OPTIMASI MULTI-OBJEKTIF UNTUK PENENTUAN UANG KULIAH TUNGGAL MENGGUNAKAN ALGORITME NON-DOMINATED SORT GENETIC ALGORITHM II (NSGA-II) YANG DIMODIFIKASI

Tesis

Untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Magister Komputer

Disusun oleh :

Farid Jauhari

NIM : 176150100111026

PROGRAM STUDI MAGISTER ILMU KOMPUTER

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

2019

PENGESAHAN

OPTIMASI MULTI-OBJEKTIF UNTUK PENENTUAN UANG KULIAH TUNGGAL MENGGUNAKAN ALGORITME NON-DOMINATED SORT GENETIC ALGORITHM II (NSGA-II) YANG DIMODIFIKASI

TESIS

Diusulkan untuk memenuhi sebagian persyaratan
Memperoleh gelar Magister Komputer

Disusun Oleh :
Farid Jauhari
NIM: 176150100111026

Tesis ini telah diuji dan dinyatakan lulus pada
30 Desember 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wayan Firdaus Mahmudy, S.Si., MT., Ph.D.	Achmad Basuki, ST., MMG., Ph.D.
NIP: 197209191997021001	NIP: 197411182003121002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, ST., MT., Ph.D.
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah Tesis ini tidak terdapat karya ilmiah yang pernah diusulkan 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 Tesis ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia Tesisini digugurkan dan gelar akademik yang telah saya peroleh (Magister) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Desember 2019

Farid Jauhari
NIM: 176150100111026

KATA PENGANTAR

Segala puji syukur untuk Allah Subhanahu Wa Ta'ala karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan Tesis yang berjudul "OPTIMASI MULTI-OBJEKTIF UNTUK PENENTUAN UANG KULIAH TUNGGAL MENGGUNAKAN ALGORITME NON-DOMINATED SORT GENETIC ALGORITHM II (NSGA-II) YANG DIMODIFIKASI". Tesis ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Master Komputer di Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan baik lahir maupun batin selama penulisan tesis ini. Penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada:

1. Allah Subhanahu Wa Ta'ala karena atas kehendak dan nikmat-Nya laporan tesis ini telah selesai dengan baik.
2. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku dosen pembimbing I tesis yang telah memberikan waktu, ilmu dan saran untuk menyelesaikan tesis ini.
3. Achmad Basuki, ST., MMG., Ph.D. selaku dosen pembimbing II tesis yang telah memberikan waktu, ilmu dan saran untuk menyelesaikan tesis ini.
4. Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika.
5. Dr. Eng. Fitra A. Bachtiar, S.T, M.Eng. selaku Ketua Program Studi Magister Ilmu Komputer.
6. Orang tua dan keluarga yang mendoakan dan telah memberikan bantuan moral maupun material kepada penulis untuk menyelesaikan tesis ini.
7. Keluarga besar Magister Ilmu Komputer yang telah memberikan dukungan dan bantuan selama masa kuliah hingga selesainya tesis ini.
8. Orang-orang yang selalu mendoakan dan membantu penulis yang tidak dapat diucapkan satu persatu, terimakasih atas semua dukungannya.

Penulis mengucapkan terimakasih dan memohon maaf apabila dalam penyusunan laporan ini terdapat banyak kekurangan. Penulis mengharapkan adanya saran maupun kritik yang berguna untuk di masa yang akan datang. Semoga tesis ini dapat memberikan manfaat bagi semua pihak.

Malang, 30 Desember 2019

Penulis
faridjauhari@ub.ac.id

ABSTRAK

Penentuan Uang Kuliah Tunggal (UKT) proporsional adalah proses penentuan besaran UKT yang harus dibayarkan mahasiswa. Penentuan besaran UKT didasarkan pada kemampuan finansial keluarga dari mahasiswa tersebut. Namun, ketika mahasiswa dalam satu fakultas terlalu banyak yang kemampuan finansialnya kurang, maka penghasilan fakultas mungkin kurang dari biaya operasional yang dibutuhkan. Pada penelitian sebelumnya, penentuan UKT diselesaikan menggunakan metode-metode data mining dan cenderung proporsional terhadap kemampuan finansial mahasiswa saja, tanpa menghiraukan penghasilan institusi. Penentuan UKT proporsional dalam penelitian ini akan diselesikan sebagai permasalahan optimasi multi-objektif. Tujuan dari proses optimasi adalah minimal jarak antara besaran UKT dengan kemampuan finansial mahasiswa, dan maksimal penghasilan institusi. Salah satu algoritme yang sering digunakan untuk menyelesaikan permasalahan multi-objektif adalah Non-dominated Sort Genetic Algorithm II (NSGA-II). Pada penelitian ini dilakukan pemodelan permasalahan penentuan UKT dengan menggunakan NSGA-II. Pemodelan NSGA II yang dimaksud adalah mencari representasi kromosom terbaik, parameter genetik terbaik, operator genetika terbaik, proses inisialisasi terbaik, hingga melakukan modifikasi metode seleksi dari NSGA-II. Hasil penelitian menunjukkan bahwa representasi kromosom kedua yang diusulkan peneliti lebih baik daripada representasi kromosom pertama. Untuk data 100 mahasiswa dalam 2 prodi, NSGA-II yang dimodifikasi berhasil memperoleh solusi yang layak hanya menggunakan jumlah populasi 200 dan maksimal iterasi 100. NSGA-II yang dimodifikasi berhasil mempercepat waktu eksekusi dari NSGA-II sebesar 20%, tanpa menurunkan nilai fitness dari solusi yang dihasilkan. Hasil optimasi multi-objektif menggunakan NSGA-II adalah sejumlah solusi, bukan satu solusi. Sehingga solusi yang dihasilkan dalam permasalahan penentuan UKT sangat beragam, mulai dari solusi yang cenderung kepada mahasiswa (solusi dengan nilai fitness 1 terbesar) sampai solusi yang cenderung kepada institusi (solusi dengan nilai fitness 2 tertinggi). Hal ini akan sangat membantu pengambil keputusan dalam penentuan UKT, untuk memilih solusi yang menguntungkan institusi atau mahasiswa, atau solusi yang berada diantara keduanya.

Kata kunci: optimasi multi-objektif, optimasi, NSGA-II, algoritme genetika, UKT, biaya pendidikan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Penentuan Uang Kuliah Tunggal (UKT) proporsional adalah proses penentuan besaran UKT yang harus dibayarkan mahasiswa. Penentuan besaran UKT didasarkan pada kemampuan finansial keluarga dari mahasiswa tersebut. UKT dibagi menjadi beberapa kategori bergantung pada jenis seleksi dari mahasiswa. Kategori I adalah kategori dengan besaran UKT paling rendah yaitu Rp. 500.000 dan Kategori II adalah Rp. 1.000.000. Besaran UKT untuk kategori III dan seterusnya berbeda untuk masing-masing program studi (Universitas Brawijaya, 2016a, 2017a) dan fakultas. Harapan dari UKT proporsional ini adalah membantu mahasiswa dari keluarga kurang mampu agar dapat melanjutkan studi. Jika dalam satu fakultas sebaran kemampuan finansial mahasiswanya cenderung kearah kurang mampu, maka kemungkinan penghasilan dari fakultas tersebut akan kurang. Sedangkan fakultas mempunyai kebutuhan dana dalam jumlah tertentu untuk menjalankan proses perkuliahan.

Proses optimasi penentuan UKT bertujuan untuk menentukan kategori UKT setiap mahasiswa dengan batasan total dari besaran UKT mahasiswa fakultas tersebut tidak kurang dari kebutuhan dana dari Fakultas dalam menjalankan proses perkuliahan (yang lebih lanjut disebut minimal penghasilan Fakultas). Peraturan Menteri Riset, Teknologi, Dan Pendidikan Tinggi Republik Indonesia (PERMERISTEKDIKTI) nomor 22 tahun 2015, memberikan batasan kepada setiap perguruan tinggi agar jumlah mahasiswa yang menerima UKT kategori I dan II masing-masing sebesar 5% dari jumlah mahasiswa yang diterima pada setiap program studi. Berdasarkan uraian tersebut, maka proses optimasi penentuan UKT dapat dikategorikan sebagai permasalahan optimasi multi-objektif. Tujuan optimasi pertama adalah minimal jarak antara kemampuan finansial mahasiswa dengan besaran kategori UKT-nya. Tujuan optimasi kedua adalah memaksimalkan penghasilan fakultas.

Optimasi multi-objektif adalah optimasi dengan tujuan optimasi (objektif) lebih dari satu dan seringkali bertentangan antara tujuan yang satu dengan yang lain (Zhou et al., 2011). Masalah utama dari optimasi multi-objektif adalah terlalu banyak jumlah solusi yang mungkin didapatkan, dikarenakan adanya pertentangan antara tujuannya, sehingga menyebabkan tingginya kompleksitas dari proses perhitungannya. Dalam survei yang dilakukan oleh Zhou dkk (2011), algoritme evolusi yang sering digunakan dalam penelitian adalah Non-Dominated Sorting Genetic Algorithm II (NSGA-II) yang diperkenalkan oleh Kalyanmoy Deb dkk (Deb, Agrawal, Pratab, & Meyarivan, 2002).

Algoritme NSGA-II adalah salah satu pengembangan algoritme genetik yang dikhususkan untuk menyelesaikan permasalahan optimasi multi-objektif. Algoritme NSGA-II menggunakan metode seleksi khusus yang disebut “Non-Dominated Sorting”. Sesuai dengan namanya metode seleksi ini berbasis *elitism,* karena metode ini selalu menyaring solusi (kromosom) terbaik saja untuk masuk ke iterasi berikutnya. Kelemahan dari algoritme NSGA-II adalah sama dengan algoritme genetik lain yang metode seleksinya berbasis *elitism,* yaitu proses seleksinya membutuhkan waktu eksekusi yang lebih lama. Penelitian untuk mempersingkat waktu eksekusi NSGA-II masih dilakukan. Modifikasi berbasis *devide and conquer* berhasil untuk menurunkan kompleksitas proses perankingan (Jensen, 2003). Fang, dkk mengembangkan metode Jensen tersebut dengan membentuk pohon (tree) dominasi (Fang, Wang, Tu, & Horstemeyer, 2008). Berdasarkan metode Jensen dan Fang juga dikembangkan kembali metode berbasis pencarian hirarki yang berhasil memperkecil waktu

eksekusi algoritme NSGA-II (Bao, Xu, Goodman, & Cao, 2017). Usaha untuk mempercepat NSGA-II juga dilakukan dengan memperbaiki "*Arena principle*" (Tang, Cai, & Zheng, 2008). Metode pengurutan *deductive* dan *climbing* juga digunakan untuk usaha mempersingkat waktu eksekusi *NSGA-II* (McClymont & Keedwell, 2012). Meskipun NSGA-II memiliki waktu eksekusi yang relatif tinggi karena kompleksitasnya, namun algoritme ini sering digunakan untuk menyelesaikan berbagai permasalahan optimasi multi-objektif.

Algoritme NSGA-II telah terbukti dapat memecahkan permasalahan optimasi multi-objektif. Algoritme NSGA-II digunakan untuk optimasi rantai pasokan (*supply chain*) dengan tiga tujuan yaitu meminimalkan biaya total pasokan, variasi jumlah pesanan dan meminimalkan penyimpanan (*inventory*) (Bandyopadhyay & Bhattacharya, 2014; Chan, Jha, & Tiwari, 2016; Dixit, Seshadrinath, & Tiwari, 2016; Pasandideh, Niaki, & Asadi, 2015). Liu, dkk (2015) berhasil mengoptimasi proses produksi minyak dan gas menggunakan algoritme NSGA-II (Liu, Gao, & Wang, 2015). Algoritme NSGA-II berhasil mengoptimasi perluasan jaringan gas dan listrik (Hu, Bie, Ding, & Lin, 2016). Tujuan dari optimasi (Hu et al., 2016) adalah meminimalisasi jumlah investasi (*investments*), dan meminimalisasi biaya produksi ditambah dengan biaya emisi karbon (*Carbon Emission Cost*). NSGA-II juga menghasilkan solusi yang optimal pada permasalahan penjadwalan (Campos Ciro, Dugardin, Yalaoui, & Kelly, 2016; Martínez-Puras & Pacheco, 2016; Wang, Xu, Wang, & Zou, 2017). Implementasi algoritme NSGA-II pada permasalahan optimasi penentuan UKT atau biaya pendidikan belum dilakukan.

Penelitian tentang pemodelan matematis tentang optimasi biaya kuliah (harga) dengan kemampuan membayar siswa telah dilakukan oleh Burer dan Fethke (2016). Namun, penelitian tersebut lebih menitikberatkan pada mencari harga biaya pendidikan, tidak untuk menentukan biaya pendidikan proporsional per mahasiswa seperti dalam penelitian ini (Burer & Fethke, 2016). Dalam penelitian terdahulu, permasalahan penentuan UKT diselesaikan menggunakan metode-metode data mining. Hal ini dikarenakan permasalahan penentuan UKT memang cenderung dianggap sebagai permasalahan klasifikasi dan *clustering*. Beberapa metode data mining yang digunakan untuk optimasi penentuan UKT antara lain : Fuzzy C-Means (Muchsin & Sudarma, 2015) dan C4.5 (Karim, Sentinuwo, & Sambul, 2017). Namun, proses penentuan UKT pada kedua penelitian tersebut hanya menitikberatkan kepada sesuainya kemampuan finansial mahasiswa dan besaran UKT yang harus dibayar. Kedua penelitian tersebut tidak memperhatikan total besaran UKT per Mahasiswa tersebut dapat memenuhi minimal penghasilan Fakultas.

Penelitian ini bertujuan untuk mencari pemodelan NSGA-II yang paling efisien untuk penentuan kategori UKT calon mahasiswa. Tujuan pertama dari proses optimasi menggunakan NSGA-II adalah minimal jarak antara kemampuan finansial mahasiswa dengan besaran UKT. Sedangkan tujuan kedua adalah maksimal penghasilan fakultas. Pemodelan NSGA-II yang efisien adalah ketika pemodelan tersebut dapat mencapai hasil optimasi terbaik (mencapai tujuan optimasinya) dengan waktu komputasi yang lebih singkat. Dengan memodelkan proses penentuan UKT ini menjadi optimasi multi-objektif, diharapkan akan terjadi keseimbangan antara penghasilan fakultas dan kemampuan mahasiswa membayar UKT.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang yang sudah disebutkan, rumusan masalah utama dalam tesis ini adalah "bagaimana memodelkan permasalahan penentuan UKT ini menjadi permasalahan optimasi multi-objektif yang diselesaikan dengan NSGA-II?". Dari permasalahan utama tersebut, maka dapat diperinci sebagai berikut:

1. Bagaimana representasi kromosom NSGA-II yang paling efisien untuk optimasi penentuan kategori UKT?
2. Bagaimana parameter algoritme genetika yang efisien untuk permasalahan optimasi penentuan UKT?
3. Bagaimana hasil modifikasi NSGA-II dengan tujuan mempercepat waktu eksekusinya?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah menyelesaikan permasalalahan penentuan UKT yang dimodelkan sebagai permasalahan optimasi multi-objektif. Optimasi multi-objektif dalam hal ini adalah mencari solusi-solusi yang layak (terhindar dari batasan optimasi). Solusi yang dihasilkan dari optimasi multi-objektif diharapkan akan sangat beragam. Sehingga hal ini akan sangat membantu pengambil keputusan dalam penentuan UKT, untuk memilih solusi yang menguntungkan institusi atau mahasiswa, atau solusi yang berada diantara keduanya.

1.4 Manfaat

Penelitian ini memberikan metode baru untuk optimasi penentuan UKT. Penelitian ini juga dapat menarik perhatian penelitian lain untuk menggunakan algoritme atau metode lain yang berbasis optimasi multi-objektif untuk menyelesaikan permasalahan pada penentuan UKT.

1.5 Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. Penelitian ini tidak menjelaskan tentang bagaimana perhitungan untuk mendapatkan kemampuan finansial mahasiswa. Kemampuan finansial mahasiswa diasumsikan adalah penghasilan orang tua perbulan.
2. Jumlah mahasiswa yang menerima UKT kategori I dan II masing-masing sebesar minimal 5% dari jumlah mahasiswa yang diterima pada setiap program studi

1.6 Sistematika Pembahasan

Gambaran pembahasan dari keseluruhan isi laporan penelitian untuk setiap bab adalah sebagai berikut:

BAB 1 PENDAHULUAN

Membahas tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan.

BAB 2 TINJAUAN PUSTAKA

Membahas tentang uraian mengenai pengertian Uang Kuliah Tunggal (UKT), algoritme genetika, membahas tentang optimasi multi-objektif, serta membahas tentang teknologi yang digunakan yaitu *Non-dominated Sort Genetic Algorithm* (NSGAII).

BAB 3 METODOLOGI

Membahas tentang metode penelitian, pemodelan matematis, pemodelan NSGA-II, metode pembanding untuk optimasi penentuan UKT, serta pengujian yang akan dilakukan pada penelitian ini.

BAB 4 PERANCANGAN

Membahas tentang perhitungan manual, perancangan algoritme, dan perancangan uji coba.

BAB 5 ANALISIS DAN HASIL

Membahas tentang hasil pengujian jumlah populasi, maksimal generasi, metode mutasi, metode repair dan metode seleksi.

BAB 6 PENUTUP

Membahas kesimpulan yang didapatkan dari hasil proses optimasi dan pengujian yang dilakukan pada penelitian ini, serta saran untuk perbaikan dan pengembangan selanjutnya.

BAB 2 TINJAUAN PUSTAKA

2.1 Uang Kuliah Tunggal

Uang Kuliah Tunggal (UKT) menurut peraturan Menteri Riset, Teknologi, Dan Pendidikan Tinggi Republik Indonesia (permeristekdikti) nomor 22 tahun 2015 pasal 2 ayat 2 adalah biaya yang ditanggung mahasiswa berdasarkan kemampuan finansial mahasiswa, orang tua mahasiswa, atau wali mahasiswa (pihak lain yang membiayai) yang selanjutnya hanya disebut kemampuan finansial mahasiswa (MENRISTEKDIKTI Republik Indonesia, 2015). Kemampuan finansial ini bisa didapatkan dari biodata yang dimasukkan mahasiswa melalui Sistem Informasi Akademik Mahasiswa Universitas Brawijaya, yang selanjutnya disebut SIAM-UB. Isian di SIAM-UB yang berhubungan dengan kemampuan finansial antara lain : penghasilan orang tua per bulan, penghasilan orang tua per tahun, jumlah saudara, pekerjaan ayah, pekerjaan ibu, penanggung biaya kuliah, kondisi ayah dan kondisi ibu (Universitas Brawijaya, 2016b).

Besaran UKT dikelompokkan menjadi beberapa kelompok (kategori) berdasarkan kemampuan finansial mahasiswa, orang tua mahasiswa, atau wali mahasiswa (pihak lain yang membiayai) (MENRISTEKDIKTI Republik Indonesia, 2015). Kelompok UKT pada Universitas Brawijaya disebut sebagai kategori UKT (Universitas Brawijaya, 2016a)(Universitas Brawijaya, 2017a)(Universitas Brawijaya, 2017b)(Universitas Brawijaya, 2017c). Jumlah kategori UKT berbeda-beda untuk setiap seleksi. Kategori UKT pertama diperuntukkan bagi mahasiswa dengan kemampuan finansial yang paling rendah dengan besaran Rp. 500.000. Kategori UKT kedua ditujukan untuk mahasiwa dengan kemampuan finansial yang rendah dengan besaran UKT 1.000.000. Sedangkan kategori UKT ketiga dan seterusnya digunakan untuk mahasiswa dengan tingkat ekonomi yang berangsur-angsur lebih tinggi dari kategori sebelumnya. Besaran nilai UKT untuk kategori ketiga dan seterusnya berbeda-beda untuk setiap program studi. Pada permenristekditi nomor 22 tahun 2015 pasal 5 ayat satu dan dua memberikan ketentuan untuk setiap perguruan tinggi agar jumlah mahasiswa yang menerima UKT kategori I dan II masing-masing sebesar minimal 5% dari jumlah mahasiswa yang diterima pada setiap program studi (MENRISTEKDIKTI Republik Indonesia, 2015).

Fakultas adalah unit dari perguruan tinggi yang mempunyai tugas menyelenggarakan dan mengelola pendidikan dari satu atau beberapa program studi (MENRISTEKDIKTI Republik Indonesia, 2016). Pada permenristekdikti nomor 4 tahun 2016, salah satu tugas fakultas adalah mengatur keuangan di lingkungan fakultas, termasuk program studi (MENRISTEKDIKTI Republik Indonesia, 2016). Fakultas bertugas untuk menentukan kategori UKT dan menentukan penghasilan minimal fakultas dari pembayaran UKT berdasarkan kebutuhan operasional fakultas dalam melaksanakan perkuliahan.

2.2 Penelitian Terkait

Penelitian tentang pemodelan matematis tentang optimasi biaya kuliah (harga) dengan kemampuan membayar siswa telah dilakukan oleh Burer dan Fethke (2016). Namun penelitian tersebut lebih menitikberatkan pada mencari harga biaya pendidikan, tidak untuk mencari kategori UKT per mahasiswa seperti dalam penelitian ini (Burer & Fethke, 2016).

Permasalahan penentuan UKT diselesaikan menggunakan metode-metode data mining. Hal ini dikarenakan permasalahan penentuan UKT memang cenderung dianggap sebagai permasalahan klasifikasi dan clustering. Muchsin dan Sudarma (2015) menggunakan Fuzzy C-Means yang biasa digunakan dalam *clustering* (Muchsin & Sudarma, 2015). Sedangkan Karim,

dkk (2017) mengunakan algoritme C4.5 yang biasa digunakan dalam permasalahan klasifikasi (Karim et al., 2017). Prasetyanti dan Listyaningrum (2017) melakukan perbandingan penentuan UKT menggunakan metode TOPSIS, SAW dan AHP-TOPSIS (Prasetyanti & Listyaningrum, 2017). Namun proses penentuan UKT pada ketiga penelitian tersebut hanya menitikberatkan kepada sesuainya kemampuan finansial mahasiswa dan besaran UKT yang harus dibayar. Kedua penelitian tersebut tidak memperhatikan total besaran UKT per Mahasiswa tersebut dapat memenuhi minimal penghasilan Fakultas.

Pada penelitian Rokhman, dkk (2017), permasalahan penentuan UKT dipandang sebagai permasalahan optimasi multi-objektif. Rokhman, dkk menggunakan metode MOORA (*Multi Objective Optimization on the Basis of Ratio Analysis*) untuk menyelesaikan permasalahan optimasi multi-objektif dalam penentuan UKT (Syaiful Rokhman, Imam Fahrur Rozi, 2017). Namun dalam publikasinya, Rokhman, dkk tidak menyebutkan secara rinci yang menjadi tujuan dari proses optimasinya. Prinsip dari metode MOORA adalah membuat daftar semua kemungkinan solusi, kemudian menghitung rasio dari nilai fungsi tujuan optimasinya.

2.3 Algoritme Genetika

Algoritme Genetika (GA) adalah salah satu algoritme pencarian yang mengadopsi langkah-langkah dari proses evolusi. GA mempunyai mekanisme yang efektif yang dapat digunakan untuk proses optimasi (Eberhart & Shi, 2007). Hal yang paling menunjang efektifitas dari mekanisme GA adalah representasi kromosomnya (Koza, 1992). Secara umum langkah-langkah dari algoritme genetika dapat dilihat pada Gambar 2-1. Berdasarkan Gambar 2-1, langkah utama algoritme genetika adalah inisialisasi, kombinasi ulang (*recombination*), evaluasi dan seleksi. Variabel *t* pada Gambar 2-1 adalah iterasi. Variabel *P*(*t*) adalah populasi pada iterasi ke-*t*. Sedangkan *C*(*t*) adalah hasil rekombinasi pada iterasi ke-(*t*).

```
Procedure AlgoritmaGenetika
begin
        t = 0
        inisialisasi P(t)
        while (bukan kondisi berhenti)
                rekombinasi C(t) dari P(t)
                evaluasi P(t) dan C(t)
                seleksi P(t+1) dari P(t) dan C(t)
                t = t + 1
        end while
end
```

Gambar 2-1 Langkah-langkah algoritme genetika

2.3.1 Representasi Kromosom

Representasi kromosom atau *encodings* adalah masalah utama dari GA yang paling berpengaruh terhadap kinerja (performa) dari GA (Koza, 1992; M. Srinivas & Patnaik, 1994). Representasi kromosom merupakan pengkodean dari pemasalahan yang akan diselesaikan. Representasi kromosom dapat membatasi jendela (*window*) yang digunakan oleh sistem dalam melihat wilayah pencarian (*search space*). Oleh sebab itu, ketepatan representasi kromosom sangat menunjang efektifitas proses pencarian solusi.

Representasi kromosom yang paling sering digunakan adalah representasi biner (Herrera, Lozano, & Verdegay, 1998). Beberapa Representasi selain bilangan biner adalah sebagai berikut:

1. Representasi bilangan pecahan (*real/floating point*)

GA dengan representasi bilangan pecahan biasa dikenal dengan Real-Coded Genetic Algorithm (RCGA) adalah representasi permasalahan optimasi menggunakan bilangan pecahan (tunggal) dan atau larik bilangan pecahan (*array of real*). Bilangan pecahan tunggal adalah hasil pengkodean (*encode*) dari deret biner representasi dari permasalahan. RCGA dengan bilangan pecahan tunggal (*real*) terbukti lebih efektif untuk menyelesaikan permasalahan penjadwalan (Mahmudy, Marian, & Luong, 2013a, 2013b, 2013c). RCGA juga terbukti dapat menyelesaikan permasalahan optimasi multi-objektif seperti permasalahan *Economic Emission Load Dispatch* (EELD) (Bhattacharjee, Bhattacharya, & Halder Nee Dey, 2014). Tujuan optimasi dari EELD adalah meminimalkan biaya produksi dan juga meminimalkan emisi yang dikeluarkan. RCGA lebih baik digunakan untuk mengoptimasi permasalahan kontinyu (continue), sedangkan representasi biner lebih baik digunakan untuk permasalahan diskrit (Herrera et al., 1998).

2. Representasi bilangan bulat (*integer*)

Representasi kromosom dengan bilangan bulat merupakan representasi kromosom menggunakan bilangan bulat tunggal dan atau larik bilangan bulat (*array of integer*). *Permutation chromosome* adalah salah satu implementasi representasi bilangan bulat yang biasa digunakan dalam permasalahan *Traveling Salesment Problem* (TSP) (Grefenstette, Gopal, Rosmaita, & Gucht, 1985). Representasi ini efektif untuk permasalahan optimasi penjadwalan (Cheng, Gen, & Tsujimura, 1996), meskipun belum dibandingkan hasilnya dengan representasi bilangan real.

3. Daftar terurut (*ordered list*)

Representasi kromosom dalam *ordered list* juga sering digunakan dalam permasalahan optimasi TSP (Grefenstette et al., 1985). Selain itu, representasi ini juga digunakan untuk permasalahan lain, misalnya pencarian reduksi (Wroblewski, 1995), optimasi pencarian dalam database (*database query*) (Kristin Bennett, Michael C. Ferris, & Yannis Ioannidis, 1991), dan penjadwalan *job-shop* (Cheng et al., 1996).

4. Ekspresi pemrograman LISP

Representasi kromosom menggunakan ekspresi-ekspresi Bahasa pemrograman LISP disebut dengan *genetic programming* (Koza, 1992). Tujuan dari genetic programming adalah membuat komputer dapat menyelesaikan permasalahan tanpa adanya campur tangan manusia (tanpa diprogram manusia).

5. Matriks dua dimensi

Representasi matriks dua dimensi sering digunakan dalam permasalahan peletakan benda dalam ruang dua dimensi, misalnya permasalahan pengemasan (Jain & Gea, 1998). Representasi ini juga digunakan untuk membuat desain dua dimensi (Jakiela, Chapman, Duda, Adewuya, & Saitou, 2000; Weile & Michielssen, 1996) dan deteksi garis tepi (Bhandarkar, Zhang, & Potter, 1994; Gudmundsson, El-Kwae, & Kabuka, 1998).

2.3.2 Proses Inisialisasi

Proses inisialisasi pada GA merupakan proses pembentukan populasi awal dengan jumlah yang ditentukan. Semakin besar jumlah populasi yang digunakan, maka GA akan dapat

mengeksplorasi lebih luas, sehingga GA akan terhindar dari konvergensi dini (M. Srinivas & Patnaik, 1994). Menurut hasil survei dari Kazimipour, dkk (2014), proses inisialisasi dapat dibedakan menjadi 3 aspek, yaitu aspek keacakannya (*randomness*), cara penyusunannya (*compositionality*), dan tingkat keumumannya (*generality*) (Kazimipour, Li, & Qin, 2014).

1. Keacakan (*randomness*)

Terdapat dua teknik yang digunakan untuk membangkitkan populasi secara acak, yaitu secara *stochastic* dan *deterministic*. Pembangkitan populasi secara *stochastic* dilakukan dengan menggunakan *Pseudo-Random Number Generator* (PRNG) dan *Chaotic Number Generators* (CNG). Dari kedua metode *stochastic* tersebut yang lebih sering digunakan adalah PRNG. Namun, inisialisasi menggunakan CNG dapat memperbaiki tingkat keragaman populasi, tingkat kesuksesan dan kecepatan dalam mencapai konvergensi dari *Evolutionary Algorithm* (EA), termasuk juga GA.

Teknik *deterministic* adalah teknik yang selalu membangkitkan populasi yang sama (menghiraukan *initial seed*). Terdapat dua pendekatan pada teknik deterministic, yaitu *Quasi-Random Sequence* (QRS) dan *Uniform Experimental Design* (UED). Pendekatan QRS hanya mempertimbangkan keseragaman dengan proyeksi 1 dimensi saja, sedangkan UED dapat mempertimbangkan dengan proyeksi 2 atau lebih dimensi. Pendekatan UED secara umum dapat membangkitkan nilai secara diskrit, sedangkan QRS dapat membangkitkan dengan sebaran kontinyu.

Terdapat banyak pendekatan untuk menghasilkan QRS dengan kelebihan dan kekurangan masing-masing (Morokoff & Caflisch, 1994). Dalam penelitian Morokoff dan Caflisch (1994) membandingkan metode Halton, Sobol dan Faure. Metode yang menghasilkan deret terbaik adalah metode Fraure. Namun waktu komputasi dari metode Faure lebih lama dibandingkan dengan metode yang lain. Waktu komputasi terbaik diperoleh oleh metode Sobol. Selain waktu komputasinya yang baik, metode sobol juga memiliki hasil yang paling baik untuk menciptakan deret dengan dimensi yang tinggi.

Joe dan Kuo menemukan permasalahan pada metode Sobol ketika membangkitkan deret dua dimensi dengan deret jumlah yang sangat banyak (Joe & Kuo, 2008). Joe dan Kuo berhasil memperbaiki metode Sobol, dan menyatakan bahwa metode Sobol yang diperbaiki memberikan hasil yang lebih baik untuk beberapa model finansial. Namun metode Sobol Joe dan Kuo ini masih memperoleh hasil yang lebih buruk untuk beberapa kasus.

2. Cara penyusunan (*compositionality*)

Inisialisasi populasi awal dibedakan menjadi 2 kategori berdasarkan cara penyusunannya, yaitu *non-composite* dan *composite*. Cara penyusunan yang termasuk kategori *non-composite* adalah cara penyusunan populasi awal dengan 1 metode saja tanpa hibridisasi. Sedangkan kategori *composite* adalah cara penyusunan dengan 2 atau lebih metode, misalnya populasi awal dibangkitkan dengan hibridisasi antara PRNG dengan QRS (Nguyen Quang Uy, Nguyen Xuan Hoai, McKay, & Tuan, 2007).

3. Tingkat keumuman (*generality*)

Aspek tingkat keumuman adalah seberapa luas metode pembangkitan populasi awal ini dapat diterapkan pada berbagai permasalahan (*domain*).

2.3.3 Proses Rekombinasi

Metode rekombinasi sering disebut juga dengan operator genetika, merupakan usaha untuk memperoleh kombinasi baru dari kromosom. Proses rekombinasi dilakukan dengan

merubah sebagian gen dari sejumlah kromosom pada populasi. Sehingga proses rekombinasi ini akan membentuk kromosom baru yang sedikit berbeda dari kromosom awalnya. Kromosom yang baru biasa disebut dengan *child* dan kromosom awal disebut dengan *parent* (Coello Coello, 2002).

Sesuai dengan genetika pada biologi, rekombinasi kromosom terdapat dua jenis yaitu *crossover* dan mutasi. Terdapat berbagai macam metode *crossover* dan mutasi. Pemilihan penggunaan metode *crossover* dan mutasi, tergantung pada representasi kromosom yang digunakan (Grefenstette et al., 1985). Pemilihan metode reproduksi juga dipengaruhi oleh besarnya jumlah populasi yang digunakan, *uniform crossover* lebih baik daripada *one-cut-poin crossover*, jika menggunakan jumlah populasi yang kecil (M. Srinivas & Patnaik, 1994).

Representasi biner dan bilangan real mempunyai metode reproduksi yang berbeda (Grefenstette et al., 1985; Herrera et al., 1998). Jumlah kromosom yang direproduksi oleh *crossover* dan mutasi ditentukan oleh prosentase peluang, yang dikenal dengan *crossover rate* (*cr*) dan *mutation rate* (*mr*). Pemilihan nilai *cr* dan *mr* merupakan permasalahan yang terbuka (M. Srinivas & Patnaik, 1994), hal ini sangat bergantung pada fungsi tujuan optimasi permasalahan, representasi kromosom dan metode reproduksi yang digunakan.

2.3.4 Proses Evaluasi

Proses evaluasi adalah proses perhitungan nilai *fitness* dari semua kromosom yang terdapat pada generasi saat ini. Perhitungan nilai *fitness* didasarkan pada fungsi *fitness*, yang dalam permasalahan optimasi sering disebut sebagai fungsi objektif (tujuan) (M. Srinivas & Patnaik, 1994). Pembentukan fungsi *fitness* merupakan konsekuensi dari representasi permasalahan kedalam model GA (representasi kromosom). Besarnya nilai fitness dari suatu kromosom dapat berarti sehatnya (*fit*) kromosom tersebut. Kromosom yang lebih sehat dinilai lebih dapat bertahan hidup dibandingkan dengan kromosom yang kurang sehat (M. Srinivas & Patnaik, 1994). Kromosom yang lebih sehat akan tetap digunakan, dan kromosom yang kurang sehat dihilangkan pada proses seleksi.

2.3.5 Proses Seleksi

Proses seleksi bertujuan untuk memilih sejumlah kromosom yang dipertahankan untuk membentuk generasi (populasi) berikutnya, dengan harapan agar mendapatkan keturunan (generasi berikutnya) yang lebih baik (nilai *fitness* lebih besar) (Razali & Geraghty, 2011). Menurut razali dan Geraghty (2011), terdapat tiga jenis metode seleksi yang sering digunakan yaitu *tournament*, *proportional roulette wheel*, dan *rank-based roulette wheel selection* (Razali & Geraghty, 2011). Dalam laporan Blickle dan Thiele (1995) terdapat 5 metode seleksi yang berbeda, yaitu *tournament*, *truncation*, *linear ranking*, *exponential ranking* dan *proportional ranking selection* (Blickle & Thiele, 1995).

Metode seleksi *tournament* dilakukan dengan memilih sejumlah *n* kromosom secara random. Kemudian membandingkan nilai *fitness* dari sejumlah n kromosom yang terpilih. Kromosom dengan nilai *fitness* yang paling tinggi dipilih sebagai kromosom yang lolos proses seleksi. Nilai *n* (*tournament size*) yang paling sering digunakan adalah dua, yang biasa disebut sebagai *binary tournament selection*. Pada metode seleksi *tournament* kromosom dengan nilai *fitness* terburuk tidak akan dipilih sebagai individu pada generasi berikutnya. Keunggulan dari metode seleksi *tournament* adalah mudah diimplementasikan, waktu komputasi yang efisien, keragaman populasi (generasi) terjaga dan tidak membutuhkan pengurutan (Razali & Geraghty, 2011). Kekurangan dari metode ini adalah waktu untuk mencapai konvergen relatif lebih lama, ketika keragaman populasi sangat tinggi (Razali & Geraghty, 2011).

Pemilihan kromosom (individu) pada metode *proportional roulette wheel selection* (PRWS) didasarkan pada nilai peluang (Razali & Geraghty, 2011). Nilai peluang didapatkan dari nilai *fitness* kromosom tersebut dibagi dengan total nilai *fitness* semua kromosom pada populasi tersebut. Berbeda dengan metode seleksi *tournament*, kromosom dengan nilai *fitness* terkecil pun masih memiliki peluang untuk terpilih menjadi generasi berikutnya. Dan sebaliknya kromosom dengan nilai *fitness* terburuk masih mempunyai peluang untuk tidak terpilih. Kekurangan dari metode PRWS adalah hanya cocok untuk fungsi dengan objektif maksimasi. Untuk fungsi objektif minimasi harus dirubah terlebih dahulu ke bentuk maksimasi.

Metode *rank-based roulette wheel selection* (RBWS) hampir sama dengan metode PRWS pada paragraf sebelumnya. Perbedaan dari RBWS adalah pada penentuan nilai probabilitas dari kromosomnya, yaitu dihitung berdasarkan peringkat nilai *fitness*-nya, tidak lagi menggunakan nilai fitnessnya secara langsung (Razali & Geraghty, 2011). Terdapat dua jenis metode RBWS yaitu linear dan non-linear. Perbedaan antara linear dan non-linear hanya pada formula dalam menentukan peluang berdasarkan peringkat kromosom (Blickle & Thiele, 1995; Razali & Geraghty, 2011). Karena peluang hanya dihitung berdasarkan rankingnya, maka kromosom dengan nilai fitness terbaik dan terburuk akan memiliki nilai peluang yang tidak banyak berbeda. Kondisi tersebut akan lebih mendukung tidak terjadinya konvergensi dini dibandingkan dengan *tournament* dan PRWS. Namun, hal tersebut juga akan menambah waktu dalam mencapai konvergen. Waktu komputasi untuk melakukan metode seleksi ini lebih besar dibandingkan dengan metode tournament dan PRWS, karena terdapat proses pengurutan berdasarkan nilai *fitness,* untuk mendapatkan peringkat dari kromosom tersebut dalam populasi.

Metode *truncation selection* atau disebut dengan *elitism* adalah metode seleksi yang memilih sejumlah kromosom dengan nilai fitness terbaik saja (Blickle & Thiele, 1995). Waktu komputasi metode ini lebih besar dibandingkan dengan metode *tournament* atau *proportional roulette wheel*, karena adanya proses pengurutan berdasarkan nilai *fitness* semua kromosom dalam satu populasi. Dengan metode ini keragaman populasi sangat kurang, karena hanya dipilih sejumlah kromosom terbaik saja. Namun untuk mencapai konvergen relative lebih cepat dibandingkan dengan tiga metode sebelumnya.

2.3.6 Metode Penanganan Batasan (*Constraint*)

Penanganan batasan (*constraint*) adalah salah satu kerumitan dalam penyelesaian permasalahan optimasi (Michalewicz, 1995). Terdapat banyak sekali metode yang sudah dikembangkan untuk menangani *constraint* (Mezura-Montes & Coello Coello, 2011). Fungsi penalti (*penalty functions*) adalah metode yang paling sederhana dan sering digunakan, terutama ketika menggunakan GA. Metode *stochastic rank* (*SR*) digunakan sebagai pengambil keputusan, ketika menangani beberapa *constraint* dengan fungsi penalti. Selain *SR*, metode yang saat ini sering digunakan adalah dengan menggunakan konsep optimasi multi-objektif.

Penanganan *constraint* dengan konsep optimasi multi-objektif dilakukan dengan menganggap *constraint* adalah objektif (Mallipeddi & Suganthan, 2010). Pengambilan keputusan pada optimasi multi-objektif adalah berdasarkan teori Pareto. Untuk lebih jelas mengenai optimasi multi-objektif, akan dijelaskan pada subbab 2.4. Metode yang paling sederhana adalah dengan memilih solusi yang paling sedikit melanggar *constraint* dan paling baik nilai objektif-nya.

Metode operator khusus (*special operators*) adalah metode yang handal dalam menangani *constraint* (Mezura-Montes & Coello Coello, 2011). Operator khusus ini dibuat untuk mengubah solusi yang melanggar *constraint* (*infeasible solution*) menjadi *feasible solution*. Metode perbaikan (*repair method*) adalah salah satu contoh metode operator khusus yang biasa digunakan dalam permasalahan kombinatorial.

Hingga saat ini, terdapat banyak jenis metode perbaikan yang telah dikembangkan (Salcedo-Sanz, 2009). Penentuan dalam menggunakan metode perbaikan sangat bergantung pada representasi solusi (kromosom) yang digunakan. Metode perbaikan yang digunakan untuk representasi *array* berbeda dengan representasi *matrix*. Metode perbaikan juga dapat dibuat sangat spesifik terhadap permasalahan. Hal ini berbeda dengan metode operator khusus yang digunakan pada permasalahan optimasi fungsi (optimasi numerik). Operator khusus yang digunakan untuk permasalahan optimasi fungsi dibuat seumum mungkin.

Hasil survey Mezura-Montes dan Coello (2011) juga mengatakan bahwa metode penanganan *constraint* menggunakan lebih dari satu metode (hibridisasi) memiliki hasil yang sangat baik (Mezura-Montes & Coello Coello, 2011). Metode penanganan constraint ini biasa disebut dengan *Ensemble of Constraint Handling Techniques* (*ECHT*). Salah satu contoh penelitian yang menggunakan ECHT adalah Mallipeddi dan Suganthan (2010). Mallipeddi dan Suganthan (2010) menggunakan empat metode penanganan *constraint* yang berbeda yaitu *Superiority of Feasible Solution*, *Self-Adaptive Penalty* (SP), *ε-Constraint* (EC), dan *Stochastic Ranking* (SR) (Mallipeddi & Suganthan, 2010).

2.3.7 Penelitian Dengan Menggunakan Algoritme Genetika

Algoritme genetika (GA) telah banyak digunakan untuk menyelesaikan permasalahan optimasi. Penelitian yang dilakukan Taufiq, Dewi, dan Mahmudy (2017) bertujuan untuk mengoptimalkan komposisi pakan ternak dengan total harga termurah namun tetap bernilai gizi tinggi. Pada penelitian ini Taufiq, dkk merepresentasikan kromosom menggunakan *real code* yaitu bilangan desimal yang dibangkitkan secara acak. Proses *crossover* mengunakan metode *extended intermediate*. Proses mutasi menggunakan metode mutasi acak. Dan proses seleksi menggunakan metode *elitism* (Taufiq, Dewi, & Mahmudy, 2017).

Penelitian yang dilakukan oleh Mahmudy, dkk (2014) bertujuan untuk menyelesaikan permasalahan optimasi pemilihan *part* dan permasalahan pemuatan mesin pada system manufaktur yang fleksibel (Mahmudy, Marian, & Luong, 2014). Representasi kromosom yang digunakan adalah *real code*. Proses *crossover* menggunakan dua metode yaitu *flat crossover* dan *extended intermediate crossover*. Proses mutasi juga menggunakan dua metode yaitu *random exchange mutation* dan *simple random mutation*. Proses seleksi menggunakan metode *replacement*. Untuk menjaga keragaman populasi ditambahkan metode *Variable Neighborhood Search* (*VNS*) (Mahmudy et al., 2014). Pada permasalahan ini terdapat 6 tujuan optimasi (*constraint*) yaitu:

1. Memastikan salah satu rencana produksi alternatif untuk salah satu jenis *part* terpilih.
2. Menjamin semua operasi dari jenis *part* yang terpilih diproses.
3. Memastikan setiap operasi dari jenis *part* selesai pada mesin yang dipilih.
4. Memastikan semua alat yang dibutuhkan dimasukkan ke mesin ketika mesin dipilih untuk memproses suatu operasi.
5. Menjamin bahwa jumlah alat yang dimasukkan ke mesin tidak melebihi ketersediaannya.

6. Memastikan bahwa jumlah slot alat yang ditempati pada mesin tidak boleh melebihi kapasitas slot alat pada mesin.

Karena pada permasalahan ini terdapat 6 tujuan optimasi, maka permasalahan ini termasuk ke dalam permasalahan optimasi multi-objektif. Namun pada penelitian yang dilakukan oleh Mahmudy, dkk (2014), 6 tujuan optimasi tersebut diformulasikan menjadi satu fungsi saja, sehingga dapat diselesaikan menggunakan GA *single-objective*.

2.4 Algoritme Genetika Untuk Optimasi Multi-Objektif

Banyak permasalahan optimasi yang menggunakan multi-objektif atau bias dikatakan mempunyai tujuan optimasi lebih dari satu dan saling bertentangan. Permasalahan optimasi multi-objektif dapat dilambangkan dengan rumus (1), dengan Ω adalah ruang pencarian solusi, m adalah jumlah fungsi optimasi (*constraint*) (Zhou et al., 2011).

$$Min/Max\ F(x) = \left(f_1(x), f_2(x), \dots, f_m(x)\right)^T \quad untuk\ x \in \Omega \qquad (1)$$

Konsep optimalisasi Pareto adalah pendekatan yang dapat digunakan untuk menyelesaikan permasalahan optimalisasi multi-objektif (Zhou et al., 2011). Konsep pareto inil diterapkan pada penyelesaian masalah optimalisasi multi-objektif menggunakan algoritme evolusi (E.Zitzler & L.Thiele, 1998) (Deb et al., 2002) (Zhou et al., 2011) (Gao et al., 2014).

2.4.1 Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

Dalam survei yang dilakukan Zhou, dkk diketahui bahwa kebanyakan penelitian pada permasalahan optimalisasi multi-objektif menggunakan algoritme evolusi, menggunakan metode yang sama dengan algoritme NSGA-II (*Non-Dominated Sorting Genetic Algorithm*) yang dihasilkan pada penelitian Deb, dkk pada tahun 2002 (Deb et al., 2002; Zhou et al., 2011). Pada dasarnya algoritme NSGA-II adalah perbaikan dari algoritme NSGA yang diperkenalkan oleh Srinivas dan Deb pada tahun 1995 (Deb et al., 2002; N. Srinivas & Deb, 1994). Perbaikan utama yang dilakukan Deb, dkk adalah pada metode *non-dominated sorting* dan operator perbandingan tingkat keramaian (*Crowded comparison operator*). Alur dari algoritme NSGA dapat dilihat pada Gambar 2-2. Proses regenerasi pada Gambar 2-2 sama dengan proses rekombinasi pada subbab 2.3.3 yaitu terdiri dari *crossover* dan mutasi.

```
1   Inisialisasi : generate random Populasi awal(P0)
2   t=1
3   Ulangi Ketika belum mencapai kondisi terminasi
4           Qt = Regenerasi(Pt-1)
6           Pt=Pt-1+Qt
7           hitungFitnessDanConstraint (Pt)
8           F = Fast-Non-Dominated-Sort(Pt)
9           HitungCrowdingDistance(F)
10          F = SortingBerdasarkan-CrowdedComparisonOperator(F)
11          Pt+1=F[0..pop_size]
12          t=t+1
```

Gambar 2-2 Diagram alir algoritme NSGA-II

Algoritme NSGA-II menawarkan solusi untuk mengurangi jumlah populasi dalam setiap pembentukan individunya (proses reproduksi), yaitu dengan membandingkan setiap solusi (individu baru) dengan himpunan dari individu yang sudah ada (parent). Jika individu baru lebih dominan dari salah satu individu, maka individu baru ini dihapus kedalam himpunan parent, jika individu baru kurang dominan dibandingkan dengan salah satu individu pada himpunan parent, maka individu pada himpunan parent ini akan dihapus dari himpunan parent (Deb et al., 2002).

Telah banyak penelitian menggunakan Algoritme NSGA-II untuk memecahkan masalah optimasi. Liu dkk (Liu et al., 2015) melakukan optimasi untuk proses produksi minyak dan gas. Optimasi dilakukan dengan tujuan memaksimalkan produksi minyak, meminimalkan produksi air dan meminimalkan konsumsi energi secara keseluruhan untuk produksi 1 ton minyak. Proses optimisasi dilakukan dengan menggunakan algoritme NSGA-II yang diimprovisasi. Improvisasi dilakukan pada proses inisialisasi populasi, yaitu dengan membuat hibridisasi chaotic mapping model. Selain itu pada penelitian Liu dkk mendefinisikan gradient operator yang dikombinasikan pada proses *crossover* dan mutasi untuk membuat generasi baru. Hu dkk (Hu et al., 2016) melakukan optimasi dalam pengembangan jaringan listrik dan gas. Tujuan dari optimasinya adalah keamanan jaringannya, meminimalkan biaya investasi dan biaya produksi. Proses optimasi menggunakan algoritme NSGA-II dengan improvisasi pada proses evaluasi, yaitu menggunakan kombinasi Primal-Dual Interior-Point (PDIP) dengan metode estimasi titik. Sebagai pengambil keputusan Hu, dkk menggunakan pendekatan fuzzy. Algoritme NSGA-II juga berhasil mengoptimasi permasalahan rantai pasokan (*supply chain*) (Bandyopadhyay & Bhattacharya, 2014; Chan, Jha, & Tiwari, 2016; Dixit, Seshadrinath, & Tiwari, 2016; Pasandideh, Niaki, & Asadi, 2015). Pada permasalahan penjadwalan, NSGA-II juga juga menghasilkan solusi yang optimal (Campos Ciro, Dugardin, Yalaoui, & Kelly, 2016; Martínez-Puras & Pacheco, 2016; Wang, Xu, Wang, & Zou, 2017).

2.4.2 Proses Seleksi pada NSGA-II

Proses seleksi dalam algoritme NSGA-II termasuk dalam kategori metode *elitist*. Proses *elitist* terdapat pada proses *non-dominated-sorting*, yaitu proses pengurutan berdasarkan operator dominasi dan proses sorting berdasarkan *crowded comparison operator*. Proses *non-dominated-sorting* dapat dilihat pada Gambar 2-3. Dari hasil sorting berdasarkan *non-dominated-sorting* dan *crowded comparison operator* ini diambil sejumlah populasi yang ditentukan (*popSize*) sehingga menghasilkan populasi yang baru. Proses perhitungan *crowding distance* dapat dilihat pada Gambar 2-4.

Langkah-langkah metode seleksi *fast-non-dominated-sort* pada Gambar 2-3 dimulai dengan membuat *array* penampungan sementara *T*. Kemudian untuk setiap individu atau kromosom *s* anggota populasi *S*, dicek apakah sudah ada pada *T*. Jika *s* belum terdapat pada *T*, maka *s* ditambahkan pada *T*. Untuk setiap *t* anggota dari *T,* jika *t = s,* maka tidak dilakukan pengecekan lebih lanjut. Jika *t != s* dan *s* mendominasi *t,* maka *t* dihapus dari daftar *T*, dan jika *t* mendominasi *s,* maka hapus *s* dari *T*. Hasil akhir dari *fast-non-dominated-sorting* adalah *T* yang berisi solusi-solusi yang paling mendominasi (sama sekali tidak didominasi solusi lain). Hasil akhir dari *fast-non-dominated-sort* ini biasa disebut dengan *Front* atau *Pareto Front*. Proses *fast-non-dominated-sort* pertama menghasilkan *Front* yang pertama. Untuk menghasilkan *Front* selanjutnya, maka proses *fast-non-dominated-sort* dilakukan kembali dengan *S = S – T*, atau bias dikatakan dilakukan untuk *S* yang belum terdapat pada *Front*.

```
1   fast-nondominated-sort( S )
2      T = {}
3      foreach S as s
4            if( notIn( s , T ) )
5                  T.add( s )
6                  foreach T as t
7                        if( t != s )
8                              if ( dominated( s, t ) )
9                                    removeElement( T , t )
10                             elseif ( dominated( t , s ))
11                                   removeElement( T , s )
```

Gambar 2-3 Algoritme *fast-non-dominated-sorting*

```
1   HitungCrowdingDistance( S )
2        foreach  S  as  s
3              s.distance = 0
4        SortingBerdasarkanFitness1( S )
5        S[ 1 ] = MAX_DOUBLE
6        S[ size(S)-1 ] = MAX_DOUBLE
7        for i = 2 to size( S )-2
8              S[ i ].distance = S[ i ].distance + ( | S[ i+1 ].fitness1 - S[ i-1 ].fitness1 | )
9        SortingBerdasarkanFitness2( S )
10       S[ 1 ] = MAX_DOUBLE
11       S[ size(S)-1 ] = MAX_DOUBLE
12       for i = 2 to size(S)-2
13             S[ i ].distance = S[ i ].distance + (|S[ i+1 ].fitness2 - S[ i-1 ].fitness2|)
```

Gambar 2-4 Algoritme perhitungan *crowding distance*

Langkah-langkah perhitungan *crowding distance* pada Gambar 2-4 dimulai dengan inisialisasi jarak = 0 untuk setiap solusi *s* anggota *S*. Kemudian dilakukan pengurutan berdasarkan nilai fitness pertama. Setelah pengurutan, dilakukan perhitungan jarak untuk solusi *s* kedua (*S[2]*) sampai terakhir-1 (*S[size(S)-1]*). Jarak dihitung dengan menjumlahkan jarak dari *s* dengan nilai *absolute* dari pengurangan nilai *fitness S[i+1]* dengan nilai *fitness S[i-1]*. Sedangkan untuk solusi pertama dan terakhir diisi dengan nilai tak hingga atau nilai yang paling besar dalam tipe data tertentu. Proses mulai dari pengurutan berdasarkan nilai *fitness* tertentu sampai perhitungan jarak diulangi untuk setiap nilai *fitness* yang ada. Proses perhitungan *crowding distance* ini dilakukan untuk setiap *Front* yang dihasilkan oleh proses *fast-non-dominated-sort*.

Setelah proses perhitungan *crowding distance*, dilakukan proses *sorting* (pengurutan) berdasarkan *crowded comparison operator*. Dengan *crowded comparison operator*, *x*

dikatakan lebih baik dari *y* adalah jika $x_{rank} > y_{rank}$ atau ($x_{rank} = y_{rank}$ dan $x_{distance} > y_{distance}$). Dengan x_{rank} adalah indeks front dari *x*, dan $x_{distance}$ adalah nilai *crowding distance* dari *x*. Jadi individu solusi yang lebih baik adalah solusi yang memiliki indeks *front* lebih kecil atau dapat dikatakan berada pada *front* lebih awal. Dan ketika dua individu solusi berada pada *front* yang sama, maka akan dibandikan nilai *crowding distance*-nya, nilai *crowding distance* yang lebih besar menunjukkan individu yang lebih baik.

2.5 Penelitian Tentang Modifikasi NSGA-II

Modifikasi algoritme NSGA-II sebagian besar dilakukan untuk mempersingkat waktu eksekusinya dengan cara menyederhanakan tingkat kompleksitas dari algoritme pada metode seleksinya. Modifikasi berbasis algoritme *devide and conquer* berhasil untuk menurunkan kompleksitas proses perankingan (Jensen, 2003). Inti dari metode Jensen tersebut adalah membagi proses sorting berdasarkan jumlah tujuan optimasinya. Ketika suatu permasalahan optimasi mempunyai 4 tujuan optimasi, maka proses sorting dibagi menjadi 2, yang masing-masing berdasarkan 2 tujuan optimasi. Kemudian hasil tersebut digabung (dibandingkan) yang dinamakan dengan proses *merge*.

Berbeda dengan metode Jensen yang membagi proses sorting berdasarkan jumlah tujuan optimasinya. Fang, dkk mengembangkan metode Jensen tersebut dengan membentuk pohon dominasi (*dominance tree*) (Fang et al., 2008). Ketika solusi *a* didominasi oleh solusi *b*, maka solusi *a* menjadi anak dari solusi *b*. Sehingga ketika solusi *b* didominasi oleh solusi *c*, maka solusi *a* menjadi turunan kedua (cucu) dari solusi *c*. Sehingga proses sorting hampir sama dengan proses *merge sort*. Selain itu, Fang, dkk juga mempercepat proses seleksi dengan mengabaikan beberapa *front* untuk tidak dilakukan perhitungan nilai *crowding distance* dan pengurutan berdasarkan nilai *crowding distnace*. *Front* yang dihitung nilai *crowding distance*-nya hanya *front* yang memiliki kemungkinan anggotanya sebagian masuk dan tidak masuk pada solusi yang diusulkan untuk iterasi berikutnya. Atau bisa dikatakan front yang dihitung nilai *crowding distance*-nya hanya front yang mengandung individu ke-*popSize* (*popSize* adalah parameter ukuran populasi).

Berdasarkan metode Jensen dan Fang juga dikembangkan kembali metode berbasis pencarian hirarki yang berhasil memperkecil waktu eksekusi algoritme NSGA-II (Bao et al., 2017). Tang, dkk (2008) mengklaim dapat mempercepat NSGA-II dengan memperbaiki "*Arena principle*" (Tang et al., 2008). Metode pengurutan *deductive* dan *climbing* juga digunakan untuk usaha mempersingkat waktu eksekusi *NSGA-II* (McClymont & Keedwell, 2012).

2.6 Metode Evaluasi

Metode evaluasi optimasi multi-objektif yang paling sederhana adalah dengan membandingkan nilai fitness yang dihasilkan, seperti yang dilakukan oleh (Hu et al., 2016; Liu et al., 2015). Perbandingan nilai fitness ini bisa diwakili oleh rerata dari nilai fitness. Karena optimasi multi-objektif memiliki lebih dari satu fitness, maka nilai-nilai fitness tersebut dapat diwakili oleh satu nilai yang merupakan bentuk vector dari beberapa nilai fitness tersebut (Schaffer, 1984). Selain perbandingan nilai fitness, waktu eksekusi juga menjadi bahan evaluasi performa dari algoritma optimasi multi-objektif.

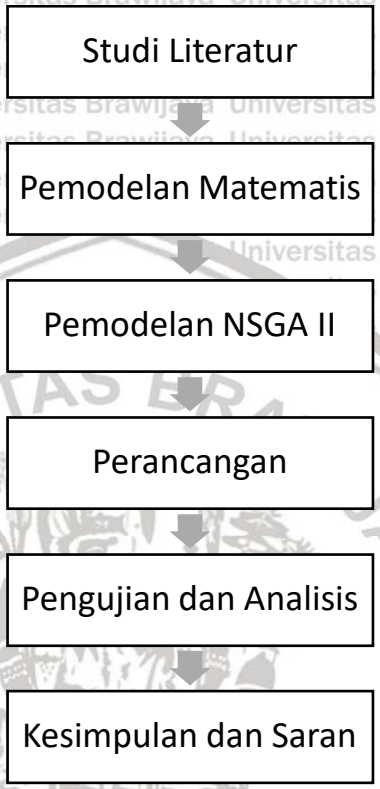
Selain metode evaluasi sederhana diatas, juga terdapat metode pengukuran yang lebih rumit yaitu *Generational Distance* (*GD*), *Inverted Generational Distance* (*IGD*), *Delta Measure* (*DM*), *Hypervolume Metric* (*HM*), dan *Inverse Hypervolume Metric* (*IHM*) (Mirjalili & Lewis, 2015). Namun, metode pengukuran tersebut memerlukan titik referensi sebagai *ground-*

truth. Sedangkan penelitian ini belum mempunyai *ground-truth,* sehingga lebih baik menggunakan metode evaluasi sederhana, yaitu menggunakan nilai fitness dan waktu eksekusi.

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan mengenai metodologi yang digunakan pada tesis ini. Bab ini terdiri dari metode penelitian, pemodelan matematis, pemodelan NSGA-II, metode pembanding untuk optimasi penentuan UKT, dan pengujian yang akan dilakukan pada penelitian ini.

3.1 Tahapan Penelitian



Gambar 3-1 Diagram alir tahapan penelitian

Tahapan penelitian yang digunakan pada penelitian ini dapat dilihat pada Gambar 3-1. Berikut adalah penjelasan dari diagram alir tahapan penelitian pada Gambar 3-1.

1. Studi Literatur
   Pada tahapan ini, penulis mencari dan mempelajari literatur tentang uang kuliah tunggal (UKT), Optimasi multi-objektif, *Non-dominated Sort Genetic Algorithm* II (NSGA-II). Literatur tentang UKT meliputi definisi UKT dan aturan mengenai UKT. Literatur tentang optimasi multi-objektif meliputi pengertian dan metode-metode optimasi multi-objektif. Literatur tentang NSGA-II meliputi langkah-langkah algoritme ini dan pemodelan-pemodelan NSGA-II.
2. Pemodelan Matematis
   Pada tahapan ini akan dibentuk model matematis tentang optimasi penentuan UKT.
3. Pemodelan NSGA-II
   Pemodelan NSGA-II terdiri dari penentuan representasi kromosom populasi, penentuan fungsi *fitness* dan *constraint* berdasarkan model matematis yang telah dibentuk, penentuan metode inisilaisasi, penentuan operator genetika (*crossover* dan mutasi), penentuan operator mutasi dan penentuan metode perbaikan (*repair*) kromosom.

4. Perancangan
   Perancangan dilakukan untuk mengaktualisasikan pemodelan matematis dan pemodelan NSGA-II yang sudah dibahas sebelumnya.
5. Pengujian dan Analisis
   Pengujian dilakukan untuk mengetahui apakah NSGA-II dapat mengoptimasi penentuan UKT.
6. Kesimpulan dan Saran
   Pada tahapan ini diberikan kesimpulan dari hasil pengujian dan analisis, serta saran untuk penelitian selanjutnya.

3.2 Pemodelan Matematis

Penentuan UKT mahasiswa didasarkan pada tingkat kemampuan finansialnya. Tingkat kemampuan finansial mahasiswa pada penelitian ini adalah penghasilan perbulan orang tua.

3.2.1 Parameter

Parameter-parameter dari penentuan UKT sebagai berikut:

$k = 1,2,3,4,5,6$	Index dari ketegori UKT
M	Jumlah seluruh mahasiswa
$m = 1,2,\dots M$	Index dari mahasiswa
S	Jumlah seluruh program studi
$s = 1,2,3,\dots S$	Index dari program studi
S_m	Program studi mahasiswa *m*
U_m	Besaran UKT mahasiswa m
U_{ks}	Besaran UKT untuk kategori *k* dalam program studi *s*
E_m	Kemampuan finansial mahasiswa *m*
$MaxE_{ks}$	Maksimal kemampuan finansial mahasiswa per kategori UKT *k* per program studi *s*
$MinE_{ks}$	Minimal kemampuan finansial mahasiswa per kategori UKT *k* per program studi *s*
H	Minimal proyeksi pengahasilan Fakultas
N_s	Jumlah mahasiswa diterima pada program studi *s*
$K_m = \{1,2,3,4,5,6\}$	Kategori UKT setiap mahasiswa *m*
NK_{ks}	Jumlah mahasiswa pada program studi *s* dan kategori *k*

3.2.2 Tujuan Optimasi

Tujuan optimasi yang pertama adalah minimal jarak antara kemampuan finansial mahasiswa dengan besaran UKT dapat diformulasikan pada Formula (2).

$$minimal \ \frac{\sum_{m=1}^{M} |U_m - E_m|}{M} \quad (2)$$

Tujuan optimasi yang kedua adalah maksimal dari proyeksi penghasilan fakultas yang didapatkan dari total dari besaran UKT mahasiswa. Tujuan optimasi kedua dapat diformulasikan pada Formula (3).

$$maksimal \sum_{m=1}^{M} U_m \quad (3)$$

3.2.3 Batasan (*Constraint*)

Batasan pertama adalah memastikan total besaran UKT mahasiswa tidak kurang dari minimal proyeksi penghasilan fakultas. Batasan pertama dapat diformulasikan pada Formula (4).

$$\sum_{m=1}^{M} U_m > H \quad (4)$$

Jumlah mahasiswa pada setiap kategori UKT diformulasikan pada Formula (5). Batasan kedua dan ketiga memastikan kategori I dan kategori II berjumlah minimal 5% dari jumlah mahasiswa diterima pada setiap program studi. Batasan kedua dan ketiga berturut-turut dapat dilihat pada Formula (6) dan (7). Formula (8) memastikan bahwa maksimal kemampuan finansial dari mahasiswa (E_m) dengan kategori *k* pada program studi *s* harus lebih besar dari minimal kemampuan finansial dari mahasiswa (E_m) dengan kategori *k-1* (kategori UKT dibawahnya).

$$NK_{ks} = \sum_{m=1}^{M} ((K_m = k) * (S_m = s)) \quad (5)$$

$$NK_{1s} \geq 5\% * N_s \quad (6)$$

$$NK_{2s} \geq 5\% * N_s \quad (7)$$

$$MinE_{ks} > MaxE_{k-1s} \quad (8)$$

3.3 Pemodelan NSGA-II

Langkah-langkah algoritme NSGA-II yang digunakan dapat dilihat pada Gambar 2-2. Proses inisialisasi adalah proses untuk membangkitkan populasi awal P_0 secara acak, dengan jumlah populasi sebesar *popSize*. Proses regenerasi menggunakan *crossover* dan mutasi. Proses regenerasi menghasilkan populasi baru sejumlah *popSize*, sehingga populasi P_t berjumlah 2**popSize*. Proses seleksi dilakukan dengan membandingkan *crowding distance* dari masing-masing kromosom.

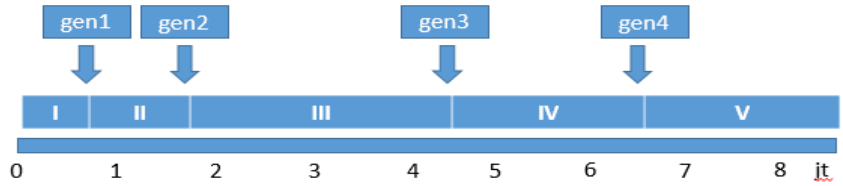
3.3.1 Representasi Kromosom

Seperti yang sudah dijelaskan pada tinjauan pustaka subbab 2.3.1, bahwa representasi kromosom adalah masalah utama dari algoritme genetika (GA) yang paling berpengaruh terhadap kinerja (performa) dari GA. Representasi kromosom merupakan pengkodean dari pemasalahan yang akan diselesaikan. Pada penelitian ini terdapat dua representasi kromosom yang akan dibandingkan. Representasi pertama (lebih singkat disebut K1) adalah representasi menggunakan bilangan bulat. Urutan gen pada kromosom mewakili mahasiswa, dan gen itu sendiri mewakili kategori UKT mahasiswa. Representasi kromosom dapat dilihat pada Gambar 3-2.

4	3	2	6	1	3	2	1	...	3
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*...*	*M*

Gambar 3-2 Representasi kromosom pertama

Representasi kromosom kedua (lebih singkat disebut K2) menggunakan representasi bilangan pecahan. Gen pada kromosom mewakili batas atas dari kemampuan finansial mahasiswa. Representasi kromosom untuk satu program studi dan dengan jumlah kategori UKT adalah 5 dapat dilihat pada Gambar 3-3. Sehingga untuk penentuan UKT mahasiswa dalam satu fakultas dengan 3 program studi representasi kromosomnya dapat dilihat pada Gambar 3-4.



Gambar 3-3 Ilustrasi representasi kromosom kedua

0.75	1.8	4.5	6.7	0.5	1.2	3	5	0.5	1.8	3.5	5.4
Program Studi A				Program Studi B				Program Studi C			

Gambar 3-4 Representasi kromosom kedua untuk 3 program studi

3.3.2 Fungsi Fitness dan Constraints

Fungsi *fitness* yang digunakan dapat dilihat pada Formula (9)-(11). Fungsi *fitness* (9) dan (10) adalah bentuk maksimal dari formula (2). Fungsi *fitness* (9) dan (10) bertujuan untuk meminimalkan jarak antara kemampuan finansial mahasiswa dengan besaran UKT yang akan diterima. Dua persamaan maksimasi dari Formula (2) tersebut, akan dibandingkan hasilnya pada tahap pengujian. Fungsi *fitness* (11) adalah pengembangan dari formula (3). Fungsi fitness (11) bertujuan untuk memaksimalkan penghasilan dari fakultas yang diperoleh dari jumlah total besaran UKT setiap mahasiswa dalam satu fakultas.

$$\max \quad f_1 = \frac{M}{\sum_{m=1}^{M} |U_m - E_m|} \tag{9}$$

$$\max \quad f_1 = -\frac{\sum_{m=1}^{M} |U_m - E_m|}{M} \tag{10}$$

$$max \; f_2 = \sum_{m=1}^{M} U_m \tag{11}$$

Fungsi *constraint* (12) adalah pengembangan dari Formula (4) yang membatasi total besaran ukt mahasiswa (U_m) dari suatu fakultas tidak kurang dari minimal proyeksi penghasilan fakultas (H) tersebut. Fungsi *constraint* (13) dan (14) secara berturut-turut adalah pengembangan dari Formula (6) dan (7), yaitu yang digunakan untuk membatasi bahwa jumlah mahasiswa yang memperoleh kategori I dan II masing-masing tidak kurang dari 5% dari jumlah mahasiswa dalam satu program studi. Fungsi *constraint* (15) bertujuan untuk

membatasi bahwa nilai minimal kemampuan finansial mahasiswa (E_m) pada kategori *k* harus lebih besar dari pada nilai maksimal dari kemampuan finansial mahasiswa pada kategori *k-1*.

$$g = \sum_{m=1}^{M} U_m - H > 0 \quad (12)$$

$$h = NK_{1s} - (5\% * N_s) \geq 0 \quad (13)$$

$$i = NK_{2s} - (5\% * N_s) \geq 0 \quad (14)$$

$$j = MinE_{ks} - MaxE_{k-1s} > 0 \quad (15)$$

Dari pendefinisian fungsi *fitness* (10) atau (11) dan (12), serta fungsi *constraint* (12)-(15) dapat diformulasikan fungsi *fitness* secara keseluruhan dapat dilihat pada formula (16).

$$max\ F = (f_1, f_2)^T \quad (16)$$
$$untuk\ g > 0$$
$$h \geq 0$$
$$i \geq 0$$
$$j > 0$$

3.3.3 Inisialisasi

Proses inisialisasi untuk representasi kromosom pertama hanya membangkitkan bilangan acak antara 1 sampai 6 untuk setiap gen (*allele*). Sedangkan untuk representasi kromosom kedua menggunakan dua metode inisialisasi khusus yaitu memanfaatkan Pseudo-Random dan Quasi-Random Sequence. Metode inisialisasi yang memanfaatkan Pseudo-Random kami beri nama inisialisasi *NCPR-Seq* (*Normalized Cumulative Pseudo Random Sequence*). Sedangkan inisialisasi yang memanfaatkan Quasi-Random Sequence kami beri nama *NCQPR-Seq* (*Normalized Cumulative Quasi-Random Sequence And Pseudo Random Sequence*).

Metode inisialisasi *NCPR-Seq* untuk satu program studi dilakukan sebagai berikut:

1. Membangkitkan sejumlah *K* bilangan secara acak (*Pseudo Random / PR*).
2. Kemudian dibentuk deret kumulatif dari *K* bilangan tersebut
3. Deret kumulatif yang terbentuk kemudian dinormalisasi . Proses normalisasi dilakukan dengan cara membagi nilai dari deret kumulatif dengan nilai tertingginya (nilai total bilangan *K* random).
4. Mengalikan deret kumulatif yang sudah dinormalisasi dengan U_{6s} yaitu nilai UKT tertinggi pada program studi tersebut.

Secara umum langkah dari *inisialisasi NCQPR-Seq* sama seperti dengan *NCPR-Seq*. Metode inisialisasi *NCQPR-Seq* untuk satu program studi dilakukan sebagai berikut:

1. Membangkitkan sejumlah bilangan dengan metode *Quasi Random Sequence* (*QRS*)
2. Memilih sejumlah *K* bilangan secara acak (*Pseudo Random / PR*) dari sejumlah bilangan hasil dari langkah 1.
3. Kemudian dibentuk deret kumulatif dari *K* bilangan tersebut.
4. Deret kumulatif yang terbentuk kemudian dinormalisasi. Proses normalisasi dilakukan dengan cara membagi nilai dari deret kumulatif dengan nilai tertingginya (nilai total bilangan *K* random).

5. Mengalikan deret kumulatif yang sudah dinormalisasi dengan U_{6s} yaitu nilai UKT tertinggi pada program studi tersebut.

3.3.4 Rekombinasi (*Genetic Operators*)

Proses rekombinasi menggunakan metode *crossover* menggunakan *one-cut-poin crossover* (*OC*) dan *uniform crossover* (*UC*). Sedangkan metode mutasi menggunakan *simple-random mutation (SM)* dan *random-exchange mutation* (*EM*). Dari dua metode *crossover* dan 2 metode mutasi tersebut, maka didapatkan 4 kombinasi metode rekombinasi yaitu :

1. *OC-SM* (*one-cut-poin crossover* dan *simple-random mutation*)
2. *OC-EM* (*one-cut-poin crossover* dan *random-exchange mutation*)
3. *UC-SM* (*uniform crossover* dan *simple-random mutation*)
4. *UC-EM* (*uniform crossover* dan *random-exchange mutation*)

Pada penelitian ini juga digunakan kombinasi acak dari masing-masing metode *crossover* dan mutasi yaitu yang disingkat *RC-RM*. Jadi pada penelitian ini akan diteliti 5 kombinasi metode rekomendasi, dan akan disimpulkan metode rekomendasi yang cocok untuk masing-masing representasi kromosom. Pada penelitian ini juga akan dilakukan penelitan tentang penggunaan metode *repair* (perbaikan). Metode repair ini dilakukan dengan merubah konfigurasi kromosom yang melanggar batasan (*constraint*).

3.3.5 Metode Perbaikan (*Repair*)

Seperti yang sudah dijelaskan sebelumnya pada subbab 2.3.6, bahwa metode perbaikan ini bertujuan untuk memperbaiki konfigurasi kromosom yang melanggar *constraint*. Maka diperlukan mekanisme untuk mencari gen (*allele*) yang melanggar *constraint*, kemudian memperbaikinya. Dikarenakan jumlah *constraint* yang banyak, maka diperlukan penyederhanaan proses pencarian gen (allele) yang melanggar *constraint*. Maka proses metode perbaikan yang digunakan adalah sebagai berikut:

1. Metode perbaikan untuk representasi kromosom pertama, dilakukan dengan mencari gen yang mempunyai nilai UKT (*Um*) yang tidak sesuai dengan kemampuan finansialnya (*Em*). Kemudian menggantinya dengan kategori UKT yang sesuai.
2. Metode perbaikan untuk representasi kromosom kedua, dilakukan dengan mencari nilai gen ke-i yang lebih besar dari nilai ke-$i+1$, ketika $i+1 \,\%\, (mod)(K-1) \neq 0$, atau bisa dikatakan *i* dan *i+1* berada dalam prodi *s* yang sama. Kemudian menukar nilai kedua gen tersebut.

3.3.6 Metode Seleksi

Metode seleksi menggunakan metode yang digunakan pada algoritme NSGA-II, yaitu menggunakan metode *elitism*. Ranking pada *elitism* didapatkan dari metode *non-dominated sort* dan pengurutan berdasarkan *crowding distance*, seperti yang sudah dijelaskan pada subbab 2.4.2. Pada tesis ini akan dilakukan modifikasi NSGA-II yaitu pada metode *non-dominated sort* dan pengurutan berdasarkan *crowding distance*. Selain itu metode seleksi selain *elitism* juga akan digunakan sebagai metode pembanding.

3.4 Perancangan

Perancangan dalam penelitian ini meliputi perhitungan manual untuk kedua representasi kromosom yang diusulkan, perancangan algoritme, dan perancangan pengujian. Perancangan algoritme terdiri dari perancangan metode inisialisasi yang diusulkan untuk representasi kromosom kedua, operator genetika, metode repair dan modifikasi metode

seleksi yang diusulkan. Sedangkan perancangan pengujian akan menunjukkan rancangan-rancangan tabel pengujian yang diperlukan.

3.5 Pengujian dan Analisis

Percobaan akan dilakukan sebanyak 20 kali untuk setiap pengujian. Tujuan dari pengulangan ini adalah agar mendapatkan nilai rata-rata dan maksimum dari indikator yang digunakan. Pengujian dalam penelitian ini adalah sebagai berikut:

1. Pengujian representasi kromosom dan pengaruh jumlah populasi
   Pengujian ini dilakukan untuk mengetahui jumlah populasi yang menghasilkan hasil yang paling optimal untuk permasalahan optimasi penentuan UKT. Pengujian berdasarkan jumlah populasi ini dilakukan untuk kedua representasi kromosom. Sehingga akan diketahui representasi kromosom yang lebih efisien, yaitu representasi yang menghasilkan nilai fitness terbaik dengan jumlah populasi yang lebih sedikit. Pada pengujian ini juga akan dibandingkan hasil antara kedua fungsi *fitness 1*.
2. Pengujian representasi kromosom dan pengaruh maksimal generasi
   Pengujian ini dilakukan untuk mengetahui jumlah maksimal generasi yang diperlukan untuk mencapai konvergensi. Pada tahap pengujian ini akan digunakan jumlah populasi sesuai dengan hasil pada pengujian jumlah populasi. Sedangkan fungsi *fitness* 1 yang digunakan adalah fungsi fitness yang lebih baik pada pengujian jumlah populasi. Pengujian berdasarkan maksimal generasi ini juga dilakukan untuk kedua representasi kromosom.
3. Pengujian metode rekombinasi
   Pengujian ini dilakukan untuk mengetahui metode rekombinasi yang paling efisien untuk masing-masing bentuk kromosom yang digunakan pada algoritme *NSGA-II* untuk optimasi penentuan UKT. Seperti yang dijelaskan pada subbab 3.3.4 bahwa pada penelitian ini akan diuji lima kombinasi metode rekombinasi.
4. Pengujian metode *repair*
   Pengujian ini bertujuan untuk mengetahui apakah dengan penambahan metode *repair* dapat meningkatkan performa dari NSGA-II dalam optimasi penentuan UKT. Pengujian metode repair ini juga akan dilakukan pada kedua bentuk kromosom. Pada pengujian ini parameter NSGA-II yang digunakan adalah jumlah populasi, maksimal generasi, metode rekombinasi, dan bentuk fungsi *fitness* 1 yang terbaik dari hasil pengujian sebelumnya.
5. Pengujian metode seleksi
   Pada pengujian metode seleksi ini performa NSGA-II yang dimodifikasi akan dibandingkan dengan algoritme NSGA-II dan algoritme genetika *non-elitism* (metode *proportional roulette wheel* dan metode *binary tournament*).
6. Analisis detail hasil penetapan UKT
   Pada tahap ini akan dilakukan analisis terhadap hasil penetapan UKT oleh NSGA-II dengan membandingkan dengan hasil penetapan UKT menggunakan metode sederhana. Metode sederhana yang dimaksud adalah dengan menggunakan operator logika sederhana yang membandingkan antara E_m (kemampuan finansial mahasiswa) dengan nilai dari besaran UKT (U_{ks}).

3.6 Kesimpulan dan Saran

Pengambilan kesimpulan didasarkan pada hasil pengujian dilakukan yaitu representasi kromosom, bentuk fungsi *fitness*, jumlah populasi, dan operator genetika yang menghasilkan

hasil yang terbaik dan efisien. Kesimpulan juga dilakukan pada penggunaan metode perbaikan. Yang terakhir adalah kesimpulan tentang metode yang diusulkan, apakah lebih baik dari metode pembanding untuk menyelesaikan permasalahan optimasi penentuan UKT?. Pemberian saran dalam penelitian ini juga didasarkan pada kekurangan pada metode yang diusulkan berdasarkan hasil pengujian.

BAB 4 PERANCANGAN

4.1 Perhitungan Manual

Parameter yang paling berpengaruh untuk optimasi penentuan UKT adalah kemampuan finansial mahasiswa (E_m), besaran UKT per kategori dan per prodi (U_{ks}), dan minimal proyeksi penghasilan fakultas (H). Jumlah mahasiswa (M) yang digunakan pada perhitungan manual ini adalah 10 mahasiswa. Nilai H yang digunakan untuk perhitungan manual ini adalah 30 juta. Parameter untuk optimasi penentuan UKT secara lengkap dapat dilihat pada subbab 3.2.1. Proses perhitungan berbeda untuk kedua representasi kromosom. Oleh karena itu, dalam perhitungan manual ini dibagi dua sesuai dengan dua rencana representasi kromosom yang dapat dilihat pada Gambar 3-2 dan Gambar 3-4.

Contoh data kemampuan finansial mahasiswa yang digunakan untuk perhitungan manual ini adalah data kemampuan finansial untuk 10 mahasiswa seperti yang dapat dilihat pada Tabel 4-1. Pada Tabel 4-1 nilai E_m (kemampuan finansial) adalah dalam juta rupiah, dan m menyatakan index mahasiswa. Kesepuluh mahasiswa ini diasumsikan adalah mahasiswa program studi yang sama. Atau dengan kata lain pada perhitungan manual ini program studi yang digunakan hanya satu yaitu $s=1$. Contoh data kategori UKT beserta besarannya yang digunakan dalam perhitungan manual ini dapat dilihat pada Tabel 4-2.

Tabel 4-1 Contoh data kemampuan finansial mahasiswa (Em)

M	1	2	3	4	5	6	7	8	9	10
E_m	6	5	3.1	2.5	0.4	1.6	0.8	3.8	4.1	2.9

Tabel 4-2 Contoh data besaran UKT per kategori

K	1	2	3	4
U_{ks}	1	2.5	3.5	5

4.1.1 Representasi Kromosom Pertama

4	3	2	1	2	4	2	1	3	4

Gambar 4-1 Kromosom pada perhitungan manual

Tabel 4-3 Bantuan perhitungan

M	1	2	3	4	5	6	7	8	9	10	Σ
E_m	6	5	3.1	2.5	0.4	1.6	0.8	3.8	4.1	2.9	
U_m	5	3.5	2.5	1	2.5	5	2.5	1	3.5	5	31.5
$\lvert U_m - E_m \rvert$	1	2.5	0.6	1.5	2.1	3.4	1.7	2.8	0.6	2.1	18.3

Representasi kromosom pertama dapat dilihat pada Gambar 3-2. Pada representasi tersebut gen merepresentasikan kategori UKT dan urutan gen adalah urutan (indeks) mahasiswa. Ketika kromosom hasil proses inisialisasi seperti yang dapat dilihat pada Tabel 4-2. Untuk mempermudah perhitungan manual, maka dibuat table pembantu perhitungan yang dapat dilihat pada Tabel 4-3. Perhitungan *fitness* dan *constrain*-nya adalah sebagai berikut:

1. *Fitness* 1

Seperti yang telah dijelaskan pada subbab 3.3.2, bahwa pada penelitian ini terdapat dua bentuk fungsi *fitness* 1. Perhitungan untuk masing-masing bentuk fungsi *fitness* 1 sebagai berikut:

- Bentuk pertama

  Perhitungan fungsi *fitness* 1 bentuk pertama sesuai dengan Formula (9). Nilai *M* adalah jumlah mahasiswa yaitu 10. Dan nilai $\sum_{m=1}^{M}|U_m - E_m|$ adalah jumlah dari selisih antara besaran UKT (U_m) dan kemampuan finansial mahasiswa (E_m) yaitu 18.3 (seperti yang dapat dilihat pada Tabel 4-3). Sehingga nilai *fitness* 1 bentuk pertama adalah $\frac{M}{\sum_{m=1}^{M}|U_m - E_m|} = \frac{10}{18.3} = 0.55.$

- Bentuk kedua

  Perhitungan fungsi *fitness* 1 bentuk kedua sesuai dengan Formula (10). Nilai *M* adalah jumlah mahasiswa yaitu 10. Dan nilai $\sum_{m=1}^{M}|U_m - E_m|$ adalah jumlah dari selisih antara besaran UKT (U_m) dan kemampuan finansial mahasiswa (E_m) yaitu 18.3 (seperti yang dapat dilihat pada Tabel 4-3). Sehingga nilai *fitness* 1 bentuk kedua adalah $-\frac{\sum_{m=1}^{M}|U_m - E_m|}{M} = -\frac{18.3}{10} = -1.83.$

2. *Fitness* 2

Nilai fungsi *fitness* 2 adalah jumlah dari seluruh besaran UKT setiap mahasiswa. Berdasarkan Tabel 4-3, maka nilai fungsi *fitness* 2 adalah 31.5.

3. *Constraint* 1

Fungsi *constraint* 1 adalah fungsi yang membatasi total besaran ukt mahasiswa (U_m) dari suatu fakultas tidak kurang dari minimal proyeksi penghasilan fakultas (*H*) tersebut. Jika kita asumsikan nilai *H* dari fakultas untuk 10 mahasiswa adalah 30 juta. Jika total U_m seperti pada Tabel 4-3 adalah 31.5, maka nilai dari fungsi *constraint* 1 adalah 31.5 – 30 = 1.5. Karena nilai dari *constraint* 1 lebih dari 0, maka kromosom ini dinyatakan tidak melanggar *constraint* 1. Atau dengan kata lain solusi (kromosom) ini dianggap *feasible* (layak).

4. *Constraint* 2 dan 3

Constraint 2 dan 3 membatasi jumlah mahasiswa yang memperoleh kategori I (NK_{1s}) dan II (NK_{2s}) masing-masing tidak kurang dari 5% dari jumlah mahasiswa (N_s) dalam satu program studi. Seperti yang dijelaskan sebelumnya bahwa data mahasiswa yang digunakan dalam perhitungan manual ini adalah dalam satu program studi yang sama, sehingga nilai N_s sama dengan M yaitu 10. Jadi 5% dari N_s adalah 0.5, jika dibulatkan maka nilai $5\% * N_s$ adalah 1. Jika diamati pada Gambar 4-1 dan atau Tabel 4-3, maka mahasiswa yang mendapat kategori 1 (NK_{1s}) adalah 2 orang. Sedangkan yang mendapat kategori 2 (NK_{2s}) adalah 3 orang. Sehingga nilai *constraint 2* adalah $NK_{1s} - (5\% * N_s) = 2 - 1 = 1$. Sedangkan nilai *constraint 3* adalah $NK_{2s} - (5\% * N_s) = 3 - 1 = 2$.

5. *Constraint* 4

Constraint 4 membatasi nilai minimal kemampuan finansial mahasiswa (E_m) pada kategori *k* harus lebih besar dari pada nilai maksimal dari kemampuan finansial mahasiswa pada kategori *k-1*. Untuk mempermudah mencermati *Constraint* 4, maka dibuatlah Tabel 4-4 yang dapat membantu. Pada Tabel 4-4 dapat dilihat bahwa minimal kemampuan finansial pada kategori 2 ($MinE_{2s}$) adalah 0.4 juta atau 400.000. Sedangkan maksimal kemampuan finansial pada kategori 1 ($MaxE_{1s}$) adalah 3.8 juta, maka solusi (kromosom) ini dinyatakan melanggar *constraint 4*.

Tabel 4-4 Kemampuan finansial mahasiswa (E_m) dan kategori UKT-nya (k_m)

M	**4**	**8**	**5**	**7**	**3**	**9**	**2**	**6**	**10**	**1**
E_m	2.5	3.8	0.4	0.8	3.1	4.1	5	1.6	2.9	6
k_m	1	1	2	2	2	3	3	4	4	4

4.1.2 Representasi Kromosom Kedua

Representasi kromosom kedua dapat dilihat pada Gambar 3-4. Untuk perhitungan manual, diberikan contoh representasi kromosom kedua yang dapat dilihat pada Gambar 4-2. Seperti yang sudah dijelaskan pada subbab 3.3.1, gen (angka) pada Gambar 4-2 adalah batas kemampuan finansial mahasiswa dari dua kategori UKT. Angka 1.2 dalam Gambar 4-2 berarti mahasiswa yang memiliki kemampuan finansial kurang dari 1.2 juta termasuk kedalam kategori I (satu). Mahasiswa yang masuk kategori 2 adalah mahasiswa yang kemampuan finansialnya lebih dari atau sama dengan 1.2 juta dan kurang dari 2.6 ($1.2 \leq Em < 2.6$).

1.2	2.6	4.2

Gambar 4-2 Contoh representasi kromosom kedua

Tabel 4-5 Perhitungan *fitness* dan *constraint* pada representasi kromosom kedua

M	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
E_m	6	5	3.1	2.5	0.4	1.6	0.8	3.8	4.1	2.9	
Logika	$E_m \geq 4.2$	$E_m \geq 4.2$	$2.6 \leq E_m < 4.2$	$1.2 \leq E_m < 2.6$	$E_m < 1.2$	$1.2 \leq E_m < 2.6$	$E_m < 1.2$	$2.6 \leq E_m < 4.2$	$2.6 \leq E_m < 4.2$	$2.6 \leq E_m < 4.2$	**Σ**
k_m	4	4	3	2	1	2	1	3	3	3	
U_m	5	5	3.5	2.5	1	2.5	1	3.5	3.5	3.5	**31**
$\lvert U_m\text{-}E_m \rvert$	1	0	0.4	0	0.6	0.9	0.2	0.3	0.6	0.6	**4.6**

Karena representasi kromosom kedua hanya berisi batas-batas kemampuan finansial saja, sedangkan nilai *fitness 1* dan *fitness 2* didapatkan dari total besaran UKT dan total kemampuan finansial mahasiswa. Sehingga diperlukan mekanisme agar kromosom tersebut menjadi nilai besaran UKT per mahasiswa. Mekanisme yang diperlukan adalah dengan menggunakan operator logika sederhana yang membandingkan antara E_m (kemampuan finansial mahasiswa) dengan nilai batas E_m seperti yang terlihat pada kromosom. Mekanisme tersebut kami tuangkan dalam Tabel 4-5. Dalam Tabel 4-5 pada baris ketiga diberikan bentuk logika dari perbandingan Em dengan batas yang ada pada kromosom. Dari baris ketiga pada

Tabel 4-5 tersebut, maka dapat dicari k_m kategori UKT per mahasiswa seperti yang dapat dilihat pada Tabel 4-5 kolom ke-4. Karena nilai k_m sudah ada. Dengan pemetaan menggunakan Tabel 4-4, maka didapatkan nilai U_m (besaran UKT per mahasiswa), seperti yang dapat dilihat pada Tabel 4-5 kolom ke-5.

1. Perhitungan Nilai *Fitness* 1

Pada representasi kromosom kedua juga diberikan contoh perhitungan manual untuk kedua bentuk *fitness* 1. Perhitungan nilai fitness 1 ini sangat terbantu dengan adanya tabel bantuan yaitu Tabel 4-5. Perhitungan untuk masing-masing bentuk fungsi *fitness* 1 adalah sebagai berikut:

- *Fitness* 1 bentuk pertama

  Perhitungan fungsi *fitness* 1 bentuk pertama sesuai dengan Formula (9). Nilai *M* adalah jumlah mahasiswa yaitu 10. Dan nilai $\sum_{m=1}^{M}|U_m - E_m|$, seperti yang dapat dilihat pada Tabel 4-5 adalah 4.6. Sehingga nilai *fitness* 1 bentuk pertama adalah $\frac{M}{\sum_{m=1}^{M}|U_m - E_m|} = \frac{10}{4.6} = 2.174.$

- *Fitness* 1 bentuk kedua

  Perhitungan fungsi *fitness* 1 bentuk kedua sesuai dengan Formula (10). Nilai *M* adalah jumlah mahasiswa yaitu 10. Dan nilai $\sum_{m=1}^{M}|U_m - E_m|$, seperti yang dapat dilihat pada Tabel 4-5 adalah 4.6. Sehingga nilai *fitness* 1 bentuk kedua adalah $-\frac{\sum_{m=1}^{M}|U_m - E_m|}{M} = -\frac{4.6}{10} = -0.46.$

2. Perhitungan Nilai Fitness 2

Nilai fungsi *fitness* 2 adalah jumlah dari seluruh besaran UKT setiap mahasiswa. Berdasarkan Tabel 4-5, maka nilai fungsi *fitness* 2 adalah 31.

3. Constraint 1

Fungsi constraint 1 adalah fungsi yang membatasi total besaran ukt mahasiswa (Um) dari suatu fakultas tidak kurang dari minimal proyeksi penghasilan fakultas (H) tersebut. Jika kita asumsikan nilai *H* dari fakultas untuk 10 mahasiswa adalah 30 juta. Jika total U_m seperti pada Tabel 4-5 adalah 31, maka nilai dari fungsi *constraint* 1 adalah 31 – 30 = 1. Karena nilai dari constraint 1 lebih dari 0, maka kromosom ini dinyatakan tidak melanggar *constraint* 1. Atau dengan kata lain solusi (kromosom) ini dianggap *feasible* (layak).

4. Constraint 2 dan 3

Constraint 2 dan 3 membatasi jumlah mahasiswa yang memperoleh kategori I (NK_{1s}) dan II (NK_{2s}) masing-masing tidak kurang dari 5% dari jumlah mahasiswa (N_s) dalam satu program studi. Seperti yang dijelaskan sebelumnya bahwa data mahasiswa yang digunakan dalam perhitungan manual ini adalah dalam satu program studi yang sama, sehingga nilai N_s sama dengan M yaitu 10. Jadi 5% dari N_s adalah 0.5, jika dibulatkan maka nilai $5\% * N_s$ adalah 1. Jika diamati pada Gambar 4-5, maka mahasiswa yang mendapat kategori 1 (NK_{1s}) adalah 2 orang. Sedangkan yang mendapat kategori 2(NK_{2s}) adalah 2 orang. Sehingga nilai *constraint 2* adalah $NK_{1s} - (5\% * N_s) = 2 - 1 = 1$. Sedangkan nilai *constraint 3* adalah $NK_{2s} - (5\% * N_s) = 2 - 1 = 1$. Karena nilai dari constraint 2 dan 3 lebih dari 0, maka kromosom ini adalah solusi yang *feasible* (layak).

5. Constraint 4

Constraint 4 membatasi nilai minimal kemampuan finansial mahasiswa (E_m) pada kategori *k* harus lebih besar dari pada nilai maksimal dari kemampuan finansial mahasiswa pada kategori *k-1*. Kromosom yang terbentuk (yang dapat dilihat pada Gambar 4-2) sudah menyatakan bahwa kromosom ini tidak akan melanggar *constraint 4*, karena nilai gen kedua lebih besar dari gen pertama dan gen ketiga lebih besar dari gen kedua.

6. Inisialisasi *NCPR-Seq*

Perhitungan manual *NCPR-Seq* dapat dilihat pada Tabel 4-6. Untuk mempermudah perhitungan manual, langkah-langkah metode inisialisasi *NCPR-Seq* untuk satu program studi dituliskan kembali sebagai berikut:

a) Membangkitkan sejumlah *K* bilangan secara acak (*Pseudo Random / PR*), untuk selanjutnya disebut *bilAcak*. Nilai *K* pada perhitungan manual ini adalah 4. Hasil dari pembangkitan ini dapat dilihat pada Tabel 4-6, pada kolom kedua yaitu kolom *pseudo-random*.

b) Kemudian dibentuk deret kumulatif dari *bilAcak*. Deret kumulatif dihasilkan dengan cara menambahkan bilangan kumulatif sebelumnya dengan *bilAcak* saat ini. Bilangan kumulatif pertama adalah hasil kumulatif sebelumnya yaitu 0 ditambah dengan *bilAcak* pertama yaitu 0.3486604. Bilangan kumulatif kedua adalah hasil kumulatif sebelumnya yaitu 0.3486604 ditambahkan dengan *bilAcak* kedua yaitu 0.6282546, sehingga hasilnya adalah 0.976915. Langkah ini dilakukan lagi sampai bilangan kumulatif ke-4 seperti yang dapat dilihat pada Tabel 4-6 kolom deret kumulatif.

c) Deret kumulatif yang terbentuk kemudian dinormalisasi. Proses normalisasi dilakukan dengan cara membagi nilai dari deret kumulatif dengan nilai tertingginya (nilai total bilangan *K* random). Hasil normalisasi dapat dilihat pada pada Tabel 4-6 kolom normalisasi.

d) Mengalikan deret kumulatif yang sudah dinormalisasi dengan U_{4s} yaitu nilai UKT tertinggi pada program studi tersebut sebesar 5 Juta (sesuai pada Tabel 4-6). Hasil perkalian sampai dengan *K-1* inilah yang menjadi kromosom (nilai dari gen-gen yang ada pada kromosom) yaitu yang bisa dilihat pada Tabel 4-6 kolom kromosom.

Tabel 4-6 Perhitungan manual NCPR-Seq

K	Pseudo-Random	Deret Kumulatif	Normalisasi	Kromosom
1	0.3486604	0.3486604	0.159621012	**0.79810506**
2	0.6282546	0.976915	0.447243682	**2.236218408**
3	0.6893976	1.6663126	0.762858367	**3.814291837**
4	0.5179888	2.1843014	1	5

7. Inisialisasi *NCQPR-Seq*

Perhitungan manual NCQPR-Seq dapat dilihat pada Tabel 4-7. Untuk mempermudah perhitungan manual, langkah-langkah metode inisialisasi Inisialisasi *NCQPR-Seq* untuk satu program studi dituliskan kembali sebagai berikut:

a) Membangkitkan sejumlah bilangan dengan metode *Quasi Random Sequence* (*QRS*)

b) Memilih sejumlah *K* bilangan secara acak (*Pseudo Random / PR*) dari sejumlah bilangan hasil dari langkah a). Hasil langkah ini selanjutnya kita sebut sebagai *QRS-PR* seperti yang dapat dilihat pada Tabel 4-7 pada kolom *QRS-PR*.

c) Kemudian dibentuk deret kumulatif dari bilangan *QRS-PR*. Langkah pembentukan deret kumulatif ini sama dengan langkah yang telah dijelaskan pada metode Inisialisasi *NCPR-Seq.* Hasil dari deret kumulatif ini dapat dilihat pada Tabel 4-7 kolom deret kumulatif.

d) Deret kumulatif yang terbentuk kemudian dinormalisasi. Proses normalisasi dilakukan dengan cara membagi nilai dari deret kumulatif dengan nilai tertingginya. Hasil proses normalisasi dapat dilihat pada Tabel 4-7 kolom normalisasi.

e) Mengalikan deret kumulatif yang sudah dinormalisasi dengan U_{4s} yaitu nilai UKT tertinggi pada program studi tersebut adalah 5 juta (sesuai pada Tabel 4-7). Hasil perkalian ini dapat dilihat pada Tabel 4-7 kolom kromosom. Nilai yang digunakan sebagai gen pada kromosom adalah *K-1* (tiga) nilai yang terdapat pada kolom kromosom tersebut.

Tabel 4-7 Perhitungan manual Inisialisasi NCQPR-Seq

K	QRS+PR	Deret Kumulatif	Normalisasi	Kromsosom
1	0.375	0.375	0.26087	**1.304348**
2	0.3125	0.6875	0.478261	**2.391304**
3	0.71875	1.40625	0.978261	**4.891304**
4	0.03125	1.4375	1	5

4.2 Perancangan Algoritme

Seperti yang telah dijelaskan pada subbab 3.3, langkah-langkah algoritme NSGA-II yang digunakan pada penelitian (Deb et al., 2002) yaitu yang dapat dilihat pada Gambar 2-2. Pada perancangan algoritme ini hanya akan dijelaskan mengenai pengembangan algoritme NSGA-II untuk optimasi penentuan UKT yaitu pada proses inisialisasi kromosom, metode rekombinasi yang digunakan, dan metode perbaikan kromosom.

4.2.1 Inisialisasi Kromosom

Proses inisialisasi untuk representasi kromosom pertama hanya membangkitkan bilangan acak antara 1 sampai 6 untuk setiap gen (*allele*). Sedangkan untuk representasi kromosom kedua menggunakan dua metode inisialisasi khusus yaitu *NCPR-Seq* (*Normalized Cumulative Pseudo Random Sequence*) dan *NCQPR-Seq* (*Normalized Cumulative Quasi-Random Sequence And Pseudo Random Sequence*).

Metode inisialisasi *NCPR-Seq* untuk satu program studi dilakukan dengan membangkitkan sejumlah *K* bilangan secara random (*Pseudo Random* yang dapat disingkat *PR*). Kemudian dibentuk deret kumulatif dari *K* bilangan tersebut berdasarkan *K* bilangan random yang telah dihasilkan dihasilkan. Deret kumulatif yang terbentuk kemudian dinormalisasi. Proses normalisasi dilakukan dengan cara membagi nilai dari deret kumulatif dengan nilai tertingginya (nilai total bilangan *K* random). Proses terakhir adalah dengan mengalikan deret kumulatif yang sudah dinormalisasi dengan U_{6s} yaitu nilai UKT tertinggi pada program studi tersebut. *NCPR-Seq* yang digunakan untuk membentuk kromosom hanya sejumlah *K-1*. Untuk membentuk kromosom, proses pembangkitan *NCPR-Seq* tersebut

diulang sejumlah *S* program studi. Sehingga panjang kromosom adalah *S* * (*K-1*). Langkah-langkah proses inisialisasi *NCPR-Seq* dapat dilihat pada Gambar 4-3.

```
1   Procedure InisialisasiNCPR-Seq
2   begin
3       for (s=0; s<=S; s++)
4           tmpCum = 0
5           for (k=0; k<=K; k++)
6               tmpCum += random()
7               cumSeq[k] = tmpCum
8           end for
9           for (k=0;k<K;k++)
10              Kromosom[s*(K-1)+k]= U6s * (cumSeq[k] / tmpCum)
11          end for
12      end for
13  end
```

Gambar 4-3 Langkah-langkah inisialisasi NCPR-Seq

Secara umum langkah dari inisialisasi *NCQPR-Seq* sama seperti dengan *NCPR-Seq*. Langkah-langkah proses inisialisasi *NCQPR-Seq* dapat dilihat pada Gambar 4-4. Perbedaan *NCQPR-Seq* dengan *NCPR-Seq* adalah *NCQPR-Seq* dilakukan dengan memanfaatkan *Quasi-Random Sequence* (*QRS*) yang dikombinasikan dengan *Pseudo-Random* (*PR*). Metode yang digunakan untuk membangkitkan QRS adalah menggunakan metode Sobol yang dihasilkan oleh (Joe & Kuo, 2008). *PR* digunakan untuk menentukan index *QRS* yang dipilih untuk menjadi kandidat deret. Kombinasi *QRS* dan *PR* ini menggantikan proses pembangkitan *K* bilangan yang sebelumnya hanya menggunakan *PR*. Langkah-langkah prosedur pembangkitan deret QRS-PR (prosedur BangkitkanQRS-PR-Seq pada Gambar 4-4) dapat dilihat pada Gambar 4-5.

```
1   Procedure InisialisasiNCQPR-Seq
2   begin
3       for (s=0; s<=S; s++)
4           tmpCum = 0
5           QRN-PR = BangkitkanQRS-PR-Seq(K)
6           for (k=0; k<=K; k++)
7               tmpCum += QRS-PR[k]
8               cumSeq[k] = tmpCum
9           end for
10          for (k=0;k<K;k++)
11              Kromosom[s*(K-1)+k]= U6s * (cumSeq[k] / tmpCum)
12          end for
13      end for
14  end
```

Gambar 4-4 Metode inisialisasi NCQPR-Seq

```
1   Procedure BangkitkanQRS-PR-Seq (Jum)
2   begin
3         sob = sobolSequence()
4         indSob = 0
5         for (i=0; i<=Jum; i++)
6               indSob += random() *10000 % 20
7               res[i] = sob[indSob]
8         end for
9         return res
10  end
```

Gambar 4-5 Langkah-Langkah membangkitkan deret QRS-PR

4.2.2 Rekombinasi (*Genetic Operator*)

Seperti yang disebutkan pada subbab 3.3.4, proses *crossover* yang digunakan adalah *one-cut-poin crossover*, dan *uniform crossover*. Sedangkan metode mutasi menggunakan *simple-random-mutation* dan *random-exchange-mutation.*

1. *One-cut-poin Crossover*

One-cut-poin crossover diawali dengan menentukan secara acak dua buah kromosom yang dilibatkan. Kemudian menentukan secara acak satu titik dalam kromosom yang digunakan untuk memotong kromosom tersebut menjadi dua bagian, untuk mempermudah kita sebut bagian pertama dan bagian kedua. Kemudian menukarkan bagian pertama pada kromosom pertama dengan bagian pertama pada kromosom kedua. Begitu juga untuk bagian kedua kromosom pertama ditukar dengan bagian kedua pada kromosom kedua. Langkah-langkah *one-cut-poin crossover* ini lebih jelas dapat dilihat pada Gambar 4-6.

```
Procedure OneCutPointCrossover( Populasi )
begin
      iP1 = random()
      iP2 = random()
      cutP = random()
      c1 = Populasi[iP1]
      c2 = Populasi[iP2]
      for(i=0 ; c1.size() ; i++)
            if(i<cutP)
                  has[i] = c1[i]
            else
                  has[i] = c2[i]
      return has
end
```

Gambar 4-6 Langkah-langkah *one-cut-point crossover*

2. *Uniform Crossover*

Langkah-langkah *uniform crossover* dimulai dengan memilih secara acak dua kromosom yang ada pada populasi. Kemudian untuk setiap gen pada kromosom secara acak ditentukan akan diisi dengan gen pada kromosom pertama atau kedua. Langkah-langkah *uniform crossover* dapat dilihat pada Gambar 4-7.

```
Procedure UniformCrossover( Populasi )
begin
        iP1 = random()
        iP2 = random()
        c1 = Populasi[iP1]
        c2 = Populasi[iP2]
        for(i=0 ; c1.size() ; i++)
                if(random() < 0.5)
                        has[i] = c1[i]
                else
                        has[i] = c2[i]
        return has
end
```

Gambar 4-7 Langkah-langkah *uniform crossover*

3. *Simple-random Mutation*

Langkah-langkah *simple-random mutation* diawali dengan memilih secara acak kromosom dari populasi. Kemudian mengganti salah satu gen dari kromosom tersebut dengan nilai acak. Nilai acak gen ini berbeda untuk representasi kromosom pertama dan kedua. Pada representasi kromosom pertama nilai random ini dipilih dari nilai *k* (kategori UKT). Sedangkan untuk representasi kromosom kedua nilai gen random ini dipilih mulai dari 0 sampai maksimal besaran UKT per kategori UKT per program studi (U_{ks}). Langkah-langkah simple-random mutation untuk representasi kromosom kedua digambarkan pada Gambar 4-8.

```
Procedure SimpleRandomMutation( Populasi )
begin
        has = Populasi[random()]
        has[random()] = random * max( Uks )
        return has
end
```

Gambar 4-8 Langkah-langkah *simple-random mutation*

4. *Random-exchange Mutation*

```
Procedure RandomExchangeMutation( Populasi )
begin
        rm1 = random()
        rm2 = random()
        has = Populasi[random()]
        tmp = has[rm1]
        has[rm1] = has[rm2]
        has[rm2] = tmp
        return has
end
```

Gambar 4-9 Langkah-langkah *random-exchange mutation*

Langkah-langkah *random-exchange mutation* dimulai dengan memilih dua titik secara acak yang digunakan untuk pertukaran. Kemudian dilanjutkan dengan memilih secara acak satu kromosom yang terdapat pada populasi. Kemudian menukarkan gen pada dua titik pada kromosom yang terpilih. Langkah-langkah *random-exchange mutation* ini dituangkan pada Gambar 4-9 agar lebih mudah dipahami.

4.2.3 Metode Perbaikan (*Repair*) Kromosom

Metode perbaikan bertujuan untuk memperbaiki konfigurasi kromosom yang melanggar *constraint* atau bisa dikatakan kromosom yang *infeasible* (tidak layak). Seperti yang telah dijelaskan pada subbab 3.3.5, bahwa metode perbaikan kromosom berbeda untuk kedua representasi kromosom.

1. Metode Perbaikan Pada Representasi Kromosom Pertama

Metode perbaikan pada representasi kromosom pertama bisa dikatakan berfokus agar kromosom tidak melanggar *constraint* ke-4, yaitu dilakukan dengan mencari gen yang mempunyai nilai UKT (U_m) yang tidak sesuai dengan kemampuan finansialnya (E_m). Kemudian menggantinya dengan kategori UKT yang sesuai. Misal dengan menggunakan contoh kasus seperti pada perhitungan manual dan salah satu mahasiswa *m* memiliki E_m = 2.8 juta. Namun karena nilai gen ditentukan acak, sehingga k_m = 4 (kategori mahasiswa *m* adalah 4) yang besaran UKT-nya adalah 5 Juta. Karena nilai E_m lebih besar dari U_{2s} (kategori 2 dengan besaran 2.5 juta) dan kurang dari U_{3s} (kategori 3 dengan besaran 3.5), maka bisa dikatakan kategori yang sesuai adalah k_m = 2 (kategori 2).

```
Procedure Repair(kromosom)
begin
        for(m=0; m< M; m++)
            propk = K
            for(k=K; k>0; k--)
                if(Em >= Uks)
                    propk = k
            kromosom[m] = propk
        return kromosom
end
```

Gambar 4-10 Metode perbaikan untuk representasi kromosom pertama

2. Metode Perbaikan Pada Representasi Kromosom Kedua

```
Procedure Repair (kromosom)
begin
    for(i=1; i < size(kromosom)-1; i++)
        if(kromosom[i]<kromosom[i-1] and i mod K-1 > 0)
            prop = kromosom[i]
            kromosom[i] = kromosom[i-1]
            kromosom[i-1] = prop
    return kromosom
end
```

Gambar 4-11 Metode perbaikan pada representasi kromosom kedua

Metode perbaikan pada representasi kromosom kedua ditujukan untuk menghindari pelanggaran pada *constraint* ke-4. Metode perbaikan ini dilakukan dengan mencari nilai gen ke-i yang lebih besar dari nilai ke-$i+1$, ketika $i+1 \,\%\, (mod)(K-1) \neq 0$, atau bisa dikatakan *i* dan *i+1* berada dalam prodi *s* yang sama. Kemudian menukar nilai kedua gen tersebut. Langkah-langkah metode perbaikan dapat dilihat pada Gambar 4-11.

4.2.4 Modifikasi Metode Seleksi Yang Diusulkan

Metode seleksi yang diusulkan adalah metode *non-dominated sorting* yang dimodifikasi. Seperti yang sudah dijelaskan pada subbab 2.4.2, terdapat dua proses *sorting* dalam metode seleksi pada NSGA-II yaitu *fast-non-dominated sorting* dan *sorting* berdasarkan *crowding distance*. Modifikasi metode seleksi NSGA-II dilakukan kepada kedua langkah tersebut. Tujuan utama modifikasi adalah untuk mempersingkat waktu eksekusi.

1. Modifikasi algoritme *fast-non-dominated sorting*

Seperti algortima *sorting* pada umumnya, untuk mengurangi waktu eksekusi pada langkah *fast-non-dominated sorting*, dilakukan dengan mengurangi jumlah pengecekan yang dilakukan. Sedangkan pengecekan yang dilakukan selalu dipengaruhi oleh jumlah individu yang terlibat dalam proses *sorting*. Sesuai dengan algortima *fast-non-dominated sorting* pada Gambar 2-3, jumlah individu yang terlibat untuk proses sorting dalam rangka mendapatkan front pertama adalah sejumlah *popSize* x 2 (jumlah populasi kali dua). Dan untuk mendapatkan front kedua, jumlah individu yang terlibat adalah *popSize* x 2 – jumlah individu pada *front 1*. Ilustrasi perhitungan jumlah individu terlibat untuk jumlah populasi 100 dapat dilihat pada Tabel 4-8. Seperti yang dapat dilihat pada Tabel 4-8 untuk jumlah populasi 100, maka total individu yang terlibat untuk mendapatkan 4 *front* adalah 500.

Tabel 4-8 Ilustrasi jumlah kromosom terlibat pada algoritme *fast-non-dominated sorting*

Front	Jumlah Kromosom	Jumlah Kromosom Terlibat
1	50	200
2	50	150
3	50	100
4	50	50
	Total	500

Modifikasi algoritme *fast-non-dominated sorting* yang dilakukan pada tesis ini mencoba untuk mengurangi jumlah kromosom yang terlibat. Sehingga dengan berkurangnya jumlah kromosom yang terlibat, maka dapat mempercepat proses *fast-non-dominated sorting*. Langkah-langkah non-dominated sorting yang dimodifikasi adalah sebagai berikut:

1) Dilakukan pengulangan untuk setiap *s* anggota *S* (seluruh individu yang terlibat yaitu sejumlah 2 x *popSize*)
2) Jika jumlah *front* lebih dari 0, maka ke langkah 3, jika tidak maka ke jadikan *s* sebagai anggota dari *front* ke-0
3) Dilakukan pengulangan untuk *i* mulai dari 0 sampai sejumlah *front* (yang sudah ada)
4) Jika *s* mendominasi *fitness* minimal dari *front* ke-*i*, maka:
5) Cari *residue*, yaitu anggota *front* ke-*i* yang didominasi oleh *s*
6) Letakkan *residu* sebagai *front* baru pada *front* ke-(*i*+1), menggeser *front* ke-(*i*+1) yang sebelumnya sudah ada

7) Jadikan *s* sebagai anggota *front* ke-*i*

Pseudocode dari langkah-langkah *non-dominated sorting* yang dimodifikasi ini dapat dilihat pada Gambar 4-12. *Pseudocode* dari langkah-langkah untuk mencari *residue* (anggota front yang terdominasi, sesuai dengan langkah ke-5) dapat dilihat pada Gambar 4-13. Dan *pseudocode* dari langkah-langkah menggeser *front* yang digunakan pada langkah ke-6 dapat dilihat pada Gambar 4-14. Dengan langkah-langkah *non-dominated sorting* yang dimodifikasi, jumlah individu yang terlibat tetap yaitu sejumlah 2 x *popSize*, untuk mendapatkan semua *front*.

```
1  Procedure Mod-NonDominatedSort()
2      foreach S as s
3          if(Fronts.size > 0 )
4              for(i=0; i<Fronts.size; i++)
5                  if(dominated(s, Fronts[i].min))
6                      residue = getDominatedMember(s, Fronts[i])
7                      shiftFronts(i+1, residue)
8                      Fronts[i].add(s)
9          else
10             Fronts[0].add(s)
```

Gambar 4-12 Langkah-langkah prosedur *non-dominated sorting* yang dimodifikasi

```
1  Procedure getDominatedMember(k, front)
2      foreach(front as f)
3          if(dominated(k, f))
4              dominatedMember.add(f)
5              removeElement(front, f)
6      return dominatedMember
```

Gambar 4-13 Prosedur untuk mendapatkan anggota *front* yang terdominasi

```
1  Procedure shiftFronts(pos, list)
2      if(pos < Fronts.size)
3          Fronts.add(Fronts[Fronts.size-1])
4          for(i=Front.size-2; i>pos+1; i--)
5              Fronts[i] = Fronts[i-1]
6          Fronts[pos] = list
```

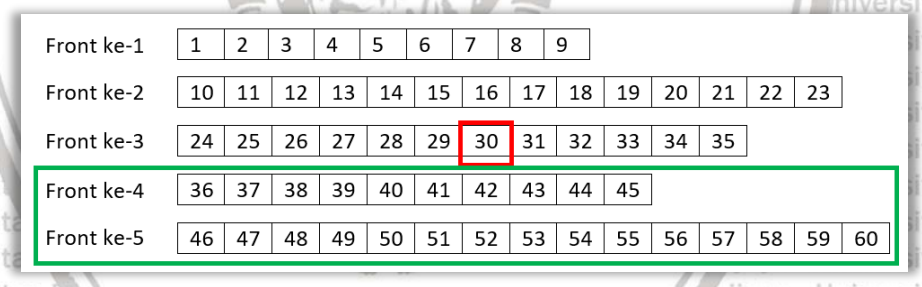
Gambar 4-14 Prosedur menggeser *front*

2. Modifikasi perhitungan *sorting* berdasarkan *crowding distance operator*

Seperti yang sudah dijelaskan pada subbab 2.4.2, bahwa diperlukan proses perhitungan nilai *crowding distance* sebelum proses *sorting* berdasarkan *crowding distance*. Dan seperti yang dapat dilihat pada Gambar 2-4, bahwa dalam proses perhitungan tersebut terdapat proses *sorting* sejumlah *fitness* yang digunakan. Sehingga pada permasalahan

tesis ini, proses sorting pada perhitungan crowding distance dilakukan sebanyak dua kali, yaitu *sorting* berdasarkan *fitness* 1 dan *fitness* 2. Dalam satu kali proses *sorting* melibatkan individu sejumlah 2 x *popSize*. Sedangkan hasil dari keseluruhan proses seleksi hanya sejumlah *popSize*. Maka dengan mengurangi jumlah individu yang terlibat dalam proses sorting ini, akan mengurangi waktu eksekusi algoritme NSGA-II secara signifikan.

Modifikasi pada proses *sorting* berdasarkan *crowding distance operator* ini dilakukan pada proses perhitungan nilai *crowding distance*. Modifikasi dilakukan dengan membatasi jumlah *front* yang dilibatkan pada proses *sorting* berdasarkan *fitness* 1 dan *fitness* 2. *Front* yang dilibatkan dalam proses *sorting* hanya dibatasi pada *front* yang mengandung individu ke-*popSize*. Ketika *front* yang mengandung individu ke-*popSize* ini adalah *front* ke-*f*, maka *front* ke-(*f+1*) dan seterusnya tidak diproses untuk pengurutan (*sorting*). Hal ini karena individu pada *front* ke-(*f+1*) dan seterusnya pasti tidak akan terpilih pada proses seleksi, yaitu ketika diambil individu sejumlah *popSize* dari hasil pengurutan (*sorting*) berdasarkan *crowded comparison operator*. Seperti yang sudah dijelaskan pada subbab 2.4.2, *crowded comparison operator* menganggap *x* lebih baik dari *y* adalah jika $x_{rank} > y_{rank}$ atau ($x_{rank} = y_{rank}$ dan $x_{distance} > y_{distance}$). Dengan x_{rank} adalah indeks *front* dari *x*, dan $x_{distance}$ adalah nilai *crowding distance* dari *x*. Ilustrasi pengaruh dari modifikasi proses perhitungan nilai *crowding distance* dapat lebih jelas dilihat pada Gambar 4-15. Gambar 4-15 mengilustrasikan ketika *popSize*=30 dan proses *non-dominated sorting* menghasil 5 buah *front*. Individu ke-30 (ke-*popSize*) ditunjukkan dengan kotak berwarna merah, terletak pada *front* ke-3. Sehingga modifikasi perhitungan nilai *crowding distance* akan menghemat waktu dengan tidak mengurutkan *front* ke-4 dan ke-5 (yang ditunjukkan dengan kotak hijau) berdasarkan nilai *fitness*.



Gambar 4-15 Ilustrasi pengaruh modifikasi proses perhitungan nilai *crowding distance*

Modifikasi proses perhitungan nilai *crowding distance* dari bentuk awal (Gambar 2-4) adalah ketika jumlah individu solusi dari *front* yang diurutkan (*sorting*) berdasarkan nilai *fitness* sudah lebih dari *popSize*, maka *front* berikutnya tidak perlu diurutkan lagi. Sehingga langkah perhitungan nilai *crowding distance* yang dimodifikasi dapat dilihat pada *Pseudocode* Gambar 4-16.

```
1   Procedure Mod-HitungCrowdingDistance ()
2     jmlInd = 0
3     for(frontIndex=0; frontIndex< Fronts.size; frontIndex++)
4         jmlInd = Fronts[frontIndex].front.size
5         if(jmlInd <= pop_size)
6             sortedByF1 = sortFrontByF1(frontIndex)
7             sortedByF2 = sortFrontByF2(frontIndex)
8         else
9             sortedByF1 = Fronts[frontIndex].front
10            sortedByF2 = Fronts[frontIndex].front
11        sortedByF1[0].distance = INFINITY
12        sortedByF1[sortedByF1.size()-1].distance = INFINITY
13        for(i=1; i<sortedByF1.size()-1; i++)
14            sortedByF1[i].distance += | sortedByF1[i+1].f1 - sortedByF1[i-1].f1 |
15        sortedByF2[0].distance = INFINITY
16        sortedByF2[sortedByF2.size()-1].distance = INFINITY
17        for(i=1; i<sortedByF2.size()-1; i++)
18            sortedByF2[i].distance += | sortedByF2[i+1].f2 - sortedByF2[i-1].f2 |
```

Gambar 4-16 Pseudocode perhitungan crowding distance yang dimodifikasi

4.3 Perancangan Pengujian

Pada perancangan pengujian ini akan lebih mendetailkan metodologi pengujian yang sudah dijelaskan pada subbab 3.4. Seperti yang sudah disebutkan pada subbab 3.4 pada penelitian ini akan dilakukan lima pengujian yaitu pengujian terhadap jumlah populasi, maksimal generasi, metode rekombinasi, metode perbaikan kromosom, dan metode seleksi. Lima pengujian tersebut ditujukan untuk mencari konfigurasi parameter NSGA-II yang terbaik. Parameter NSGA-II yang dimaksud adalah representasi kromosom, metode inisialisasi kromosom, metode rekombinasi (operator genetika), jumlah populasi, dan maksimal generasi. Parameter NSGA-II yang terbaik didapatkan ketika dengan parameter yang digunakan dapat mencapai nilai *fitness* yang terbaik, membutuhkan memori penyimpanan yang kecil dan dengan waktu penyelesaian yang singkat. Parameter NSGA-II yang terbaik ini juga bisa disebut parameter yang efisien.

Sudah dijelaskan sebelumnya bahwa indikator yang digunakan adalah nilai *fitness* yang terbaik dan waktu komputasi yang paling singkat. Sedangkan indikator besar memori sudah jelas terlihat dari representasi kromosom yang digunakan. Representasi kedua jelas lebih memiliki panjangan larik (*array*) yang lebih pendek dibandingkan dengan representasi kromosom pertama. Untuk lebih mempertegas hasil indikator nilai *fitness*, maka akan dibandingkan juga jarak *Euclidean* dari dua nilai fitness ini. Untuk menghitung jarak *Euclidean* (*ED*), maka dua nilai *fitness* ini dianggap sebagai sebuah titik dalam ruang dua dimensi. Sehingga jarak *Euclidean* adalah jarak antara titik (0, 0) ke titik dua nilai *fitness* (*f1, f2*). Sebelum dihitung jarak *Euclidean*-nya, dua nilai *fitness* ini perlu untuk diskalakan menjadi bilangan antara 0 dan 1.

4.3.1 Pengujian Representasi Kromosom Dan Pengaruh Jumlah Populasi

Pengujian terhadap jumlah populasi (lebih singkat disebut *popSize*) ini ditujukan untuk mencari jumlah populasi yang efisien untuk penentuan UKT. Jumlah populasi yang efisien adalah jumlah populasi yang menggunakan memori kecil dan waktu penyelesaian yang cepat,

tetapi memperoleh hasil yang baik. Optimasi memperoleh hasil yang baik adalah ketika mencapai nilai fitness yang tertinggi (atau terkecil tergantung pada tujuan optimasi).

Pada pengujian terhadap jumlah populasi ini akan dibandingkan antara representasi kromosom pertama dan kedua. Sehingga pada tahap ini juga akan dihasilkan representasi mana yang lebih efisien. Representasi yang lebih efisien adalah representasi yang membutuhkan jumlah populasi yang lebih sedikit dan menghasilkan nilai fitness yang terbaik, dan tentu saja membutuhkan waktu komputasi yang lebih singkat.

Pada pengujian ini akan dicari jumlah populasi yang paling efisien mulai dari 20 sampai dengan 200 dengan kelipatan 20. Untuk lebih memastikan hasil pengujian, pada pengujian ini juga akan dicari menggunakan jumlah populasi yang lebih besar yaitu kelipatan 200 mulai dari 400 sampai 2000. Karena maksimal generasi yang paling efisien masih belum diketahui, maka pada tahap ini menggunakan jumlah generasi yang besar yaitu 3000. Metode reproduksi yang digunakan adalah *one-cut-poin crossover* dan *simple-random mutation* (*OC-SM*). Pengujian ini akan dilakukan 20 kali untuk diambil rata-rata dan maksimum nilai *fitness* serta rata-rata dan maksimum jarak *Euclidean*-nya (*ED*). Hasil pengujian untuk setiap repesentasi kromosom akan dituangkan pada tabel seperti yang diperlihatkan pada Tabel 4-9. Tabel 4-9 menampilkan rerata nilai *fitness* (*f1*, *f2*, dan *ED*), maksimal nilai *fitness* (maks *f1*, maks *f2*, dan maks *ED*), rerata waktu eksekusi dan rerata jumlah solusi yang layak (*Feasible Solution* disingkat *FeaSol*). Untuk lebih memudahkan melihat dan menganalisis kecenderungan dan perbandingan, maka data hasil pengujian juga akan ditampilkan dalam bentuk grafik.

Tabel 4-9 Rancangan tabel hasil pengujian terhadap jumlah populasi

popSize	**Rerata *f1***	**Rerata *f2***	**Rerata *ED***	**Maks *f1***	**Maks *f2***	**Maks *ED***	**Rerata waktu (detik)**	**Rerata *FeaSol***
20								
40								
…								
200								

4.3.2 Pengujian Representasi Kromosom Dan Pengaruh Maksimal Generasi

Pengujian terhadap maksimal generasi (lebih singkat disebut *maxGen*) dilakukan untuk mengetahui jumlah maksimal generasi yang diperlukan untuk mencapai nilai *fitness* yang terbaik. Pada tahap pengujian ini akan digunakan jumlah populasi sesuai dengan hasil pada pengujian terhadap jumlah populasi. Sedangkan fungsi *fitness* 1 yang digunakan adalah fungsi *fitness 1* bentuk pertama. Pengujian berdasarkan maksimal generasi ini juga dilakukan untuk kedua representasi kromosom.

Rancangan pengujian terhadap maksmal generasi untuk masing-masing representasi kromosom dapat dilihat pada Tabel 4-10. Hasil pengujian juga akan ditampilkan dalam bentuk grafik garis, dengan tujuan untuk mempermudah dalam menganalisis.

Tabel 4-10 Rancangan tabel pengujian terhadap maksimal generasi (*maxGen*)

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
3000								
2000								
...								
10								

4.3.3 Pengujian Metode Rekombinasi (*Genetic Operator*)

Pengujian terhadap metode rekombinasi (*Genetic Operator* lebih singkat disebut *GO*) dilakukan untuk mengetahui kombinasi *GO* yang paling efisien. *GO* yang paling efisien adalah *GO* yang membuat algoritme genetika memiliki nilai *fitness* yang paling baik dalam waktu yang singkat. Pada tahap pengujian ini akan digunakan jumlah populasi dan jumlah maksimal generasi sesuai dengan hasil pada pengujian sebelumnya (yaitu pengujian terhadap jumlah populasi dan maksimal generasi). Sedangkan fungsi *fitness* 1 yang digunakan adalah fungsi *fitness1* bentuk pertama. Pengujian terhadap metode rekomendasi ini dilakukan terhadap kedua representasi kromosom. Sehingga hasil akhirnya dapat memberikan rekomendasi penggunaan *GO* yang paling cocok untuk masing-masing representasi kromosom.

Seperti yang sudah dijelaskan pada subbab 3.3.4, terdapat lima kombinasi metode rekombinasi (*GO*) yang diakan diuji, yaitu:

1. *OC-SM* (*one-cut-poin crossover* dan *simple-random mutation*)
2. *OC-EM* (*one-cut-poin crossover* dan *random-exchange mutation*)
3. *UC-SM* (*uniform crossover* dan *simple-random mutation*)
4. *UC-EM* (*uniform crossover* dan *random-exchange mutation*)
5. *RC-RM* (*random crossover* yaitu random antara *one-cut-poin crossover dan uniform crossover* serta *random mutation* yaitu random antara *simple-random mutation* dan *random-exchange mutation*)

Tabel 4-11 Rancangan tabel pengujian terhadap metode reproduksi (*GO*)

GO	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
OC-SM								
OC-EM								
UC-SM								
UC-EM								
RC-RM								

Hasil dari kelima kombinasi metode rekombinasi tersebut akan ditampilkan dalam tabel seperti yang dapat dilihat pada Tabel 4-11. Selain itu, pengujian juga dilakukan untuk mencari kombinasi dari nilai *Crossover Rate* (*Cr*) dan *Mutation Rate* (*Mr*) terbaik. Hasil pengujian *Cr* dan *Mr* ditampilkan pada tabel seperti yang dapat dilihat pada **Error! Not a valid bookmark self-reference.**.

Tabel 4-12 Rancangan tabel pengujian nilai *Cr* dan *Mr* terbaik

Cr	*Mr*	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
0	1								
0.1	0.9								
…	…								
1	0								

4.3.4 Pengujian Pengaruh Metode *Repair*

Pengujian ini bertujuan untuk mencari tahu pengaruh metode *repair* kaitannya dengan dua bentuk representasi kromosom yang diusulkan. Parameter algoritme genetika yang digunakan pada pengujian ini adalah parameter yang terbaik yang telah dihasilkan pada tahapan pengujian sebelumnya. Hasil dari pengujian akan dituangkan pada tabel yang dapat dilihat pada Tabel 4-13 dan Tabel 4-14. Kolom rerata *DistictSol* pada Tabel 4-14 akan berisi rerata jumlah solusi (kromosom) yang berbeda. Rerata jumlah solusi (kromosom) yang berbeda akan menunjukkan tingkat keragaman dari solusi yang dihasilkan dari rangkaian metode yang diuji.

Tabel 4-13 Rancangan pengujian metode *repair* terhadap nilai *fitness* dan waktu eksekusi

Representasi dan Repair	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)
K1-*NoRepair*							
K1-*Repair*							
K2-*NoRepair*							
K2-*Repair*							

Tabel 4-14 Rancangan pengujian pengaruh metode *repair* pada keragaman solusi

Representasi dan Repair	Rerata *FeaSol*	Rerata *DistinctSol*
K1-*NoRepair*		
K1-*Repair*		
K2-*NoRepair*		
K2-*Repair*		

4.3.5 Pengujian Bentuk Fungsi *Fitness 1*

Pengujian bentuk fungsi *fitness 1* bertujuan untuk mencari pengaruh bentuk *fitness 1* untuk kedua representasi kromosom yang diusulkan. Parameter algoritme genetika yang digunakan pada pengujian ini adalah parameter terbaik yang dihasilkan pada tahapan pengujian sebelumnya. Hasil pengujian bentuk fungsi *fitness 1* akan ditampilkan seperti Tabel 4-15. Berbeda dengan tabel-tabel pengujian sebelumnya yang selalu menunjukkan rerata dan maksimal *f1* (*fitness 1*), pada Tabel 4-15 yang ditampilkan adalah rerata dari nilai total jarak antara besaran UKT U_m dengan kemampuan finansial mahasiswa E_m. Untuk mempersingkat penyebutan, maka dapat disebut rerata nilai $\sum_{m=1}^{M}|U_m - E_m|$.

Tabel 4-15 Rancangan tabel pengujian terhadap bentuk fungsi *fitness 1*

Representasi Kromosom dan Bentuk *f1*	Rerata $\sum_{m=1}^{M}\|U_m - E_m\|$	Rerata *f2*	Maks $\sum_{m=1}^{M}\|U_m - E_m\|$	Maks *f2*	Rerata waktu (detik)	Rerata *FeaSol*
K1-*f1*V1						
K1-*f1*V2						
K2-*f1*V1						
K2-*f1*V2						

4.3.6 Pengujian Metode Inisialisasi

Pengujian metode inisialisasi bertujuan untuk mencari metode inisialisasi yang menghasilkan nilai *fitness* (*f1 dan f2*) terbaik dengan waktu yang paling cepat. Pengujian metode inisialisasi ini hanya dilakukan untuk representasi kromosom kedua. Pengujian dilakukan mulai dari nilai maksimal generasi 10 sampai dengan 100 untuk masing-masing metode inisialisasi. Sedangkan parameter algoritme genetika yang lain menggunakan parameter terbaik yang dihasilkan pada tahapan pengujian sebelumnya. Hasil dari pengujian untuk masing-masing metode inisialisasi ditampilkan menggunakan tabel seperti yang dapat dilihat pada Tabel 4-16.

Tabel 4-16 Rancangan tabel pengujian metode inisialisasi

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
100								
90								
...								
10								

4.3.7 Pengujian Metode Seleksi

Tahap pengujian ini bertujuan untuk mencari tahu tentang algoritme yang lebih efisien dalam optimasi multi-objektif penentuan UKT. Seperti yang sudah dijelaskan sebelumnya bahwa efisien berarti yang mencapai nilai *fitness* (*f1 dan f2*) terbaik dan dengan waktu yang lebih singkat. Pengujian ini dilakukan untuk kedua representasi kromosom. Sedangkan parameter algoritme genetika yang lain menggunakan parameter terbaik yang dihasilkan pada tahapan pengujian sebelumnya. Hasil dari tahapan pengujian ini akan ditampilkan menggunakan tabel seperti yang dapat dilihat pada Tabel 4-17.

Tabel 4-17 Rancangan pengujian terhadap metode seleksi

Metode	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
GA-BT								
GA-PRW								
NSGA-II								
NSGA-II-Mod								

BAB 5 HASIL DAN PEMBAHASAN

5.1 Pengujian Representasi Kromosom Dan Pengaruh Jumlah Populasi

Pengujian terhadap jumlah populasi untuk representasi kromosom pertama disajikan pada Tabel 5-1. Sedangkan untuk representasi kromosom kedua ditampilkan pada Tabel 5-2. Kedua pengujian tersebut dilakukan menggunakan maksimal iterasi 3000, bentuk *fitness 1* pertama, metode *one-cut-point crossover* dan metode *simple-random mutation*. Nilai *Euclidean Distance (ED)* dihitung dari penskalaan nilai *fitness 1* (*f1*) dan *fitness 2* (*f2*). Penskalaan *f1* dilakukan dengan nilai maksimal *f1* adalah 0.5 dan nilai minimal *f1* adalah 0.1. Sedangkan nilai maksimal *f2* adalah 800 dan niai minimalnya adalah 300. Seperti yang dapat dilihat pada Tabel 5-1 dan Tabel 5-2, nilai yang digunakan untuk analisis adalah rerata *f1*, *f2* dan *ED*. Ketiga nilai rerata ini akan dapat menunjukkan garis besar dari nilai *fitness* yang dihasilkan oleh kedua representasi kromosom. Sedangkan nilai maksimal *f1*, *f2* dan *ED* menunjukkan hasil terbaik yang diperoleh oleh kedua bentuk kromosom. Tabel 5-1 dan Tabel 5-2 juga menunjukkan rerata waktu (dalam detik) yang diperlukan ketika menggunakan jumlah populasi tertentu. Rerata waktu bersama dengan nilai *fitness* ini menunjukkan efisiensi dari kedua bentuk kromosom. Selain itu Tabel 5-1 dan Tabel 5-2 juga menyajikan jumlah solusi (kromosom) yang sama sekali tidak melanggar constraint, jumlah ini ditampilkan pada kolom Rerata *FeaSol* (*Feasible Solution*).

Tabel 5-1 Hasil K1 ketika *popSize* 20 sampai 200

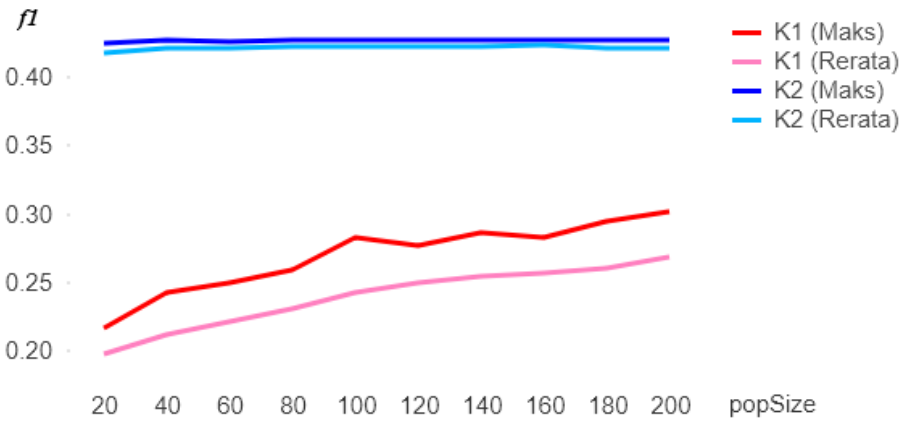
popSize	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
20	0.198	521.583	0.508	0.217	569.310	0.584	2.390	0
40	0.213	550.781	0.576	0.243	609.500	0.695	3.701	0
60	0.222	573.499	0.627	0.251	640.775	0.744	5.902	0
80	0.231	591.341	0.669	0.260	644.415	0.772	8.094	0
100	0.243	603.659	0.705	0.283	666.828	0.824	15.289	0
120	0.250	617.745	0.738	0.278	672.160	0.837	12.254	0
140	0.254	619.638	0.747	0.286	667.605	0.826	14.764	0
160	0.257	625.123	0.761	0.282	668.615	0.829	18.519	0
180	0.260	631.938	0.776	0.295	686.278	0.884	21.774	0
200	0.268	637.547	0.796	0.302	698.095	0.902	24.137	0

Untuk lebih memperlihatkan perbandingan dari dua buah bentuk kromosom, dari Tabel 5-1 dan Tabel 5-2 ditampilkan dalam bentuk grafik. Grafik perbandingan *f1* (*fitness 1*) antara representasi kromosom pertama (K1) dan kedua (K2) ditampilkan pada Gambar 5-1. Sedangkan grafik perbandingan *f2* (*fitness 2*) antara representasi pertama (K1) dan kedua (K2) dapat dilihat pada Gambar 5-2. Gambar 5-1 dan Gambar 5-2 menunjukkan bahwa representasi kromosom kedua (K2) menghasilkan nilai maksimal dan rerata *f1* dan *f2* lebih

baik dibandingkan dengan representasi pertama (K1). Selain itu representasi krosomsom pertama (K1) tidak menghasilkan solusi yang layak (*feasible*). Semua solusi yang dihasilkan representasi kromosom pertama selalu melanggar *constraint* ke-4, yaitu nilai minimal kemampuan finansial mahasiswa (E_m) pada kategori *k* harus lebih besar dari pada nilai maksimal dari kemampuan finansial mahasiswa pada kategori sebelumnya (*k-1*). Sedangkan representasi kedua (K2) selalu menghasilkan solusi yang layak sejumlah populasinya.

Tabel 5-2 Hasil K2 ketika *popSize* 20 sampai 200

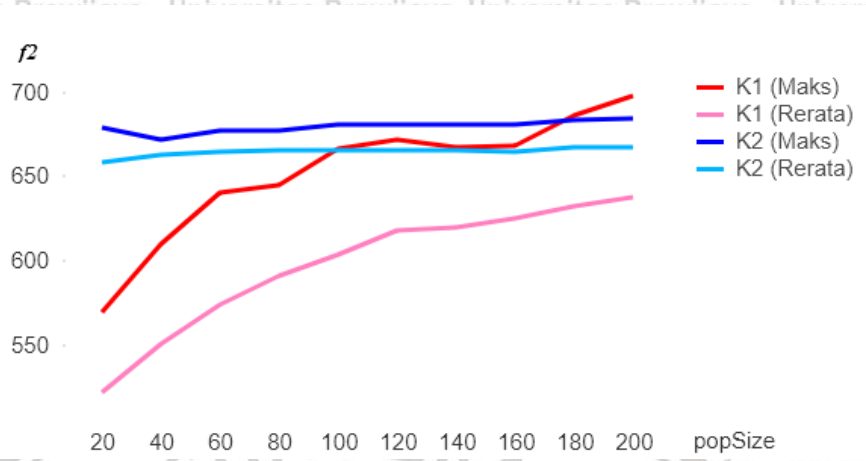
popSize	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
20	0.417	658.028	1.069	0.425	679	1.090	2.623	20
40	0.421	663.038	1.083	0.426	672	1.098	3.720	40
60	0.421	664.463	1.085	0.426	677	1.093	5.482	60
80	0.423	665.274	1.089	0.426	677	1.098	7.648	80
100	0.422	665.135	1.087	0.426	681	1.098	9.857	100
120	0.423	665.402	1.088	0.426	681	1.098	11.790	120
140	0.423	665.471	1.089	0.426	681	1.098	14.182	140
160	0.423	664.700	1.089	0.426	681	1.098	17.107	160
180	0.422	667.349	1.089	0.426	683	1.098	20.143	180
200	0.421	667.403	1.089	0.426	684	1.098	23.483	200



Gambar 5-1 Perbandingan *f1* ketika *popSize* 20 sampai dengan 200

Hasil representasi kromosom pertama (K1) pada Gambar 5-1 dan Gambar 5-2 digambarkan dengan garis warna merah dan warna merah muda. Garis warna merah adalah nilai maksimal dari *f1* atau *f2* yang dihasilkan oleh representasi kromosom pertama (K1). Sedangkan garis warna merah muda menggambarkan rerata dari nilai *f1* maupun *f2* dari K1. Nilai *fitness* (*f1*, *f2*) yang tertinggi dihasilkan ketika jumlah populasi (*popSize*) adalah 200. Demikian juga untuk nilai rerata fitness tertinggi juga diperoleh ketika jumlah populasi (*popSize*) adalah 200. Jika cermati Gambar 5-1 dan Gambar 5-2, nilai *fitness* yang dihasilkan

oleh representasi kromosom pertama memiliki kecenderungan untuk naik searah dengan bertambahnya jumlah populasi (*popSize*). Sehingga masih memiliki kemungkinan untuk mendapatkan nilai *fitness* yang lebih baik dan menghasilkan solusi yang layak, dengan memperbesar jumlah populasi (*popSize*) dengan nilai yang lebih dari 200. Untuk itu pengujian terhadap jumlah populasi akan dikembangkan kembali untuk jumlah populasi (*popSize*) lebih dari 200 dengan kelipatan 200 sampai dengan jumlah populai (*popSize*) 2000.

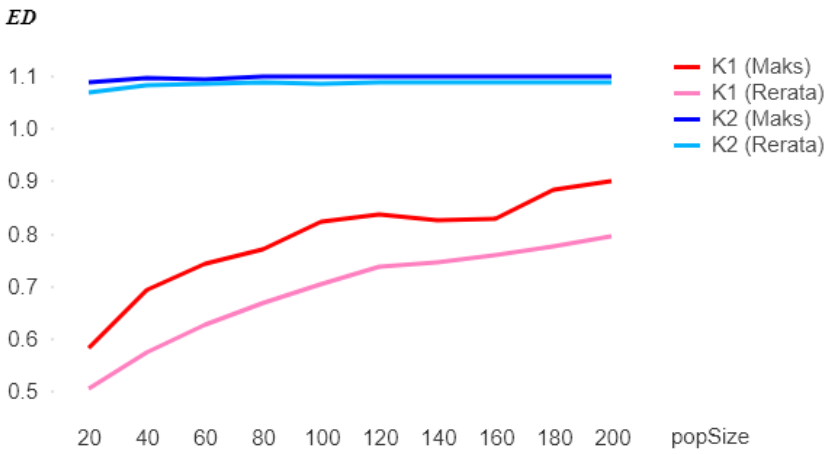


Gambar 5-2 Perbandingan *f2* ketika *popSize* 20 sampai dengan 200

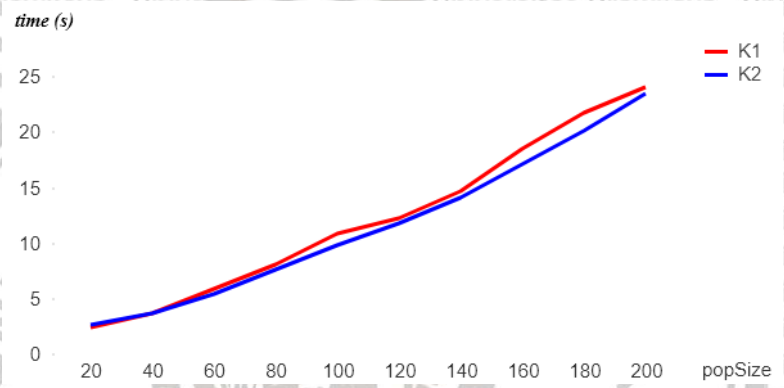
Pada Gambar 5-1 hampir tidak terlihat kenaikan nilai rata-rata (garis warna biru muda) maupun nilai maksimal (garis warna biru tua) *f1* yang dihasilkan oleh representasi kromosom kedua (K2). Namun jika dilihat pada Tabel 5-2, perubahan nilai rerata dan maksimal *f1* yang sangat kecil sekali. Nilai maksimal *f1* untuk representasi kromosom kedua (K2) adalah 0.426, dan nilai ini didapatkan mulai dari jumlah populasi 40 sampai 200. Sedangkan nilai maksimal *f2* untuk representasi kedua adalah 684 yang didapatkan ketika jumlah populasinya 200, seperti yang bisa dilihat pada Gambar 5-2. Sehingga untuk representasi kromosom kedua, jumlah populasi paling kecil yang mencapai nilai *fitness* terbaik adalah 200. Namun hal ini akan berbeda ketika kita menganalisis melalui *euclidean distance* dari *f1* dan *f2* yaitu nilai *ED*.

Perbandingan nilai rerata dan maksimal *Euclidean Distance* (*ED*) nilai *fitness* (*f1*, *f2*) terhadap sumbu (0, 0) dari kedua representasi kromosom (K1 dan K2) ditunjukkan pada Gambar 5-3. Pada Gambar 5-3 juga menunjukkan dominasi hasil dari representasi kromosom kedua (K2) terhadap representasi kromosom pertama (K1). Pada Tabel 5-2 menunjukkan bahwa nilai *ED* terbaik yang dicapai oleh representasi kromosom kedua (K2) adalah 1.098. Nilai *ED* terbaik ini diperoleh mulai dari jumlah populasi 80 sampai 200. Sampai disini dapat disimpulkan bahwa jumlah populasi paling efisien adalah 80.

Gambar 5-4 menampilkan grafik perbandingan waktu eksekusi (dalam detik) *NSGA-II* antara kedua representasi kromosom terhadap jumlah populasi (*popSize*). Dalam Gambar 5-4 menunjukkan bahwa waktu eksekusi representasi kromosom pertama dan kedua hampir sama. Namun jika dihubungkan dengan nilai *f1* dan *f2* serta nilai *ED*, maka representasi kromosom kedua bisa dikatakan lebih efisien. Dengan jumlah populasi yang sama dan waktu yang hampir sama, representasi kromosom kedua dapat menghasilkan nilai *fitness* yang jauh lebih baik.



Gambar 5-3 Perbandingan *ED* ketika *popSize* 20 sampai dengan 200



Gambar 5-4 Perbandingan waktu eksekusi ketika *popSize* 20 sampai dengan 200

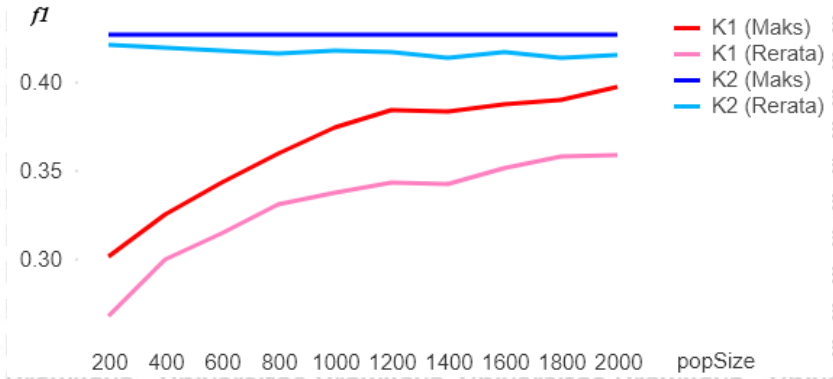
Sesuai dengan yang dituliskan sebelumnya, bahwa agar lebih meyakinkan hasil dari pengujian jumlah populasi, maka dilakukan pengujian kembali untuk jumlah populasi 400 sampai 2000, dengan kelipatan 200. Hasil pengujian terhadap jumlah populasi 200 sampai dengan 2000 untuk representasi kromosom dapat dilihat pada Tabel 5-3. Sedangkan untuk representasi kedua ditampilkan pada Tabel 5-4. Dari Tabel 5-3 dapat diketahui bahwa dengan representasi kromosom pertama (K1), nilai maksimal *f1* (*fitness 1*) terus meningkat hingga *popSize* 2000, atau bisa dikatakan dengan *popSize* lebih besar masih akan mendapatkan nilai *f1* yang lebih besar pula. Namun representasi kromosom pertama masih tidak dapat menghasilkan solusi yang layak (*feasible)*. Hal ini juga terjadi pada nilai rerata *f1*, yang cenderung naik hingga *popSize* 2000. Berbeda dengan hasil dari representasi kromosom kedua (K2) yang dapat dilihat pada Tabel 5-4, nilai maksimal *f1* dan *ED* tetap sejak jumlah populasi (*popSize*) adalah 200. Sedangkan nilai rata-rata *f1* cenderung menurun. Agar dapat memperlihatkan perbandingkan hasil *f1* dan *f2* dari kedua representasi kromosom (K1 dan K2), maka dari Tabel 5-3 dan Tabel 5-4 digambarkan dalam bentuk grafik pada Gambar 5-5 dan Gambar 5-6. Sedangkan untuk memperjelas perbedaan rerata dan maksimal jarak *Euclidean* antara *f1* dan *f2* (*ED*) antara kedua representasi kromosom terhadap peningkatan jumlah populasi digambarkan pada Gambar 5-7.

Tabel 5-3 Hasil K1 ketika jumlah populasi 200 sampai dengan 2000

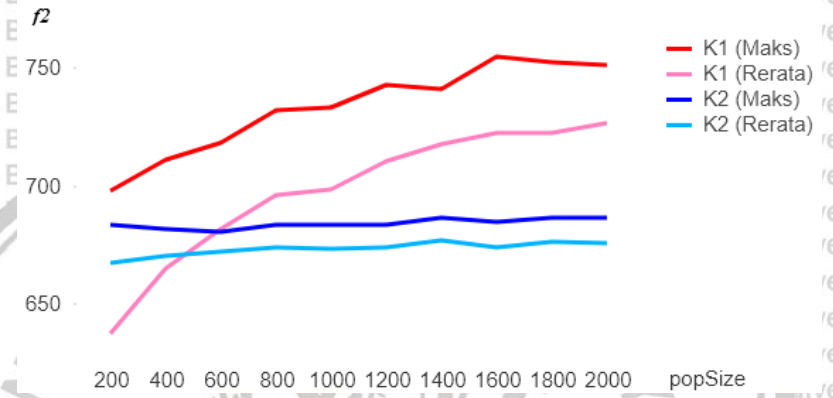
popSize	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
200	0.268	637.547	0.796	0.302	698.095	0.902	24.137	0
400	0.300	665.256	0.886	0.325	711.298	0.965	64.332	0
600	0.315	681.846	0.934	0.344	718.595	1.001	134.914	0
800	0.331	696.249	0.981	0.359	732.130	1.037	234.380	0
1000	0.337	698.506	0.995	0.375	733.565	1.067	409.604	0
1200	0.343	710.791	1.022	0.384	743.060	1.093	546.551	0
1400	0.343	718.023	1.034	0.384	741.127	1.096	705.280	0
1600	0.352	722.587	1.054	0.388	755.073	1.094	959.785	0
1800	0.358	722.727	1.064	0.390	752.305	1.113	1280.011	0
2000	0.359	726.803	1.072	0.398	751.443	1.124	1537.026	0

Tabel 5-4 Hasil K2 ketika *popSize* 200 sampai dengan 2000

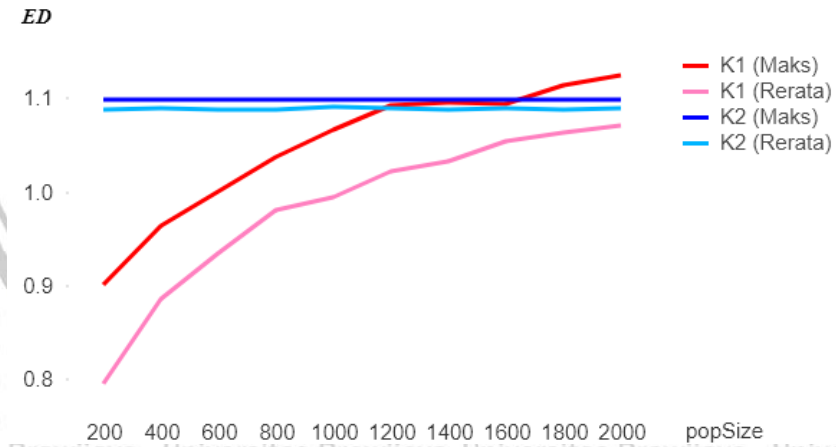
popSize	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
200	0.421	667.403	1.089	0.426	684	1.098	23.483	200
400	0.419	670.685	1.090	0.426	682	1.098	69.036	400
600	0.418	672.264	1.089	0.426	681	1.098	146.082	600
800	0.416	674.196	1.089	0.426	684	1.098	258.489	800
1000	0.418	673.888	1.091	0.426	684	1.098	402.457	1000
1200	0.417	673.958	1.089	0.426	684	1.098	614.489	1200
1400	0.414	677.213	1.089	0.426	687	1.098	887.443	1400
1600	0.417	674.402	1.090	0.426	685	1.098	1326.240	1600
1800	0.414	676.859	1.088	0.426	687	1.098	1598.538	1800
2000	0.415	676.107	1.090	0.426	687	1.098	2039.133	2000



Gambar 5-5 Perbandingan *f1* ketika *popSize* 200 sampai dengan 2000



Gambar 5-6 Perbandingan nilai *f2* ketika *popSize* 200 sampai dengan 2000



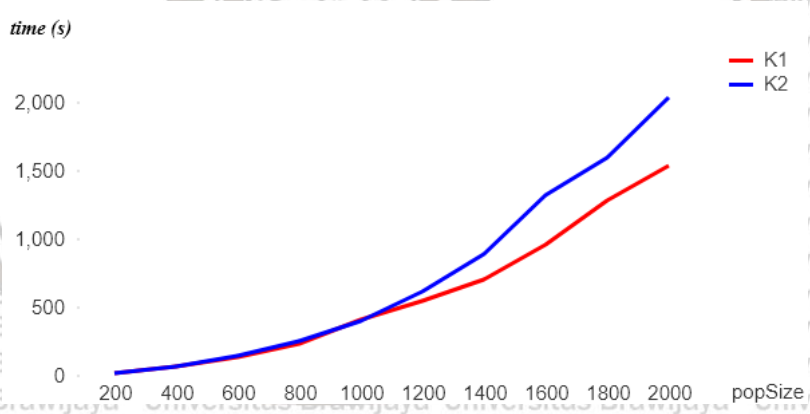
Gambar 5-7 Perbandingan *ED* ketika *popSize* 200 sampai dengan 2000

Tabel 5-3 menunjukkan bahwa dengan menggunakan representasi kromosom pertama NSGA-II tidak menghasilkan solusi yang layak (*feasible*) hingga jumlah populasinya 2000. *Constraint* yang dilanggar adalah *constraint* ke-4, yaitu nilai minimal kemampuan finansial mahasiswa (E_m) pada kategori *k* harus lebih besar dari pada nilai maksimal dari kemampuan finansial mahasiswa pada kategori sebelumnya (*k-1*). Hal ini terjadi karena terlalu acaknya pembentukan kromosom ketika proses inisialisasi. Begitu juga dengan metode rekombinasi (operator genetika) yang digunakan yaitu *One-cut-poin Crossover* dan *Simple-random Mutation* yang tidak mengarahkan konfigurasi kromosom untuk menghindari *constraint* ke-4. Operator genetika yang digunakan pun mempunyai kemungkinan untuk merusak

konfigurasi kromosom yang layak. Berdasarkan hasil ini dan sesuai dengan yang dibahas pada subbab 2.3.6, maka dapat disimpulkan bahwa ketika menggunakan representasi kromosom pertama mungkin diperlukan metode perbaikan (*repair method*) kromosom.

Gambar 5-5 memperlihatkan bahwa representasi kromosom pertama (K1) tidak dapat mengungguli representasi kromosom kedua (K2) dalam perolehan nilai *f1* (*fitness 1*) baik secara rerata dan maksimal. Hasil yang berbeda ditunjukkan oleh Gambar 5-6, representasi kromosom pertama (K1) mengungguli representasi kromosom kedua dalam rerata *f2* (nilai *fitness 2*) ketika menggunakan jumlah populasi (*popSize*) diatas 600. Demikian juga dengan nilai maksimal *f1*, representasi kromosom pertama unggul ketika jumlah populasi diatas 200. Karena adanya perbedaan hasil *f1* dan *f2* antara representasi kromosom pertama (K1) dan kedua (K2), maka jarak *Ecluidean* antara *f1* dan *f2* (*ED*) yang dapat memberikan pandangan yang lebih baik.

Gambar 5-7 menunjukkan bahwa representasi kromosom pertama (K1) mengungguli nilai rerata *ED* (*Euclidean* distance dari *f1* dan *f2*) dari representasi kromosom kedua (K2) ketika menggunakan jumlah populasi 1800 atau lebih, meskipun K1 tidak menghasilkan solusi yang layak (*feasible*). Sedangkan nilai maksimal *ED* dari representasi kromosom kedua (K2) tidak berubah mulai jumlah populasi (*popSize*) 200 (seperti yang dapat dilihat pada Tabel 4-1, Tabel 5-4, Gambar 5-3 dan Gambar 5-6). Sedangkan kita dapat membandingkan waktu eksekusi menggunakan Gambar 5-8, representasi kromosom pertama (K1) dengan jumlah populasi 1800 membutuhkan waktu 1280.011 detik atau sekitar 21 menit 20 detik. Dan representasi kromosom kedua (K2) dengan jumlah populasi 200 hanya sekitar 23.483 detik. Fakta tersebut masih menunjukkan bahwa representasi kromosom kedua (K2) lebih efisien dari pada representasi kromosom pertama (K1).



Gambar 5-8 Perbandingan waktu eksekusi ketika *popSize* 200 sampai dengan 2000

Pada Gambar 5-8 terlihat bahwa representasi kromosom kedua (K2) memiliki waktu eksekusi yang lebih tinggi dibanding dengan representasi kromosom pertama (K1) yaitu mulai jumlah populasi ke 1200 sampai 2000. Seperti yang sudah diketahui bahwa representasi kromosom kedua (K2) lebih pendek dari pada representasi pertama (K1). Panjang kromosom representasi kedua (K2) adalah 10 dan panjang representasi pertama (K1) adalah 100. Alasan tentang representasi kromosom kedua (K2) memiliki waktu eksekusi yang lebih lama, meskipun dengan kromosom yang lebih pendek adalah karena terjadi pemetaan antara mahasiswa dengan kategori UKT-nya (U_m) berdasarkan batas-batas kemampuan finansial mahasiswa (E_m) setiap kategori yang tersirat pada konfigurasi kromosom. Pemetaan tersebut terjadi sebelum proses perhitungan *fitness* dan *constraint* terjadi. Sedangkan untuk

representasi kromosom pertama (K1), konfigurasi kromosom sudah menyatakan kategori UKT dari mahasiswa (K_m).

Tabel 5-5 menunjukkan sebaran hasil optimasi dari NSGA-II menggunakan kedua representasi kromosom. Sebaran hasil yang ditunjukkan pada Tabel 5-5 dihasilkan ketika menggunakan jumlah populasi 2000, jumlah iterasi 3000, serta operator genetika *One-cut-poin Crossover* dan *Simple-random Mutation*. Jika kita cermati Tabel 5-5, hasil K2 lebih variatif dibandingkan dengan hasil K1. Hal ini ditunjukan dengan total kolom jumlah (*distinct*) pada K2 lebih besar dibandingkan K1, sedangkan jumlah kromosom yang dihasilkan sama yaitu sesuai dengan jumlah populasinya (*popSize*). Sedangkan konfigurasi kromosom terbaik (yang diurutkan berdasarkan *crowding distance*) dapat dilihat pada Tabel 5-6.

Tabel 5-5 Contoh sebaran hasil K1 dan K2 ketika *popSize* 2000

K1					K2				
f1	***f2***	***ED***	**Jumlah**	**Jumlah (*distinct*)**	***f1***	***f2***	***ED***	**Jumlah**	**Jumlah (*distinct*)**
0.361	751.185	1.114	3	1	0.4	687	1.078	5	5
0.362	751.083	1.115	5	1	0.401	686	1.077	5	5
0.363	750.408	1.115	2	1	0.404	685	1.082	10	10
0.363	750.305	1.115	15	2	0.404	684	1.081	7	6
0.364	749.63	1.115	4	1	0.406	683	1.083	1865	1306
0.364	749.528	1.115	1747	1	0.407	682	1.083	13	13
0.365	748.853	1.116	8	1	0.408	681	1.084	7	5
0.367	748.733	1.118	7	1	0.409	680	1.084	4	1
0.367	748.63	1.119	8	1	0.412	679	1.088	2	2
0.368	747.955	1.119	8	1	0.413	678	1.088	5	5
…	…	…	…	…	0.415	677	1.09	24	20
0.382	738.368	1.124	3	1	0.417	676	1.092	4	4
…	…	…	…	…	0.417	675	1.092	2	2
0.391	725.15	1.119	6	1	0.419	674	1.094	10	9
0.391	724	1.118	4	1	0.421	673	1.096	3	3
0.392	723.578	1.118	2	1	0.422	672	1.096	3	3
0.393	723.223	1.119	7	1	0.424	671	1.098	11	11
0.393	722.8	1.119	2	1	0.424	670	1.098	3	1
0.393	722.125	1.118	2	1	0.425	669	1.098	5	3
0.394	720.553	1.118	4	1	0.426	667	1.096	2	2
0.396	719.775	1.119	3	1	0.426	666	1.096	4	4
0.396	717.003	1.115	5	1	0.426	663	1.092	3	3
0.398	716.225	1.116	8	1	0.426	660	1.088	3	3
Total			**2000**	**56**	**Total**			**2000**	**1426**

Tabel 5-6 Kromosom urutan 1-10 berdasarkan *crowding distance* ketika *popSize*=2000

No	Kromosom
1	[1475138, 2616029, 4621095, 6800742, 8750000, 1360303, 3602218, 6095465, 6428826, 8775000]
2	[1383580, 2477840, 3305861, 6282805, 8750000, 1360303, 1974321, 2514255, 7602531, 8775000]
3	[1213575, 2730631, 5136816, 6282805, 8750000, 1360303, 2071241, 2514255, 7996333, 8775000]
4	[1213575, 2720594, 5136816, 6654867, 8750000, 1379404, 2799712, 5284740, 7602531, 8775000]
5	[1577990, 2477840, 4858094, 5565178, 8750000, 1360303, 1974321, 2514255, 7602531, 8775000]
6	[1213575, 2720594, 6135643, 6460338, 8750000, 1360303, 1974321, 2514255, 6647969, 8775000]
7	[1213575, 2477840, 4597662, 6572336, 8750000, 1360303, 1974321, 2514255, 6647969, 8775000]
8	[1586387, 2730631, 4405477, 6282805, 8750000, 1360303, 1974321, 2514255, 6294639, 8775000]
9	[1213575, 2477840, 5136816, 6282805, 8750000, 1360303, 1974321, 2514255, 6380504, 8775000]
10	[1550657, 2604193, 4858094, 6282805, 8750000, 1368856, 1974321, 2514255, 6647969, 8775000]

Tabel 5-7 Hasil K1 terhadap maksimal generasi (*maxGen*)

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
3000	0.271	636.115	0.797	0.295	682.595	0.864	18.440	0
2000	0.268	631.270	0.785	0.299	680.843	0.878	12.219	0
1000	0.267	635.995	0.792	0.288	685.603	0.872	6.207	0
800	0.267	633.259	0.788	0.292	678.298	0.864	5.092	0
600	0.272	638.406	0.803	0.300	689.603	0.898	3.892	0
400	0.277	642.894	0.816	0.303	688.140	0.898	2.743	0
200	0.269	633.797	0.791	0.293	683.928	0.879	1.621	0
100	0.263	643.803	0.800	0.300	689.473	0.894	1.042	0
80	0.254	632.443	0.768	0.282	674.963	0.845	0.935	0
60	0.239	642.643	0.769	0.266	663.590	0.818	0.831	0
40	0.229	617.680	0.714	0.253	655.875	0.773	0.725	0
20	0.209	581.056	0.626	0.230	631.080	0.735	0.628	0
10	0.194	538.736	0.533	0.210	576.943	0.597	0.563	0

5.2 Pengujian Representasi Kromosom Dan Pengaruh Maksimal Generasi

Pengujian terhadap maksimal generasi (lebih lanjut disebut *maxGen*) bertujuan untuk mengetahui jumlah iterasi yang paling sedikit, tetapi bisa menghasilkan nilai *fitness* terbaik. Jumlah populasi (*popSize*) yang digunakan dalam pengujian ini adalah 200. Sedangkan operator genetika yang digunakan masih sama yaitu (*one-cut-point crossover* dan *simple-random mutation*). Dalam pengujian terhadap jumlah populasi didapatkan bahwa representasi kromosom pertama (K1) tidak dapat menghasilkan solusi yang layak (*feasible*).

Meskipun demikian representasi kromosom pertama (K1) masih dilibatkan dalam pengujian terhadap *maxGen*, agar dapat diketahui pengaruh *maxGen* terhadap pergerakan perolehan nilai *fitness*.

Hasil pengujian terhadap *maxGen* untuk representasi kromosom pertama (K1) ditampilkan pada Tabel 5-7. Sedangkan hasil representasi kromosom kedua (K2) ditampilkan pada Tabel 5-8. Jika dilihat pada Tabel 5-7, hasil pengujian terhadap *maxGen* pada representasi kromosom pertama (K1) sama dengan hasil pengujian terhadap jumlah populasi yaitu representasi kromosom (K1) tidak menghasilkan solusi yang layak. Pada Tabel 5-8, representasi kromosom kedua (K2) menghasilkan solusi yang layak, bahkan ketika maksimal iterasinya hanya 10 saja. Untuk mengetahui pergerakan jumlah solusi layak per iterasi, maka dilakukan pengujian untuk iterasi 1 sampai 10.

Pada Tabel 5-8 rerata *ED* (*Euclidean Distance* dari *f1* dan *f2*) yang tertinggi yaitu 1.091, diperoleh ketika nilai maksimal iterasi adalah 100. Hal ini juga terjadi pada nilai rerata *f1* dan *f2*. Sedangkan nilai terbaik dari *ED* yaitu 1.098, diperoleh bahkan mulai nilai maksimal iterasi 10. Nilai *ED* terbaik ini tidak diperoleh oleh representasi kromosom pertama bahkan hingga iterasi ke 3000. Sehingga bisa diambil kesimpulan sementara bahwa nilai maksimal iterasi yang terbaik untuk representasi kromosom kedua (K2) adalah 100.

Tabel 5-8 Pengaruh *maxGen* terhadap hasil K2

maxGen	**Rerata *f1***	**Rerata *f2***	**Rerata *ED***	**Maks *f1***	**Maks *f2***	**Maks *ED***	**Rerata waktu (detik)**	***Rerata FeaSol***
3000	0.422	667.421	1.090	0.426	680	1.098	17.599	200
2000	0.421	667.659	1.089	0.426	681	1.098	11.872	200
1000	0.422	666.920	1.089	0.426	687	1.098	5.881	200
800	0.422	666.938	1.090	0.426	682	1.098	5.485	200
600	0.422	667.381	1.090	0.426	681	1.098	3.608	200
400	0.423	666.220	1.090	0.426	679	1.098	2.522	200
200	0.423	666.322	1.091	0.426	682	1.098	1.603	200
100	0.423	667.210	1.091	0.426	681	1.098	1.053	200
80	0.423	666.669	1.090	0.426	681	1.098	0.936	200
60	0.423	666.293	1.090	0.426	683	1.098	0.834	200
40	0.423	665.174	1.089	0.426	687	1.098	0.710	200
20	0.422	665.430	1.088	0.426	680	1.098	0.595	200
10	0.420	665.783	1.084	0.426	678	1.098	0.548	200

Pengujian terhadap nilai *maxGen* 1 sampai 10 ditampilkan dalam Tabel 5-9. Seperti yang tampil pada Tabel 5-9, bahwa ketika nilai *maxGen* adalah sembilan, jumlah solusi (kromosom) yang layak (*feasible*) sama dengan jumlah populasi yaitu 200. Kita juga bisa melihat kenaikan rata-rata jumlah solusi yang layak (pada kolom *Rerata FeaSol*) seiring dengan naiknya nilai

maxGen. Hal ini menunjukkan bahwa representasi kromosom kedua (K2) adalah representasi yang efisien untuk penentuan UKT, karena hanya dengan 9 kali iterasi, NSGA-II sudah menghasilkan solusi yang layak sejumlah sama dengan jumlah populasinya (*popSize*).

Tabel 5-9 Hasil K2 ketika *maxGen* 1 sampai 10

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
10	0.420	665.783	1.084	0.426	678	1.098	0.548	200
9	0.421	663.515	1.084	0.426	677	1.093	0.526	200
8	0.421	662.871	1.083	0.426	674	1.093	0.534	196.8
7	0.420	664.445	1.082	0.426	675	1.093	0.510	191.1
6	0.420	664.127	1.082	0.426	677	1.093	0.518	181.15
5	0.420	663.784	1.082	0.426	675	1.093	0.515	127.85
4	0.420	662.993	1.080	0.425	674	1.091	0.491	91.7
3	0.420	663.031	1.081	0.426	671	1.091	0.482	71.45
2	0.420	662.103	1.078	0.426	672	1.093	0.483	45.9
1	0.418	662.000	1.075	0.425	675	1.088	0.467	31.75

5.3 Pengujian Metode Rekombinasi (*Genetic Operator*)

Pada tahap pengujian metode rekombinasi ini bertujuan untuk mencari kombinasi metode rekomendasi terbaik untuk optimasi penentuan UKT. Pengujian dilakukan untuk kedua representasi kromosom dengan menggunakan jumlah populasi (*popSize*) 200 dan nilai maksimal generasi (*maxGen*) 100. Pengujian juga bertujuan untuk mencari kombinasi nilai *Cr* dan *Mr* yang terbaik. Nilai *Cr* mulai 0 sampai 1 dengan kelipatan 0.1, dan nilai *Mr=1-Cr*. Masing-masing pengujian diulang sebanyak 20 kali. Sehingga untuk satu GO terdiri dari 220 percobaan, yaitu 11 kombinasi *Cr* dan *Mr* dan diulang sebanyak 20 kali.

Hasil pengujian metode rekomendasi untuk representasi kromosom pertama (K1) ditunjukkan pada Tabel 5-10. Tabel 5-10 menunjukkan bahwa kombinasi *UC-EM* (*uniform crossover* dan *random-exchange mutation*) memiliki hasil yang terbaik untuk representasi kromosom pertama. Pada Tabel 5-10 juga terlihat bahwa representasi kromosom pertama (K1) dengan kelima kombinasi *GO* masih belum bisa menghasilkan solusi yang *feasible*. Jika kita perhatikan lebih detail, untuk representasi kromosom pertama (K1), metode *uniform crossover (UC)* lebih baik dibandingkan dengan metode *one-cut-poin crossover (OC)*. *UC* lebih baik dibandingkan dengan *OC*, karena untuk representasi kromosom pertama (K1) yang memiliki kromosom yang panjang (jumlah *allele* yang banyak), maka diperlukan pertukaran dengan titik potong lebih banyak. Namun, waktu eksekusi *UC* lebih lama dibandingkan dengan *OC*, hal ini dikarenakan untuk setiap *allele* dibangkitkan bilangan random, bilangan random inilah yang digunakan sebagai patokan untuk mengisi *allele* baru ini dengan *allele* pada kromosom induk pertama atau kedua.

Tabel 5-10 juga menunjukkan bahwa untuk mutasi, *EM* (*random-exchange mutation*) cenderung memiliki hasil yang lebih baik dibandingkan dengan *SM* (*simple-random mutation*). Metode mutasi *SM* hanya merubah satu *allele* dengan bilangan acak, sedangkan *EM* menukarkan 2 titik *allele*. Hal ini memperkuat pernyataan sebelumnya bahwa, karena representasi kromosom pertama (K1) cenderung memiliki jumlah allele yang banyak, maka diperlukan perubahan pada titik yang lebih banyak juga. Sedangkan waktu eksekusi *SM* dan *EM* tidak menunjukkan perbedaan yang signifikan.

Tabel 5-10 Hasil pengujian metode rekombinasi untuk K1

GO	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
OC-SM	0.256	635.547	0.777	0.315	702.590	0.926	**0.920**	0
OC-EM	0.281	671.196	0.870	0.363	738.378	1.073	**0.920**	0
UC-SM	0.353	712.564	1.041	0.423	768.925	1.182	0.952	0
UC-EM	**0.375**	**732.788**	**1.106**	**0.429**	**773.053**	**1.189**	0.954	0
RC-RM	0.355	718.903	1.054	0.415	764.465	1.164	0.957	0

Sedangkan untuk representasi kromosom kedua (K2) ditunjukkan pada Tabel 5-11. Pada Tabel 5-11 terlihat bahwa kombinasi *GO* terbaik adalah *RC-RM.* Hal ini terlihat dari hasil rerata *f1*, rerata *f2*, rerata *ED*, maksimal *f1*, dan maksimal ED yang dihasilkan *RC-RM* lebih dari kombinasi *GO* yang lain, meskipun dengan perbedaan yang kecil. Tabel 5-11 terlihat bahwa kombinasi yang melibatkan *SM* tidak mencapai nilai *ED* 1.114 (Maksimal *ED*). Kombinasi yang melibatkan *SM* juga tidak selalu menghasilkan jumlah *feasible solution* (solusi yang layak) yang sama dengan jumlah populasi yang digunakan. Hal ini ditunjukkan pada nilai *Rerata FeaSol* yang dihasilkan yaitu 184.273 dan 184.636*.* Untuk mengetahui lebih dalam tentang kombinasi *GO* dan pengaruhnya terhadap hasil NSGA-II untuk penentuan UKT, maka akan ditunjukkan hasil pengujian untuk setiap kombinasi *Cr* dan *Mr*.

Tabel 5-11 Hasil pengujian metode rekombinasi untuk K2

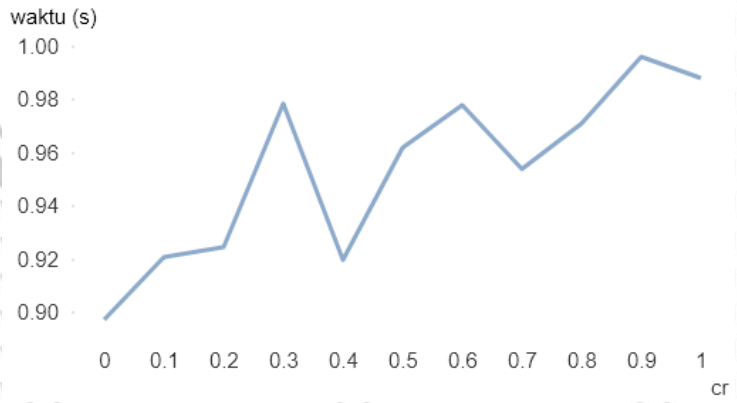
GO	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
OC-SM	0.423	665.464	1.090	0.426	679	1.098	1.111	184.273
OC-EM	0.426	670.005	1.101	**0.432**	**691**	1.114	1.144	**200**
UC-SM	0.424	664.843	1.091	0.426	681	1.098	1.111	184.636
UC-EM	**0.427**	668.759	1.101	**0.432**	686	1.114	1.112	**200**
RC-RM	**0.427**	**670.442**	**1.103**	**0.432**	688	**1.114**	**1.103**	**200**

Hasil pengujian *Cr* dan *Mr* untuk representasi kromosom pertama (K1) dan menggunakan metode rekombinasi *UC-EM* ditampilkan pada Tabel 5-12. Nilai rerata *fitness* terbaik ditampilkan dengan huruf yang ditebalkan, demikian juga untuk rerata waktu tersingkat. Rerata nilai *fitness* terbaik diperoleh ketika nilai *Cr* = 0.8 dan *Mr* = 0.2. *Nilai fitness* terbaik (Maks *f1*, *f2* dan *ED*) diperoleh pada nilai *Cr* dan *Mr* yang berbeda-beda. Sedangkan

rerata waktu eksekusi tersingkat diperoleh ketika *Cr* = 0 dan *Mr* = 1. Dengan kata lain waktu yang paling singkat adalah ketika hanya menggunakan mutasi *EM* (*Exchange Mutation*). Hal ini diperkuat dari kecenderungan naiknya waktu eksekusi seiring dengan bertambahnya nilai *Cr* yang ditunjukkan pada Gambar 5-9. Hal ini membuktikan bahwa waktu eksekusi metode *crossover UC* (*Uniform Crossover*) memiliki waktu eksekusi yang lebih lama dibandingkan dengan metode mutasi *EM* (*Exchange Mutation*).

Tabel 5-12 Pengaruh *Cr* dan *Mr* pada hasil K1 dengan metode rekombinasi *UC-EM*

Cr	*Mr*	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
0	**1**	0.212	509.952	0.506	0.228	520.468	0.536	**0.898**	0
0.1	**0.9**	0.296	687.320	0.917	0.329	720.878	1.008	0.921	0
0.2	**0.8**	0.329	716.064	1.011	0.373	741.043	1.081	0.925	0
0.3	**0.7**	0.357	731.070	1.076	0.383	753.46	1.118	0.979	0
0.4	**0.6**	0.367	738.753	1.103	0.405	766.575	1.151	0.920	0
0.5	**0.5**	0.378	745.928	1.132	0.410	771.258	1.167	0.962	0
0.6	**0.4**	0.386	750.032	1.150	0.412	772.052	1.175	0.978	0
0.7	**0.3**	0.386	753.182	1.155	0.415	**773.053**	1.175	0.954	0
0.8	**0.2**	**0.394**	**753.436**	**1.168**	0.424	771.915	1.188	0.971	0
0.9	**0.1**	0.393	752.263	1.165	0.421	772.932	**1.189**	0.996	0
1	**0**	0.389	745.317	1.147	**0.429**	769.045	1.183	0.988	0



Gambar 5-9 Pengaruh *Cr dan Mr* terhadap waktu eksekusi K1 dan *UC-EM*

Hasil pengujian *Cr* dan *Mr* untuk representasi kromosom kedua (K2) dan menggunakan metode rekombinasi *RC-RM* ditampilkan pada Tabel 5-13. Angka yang ditebalkan pada Tabel 5-13, menunjukkan nilai rerata dan maksimal *fitness* terbaik, serta rerata waktu tersingkat. Rerata nilai *f1* terbaik diperoleh ketika nilai *Cr* = {0.4, 0.5, 0.6, 0.9} dan *Mr* = {0.6, 0.5, 0.4, 0.1}. Rerata nilai *f2* terbaik diperoleh ketika nilai *Cr* = 0.7 dan *Mr* = 0.3, dan rerata *ED* terbaik diperoleh ketika nilai *Cr* = {0.4, 0.7} dan *Mr* = {0.6, 0.3}. *Nilai f1* dan terbaik diperoleh ketika

nilai *Cr* = 0.1 sampai *Cr* = 0.9. *Nilai f2* terbaik diperoleh ketika nilai *Cr* = 0.5 dan *Mr* = 0.5. Sedangkan rerata waktu eksekusi tersingkat diperoleh ketika *Cr* = 0 dan *Mr* = 1. Dengan kata lain waktu yang paling singkat adalah ketika hanya menggunakan mutasi *EM* (*Exchange Mutation*). Hal ini membuktikan bahwa waktu eksekusi metode *crossover UC* (*Uniform Crossover*) memiliki waktu eksekusi yang lebih lama dibandingkan dengan metode mutasi *EM*.

Tabel 5-13 Pengaruh *Cr* dan *Mr* pada hasil K2 dengan metode rekombinasi *RC-RM*

Cr	*Mr*	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	*Rerata FeaSol*
0	**1**	0.423	666.526	1.092	0.431	680	1.105	1.106	200
0.1	**0.9**	0.428	668.844	1.104	**0.432**	687	**1.114**	1.100	200
0.2	**0.8**	0.428	670.626	1.106	**0.432**	695	**1.114**	1.063	200
0.3	**0.7**	0.428	671.256	1.106	**0.432**	688	**1.114**	1.097	200
0.4	**0.6**	**0.429**	671.135	**1.107**	**0.432**	691	**1.114**	1.062	200
0.5	**0.5**	**0.429**	670.390	1.106	**0.432**	**695**	**1.114**	1.063	200
0.6	**0.4**	**0.429**	670.216	1.106	**0.432**	689	**1.114**	1.098	200
0.7	**0.3**	0.428	**671.624**	**1.107**	**0.432**	691	**1.114**	1.062	200
0.8	**0.2**	0.428	670.842	1.106	**0.432**	693	**1.114**	1.088	200
0.9	**0.1**	**0.429**	669.160	1.106	**0.432**	689	**1.114**	1.057	200
1	**0**	0.424	664.108	1.090	0.426	683	1.098	**1.041**	200

5.4 Pengujian Pengaruh Metode *Repair*

Hasil pengujian pengaruh metode repair disajikan pada Tabel 5-14 dan Tabel 5-15. Pengujian ini dilakukan dengan *popSize* = 200, iterasi maksimal = 100. Sedangkan metode rekombinasi (operator genetika) berbeda untuk kedua representasi kromosom. Untuk representasi kromosom pertama (K1) menggunakan metode rekombinasi *UC-EM* yaitu perpaduan antara metode *crossover* = *Uniform crossover*, dan metode *mutasi* = random *exchange*. Metode rekombinasi *UC-EM* tersebut dijalankan dengan *crossover rate* (*cr*) = 0.8 dan *mutation rate* (*mr*) = 0.2. Sedangkan untuk representasi kromosom kedua (K2) menggunakan metode *RC-RM* yaitu perpaduan antara metode *crossover* = *Random crossover* dengan *crossover rate* (*cr*) = 0.8 dan metode mutasi = *random exchange* dengan *mutation rate* (*mr*) = 0.2. Tabel 5-14 dan Tabel 5-15, K1-*NoRepair* mewakili *NSGA-II* menggunakan representasi kromosom pertama tanpa metode *repair*. Sedangkan K1-*Repair* mewakili *NSGA-II* menggunakan representasi kromosom pertama dan menggunakan metode repair.

Tabel 5-14 Pengaruh metode *repair* pada perolehan nilai *fitness* dan waktu eksekusi

Representasi dan Repair	**Rerata *f1***	**Rerata *f2***	**Rerata *ED***	**Maks *f1***	**Maks *f2***	**Maks *ED***	**Rerata waktu (detik)**
K1-*NoRepair*	0.394	753.436	1.168	0.424	771.915	1.188	0.971
K1-*Repair*	0.411	706.287	1.124	0.423	725.005	1.158	1.460
K2-*NoRepair*	0.429	670.390	1.106	0.432	695	1.114	1.063
K2-*Repair*	0.428	671.300	1.106	0.432	695	1.114	1.081

Tabel 5-15 Pengaruh metode repair pada keragaman solusi yang dihasilkan

Representasi dan Repair	**Rerata *FeaSol***	**Rerata *DistinctSol***
K1-*NoRepair*	0	0
K1-*Repair*	200	2.55
K2-*NoRepair*	200	168.35
K2-*Repair*	200	165.4

Seperti yang dapat dilihat pada Tabel 5-15 bahwa metode *repair* sangat berpengaruh pada representasi kromosom pertama (K1). Hal ini dapat dilihat ketika tidak menggunakan metode *repair* (*K1-Repair*), representasi kromosom pertama (K1) tidak menghasilkan solusi yang layak. Namun, ketika menggunakan metode *repair* representasi pertama menghasilkan solusi yang layak sejumlah dengan jumlah *popSize*. Jika dilihat pada Tabel 5-14, metode repair pada representasi kromosom pertama juga meningkatkan waktu eksekusi yaitu sebesar 0.489 detik. Jadi metode *repair* berhasil memperbaiki hasil dari representasi kromosom pertama dengan konsekuensi penambahan waktu eksekusi.

Dapat dilihat pada Tabel 5-14, bahwa metode *repair* pada representasi kromosom kedua (K2) tidak dapat meningkatkan nilai *fitness* yang dihasilkan (bandingkan antara K2-*NoRepair* dan K2-*Repair*). Metode *repair* pada K2 hanya meningkatkan waktu eksekusi sebesar 0.018. Tabel 5-15 menjelaskan bahwa metode *repair* pada K2 tidak meningkatkan keragaman dari solusi yang dihasilkan. Maka untuk representasi kromosom kedua (K2), tidak diperlukan metode *repair* yang diusulkan. Karena metode *repair* hanya meningkatkan waktu eksekusi tanpa ada kenaikan nilai *fitness* ataupun jumlah solusi yang layak dan keragaman solusi.

Secara umum hasil *NSGA-II* yang terbaik diperoleh ketika menggunakan representasi kromosom kedua (*K2*) tanpa menggunakan metode *repair* (K2-*NoRepair*). Meskipun pada *K1-Repair* menghasilkan nilai *fitness* kedua (*f2*) lebih baik dari pada yang dihasilkan oleh K2-*NoRepair* (lihat Tabel 5-14). Dengan perbedaan nilai *fitness 2* (*f2*) yang lebih besar antara *K1-Repair* dan K2-*NoRepair*, dibandingkan dengan perbedaan nilai *fitness* 1 (*f1*), sehingga nilai *ED* (baik rerata maupun maksimal) dari *K1-Repair* lebih baik. Namun keunggulan dari *K1-Repair* tersebut menjadi tidak berarti ketika dilihat nilai rerata nilai *fitness 1* (*f1*) dari K2-*NoRepair* lebih baik dari pada *K1-Repair*. Jika kita bandingkan menggunakan metode dominasi Pareto, maka secara rerata solusi dari *K1-Repair* tidak mendominasi solusi dari K2-*NoRepair*. Demikian juga ketika dibandingkan berdasarkan waktu eksekusi dan juga keragaman solusi (Rerata *DistinctSol*) yang dihasilkan. Rerata waktu eksekusi dari K1-*Repair* 1.460 detik dan K2-*NoRepair* adalah 1.063, atau bisa dikatakan waktu eksekusi K2-*NoRepair* lebih cepat 0.397

detik dibandingkan dengan K1-*Repair*. Seperti yang dapat dilihat pada Tabel 5-15, *K1-Repair* hanya rata-rata menghasilkan 2.55 solusi yang berbeda. Sedangkan K2-*NoRepair* rata-rata menghasilkan 168.35 solusi yang berbeda.

Kecilnya rerata jumlah solusi yang berbeda (Rerata *DistinctSol*) hasil dari K1-*Repair*, menunjukkan bahwa sangat sulitnya representasi kromosom pertama (K1) mencari kombinasi nilai kromosom yang layak (tidak melanggar *constraint*). Sehingga solusi yang layak tersebut mayoritas berasal dari metode *repair*. Sedangkan jika kita lihat algoritme metode *repair* pada Gambar 4-10, metode *repair* pada K1 ini akan menghasilkan kromosom (kombinasi gen) yang sama setiap kali metode ini dijalankan. Sulitnya *NSGA-II* mencari kromosom (solusi) yang layak ketika menggunakan K1 ini dipengaruhi oleh jumlah gen (panjang kromosom) yang terlalu banyak. Semakin banyak jumlah gen, maka semakin banyak pula kemungkinan kombinasi gen yang dihasilkan. Pada K1, satu gen memiliki kemungkinan 6 kombinasi (yaitu sesuai dengan kemungkinan kategori UKT yang diterima oleh satu mahasiswa). Maka untuk 100 mahasiswa (yang digunakan pada data pengujian), secara keseluruhan akan terdapat 6^{100} kombinasi gen (kromosom).

5.5 Pengujian Bentuk Fungsi *Fitness 1* (*f1*)

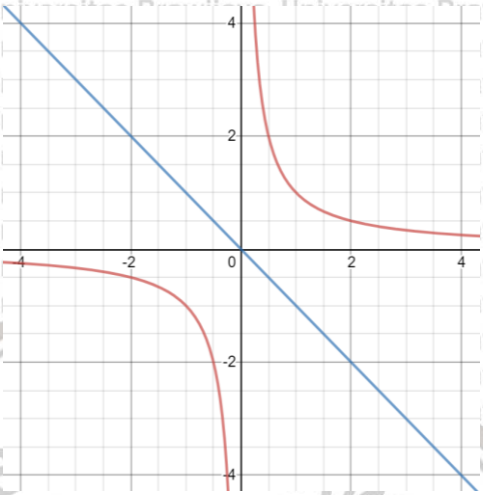
Hasil pengujian pengaruh bentuk fungsi *fitness 1* (*f1*) disajikan pada Tabel 5-16. Pengujian ini dilakukan dengan *popSize* = 200, iterasi maksimal = 100. Sedangkan metode rekombinasi (operator genetika) berbeda untuk kedua representasi kromosom. Untuk representasi kromosom pertama (K1) menggunakan metode rekombinasi *UC-EM* yaitu perpaduan antara metode *crossover* = *Uniform crossover*, dan metode *mutasi* = random *exchange*. Metode rekombinasi *UC-EM* tersebut dijalankan dengan *crossover rate* (*cr*) = 0.8 dan *mutation rate* (*mr*) = 0.2. Sedangkan untuk representasi kromosom kedua (K2) menggunakan metode *RC-RM* yaitu perpaduan antara metode *crossover* = *Random crossover* dengan *crossover rate* (*cr*) = 0.8 dan metode mutasi = *random exchange* dengan *mutation rate* (*mr*) = 0.2. Pada Tabel 5-16 K1-*f1*V1 mewakili NSGA-II yang menggunakan representasi kromosom pertama (K1) dan bentuk *f1* pertama. K2-*f1*V2 mewakili NSGA-II yang menggunakan representasi kromosom kedua (K2) dan bentuk *f1* kedua.

Tabel 5-16 Pengaruh bentuk *fitness 1* (*f1*)

Representasi Kromosom dan Bentuk *f1*	Rerata $\sum_{m=1}^{M}\|U_m - E_m\|$	Rerata *f2*	Maks $\sum_{m=1}^{M}\|U_m - E_m\|$	Maks *f2*	Rerata waktu (detik)	Rerata *FeaSol*
K1-*f1*V1	243.586	706.287	236.565	725.005	1.4601	200
K1-*f1*V2	243.799	706.239	239.12	723.645	1.5781	200
K2-*f1*V1	233.746	671.299	231.425	695	1.0811	200
K2-*f1*V2	233.488	670.862	231.425	691	1.0814	200

Berdasarkan Tabel 5-16 diketahui bahwa bentuk *fitness 1* (*f1*) tidak memiliki pengaruh yang signifikan terhadap nilai rerata $\sum_{m=1}^{M}|U_m - E_m|$, rerata *f2*, maksimal $\sum_{m=1}^{M}|U_m - E_m|$, maksimal *f2*, rerata waktu eksekusi, dan rerata jumlah solusi yang layak (*FeaSol*). Hal ini dikarenakan bentuk *f1* tidak mempengaruhi pergerakan pencarian solusi. Namun hanya sebagai acuan untuk menilai solusi yang lebih baik. Ilustrasi bentuk fungsi *fitness 1* (*f1*) dapat dilihat pada Tabel 5-10. Jika kita asumsikan $x = \sum_{m=1}^{M}|U_m - E_m|$, maka Formula (9) bisa dipersingkat menjadi $f(x) = m/x$, sehingga bentuk grafik akan menyerupai garis merah pada

Gambar 5-10. Demikian juga untuk Formula (10), dapat dipersingkat menjadi $f(x) = -x/m$ dan bentuk grafik akan menyerupai garis biru pada Gambar 5-10. Jika $x = \sum_{m=1}^{M} |U_m - E_m|$, maka x tidak mungkin bernilai negatif, sehingga kita bisa mengabaikan sumbu horizontal bernilai negatif. Dengan diabaikannya sumbu horizontal bernilai negative, maka secara umum arah grafik dari kedua fungsi tersebut sama, yaitu ketika *x* lebih besar, maka nilai fungsinya (*f(x)*) lebih kecil, demikian pula sebaliknya.



Gambar 5-10 Ilustrasi grafik bentuk *fitness 1*

5.6 Pengujian Metode Inisialisasi

Tabel 5-17 Hasil NSGA-II dengan NCPR untuk maksimal generasi 10-100

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
100	-2.332	670.417	1.201	-2.315	687	1.208	0.926	200
90	-2.337	671.034	1.201	-2.315	692	1.209	0.866	200
80	-2.337	670.713	1.200	-2.315	691	1.208	0.827	200
70	-2.327	669.594	1.200	-2.314	687	1.208	0.720	200
60	-2.335	671.212	1.201	-2.315	693	1.208	0.697	200
50	-2.332	670.712	1.201	-2.314	688	1.208	0.640	200
40	-2.336	671.786	1.202	-2.314	692	1.209	0.568	200
30	-2.343	671.133	1.200	-2.315	687	1.208	0.516	200
20	-2.352	668.050	1.195	-2.315	686	1.207	0.460	200
10	-2.376	664.969	1.188	-2.322	685	1.204	0.423	200

Hasil pengujian pengaruh metode inisialisasi NCPR disajikan pada Tabel 5-17. Sedangkan Tabel 5-18 menyajikan hasil metode inisialisasi NCQPR. Pengujian ini dilakukan dengan *popSize* = 200, iterasi maksimal mulai 10 sampai dengan 100. Sedangkan metode rekombinasi (operator genetika) metode *RC-RM* yaitu perpaduan antara metode *random*

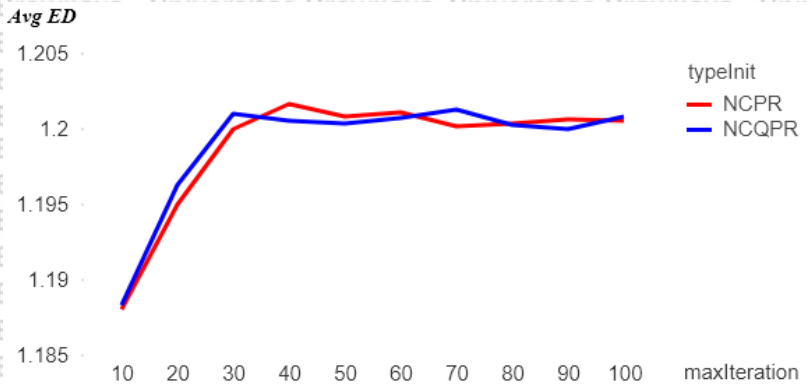
crossover dengan *crossover rate* (*cr*) = 0.8 dan metode mutasi *random exchange* dengan *mutation rate* (*mr*) = 0.2. Untuk mempermudah perbandingan dari kedua metode inisialisasi, maka dibuatlah grafik perbandingan rerata ED yang dapat dilihat pada Gambar 5-11. Selain itu untuk membandingkan waktu eksekusi antara kedua metode inisialisasi dapat dilihat pada Gambar 5-12.

Tabel 5-18 Hasil NSGA-II dengan NCQPR untuk maksimal generasi 10-100

maxGen	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
100	-2.329	670.350	1.201	-2.315	691	1.208	0.946	200
90	-2.329	669.697	1.200	-2.315	686	1.208	0.887	200
80	-2.331	670.141	1.200	-2.314	687	1.208	0.834	200
70	-2.334	671.217	1.201	-2.315	691	1.209	0.783	200
60	-2.332	670.549	1.201	-2.315	689	1.208	0.698	200
50	-2.334	670.506	1.200	-2.315	689	1.208	0.652	200
40	-2.335	670.707	1.201	-2.314	686	1.208	0.598	200
30	-2.339	671.535	1.201	-2.315	691	1.208	0.531	200
20	-2.354	669.330	1.196	-2.318	686	1.207	0.484	200
10	-2.375	665.030	1.188	-2.320	677	1.199	0.444	200

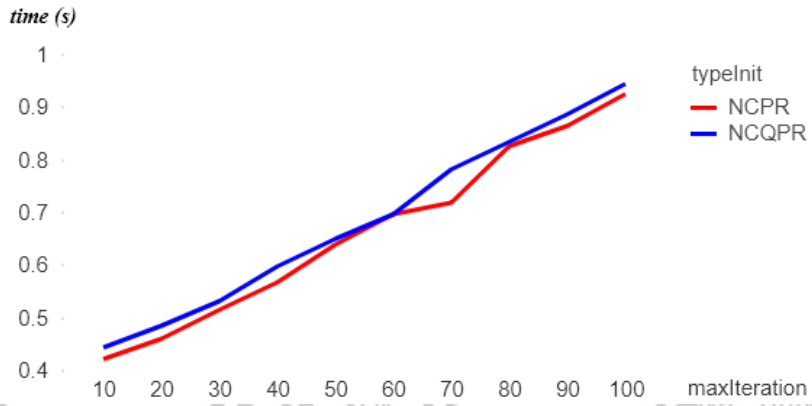
Berdasarkan Tabel 5-17 dan Hasil pengujian pengaruh metode inisialisasi NCPR disajikan pada Tabel 5-17. Sedangkan Tabel 5-18 menyajikan hasil metode inisialisasi NCQPR. Pengujian ini dilakukan dengan *popSize* = 200, iterasi maksimal mulai 10 sampai dengan 100. Sedangkan metode rekombinasi (operator genetika) metode *RC-RM* yaitu perpaduan antara metode *random crossover* dengan *crossover rate* (*cr*) = 0.8 dan metode mutasi *random exchange* dengan *mutation rate* (*mr*) = 0.2. Untuk mempermudah perbandingan dari kedua metode inisialisasi, maka dibuatlah grafik perbandingan rerata ED yang dapat dilihat pada Gambar 5-11. Selain itu untuk membandingkan waktu eksekusi antara kedua metode inisialisasi dapat dilihat pada Gambar 5-12.

Tabel 5-18, metode inisialisasi tidak memiliki pengaruh yang berarti pada perolehan nilai *fitness* dari NSGA-II. Hal ini dapat lebih jelas dilihat pada Gambar 5-11, yaitu grafik rerata *ED* (*euclidean distance*) dari nilai *fitness* (*f1* dan *f2*) hasil dari NSGA-II dengan kedua metode inisialisasi. Garis merah pada Gambar 5-11 merupakan hasil dari metode inisialisasi NCPR, sedangkan garis biru menunjukkan hasil dari metode inisialisasi NCQPR. Pada Gambar 5-11 terlihat bahwa nilai rata-rata *ED* hasil dari NSGA-II menggunakan kedua metode inisialisasi tersebut hasilnya relatif sama.



Gambar 5-11 Perbandingan rerata *ED* hasil dari NCPR dan NCQPR

Perbandingan waktu eksekusi dari NSGA-II dengan menggunakan metode inisialisasi NCPR dan NCQPR digambarkan pada grafik yang dapat dilihat pada Gambar 5-12. Garis merah pada Gambar 5-12 merupakan waktu dari metode inisialisasi NCPR, sedangkan garis biru menunjukkan waktu dari metode inisialisasi NCQPR. Terlihat jelas pada Gambar 5-12 bahwa garis biru selalu berada diatas garis merah. Hal tersebut dapat diartikan waktu yang dibutuhkan NCQPR lebih banyak diabandingkan dengan NCPR. Waktu yang dibutuhkan NCQPR lebih banyak dikarenakan kebutuhan waktu lebih dalam proses pembentukan deret *Quasi Random Sequence* (*QRS*).



Gambar 5-12 Perbedaan waktu eksekusi dari penggunaan NCPR dan NCQPR

5.7 Pengujian Metode Seleksi

Hasil pengujian pengaruh metode seleksi untuk representasi kromosom pertama (K1) disajikan pada Tabel 5-19 dan Tabel 5-20 untuk representasi kromosom kedua (K2). Pengujian ini dilakukan dengan *popSize* = 200, iterasi maksimal = 100. Sedangkan metode rekombinasi (operator genetika) berbeda untuk kedua representasi kromosom. Untuk representasi kromosom pertama (K1) menggunakan metode rekombinasi *UC-EM* yaitu perpaduan antara metode *crossover* = *Uniform crossover*, dan metode *mutasi* = random *exchange*. Metode rekombinasi *UC-EM* tersebut dijalankan dengan *crossover rate* (*cr*) = 0.8 dan *mutation rate* (*mr*) = 0.2. Sedangkan untuk representasi kromosom kedua (K2) menggunakan metode *RC-RM* yaitu perpaduan antara metode *crossover* = *Random crossover* dengan *crossover rate* (*cr*) = 0.8 dan metode mutasi = *random exchange* dengan *mutation rate* (*mr*) = 0.2. Pada Tabel 5-19, *GA-BT* menyatakan algoritme genetika (*GA*) menggunakan metode seleksi *binary tournament* (*BT*). *GA-PRW* mewakili algoritme genetika (*GA*) menggunakan metode seleksi *proportional roulette wheel* (*PRW*). *NSGA-II* adalah menggunakan *non-dominated sorting*

genetic algorithm yang belum dimodifikasi. *NSGA-II-Mod* menggunakan *non-dominated sorting genetic algorithm* yang dimodifikasi.

Berdasarkan Tabel 5-19, metode seleksi terbaik untuk representasi kromosom pertama adalah *NSGA-II-Mod*. *NSGA-II-Mod* menghasilkan nilai *fitness* terbaik jika dibandingkan dengan ketiga metode seleksi lainnya. Meskipun untuk waktu eksekusi masih lebih lama jika dibandingkan dengan *GA-BT* dan *GA-PRW*. *GA-BT* dan *GA-PRW* adalah metode seleksi *non-elitism* yang didalamnya tidak diperlukan proses sorting, sehingga sangat wajar jika memiliki waktu yang lebih cepat dibandingkan dengan metode seleksi berbasis *elitism* yang dalam hal ini adalah *NSGA-II* dan *NSGA-II-Mod*. Meskipun *GA-BT* dan *GA-PRW* lebih cepat, namun dalam kasus optimasi multi-objektif penentuan UKT tidak menghasilkan solusi yang layak sejumlah *popSize*. Selain itu nilai *fitness* yang diraih *GA-BT* dan *GA-PRW* tidak sebaik *NSGA-II* dan *NSGA-II-Mod*. Jika dibandingkan waktu eksekusi antara *NSGA-II* dan *NSGA-II-Mod*, *NSGA-II-Mod* lebih cepat 0.154 detik. Oleh Karena itu, bisa dikatakan tujuan dari *NSGA-II-Mod* tercapai, yaitu mempercepat waktu eksekusi dari *NSGA-II*.

Tabel 5-19 Pengaruh metode seleksi untuk K1

Metode	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
GA-BT	-2.444	702.068	1.226	-2.419	708.095	1.236	**1.294**	199.9
GA-PRW	-2.474	700.920	1.221	-2.438	**716.768**	1.237	1.348	144.5
NSGA-II	-2.435	703.016	1.229	-2.396	715.110	1.244	1.508	**200**
NSGA-II-Mod	**-2.429**	**705.226**	**1.232**	**-2.387**	716.315	**1.247**	1.354	**200**

Tabel 5-20 menunjukkan bahwa secara umum *NSGA-II-Mod* menghasilkan nilai *fitness* terbaik untuk representasi kromosom kedua (K2). Hal tersebut dapat diamati pada kolom rerata *ED* dan Maks *ED* yang memberikan gambaran kompromi dari dua tujuan dari optimasi yaitu *f1* dan *f2*. Meskipun nilai rerata *ED* antara *NSGA-II-Mod* dan *NSGA-II* hanya berselisih 0.001 dan nilai Maks *ED* yang sama. Namun rerata waktu yang dihasilkan *NSGA-II-Mod* lebih cepat 0.193 detik atau 20% lebih cepat dari *NSGA-II*. Lebih cepatnya waktu *NSGA-II-Mod* dibandingkan dengan *NSGA-II* menunjukkan bahwa tujuan dari modifikasi algoritme ini tercapai, yaitu mempercepat waktu eksekusi.

Tabel 5-20 Pengaruh metode seleksi untuk K2

Metode	Rerata *f1*	Rerata *f2*	Rerata *ED*	Maks *f1*	Maks *f2*	Maks *ED*	Rerata waktu (detik)	Rerata *FeaSol*
GA-BT	-2.335	668.673	1.198	**-2.315**	680	1.205	**0.585**	191.4
GA-PRW	-2.407	655.051	1.172	-2.324	670	1.192	0.695	199.15
NSGA-II	-2.336	**670.987**	**1.201**	**-2.315**	**687**	**1.208**	0.974	**200**
NSGA-II-Mod	**-2.333**	670.502	1.200	**-2.315**	685	**1.208**	0.781	**200**

Tabel 5-20 juga menunjukkan bahwa metode seleksi berbasis *elitism* lebih unggul dalam optimasi multi-objektif penentuan UKT. Hal ini ditunjukkan pada kolom rerata *FeaSol* metode seleksi *non-elitism* tidak menghasilkan solusi yang layak sejumlah *popSize*. Hal ini

menunjukkan bahwa dalam optimasi multi-objektif penentuan UKT dengan representasi kromosom yang digunakan, solusi yang layak lebih memiliki peluang menghasilkan solusi yang layak juga ketika proses rekombinasi. Atau solusi yang tidak layak memiliki peluang kecil untuk menghasilkan solusi yang layak. Hal ini dikarenakan konstruksi representasi kromosom kedua (K2) membutuhkan hubungan tertentu pada gen agar solusi (kromosom) itu layak, yaitu gen yang lebih awal harus lebih kecil dari sebelumnya untuk batasan pada 1 program studi.

5.8 Analisis Hasil Penetapan UKT

Dalam tahap ini akan dilakukan perbandingan hasil penetapan UKT (solusi/kromosom) menggunakan metode NSGA-II-Mod dengan metode sederhana. Metode sederhana dalam hal ini adalah menggunakan operator logika sederhana yang membandingkan antara E_m (kemampuan finansial mahasiswa) dengan nilai dari besaran UKT (U_{ks}). Jika E_m kurang dari U_{1s} (Besaran UKT kategori 1 untuk prodi *s*), maka K_m = 1. Jika E_m kurang dari U_{2s} (Besaran UKT kategori 2 untuk prodi *s*), maka K_m = 2, dan seterusnya.

Tabel 5-21 Hasil penentuan UKT menggunakan metode sederhana (sol-sed)

S	*K*	*Min*(E_m)	*Max*(E_m)	NK_{ks}	*Sum*(U_m)	avg(|U_m-E_m|)
0	1	410,000	600,000	2	1,000,000	95,000.00
	2	1,050,000	3,870,000	6	6,000,000	1,316,666.67
	3	4,490,000	4,490,000	1	4,050,000	440,000.00
	4	6,560,000	7,060,000	4	25,600,000	390,000.00
	5	7,900,000	8,720,000	4	28,300,000	1,215,000.00
	6	9,440,000	28,400,000	35	306,250,000	1,227,428.57
Total Prodi 0				**52**	**371,200,000**	**1,113,653.85**
1	1	850,000	980,000	2	1,000,000	415,000.00
	2	1,360,000	4,530,000	8	8,000,000	1,578,750.00
	3	5,920,000	6,200,000	3	13,995,000	1,421,666.67
	6	9,590,000	29,850,000	35	307,125,000	1,206,714.29
Total Prodi 1				**48**	**330,120,000**	**1,249,166.67**
Total				**100**	**701,320,000**	**1,178,700.00**

Solusi (hasil penetapan UKT) dengan metode sederhana yang lebih singkat disebut "sol-sed" ditampilkan pada Tabel 5-21. Hasil penetapan UKT yang digunakan untuk analisis pada tahap ini terdiri dari 3 variasi, yaitu solusi dengan nilai *f1* tertinggi, solusi dengan nilai *f2* tertinggi dan solusi yang memiliki nilai tengah *f1.* Hasil dari 3 variasi solusi tersebut secara berurutan ditampilkan pada Tabel 5-22, Tabel 5-23, dan Tabel 5-24. Tabel 5-21, Tabel 5-22, Tabel 5-23, dan Tabel 5-24 menunjukkan minimal dan maksimal kemampuan finansial mahasiswa (E_m) berdasarkan kategori *k* dan program studi *s*. Pada Tabel 5-21 juga terdapat informasi tentang jumlah mahasiswa (NK_{ks}), jumlah besaran UKT mahasiswa (*Sum*(U_m)) , dan rerata jarak antara besaran UKT mahasiswa dengan kemampuan finansial mahasiswa (*avg*(|U_m-E_m|)).

Dapat dilihat pada Tabel 5-21, bahwa metode sederhana menghasilkan total besaran UKT mahasiswa (*Sum*(U_m)) sebesar 701,320,000 dan dan rerata jarak antara besaran UKT mahasiswa dengan kemampuan finansial mahasiswa (*avg*(|U_m-E_m|)) sebesar 1,178,700.00. Sedangkan pada Tabel 5-22 menunjukkan bahwa NSGA-II-Mod dapat memperbesar *Sum*(U_m) menjadi sebesar 728,965,000 atau lebih besar 27,645,000.00. Disamping itu NSGA-II-Mod juga dapat memperkecil nilai *avg*(|U_m-E_m|) menjadi sebesar 983,846.15 atau lebih rendah

127,850.00. Hal ini menunjukkan bahwa solusi sol-*f1* lebih baik daripada solusi yang dihasilkan oleh metode sederhana.

Tabel 5-22 Solusi hasil *NSGA-II-Mod* dengan nilai *f1* tertinggi (sol-*f1*)

S	*K*	*Min*(E_m)	*Max*(E_m)	NK_{ks}	*Sum*(U_m)	avg(|U_m-E_m|)
0	1	410,000	1,050,000	3	1,500,000	246,666.67
	2	1,690,000	2,420,000	3	3,000,000	1,076,666.67
	3	2,750,000	4,490,000	3	12,150,000	640,000.00
	4	6,560,000	6,580,000	2	12,800,000	170,000.00
	5	6,960,000	7,060,000	2	14,150,000	65,000.00
	6	7,900,000	28,400,000	39	341,250,000	1,148,717.95
Total Prodi 0				**52**	**384,850,000**	**983,846.15**
1	1	850,000	1,380,000	4	2,000,000	642,500.00
	2	1,920,000	2,760,000	4	4,000,000	1,285,000.00
	3	4,220,000	4,530,000	2	9,330,000	290,000.00
	4	5,920,000	6,200,000	3	21,660,000	1,133,333.33
	6	9,590,000	29,850,000	35	307,125,000	1,206,714.29
Total Prodi 1				**48**	**344,115,000**	**1,123,437.50**
Total				**100**	**728,965,000**	**1,050,850.00**

Tabel 5-23 Solusi hasil NSGA-II-Mod dengan nilai *f2* tertinggi (sol-*f2*)

S	*K*	*Min*(E_m)	*Max*(E_m)	NK_{ks}	*Sum*(U_m)	avg(|U_m-E_m|)
0	1	410,000	1,050,000	3	1,500,000	246,666.67
	2	1,690,000	2,420,000	3	3,000,000	1,076,666.67
	3	2,750,000	2,750,000	1	4,050,000	1,300,000.00
	4	3,870,000	4,490,000	2	12,800,000	2,220,000.00
	5	6,560,000	7,060,000	4	28,300,000	285,000.00
	6	7,900,000	28,400,000	39	341,250,000	1,148,717.95
Total Prodi 0				**52**	**390,900,000**	**1,070,192.31**
1	1	850,000	1,380,000	4	2,000,000	642,500.00
	2	1,920,000	2,530,000	3	3,000,000	1,126,666.67
	3	2,760,000	2,760,000	1	4,665,000	1,905,000.00
	4	4,220,000	6,200,000	5	36,100,000	1,818,000.00
	6	9,590,000	29,850,000	35	307,125,000	1,206,714.29
Total Prodi 1				**48**	**352,890,000**	**1,232,916.67**
Total				**100**	**743,790,000**	**1,148,300.00**

Tabel 5-23 menunjukkan bahwa NSGA-II-Mod dapat memperbesar *Sum*(U_m) menjadi sebesar 743,790,000 atau lebih besar 42,470,000. Disamping itu NSGA-II-Mod juga dapat memperkecil nilai *avg*(|U_m-E_m|) menjadi sebesar 1,148,300 atau lebih rendah 30,400. Hal ini menunjukkan bahwa solusi sol-*f2* pun lebih baik daripada solusi yang dihasilkan oleh metode sederhana. Demikian juga yang dapat dilihat pada Tabel 5-24 menunjukkan bahwa NSGA-II-Mod dapat memperbesar *Sum*(U_m) menjadi sebesar 733,980,000 atau lebih besar 32,660,000. Disamping itu NSGA-II-Mod juga dapat memperkecil nilai *avg*(|U_m-E_m|) menjadi sebesar 1,059,000 atau lebih rendah 119,700. Hal ini menunjukkan bahwa solusi sol-*med* pun

lebih baik daripada solusi yang dihasilkan oleh metode sederhana. Perbandingan nilai *Sum(U_m)* dan *avg($|U_m-E_m|$)* untuk keempat solusi tersebut dapat dilihat pada Tabel 5-25.

Tabel 5-24 Solusi hasil NSGA-II-Mod dengan nilai *f1* dan *f2* tengahan (sol-med)

S	*K*	*Min*(E_m)	*Max*(E_m)	NK_{ks}	*Sum*(U_m)	avg($\|U_m-E_m\|$)
0	1	410,000	1,050,000	3	1,500,000	246,666.67
	2	1,690,000	2,420,000	3	3,000,000	1,076,666.67
	3	2,750,000	4,490,000	3	12,150,000	640,000.00
	5	6,560,000	7,060,000	4	28,300,000	285,000.00
	6	7,900,000	28,400,000	39	341,250,000	1,148,717.95
Total Prodi 0				**52**	**386,200,000**	**996,730.77**
1	1	850,000	1,380,000	4	2,000,000	642,500.00
	2	1,920,000	2,530,000	3	3,000,000	1,126,666.67
	3	2,760,000	4,530,000	3	13,995,000	828,333.33
	4	5,920,000	6,200,000	3	21,660,000	1,133,333.33
	6	9,590,000	29,850,000	35	307,125,000	1,206,714.29
Total Prodi 1				**48**	**347,780,000**	**1,126,458.33**
Total				**100**	**733,980,000**	**1,059,000.00**

Tabel 5-25 Perbandingan solusi metode sederhana dan 3 solusi hasil NSGA-2-Mod

Solusi	*Sum(U_m)*	*avg*($\|U_m-E_m\|$)
sol-sed	701,320,000	1,178,700.00
sol-f1	728,965,000	**1,050,850.00**
sol-f2	**743,790,000**	1,148,300.00
sol-med	733,980,000	1,059,000.00

5.9 NSGA-II dan Penentuan UKT

Permasalahan optimasi penentuan UKT merupakan permasalahan optimasi kombinatorial. Permasalahan optimasi kombinatorial adalah proses pencarian titik (solusi) optimal dari sejumlah kombinasi solusi. Optimasi penentuan UKT secara umum adalah proses mencari kombinasi kategori UKT untuk setiap mahasiswa. Algoritme genetika sangat cocok untuk menyelesaikan permasalahan optimasi penentuan UKT. Karena algoritme genetika merupakan algoritme yang diciptakan untuk menyelesaikan permasalahan optimasi kombinatorial.

Seperti yang dijelaskan sebelumnya bahwa permasalahan optimasi penentuan UKT adalah permasalahan optimasi multi-objektif. Tujuan pertama optimasi penentuan UKT adalah minimal jarak antara kemampuan finansial mahasiswa dengan besaran UKT atau lebih singkat disebut objektif yang cenderung kepada mahasiswa. Tujuan kedua adalah maksimal penghasilan fakultas yang didapatkan dari total besaran UKT mahasiswa atau dengan kata lain disebut objektif yang cenderung kepada institusi. Algoritme genetika yang paling sering digunakan untuk menyelesaikan permasalahan multi-objektif adalah NSGA-II.

Hasil dari pengujian metode seleksi menunjukkan bahwa metode seleksi *elitism* (baik NSGA-II maupun NSGA-II yang dimodifikasi) menghasilkan nilai *fitness* lebih tinggi dan jumlah *feasible solution* lebih banyak dari pada metode seleksi *non-elitism*. Hal ini menunjukkan

bahwa dalam optimasi multi-objektif penentuan UKT, solusi yang layak lebih memiliki peluang menghasilkan solusi yang layak juga ketika proses rekombinasi. Dan sebaliknya solusi yang tidak layak memiliki peluang kecil untuk menghasilkan solusi yang layak. Hal ini menunjukkan bahwa NSGA-II sangat cocok untuk menyelesaikan permasalahan optimasi penentuan UKT.

Hasil optimasi multi-objektif menggunakan NSGA-II adalah sejumlah solusi, bukan satu solusi. Sehingga solusi yang dihasilkan dalam permasalahan penentuan UKT sangat beragam, mulai dari solusi yang cenderung kepada mahasiswa (solusi dengan nilai *fitness* 1 terbesar) sampai solusi yang cenderung kepada institusi (solusi dengan nilai *fitness* 2 tertinggi). Hal ini akan sangat membantu pengambil keputusan dalam penentuan UKT, untuk memilih solusi yang menguntungkan institusi atau mahasiswa, atau solusi yang berada diantara keduanya.

BAB 6 PENUTUP

6.1 Kesimpulan

NSGA-II yang dimodifikasi telah berhasil mengoptimasi permasalahan penentuan UKT. NSGA-II yang dimodifikasi menghasilkan solusi sejumlah 200 (sesuai dengan *popSize*). Solusi tersebut beragam, mulai dari solusi yang cenderung lebih mengutamakan tujuan dari sisi mahasiswa (*fitness 1* bernilai paling tinggi) sampai solusi yang sangat menguntungkan institusi (*fitness 2* bernilai paling tinggi). Dengan beragamnya solusi yang dihasilkan, NSGA-II yang dimodifikasi dapat membantu pengambil keputusan dalam memilih solusi penentuan UKT. Berdasarkan hasil pengujian dan disertai dengan analisis hasil pengujian optimasi multi-objektif untuk penentuan UKT menggunakan NSGA-II yang dimodifikasi, maka dapat ditarik kesimpulan sebagai berikut:

1. Representasi kromosom yang paling efisien adalah representasi kromosom kedua (K2), karena K2 mencapai nilai *fitness* terbaik dengan waktu eksekusinya lebih singkat (K2 lebih singkat 42%). Hal ini dikarenakan K2 lebih membatasi wilayah pencarian, atau mempersempit kombinasi yang dihasilkan dengan memperkecil jumlah gen dari kromosom. Meskipun variasi nilai satu gen masih bersifat kontinyu tidak diskrit. Representasi kromosom pertama (K1) sangat memerlukan metode repair untuk menghasilkan solusi yang layak (*feasible*), meskipun masih menghasilkan sedikit jumlah solusi yang layak. K1 sangat sulit untuk mencari solusi yang layak karena terlalu luasnya semesta pencariannya atau dengan kata lain terlalu banyak variasi kombinasi yang dihasilkan.
2. Parameter terbaik untuk optimasi multi-objektif penentuan UKT adalah jumlah populasi (*popSize*) 200 dan maksimal iterasi = 100. Untuk representasi kromosom pertama (K1) menggunakan metode rekombinasi *UC-EM* yaitu perpaduan antara metode *crossover* = *Uniform crossover*, dan metode *mutasi* = random *exchange*. Metode rekombinasi *UC-EM* tersebut dijalankan dengan *crossover rate* (*cr*) = 0.8 dan *mutation rate* (*mr*) = 0.2. Sedangkan untuk representasi kromosom kedua (K2) menggunakan metode *RC-RM* yaitu perpaduan antara metode *crossover* = *Random crossover* dengan *crossover rate* (*cr*) = 0.5 dan metode mutasi = *random exchange* dengan *mutation rate* (*mr*) = 0.5.
3. Modifikasi metode seleksi NSGA-II yang diusulkan terbukti dapat mempersingkat waktu eksekusi dari NSGA-II sebesar 20%, tanpa menurunkan nilai *fitness* yang diperoleh.

6.2 Saran

Optimasi penentuan UKT telah menghasilkan solusi yang layak, modifikasi NSGA-II juga sudah berhasil memperkecil waktu eksekusi-nya. Namun permasalahan optimasi multi-objektif untuk penentuan UKT menggunakan NSGA-II masih terbuka untuk penelitian lebih lanjut. Saran untuk peneliti yang ingin melanjutkan penelitian ini adalah sebagai berikut:

1. Representasi kromosom kedua (K2) lebih baik dari pada representasi kromosom pertama (K1), karena semesta pencarian (kombinasi kromosom) lebih sempit. Seperti yang sudah diketahui bahwa nilai gen K2 masih bersifat kontinyu. Jika nilai gen bersifat diskrit, maka akan lebih mempersempit wilayah pencarian.

2. Seperti yang sudah diketahui dari hasil penelitian bahwa metode rekombinasi (operator genetika yang terdiri dari *crossover* dan mutasi) memiliki pengaruh terhadap hasil dari optimasi. Maka bisa diteliti lebih lanjut ketika menggunakan kombinasi operator genetika yang lain atau bahkan membuat operator genetika khusus untuk representasi kromosom yang digunakan.

DAFTAR PUSTAKA

Bandyopadhyay, S., & Bhattacharya, R. (2014). Solving a tri-objective supply chain problem with modified NSGA-II algorithm. *Journal of Manufacturing Systems*, *33*(1), 41–50. The Society of Manufacturing Engineers. Retrieved from http://dx.doi.org/10.1016/j.jmsy.2013.12.001

Bao, C., Xu, L., Goodman, E. D., & Cao, L. (2017). A novel non-dominated sorting algorithm for evolutionary multi-objective optimization. *Journal of Computational Science*, *23*, 31–43. Elsevier B.V. Retrieved from http://dx.doi.org/10.1016/j.jocs.2017.09.015

Bhandarkar, S. M., Zhang, Y., & Potter, W. D. (1994). An edge detection technique using genetic algorithm-based optimization. *Pattern Recognition*, *27*(9), 1159–1180.

Bhattacharjee, K., Bhattacharya, A., & Halder Nee Dey, S. (2014). Solution of Economic Emission Load Dispatch problems of power systems by Real Coded Chemical Reaction algorithm. *International Journal of Electrical Power and Energy Systems*, *59*, 176–187. Elsevier Ltd. Retrieved from http://dx.doi.org/10.1016/j.ijepes.2014.02.006

Blickle, T., & Thiele, L. (1995). *A Comparison of Selection Schemes used in Genetic Algorithm (2nd Edition). TIK Report No. 11.* (Vol. 2).

Burer, S., & Fethke, G. (2016). Nearly-efficient tuitions and subsidies in American public higher education. *Economics of Education Review*, *55*, 182–197. Elsevier Ltd. Retrieved from http://dx.doi.org/10.1016/j.econedurev.2016.09.003

Campos Ciro, G., Dugardin, F., Yalaoui, F., & Kelly, R. (2016). A NSGA-II and NSGA-III comparison for solving an open shop scheduling problem with resource constraints. *IFAC-PapersOnLine*, *49*(12), 1272–1277.

Chan, F. T. S., Jha, A., & Tiwari, M. K. (2016). Bi-objective optimization of three echelon supply chain involving truck selection and loading using NSGA-II with heuristics algorithm. *Applied Soft Computing Journal*, *38*, 978–987. Elsevier B.V.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms - I. Representation. *Computers and Industrial Engineering*, *30*(4), 983–997.

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11–12), 1245–1287.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2002). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Dixit, V., Seshadrinath, N., & Tiwari, M. K. (2016). Performance measures based optimization of supply chain network resilience: A NSGA-II + Co-Kriging approach. *Computers and Industrial Engineering*, *93*, 205–214. Elsevier Ltd. Retrieved from http://dx.doi.org/10.1016/j.cie.2015.12.029

E.Zitzler, & L.Thiele. (1998). An evolutionary algorithm for multiobjective optimization:The strength Pareto approach. *TIK-Rep*, *43*(43).

Eberhart, R. C., & Shi, Y. (2007). *Computational Intelligence: Concepts to Implementations*. *Knowledge Creation Diffusion Utilization*.

Fang, H., Wang, Q., Tu, Y. C., & Horstemeyer, M. F. (2008). An efficient non-dominated sorting method for evolutionary algorithms. *Evolutionary Computation*, *16*(3), 355–384.

Gao, K. Z., Suganthan, P. N., Pan, Q. K., Chua, T. J., Cai, T. X., & Chong, C. S. (2014). Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, *289*(1), 76–90. Elsevier Inc. Retrieved from http://dx.doi.org/10.1016/j.ins.2014.07.039

Grefenstette, J. J., Gopal, R., Rosmaita, B., & Gucht, D. Van. (1985). Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, *1*(13), 160–168.

Gudmundsson, M., El-Kwae, E. A., & Kabuka, M. R. (1998). Edge detection in medical images using a genetic algorithm. *IEEE Transactions on Medical Imaging*, *17*(3), 469–474.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis. *Artificial Intelligence Review*, *12*(4), 265–319.

Hu, Y., Bie, Z., Ding, T., & Lin, Y. (2016). An NSGA-II based multi-objective optimization for combined gas and electricity network expansion planning. *Applied Energy*, *167*, 280–293. Elsevier Ltd. Retrieved from http://dx.doi.org/10.1016/j.apenergy.2015.10.148

Jain, S., & Gea, H. C. (1998). Two-Dimensional Packing Problems Using Genetic Algorithms. *Engineering with Computers*, *14*(3), 206–213.

Jakiela, M. J., Chapman, C., Duda, J., Adewuya, A., & Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2–4), 339–356.

Jensen, M. T. (2003). Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(5), 503–515.

Joe, S., & Kuo, F. Y. (2008). Constructing Sobol Sequences with Better Two-Dimensional Projections. *SIAM Journal on Scientific Computing*, *30*(5), 2635–2654.

Karim, B., Sentinuwo, S. R., & Sambul, A. M. (2017). Penentuan Besaran Uang Kuliah Tunggal untuk Mahasiswa Baru di Universitas Sam Ratulangi Menggunakan Data Mining. *E-journal teknik informatika*, *11*(1).

Kazimipour, B., Li, X., & Qin, A. K. (2014). A review of population initialization techniques for evolutionary algorithms. *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014* (pp. 2585–2592).

Koza, J. R. (1992). *Genetic Programming*. *MIT Press* (Vol. 1).

Kristin Bennett, Michael C. Ferris, & Yannis Ioannidis. (1991). A genetic algorithm for database query optimization. *4th Int'l Conference on Genetic Algorithms* (pp. 400–407).

Liu, T., Gao, X., & Wang, L. (2015). Multi-objective optimization method using an improved NSGA-II algorithm for oil–gas production process. *Journal of the Taiwan Institute of Chemical Engineers*, *000*, 1–12. Elsevier Ltd.

Mahmudy, W. F., Marian, R. M., & Luong, L. H. S. (2013a). Real Coded Genetic Algorithms for

Solving Flexible Job-Shop Scheduling Problem - Part II: Optimization. *Advanced Materials Research*, *701*, 364–369.

Mahmudy, W. F., Marian, R. M., & Luong, L. H. S. (2013b). Real Coded Genetic Algorithms for Solving Flexible Job-Shop Scheduling Problem - Part I: Modelling. *Advanced Materials Research*, *701*, 359–363.

Mahmudy, W. F., Marian, R. M., & Luong, L. H. S. (2013c). Hybrid genetic algorithms for multi-period part type selection and machine loading problems in flexible manufacturing system. *Proceeding - IEEE CYBERNETICSCOM 2013: IEEE International Conference on Computational Intelligence and Cybernetics*, 126–130.

Mahmudy, W. F., Marian, R. M., & Luong, L. H. S. (2014). Hybrid Genetic Algorithms for Part Type Selection and Machine Loading Problems with Alternative Production Plans in Flexible Manufacturing System. *ECTI Transaction On Computer And Information Technology*, *8*(1), 80–93.

Mallipeddi, R., & Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, *14*(4), 561–579.

Martínez-Puras, A., & Pacheco, J. (2016). MOAMP-Tabu search and NSGA-II for a real Bi-objective scheduling-routing problem. *Knowledge-Based Systems*, *112*, 92–104.

McClymont, K., & Keedwell, E. (2012). Deductive sort and climbing sort: New methods for non-dominated sorting. *Evolutionary Computation*, *20*(1), 1–26.

MENRISTEKDIKTI Republik Indonesia. (2015). *Peraturan Menteri Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia Nomor 22 Tahun 2015 Tentang Biaya Kuliah dan Uang Kuliah Tunggal Pada Perguruan Tinggi Negeri Di Lingkungan Kementrian Riset, Teknologi dan Pendidikan*. Indonesia.

MENRISTEKDIKTI Republik Indonesia. (2016). *Peraturan Menteri Riset, Teknologi, Dan Pendidikan Tinggi Republik Indonesia Nomor 4 Tahun 2016 Tentang Organisasi Dan Tata Kerja Universitas Brawijaya*. Republik Indonesia.

Mezura-Montes, E., & Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173–194. Elsevier B.V. Retrieved from http://dx.doi.org/10.1016/j.swevo.2011.10.001

Michalewicz, Z. (1995). A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. *Evolutionary Programming*, *4*, 135–155.

Mirjalili, S., & Lewis, A. (2015). Novel performance metrics for robust multi-objective optimization algorithms. *Swarm and Evolutionary Computation*, *21*, 1–23. Elsevier. Retrieved from http://dx.doi.org/10.1016/j.swevo.2014.10.005

Morokoff, W. J., & Caflisch, R. E. (1994). Quasi-Random Sequences and Their Discrepancies. *SIAM Journal on Scientific Computing*, *15*(6), 1251–1279. Retrieved from http://epubs.siam.org/doi/10.1137/0915077

Muchsin, A. K., & Sudarma, M. (2015). Penerapan Fuzzy C-Means Untuk Penentuan Besar Uang Kuliah Tunggal Mahasiswa Baru. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, *6*(3), 175. Retrieved from https://ojs.unud.ac.id/index.php/lontar/article/view/32368

Nguyen Quang Uy, Nguyen Xuan Hoai, McKay, R., & Tuan, P. M. (2007). Initialising PSO with randomised low-discrepancy sequences: the comparative results. *2007 IEEE Congress on Evolutionary Computation* (Vol. 22, pp. 1985–1992). IEEE. Retrieved from http://ieeexplore.ieee.org/document/4424717/

Pasandideh, S. H. R., Niaki, S. T. A., & Asadi, K. (2015). Bi-objective optimization of a multi-product multi-period three-echelon supply chain problem under uncertain environments: NSGA-II and NRGA. *Information Sciences*, *292*, 57–74. Elsevier Inc. Retrieved from http://dx.doi.org/10.1016/j.ins.2014.08.068

Prasetyanti, D. N., & Listyaningrum, R. (2017). Kaji Banding Metode TOPSIS, Saw Dan AHP-TOPSIS Guna Menentukan UKT Mahasiswa Baru Di Politeknik Negeri Cilacap. *Jurnal Infotekmes*, *8*(1), 9–14.

Razali, N. M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategiesin solving TSP. *Proceedings of the World Congress on Engineering 2011*, *2*, 1134–1139.

Salcedo-Sanz, S. (2009). A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Science Review*, *3*(3), 175–192. Elsevier Inc. Retrieved from http://dx.doi.org/10.1016/j.cosrev.2009.07.001

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*. *Ph.D. dissertation, Vanderbilt University, Nashville, TN.* Vanderbilt University, Nashville, TN. Retrieved from http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=5673304

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: a survey. *Computer*, *27*(6), 17–26. Retrieved from http://ieeexplore.ieee.org/document/294849/

Srinivas, N., & Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, *2*(3), 221--248.

Syaiful Rokhman, Imam Fahrur Rozi, R. A. A. (2017). Pengembangan sistem penunjang keputusan penentuan ukt mahasiswa dengan menggunakan metode moora studi kasus politeknik negeri malang. *Jurnal Informatika Polinema*, *3*, 36–42.

Tang, S., Cai, Z., & Zheng, J. (2008). A fast method of constructing the non-dominated set: Arena's principle. *Proceedings - 4th International Conference on Natural Computation, ICNC 2008*, *1*, 391–395.

Taufiq, M. N., Dewi, C., & Mahmudy, W. F. (2017). Optimasi Komposisi Pakan Untuk Penggemukan Sapi Potong Menggunakan Algoritma Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *1*(January), 571–582.

Universitas Brawijaya. (2016a). UKT bagi Mahasiswa Baru UB Tahun Akademik 2016/2017 Jalur SNMPTN dan SBMPTN. Retrieved September 14, 2017, from https://selma.ub.ac.id/ukt-bagi-mahasiswa-baru-ub-tahun-akademik-20162017-jalur-snmptn-dan-sbmptn/

Universitas Brawijaya. (2016b). Panduan Mengisi Berkas Online. *https://selma.ub.ac.id*. Retrieved January 1, 2017, from https://selma.ub.ac.id/wp-content/uploads/2016/07/Panduan-Mengisi-Berkas-SIAM-3.1.2-Dokumen-Laman-Infokom-Juli-2016.pdf

Universitas Brawijaya. (2017a). Biaya Pendidikan S1 Jalur Seleksi Mandiri 2017/2018. Retrieved September 14, 2017, from https://selma.ub.ac.id/biaya-seleksi-mandiri-20172018/

Universitas Brawijaya. (2017b). Biaya Pendidikan Bagi Mahasiswa Baru Jalur Seleksi Mandiri UB Kampus III Kediri Tahun Akademik 2017/2018. Retrieved September 14, 2017, from https://selma.ub.ac.id/biaya-pendidikan-bagi-mahasiswa-baru-program-s1-jalur-mandiri-kampus-iv-kediri-20172018/

Universitas Brawijaya. (2017c). Uang Kuliah Tunggal Mahasiswa Baru Program Pendidikan Vokasi 2017/2018. Retrieved September 14, 2017, from https://selma.ub.ac.id/uang-kuliah-tinggal-mahasiswa-baru-program-pendidikan-vokasi-20172018/

Wang, G., Xu, T., Wang, H., & Zou, Y. (2017). AdaBoost and Least Square Based Failure Prediction of Railway Turnouts. *Proceedings - 2016 9th International Symposium on Computational Intelligence and Design, ISCID 2016*, *1*, 434–437.

Weile, D. S., & Michielssen, E. (1996). Design of Constrained Antenna Arrays. *Electronics Letters*, *32*(19), 1744–1745.

Wroblewski, J. (1995). Finding minimal reducts using genetic algorithms. *Proceedings of Second International Joint Conference on Information Science*, 186–189.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, *1*(1), 32–49. Elsevier B.V.