j]) + keDbl(Vid[i][j]));
37              tmp = keInt(Xid[i][j]);
38              //pembulatan dan handling jika lebih dari 20 atau kurang dari 0
39              Xid[i][j]
40                      = tmp < 0 ? Math.abs(tmp) > 20 ? Math.abs(tmp) % 20
41                              : Math.abs(tmp) % 20 == 0 ? 1
42                                      : Math.abs(tmp)
43                      : tmp == 0 ? 1
44                              : tmp > 20 ? tmp % 20 != 0
```

```
45                                      ? tmp % 20 : 1
46                                  : tmp;
47          }
48  //          vardump(Xid[i]);
49          //hitung nilai fitnes dari hasil pembentukan kromosom baru
50          Xid[i][jumlahSKS + 1] = fitness(Xid[i]);
51      }
52      //iterasi 0 selesai
53      //iterasi PSO ke 1 sampai ke n
54      for (int m = 0; m < iterMax; m++) {
55          //cek pbest
56          max = 0;
57          for (Object[] Xid1 : Xid) {
58              if (max < keInt(Xid1[jumlahSKS + 1])) {
59                  System.arraycopy(Xid1, 0, pbest, 0, Xid1.length);
60                  max = keInt(Xid1[jumlahSKS + 1]);
61              }
62          }
63          //cek gbest
64          if (keInt(pbest[jumlahSKS + 1]) > keInt(gbest[jumlahSKS + 1])) {
65              System.arraycopy(pbest, 0, gbest, 0, gbest.length);
66          }
67
68          for (int i = 0; i < child.length; i++) {
69              System.arraycopy(Xid[i], 0, temp[i], 0, child[i].length);
70              Vid[i][0] = Xid[i][0];
71              for (int j = 1; j < child[0].length - 1; j++) {
72                  Vid[i][j] = (W * keDbl(Vid[i][j]))
73                          + (c1 * r1 * (keDbl(pbest[j]) - keDbl(Xid[i][j])))
74                          + (c2 * r2 * (keDbl(pbest[j]) - keDbl(Xid[i][j])));
75              }
76          }
77
78          for (int i = 0; i < child.length; i++) {
79              for (int j = 1; j < child[0].length - 1; j++) {
                    Xid[i][j] = Math.abs(((int) (keDbl(temp[i][j]) + keDbl(Vid[i][j]))) % 20)
80  + 1;
81                  tmp = keInt(Xid[i][j]);
82                  Xid[i][j]
83                          = tmp < 0 ? Math.abs(tmp) > 20 ? Math.abs(tmp) % 20
84                                  : Math.abs(tmp) % 20 == 0 ? 1
85                                          : Math.abs(tmp)
86                                  : tmp == 0 ? 1
87                                          : tmp > 20 ? tmp % 20 != 0
```

```
88                                      ? tmp % 20 : 1
89                                    : tmp;
90              }
91              Xid[i][jumlahSKS + 1] = fitness(Xid[i]);
92            }
93        }
94
95        return Xid;
96
97    }
```

Source Code 5.5 Implementasi Local Search dan Hitung Fitness

Setelah sebelumnya didapatkan nilai cost mata kuliah, cost SKS dan cost kompetensi pada setiap gen dari setiap kromosom child kemudian dilakukan perbaikan kromosom dan dihitung fitness dari setiap kromosom dengan cara 100000 dikurangi dengan total keseluruhan cost.

5.3.6 Proses Seleksi

Langkah terakhir perhitungan algoritma memetic adalah melakukan seleksi untuk mendapatkan individu terbaik. Implementasi proses seleksi ditunjukan pada Source Code 5.6.

```
1   Object[][] evaluasiSeleksi(Object[][] parent, Object[][] child) {
        int offspring = (int) Math.round(mr * popsize) + (int) Math.round(cr *
2   popsize);
        Object newParent[][], gabungan[][] = new Object[popsize +
3   offspring][parent[0].length];
4       //gabung parent+child lalu disimpan pada array gabungan
        System.arraycopy(parent, 0, gabungan, 0, popsize);//array gabungan pada
5   indeks 0 sampai popsize diisi parent
        System.arraycopy(child, 0, gabungan, popsize, offspring);//array gabungan
6   pada indeks popsize sampai offspirng diisi child
7       //seleksi
        newParent = insertionSort(gabungan);//sort fitnsess menggunakan
8   algoritma insertionsort
        for (int i = 0; i < popsize; i++) {//set nama baru untuk parent yg telah
9   diurutkan
10          newParent[i][0] = "P" + (i + 1);
11      }
        System.arraycopy((insertionSort(gabungan)), 0, newParent, 0,
    popsize);//10 fitness tertinggi dijadikan parent pada generasi selanjutnya jika
12  popsize=10
13      return newParent;
14    }
15  Object[][] insertionSort(Object array[][]) {
        int n = array.length, m = array[0].length, jumlahSKS =
16  getTotalKelas(randomMatkul);
```

```
17      Object key[] = new Object[m];
18      for (int j = 1; j < n; j++) {
19         System.arraycopy(array[j], 0, key, 0, m);
20         int i = j - 1;
           while ((i > -1) && (Double.valueOf(array[i][jumlahSKS + 1].toString()) <
21  Double.valueOf(key[jumlahSKS + 1].toString()))) {
22            System.arraycopy(array[i], 0, array[i + 1], 0, m);
23            i--;
24         }
25         System.arraycopy(key, 0, array[i + 1], 0, m);
26      }
27      return array;
28    }
```

Source Code 5.6 Proses Seleksi

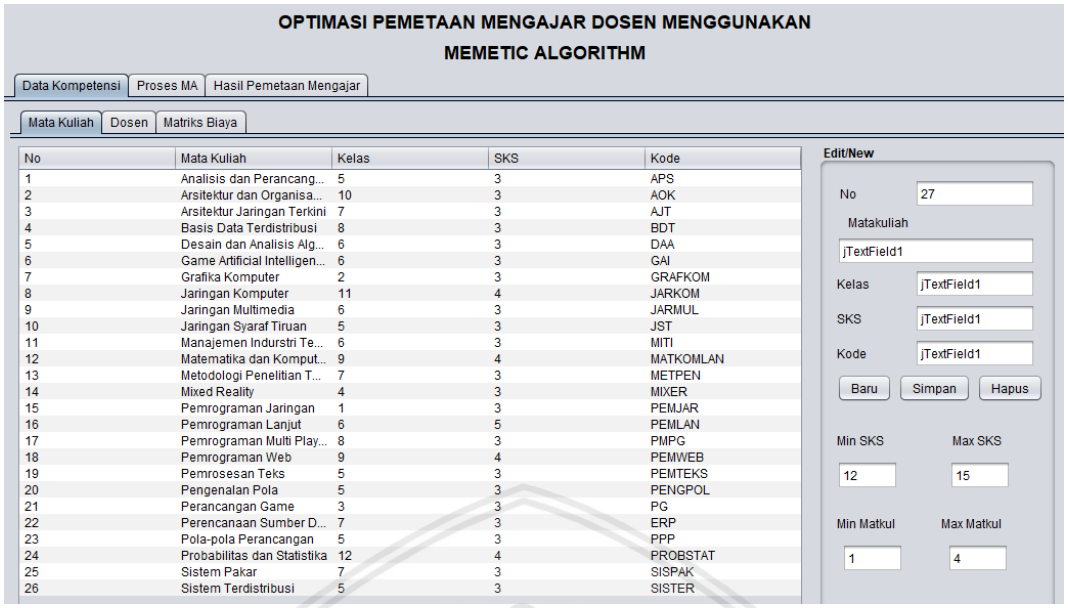
Tahapan terakhir adalah melakukan seleksi dengan cara mencari fitness terbaik dari populasi awal dan child yang telah dihasilkan. Pada penelitian ini seleksi yang digunakan adalah seleksi elitism yaitu mengurutkan fitness dari yang tertinggi hingga terendah kemudian mengambil kromosom sebanyak populasi awal.

5.4 Implementasi Antarmuka

Antarmuka bertujuan untuk memudahkan interaksi pengguna dengan sistem, Implementasi antarmuka terdiri dari tampilan antarmuka awal, data uji, proses MA.

5.4.1 Implementasi Antarmuka Data Kompetensi

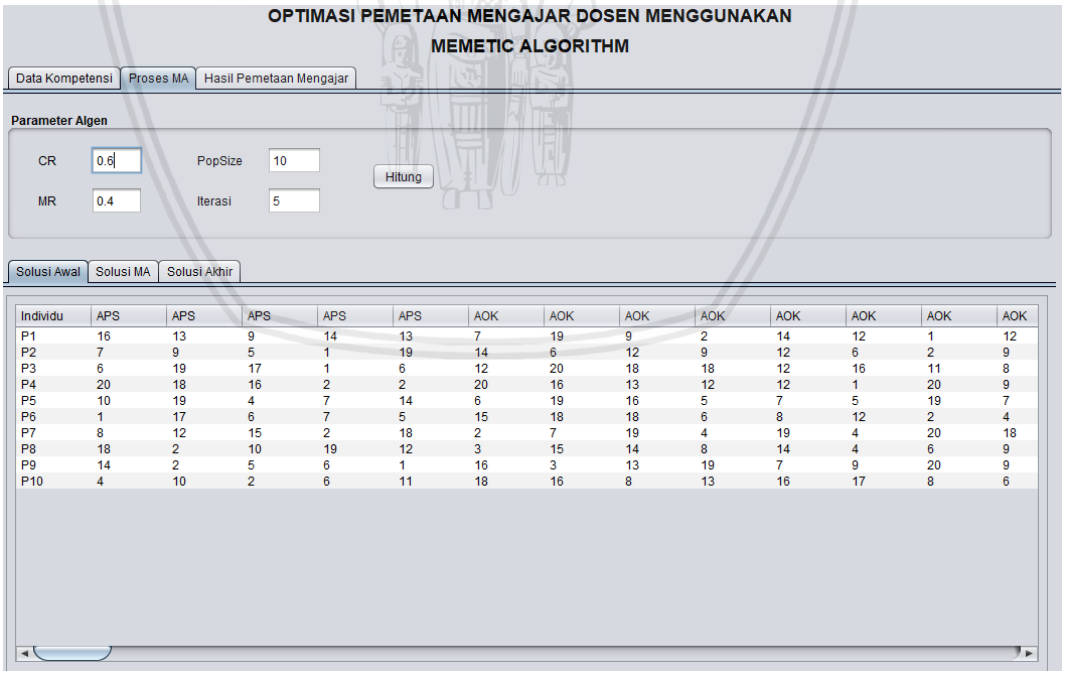
Tampilan implementasi antarmuka data kompetensi merupakan tampilan pertama saat pengguna menjalankan aplikasi. Pada halaman antarmuka data kompetensi terdapat list data berupa daftar mata kuliah, daftar dosen dan daftar matriks biaya. Implementasi antarmuka data kompetensi ditunjukan pada Gambar 5.2.



Gambar 5.2 Antarmuka Data Kompetensi

5.4.2 Implementasi Antarmuka Proses Algoritma Memetic

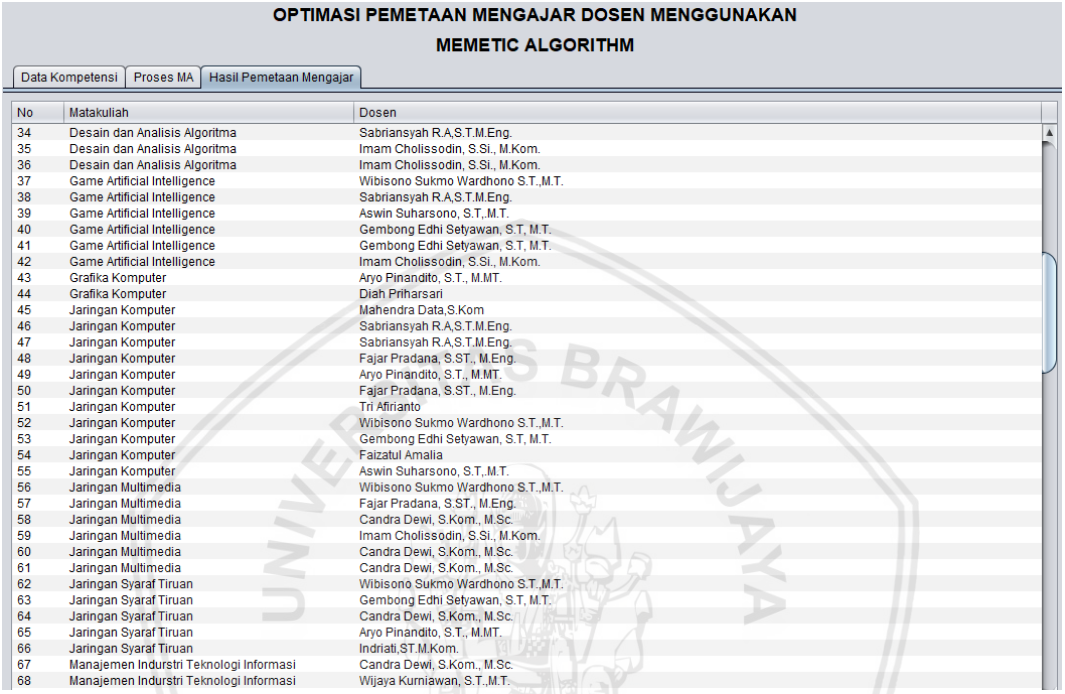
Menu proses algoritma memetic yang berfungsi untuk memasukkan parameter algoritma dan memproses perhitungan algoritma memetic berupa solusi awal, solusi algoritma memetic dan solusi akhir. Desain antarmuka proses algoritma memetic pada Gambar 5.3.



Gambar 5.3 Antarmuka proses memetic algorithm

5.4.3 Implementasi Antarmuka Hasil Pemetaan Mengajar

Halaman terakhir adalah implementasi antarmuka hasil pemetaan mengajar yang berisi daftar mata kuliah dan dosen yang mengajar. Tampilan implementasi antarmuka hasil pemetaan mengajar dapat dilihat pada Gambar 5.4.



No	Matakuliah	Dosen
34	Desain dan Analisis Algoritma	Sabriansyah R.A,S.T.M.Eng.
35	Desain dan Analisis Algoritma	Imam Cholissodin, S.Si., M.Kom.
36	Desain dan Analisis Algoritma	Imam Cholissodin, S.Si., M.Kom.
37	Game Artificial Intelligence	Wibisono Sukmo Wardhono S.T.,M.T.
38	Game Artificial Intelligence	Sabriansyah R.A,S.T.M.Eng.
39	Game Artificial Intelligence	Aswin Suharsono, S.T.,M.T.
40	Game Artificial Intelligence	Gembong Edhi Setyawan, S.T, M.T.
41	Game Artificial Intelligence	Gembong Edhi Setyawan, S.T, M.T.
42	Game Artificial Intelligence	Imam Cholissodin, S.Si., M.Kom.
43	Grafika Komputer	Aryo Pinandito, S.T., M.MT.
44	Grafika Komputer	Diah Priharsari
45	Jaringan Komputer	Mahendra Data,S.Kom
46	Jaringan Komputer	Sabriansyah R.A,S.T.M.Eng.
47	Jaringan Komputer	Sabriansyah R.A,S.T.M.Eng.
48	Jaringan Komputer	Fajar Pradana, S.ST., M.Eng.
49	Jaringan Komputer	Aryo Pinandito, S.T., M.MT.
50	Jaringan Komputer	Fajar Pradana, S.ST., M.Eng.
51	Jaringan Komputer	Tri Afirianto
52	Jaringan Komputer	Wibisono Sukmo Wardhono S.T.,M.T.
53	Jaringan Komputer	Gembong Edhi Setyawan, S.T, M.T.
54	Jaringan Komputer	Faizatul Amalia
55	Jaringan Komputer	Aswin Suharsono, S.T.,M.T.
56	Jaringan Multimedia	Wibisono Sukmo Wardhono S.T.,M.T.
57	Jaringan Multimedia	Fajar Pradana, S.ST., M.Eng.
58	Jaringan Multimedia	Candra Dewi, S.Kom., M.Sc.
59	Jaringan Multimedia	Imam Cholissodin, S.Si., M.Kom.
60	Jaringan Multimedia	Candra Dewi, S.Kom., M.Sc.
61	Jaringan Multimedia	Candra Dewi, S.Kom., M.Sc.
62	Jaringan Syaraf Tiruan	Wibisono Sukmo Wardhono S.T.,M.T.
63	Jaringan Syaraf Tiruan	Gembong Edhi Setyawan, S.T, M.T.
64	Jaringan Syaraf Tiruan	Candra Dewi, S.Kom., M.Sc.
65	Jaringan Syaraf Tiruan	Aryo Pinandito, S.T., M.MT.
66	Jaringan Syaraf Tiruan	Indriati,ST.M.Kom.
67	Manajemen Indurstri Teknologi Informasi	Candra Dewi, S.Kom., M.Sc.
68	Manajemen Indurstri Teknologi Informasi	Wijaya Kurniawan, S.T.,M.T.

Gambar 5.4 Antarmuka hasil output pemetaan mengajar

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab pengujian dan analisis ini menjelaskan tentang pengujian serta analisis terhadap hasil analisis terhadap hasil implementasi yang sudah dirancang pada bab sebelumnya. Pengujian ini dilakukan untuk mengetahui parameter algoritma memetic yang optimal dilihat dari nilai *fitness* terbaik dari setiap pengujian yang dilakukan. Pengujian pada bab ini meliputi pengujian parameter algoritma memetic yaitu pengujian ukuran populasi, pengujian iterasi serta pengujian crossover rate dan mutation rate.

6.1 Sistematika Pengujian

Pengujian dilakukan untuk mengetahui parameter algoritma memetic yang dianggap optimal dalam menghasilkan nilai *fitness* terbaik sehingga dapat ditemukan parameter optimal yang digunakan untuk menyelesaikan permasalahan pemetaan tugas mengajar dosen. Pengujian ini dilakukan dengan masukan sebagai berikut:

a. Data Mata Kuliah
b. Data Nama Dosen
c. Data matriks biaya

6.2 Analisis dan Pembahasan

Proses pengujian meliputi ukuran populasi, pengujian iterasi serta pengujian crossover rate dan mutation rate. Data yang digunakan pada pengujian ini adalah data mata kulaih, nama dosen, dan matriks biaya.

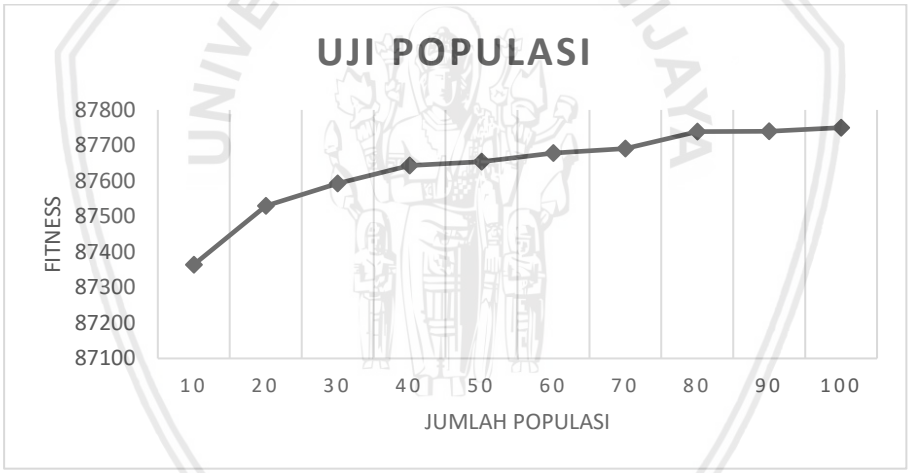
6.3 Pengujian dan Analisis Ukuran Populasi

Pada pengujian pertama dilakukan pengujian terhadap parameter ukuran populasi. Pengujian ukuran populasi dilakukan untuk mengetahui pengaruh jumlah populasi terhadap nilai *fitness* yang dihasilkan. Untuk menguji ukuran populasi yang digunakan ukuran populasi kelipatan 10, dimulai dari 10 sampai dengan 100. Untuk mendapatkan hasil yang lebih *valid* maka setiap percobaan akan dilakukan sebanyak 10 kali untuk diambil nilai rata-rata *fitness*nya. Pada percobaan ini digunakan iterasi sebanyak 50 kali, crossover rate sebesar 0,8 dan mutation rate 0,2. Tabel hasil uji coba ukuran populasi dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Uji Coba Ukuran populasi

pop/uji	1	2	3	4	5	6	7	8	9	10	Rata-Rata
10	87335	87157	87426	87397	87261	87461	87278	87491	87578	87257	87364,1
20	87380	87670	87224	87595	87616	87524	87641	87706	87442	87499	87529,7
30	87622	87694	87533	87854	87520	87559	87437	87718	87433	87562	87593,2
40	87364	87566	87623	87821	87511	87718	87633	87634	87709	87857	87643,6
50	87739	87704	87541	87551	87621	87757	87627	87564	87797	87636	87653,7
60	87536	87684	87750	87657	87631	87659	87732	87680	87763	87697	87678,9
70	87596	87718	87660	87752	87681	87744	87717	87661	87652	87728	87690,9
80	87527	87734	87717	87813	87552	87669	88119	87389	87920	87949	87738,9
90	87550	88116	87552	87530	87830	87830	87698	87610	87850	87836	87740,2
100	87727	87645	87690	87694	87741	87945	87740	87878	87744	87697	87750,1

Dari Tabel 6.1 dapat dibuat sebuah grafik untuk melihat dengan jelas hasil dari uji coba ukuran populasi terhadap nilai *fitness*. Dan grafik hasil uji coba ukuran populasi dapat dilihat pada Gambar 6.1.



Gambar 6.1 Hasil Uji Coba Ukuran populasi

Dari hasil percobaan diatas dapat dilihat bahwa dari awal percobaan hingga akhir percobaan nilai fitness yang dihasilkan cenderung stabil dari percobaan ketika jumlah populasi 10 hingga jumlah populasi 100 terdapat kecenderungan bahwa semakin besar jumlah populasi maka kemungkinan fitness yang dihasilkan akan lebih baik.Hasil terbaik didapatkan ketika jumlah populasi 100 yaitu dengan nilai 87750.

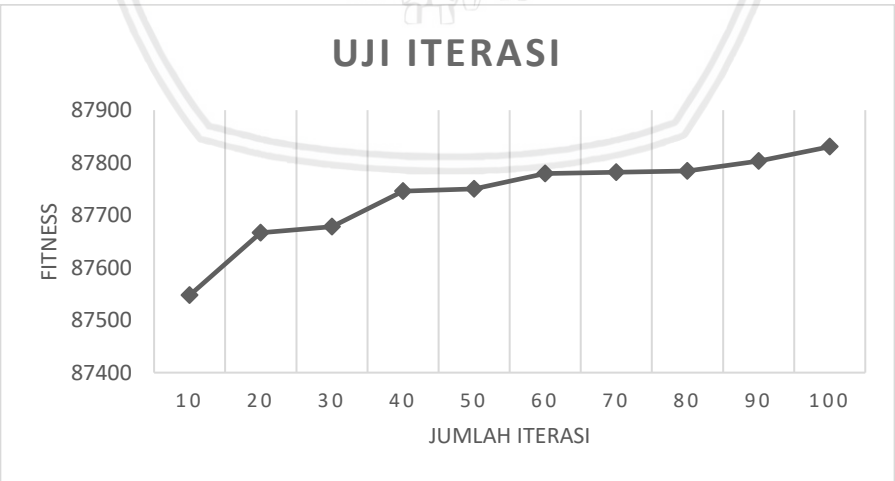
6.4 Pengujian dan Analisis Jumlah Iterasi

Pada pengujian kedua dilakukan pengujian terhadap parameter jumlah iterasi. Pengujian jumlah iterasi dilakukan untuk mengetahui pengaruh jumlah iterasi terhadap nilai *fitness* yang dihasilkan. Untuk menguji jumlah iterasi yang digunakan jumlah iterasi kelipatan 10, dimulai dari 10 sampai dengan 100. Untuk mendapatkan hasil yang lebih *valid* maka setiap percobaan akan dilakukan sebanyak 10 kali untuk diambil nilai rata-rata *fitness*nya. Pada percobaan ini digunakan populasi yang menghasilkan fitness terbaik dari pengujian sebelumnya yaitu 100, crossover rate sebesar 0,8 dan mutation rate 0,2. Tabel hasil uji coba jumlah iterasi dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Uji Jumlah Iterasi

iter/uji	1	2	3	4	5	6	7	8	9	10	Rata-Rata
10	87741	87560	87807	87204	87746	87338	87562	87416	87431	87669	87547,4
20	87674	87719	87519	87541	87494	87634	87902	87626	87815	87742	87666,6
30	87574	87774	87619	87580	87474	87634	87892	87636	87714	87882	87677,9
40	87743	87731	87731	87866	87739	87692	87796	87626	87738	87797	87745,9
50	87727	87645	87690	87694	87741	87945	87740	87878	87744	87697	87750,1
60	87831	87628	87924	87850	87812	87886	87629	87618	87790	87826	87779,4
70	87557	87644	87716	87782	87936	87647	87752	88270	87910	87600	87781,4
80	87864	87631	87885	87749	87727	87637	87587	88147	87807	87810	87784,4
90	87938	88098	87749	87939	87855	87632	87809	87624	87656	87733	87803,3
100	88123	87748	87753	87838	87755	87632	87901	87724	87696	88133	87830,3

Dari Tabel 6.2 dapat dibuat sebuah grafik untuk melihat dengan jelas hasil dari uji coba jumlah iterasi terhadap nilai *fitness*. Dan grafik hasil uji coba jumlah iterasi dapat dilihat pada Gambar 6.2.



Gambar 6.2 Hasil Uji Jumlah Iterasi

Dari hasil percobaan diatas dapat dilihat bahwa dari awal percobaan hingga akhir percobaan nilai fitness yang dihasilkan cenderung stabil dari percobaan ketika jumlah iterasi 10 hingga jumlah iterasi 100 terdapat kecenderungan bahwa semakin besar jumlah populasi maka kemungkinan fitness yang dihasilkan akan lebih baik.Hasil terbaik didapatkan ketika jumlah iterasi 100 yaitu dengan nilai 87830.

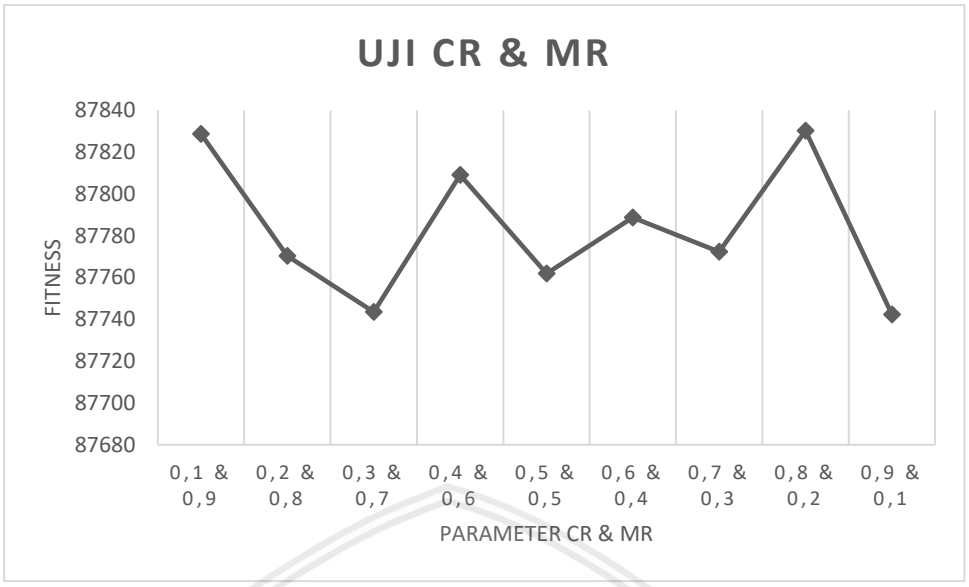
6.5 Pengujian dan Analisis Cr/Mr

Pada pengujian ketiga dilakukan pengujian terhadap parameter nilai Cr/Mr Pengujian nilai Cr/Mr dilakukan untuk mengetahui pengaruh nilai Cr/Mr terhadap nilai *fitness* yang dihasilkan. Untuk menguji nilai Cr/Mr yang digunakan nilai Cr/Mr kelipatan 0,1, dimulai dari 0,1 sampai dengan 0,9. Untuk mendapatkan hasil yang lebih *valid* maka setiap percobaan akan dilakukan sebanyak 10 kali untuk diambil nilai rata-rata *fitness*nya. Pada percobaan ini digunakan populasi yang menghasilkan fitness terbaik dari pengujian sebelumnya yaitu 100, iterasi yang menghasilkan fitness terbaik dari pengujian sebelumnya yaitu 100. Tabel hasil uji coba nilai Cr/Mr dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Uji Coba nilai Cr/Mr

crmr/uji		1	2	3	4	5	6	7	8	9	10	Rata-Rata
0,1	0,9	87906	88292	87676	87523	87820	87721	87861	87706	87760	88022	87828,7
0,2	0,8	87967	87662	87897	87658	87734	87807	87749	87649	87751	87829	87770,3
0,3	0,7	87655	87679	87846	87652	87865	87673	87673	87732	87774	87887	87743,6
0,4	0,6	87742	87788	87721	87806	87848	87653	87941	87890	87894	87808	87809,1
0,5	0,5	87642	87770	87812	87720	87843	87655	87562	87836	87844	87936	87762
0,6	0,4	87855	87636	88195	87763	87713	87628	87621	87890	87840	87746	87788,7
0,7	0,3	87728	87784	88008	87731	87773	87797	87990	87525	87784	87603	87772,3
0,8	0,2	88123	87748	87753	87838	87755	87632	87901	87724	87696	88133	87830,3
0,9	0,1	87845	87877	87650	87648	87832	87915	87888	87907	87435	87427	87742,4

Dari Tabel 6.2 dapat dibuat sebuah grafik untuk melihat dengan jelas hasil dari uji coba nilai Cr/Mr terhadap nilai *fitness*. Dan grafik hasil uji coba jumlah lamda dapat dilihat pada Gambar 6.2.



Gambar 6.3 Hasil Uji Coba nilai Cr/Mr

Dari hasil percobaan diatas dapat dilihat bahwa dari awal percobaan hingga akhir percobaan nilai fitness yang dihasilkan tidak stabil, dari awal hingga akhir nilai fitness yang dihasilkan selalu naik turun. Hasil terbaik didapatkan ketika Nilai Crossover rate = 0,1 dan mutation rate = 0,9 yaitu dengan nilai fitness 87829.

6.6 Analisis Hasil Pengujian

Algoritma algoritma memetic merupakan salah satu algoritma optimasi yang tugasnya adalah mencari solusi optimum dari suatu permasalahan. Pada penelitian ini, nilai *fitness* terbaik yang didapatkan yaitu 87830. Solusi ini didapatkan dengan nilai parameter ukuran populasi = 100, jumlah generasi = 100, dan Cr = 0,8 Mr 0,2.

Nilai *fitness* yang dihasilkan ini masih dapat bertambah baik dengan melihat bahwa konvergensi nilai belum terlihat sehingga penambahan jumlah maksimum parameter dapat dilakukan. Namun, penambahan jumlah maksimum parameter ini juga akan menyebabkan waktu perhitungan yang semakin lama dan alokasi memori yang semakin besar sehingga diperlukan spesifikasi mesin dengan memori RAM yang lebih besar dengan alokasi memori yang lebih baik. Banyaknya jumlah nilai parameter uji yang digunakan disini adalah pertimbangan penulis dengan melihat kemampuan perangkat yang digunakan.

Sedangkan untuk hasil *output* dari sistem, nilai *error* yang dihasilkan semakin berkurang seiring bertambahnya jumlah parameter yang digunakan. Dari nilai *fitness* yang dihasilkan, terdapat kesalahan penempatan dosen terhadap mata kuliah dengan rata-rata sebanyak 61 dari 165 mata kuliah. Hal ini dapat diperkecil dengan menambah nilai parameter. Bertambahnya nilai parameter seperti jumlah populasi, menyebabkan semakin luasnya ruang lingkup pencarian sehingga posisi optimum lebih cepat dicapai dan variasi nilai *fitness* yang didapat semakin banyak.

Hal ini menyebabkan berkurangnya terjadi konvergensi dini pada nilai *fitness* sehingga pencarian solusi dapat diteruskan. Bertambahnya jumlah iterasi menyebabkan semakin banyaknya proses *update* nilai *fitness* dilakukan karena pergerakan populasi terhadap suatu posisi lebih sering. Arah grafik nilai *fitness* yang dihasilkan pada bertambahnya jumlah iterasi cenderung lebih tinggi. Hal ini dipengaruhi pergerakan populasi ke arah posisi yang lebih optimum

Setelah melakukan beberapa pengujian terhadap parameter algoritma memetic, MA mampu menyelesaikan permasalahan optimasi pemetaan tugas mengajar dosen. Dengan solusi yang diberikan maka dosen dapat mendapatkan tugas mengajar sesuai porsi dan prioritas masing-masing.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Kesimpulan yang diperoleh melalui hasil uji coba yang telah dilakukan mengenai penerapan algoritma memetic untuk menyelesaikan permasalahan optimasi pemetaan tugas mengajar dosen yaitu sebagai berikut:

1. Hasil solusi dalam menerapkan Memetic Algorithm yang diberikan cukup baik dengan jumlah kesalahan yang terus menurun seiring dengan bertambahnya jumlah parameter populasi. Nilai fitness yang didapat semakin tinggi dan belum terjadinya konvergensi nilai.
2. Untuk mengukur kualitas solusi terbaik pada permasalahan optimasi pemetaan tugas mengajar dosen yaitu dengan melihat nilai *fitness* tertinggi. Pada pengujian didapatkan nilai *fitness* tertinggi sebesar 87830.
3. Pengujian parameter algoritme *memetic* memiliki pengaruh terhadap nilai *fitness* yang dihasilkan. Pengujian pada parameter algoritma memetic dilakukan dengan melakukan perubahan jumlah populasi dari 10 hingga 100 dan menghasilkan fitness sebesar 87750, pada jumlah populasi 100. Parameter algoritma memetic yang juga diuji terhadap penelitian ini yaitu generasi dan populasi. Kedua parameter tersebut berpengaruh terhadap nilai *fitness* yang diperoleh dari proses algoritma memetic. Ukuran populasi yang optimal terdapat pada ukuran 100 dengan nilai *fitness* sebesar 87850. Sehingga dapat disimpulkan bahwa nilai *fitness* yang dihasilkan oleh sistem dipengaruhi oleh ukuran populasi yang semakin tinggi, namun jika ukuran populasi semakin tinggi maka tidak didapatkan kenailkan *fitness* yang signifikan dan waktu komputasinya akan berjalan semakin lama pula. Pengujian jumlah generasi optimal dihasilkan *fitness* terbaik dengan nilai *fitness* 87830 dan terdapat pada jumlah generasi ke 100. Sehingga disimpulkan pada jumlah generasi yang terlalu rendah tidak didapatkan generasi terbaik sebagai solusi. Pengujian nilai Cr/Mr optimal dihasilkan *fitness* terbaik dengan nilai 87830 dan terdapat pada jumlah Cr= 0,8 Mr = 0,2. Parameter tidak tetap mempengaruhi jika nilai Cr semakin Kecil dan Nilai Mr semakin besar atau nilai Mr lebih besar dari Cr akan menghasilkan fitness yang semakin baik tetapi tergantung pada penyesuaian parameter terhadap representasi kromosom.

.

7.2 Saran

Pada penelitian ini, terdapat hal yang dapat ditambahkan dan dikembangkan untuk penelitian selanjutnya antara lain :

1. Nilai *fitness* masih dapat ditingkatkan dengan menambah jumlah maksimum parameter iterasi dan populasi pengujian walaupun membutuhkan waktu komputasi yang lebih lama dan alokasi memori perangkat yang lebih banyak, sehingga akan lebih baik apabila menggunakan perangkat keras dengan spesifikasi yang lebih besar untuk pengalokasian memori yang lebih baik.
2. Menambah data keminatan, sehingga kesalahan penempatan mata kuliah pada perhitungan dapat diminimalisir dengan semakin banyaknya data.
3. Mungkin bisa ditambahkan *fitur* atau *batasan* cost yang lain.

DAFTAR PUSTAKA

Albar, 2013. *Algoritma Genetik Tabu Search Dan Memetika Pada Permasalahan Penjadwalan Kuliah.* Seminar Nasional Teknologi Informasi dan Multimedia 2013.

Eberhart, R. C., Shi, Y., 2001. *Particle Swarm Optimization: Developments, Applications, Resources*. Evolutionary Computation, 2001. Proceedings of the 2001 Congress on Vol. 1, Mei 2001.

Ghaziabad, 2008. *A Comparison Of Memetic & Tabu Search For The Cryptanalysis Of Simplified Data Encryption Standard Algorithm*. Journal of Theoretical and Applied Information Technology.

Hetari, R., Cholissodin, I., Mahmudy, W. F., 2016. *Implementasi Hybrid Genetic Algorithm dan Simulated Annealing untuk Pembagian Tugas Mengajar Dosen PTIIK.* Repositori Jurnal Mahasiswa PTIIK UB Vol. 7, No. 15.

Hsieh, H. I., Lee T. P., Lee, T. S., 2011. *A Hybrid Particle Swarm Optimization and Support Vector Regression Model for Financial Time Series Forecasting*. International Journal of Business Administration Vol.2, No. 2; May 2011.

Mahmudy, W. F., 2006. *Penerapan Algoritma Genetika Pada Optimasi Model Penugasan*. Natural Vol.10, No. 3, p. 197-207.

Mansur., Prahasto, T., Farikhin., 2014. *Particle Swarm Optimization Untuk Sistem Informasi Penjadwalan Resource Di Perguruan Tinggi*. Jurnal Sistem Informasi Bisnis 01 (2014).

Marini, F., Walczak, B., 2015. *Particle Swarm Optimization (PSO). A Tutorial*. Chemometrics and Intelligent Laboratory Systems.

Paendong, M., Prang, J. D., 2011. *Optimasi Pembagian Tugas Karyawan Menggunakan Metode Hungarian*. Jurnal Ilmiah Sains Vol. 11 No.1, April 2011.

Nurrochma, 2018. *Penerapan Hybrid Algoritma Genetika dan Particle Swarm Optimization (PSO) untuk menyelesaikan Multi Trip Vehicle Routing Problem.*, Skripsi Thesis Universitas Airlangga.

Rachman, R. A., Syarif, D., Sari, R. P., 2012. *Analisa dan Penerapan Metode Particle Swarm Optimization Pada Optimasi Penjadwalan Kuliah*. Jurnal Teknik Informatika, Vol 1 September 2012.

Taha, H. A., 2007. *Operations Research: An Introduction 8th Edition.* Pearson Prentice Hall. Pearson Education, Inc.