PENENTUAN PORTOFOLIO SAHAM OPTIMAL MENGGUNAKAN ALGORITME GENETIKA TERDISTRIBUSI

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Talitha Raissa
NIM: 145150201111045

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENENTUAN PORTOFOLIO SAHAM OPTIMAL MENGGUNAKAN ALGORITME GENETIKA TERDISTRIBUSI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Talitha Raissa
NIM: 145150201111045

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agus Wahyu Widodo, S.T, M.Cs	Budi Darma Setiawan, S.Kom, M.Cs
NIP: 19740805 200112 1 001	NIP: 19841015 201404 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 27 Mei 2018

Talitha Raissa

NIM: 145150201111045

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT atas segala rahmat dan ata izin-Nya penulis dapat menyelesaikan skripsi yang berjudul "Penentuan Portofolio Saham Menggunakan Algoritme Genetika Terdistribusi".

Selama pengerjaan skripsi ini tidak lepas dari bantuan, bimbingan, semangat serta doa yang diberikan oleh berbagai pihak kepada penulis. Oleh karena itu, ucapan terimakasih akan disampaikan kepada:

1. Kedua orang tua dan adik tercinta Neyzsa Amanda yang selalu memberikan dukungan, semangat, dan doa selama proses pengerjaan skripsi dan selalu sabar menampung segala keluh kesah penulis selama mengerjakan skripsi.
2. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku dosen pembimbing 1 yang selalu meluangkan waktu untuk membimbing dan mengarahkan penulis dengan sabar, segala kritik dan saran yang membangun, dukungan dan doa yang diberikan kepada penulis.
3. Bapak Budi Darma Setiawan, M.Cs selaku dosen pembimbing 2 yang selalu meluangkan waktu untuk membimbing dan mengarahkan penulis dalam mengerjakan dokumen skripsi serta kritik dan saran yang sangat membantu penulis.
4. Reza Muhammad Rizky yang selalu bersedia menemani, membantu dan memberikan dukungan kepada penulis selama mengerjakan skripsi.
5. Kholifaul Khoirin yang dengan sabar banyak membantu, dan memberikan semangat kepada penulis dalam mengerjakan skripsi, Muhammad Dimas S. yang membantu dan memberikan semangat kepada penulis, Falih Gozi F. yang bersedia membantu dan memberikan semangat kepada penulis.
6. Nur Afdaliyah Anwar, Savira Sulistyo, dan Cusen Mosabeth yang selalu membantu, memberi dukunga, dan mendoakan dalam pengerjaan skripsi.
7. Sahabat-sahabat penulis, Elizabeth Tiffany S., Hanny Kurniawati G., Grahadenata Hana Putra, Catur Sandhy D. yang selalu mendoakan penulis.
8. Adik-adik saya di Departemen Internal Eksekutif Mahasiswa Informatika Filkom UB yaitu Diajeng Tania Ananda, Hafiz Maulana, Arsya Monica P., Siti Alfina P.S, Made Rezananda P, Maulana Yoga W, Naufal Akbar E, Nisrina Rahmania, Citra Nadya D.I, Dhanika Jeihan A, Dewi Syafira, dan Dwi Jan Prayogi yang selalu memberikan semangat dan doa kepada penulis.
9. Teman-teman seperjuangan Griya Muslimah yang memberi dukungan dan doa kepada penulis Khairinnisa Rifna, Savira Sulistyo, Adia Margustia,

Sarah F. Rangkuti, Bella Julia, Deniarsha Putri, Namira Yekti, dan Usy Hapsari.

10. Semua pihak yang tidak dapat disebutkan satu persatu, terimakasih banyak atas bantuan, semangat dan doa yang diberikan kepada penulis.

Penulis menyadari bahwa skripsi yang dibuat masih jauh dari kata sempurna, maka dari itu kritik dan saran yang membangun sangat diharapkan penulis untuk perbaikan kedepannya. Akhir kata, semoga skripsi ini bermanfaat bagi pembaca.

Malang, 27 Mei 2018

Penulis

talitharaissa96@gmail.com

ABSTRAK

Dalam melakukan investasi saham diperlukan ada diversifikasi atau penyebaran investasi sehingga terbentuk portofolio saham dengan proporsi atau bobot masing-masing saham yang optimal, keuntungan yang baik dan resiko yang bisa ditanggung investor. Maka dari itu, dibuat sistem yang dapat menentukan portofolio saham optimal dengan menerapkan algoritme genetika terdistribusi.

Algoritme genetika terdistribusi membangkitkan kromosom secara acak pada interval tertentu sebagai representasi dari proporsi saham. Kemudian dilakukan reproduksi, evaluasi dan seleksi berdasarkan *fitness* terbesar yang didapat dari hasil perhitungan dengan *single index model* untuk mencari *return* dan resiko. Algoritme genetika terdistribusi memiliki mekanisme migrasi yang dapat menjaga keragaman variasi individu. Hal ini diperlukan agar pencarian solusi lebih luas sehingga dapat menghasilkan portofolio saham yang beragam dan optimal.

Dari hasil pengujian, algoritme genetika terdistribusi dapat diterapkan dengan baik dan menghasilkan portofolio saham yang optimal. Parameter terbaik dari hasil pengujian untuk jumlah ukuran populasi sebesar 80, jumlah generasi 150, *crossover rate* 0,8 dan *mutation rate* 0,2 serta jumlah sub populasi optimal sebanyak 10 untuk menghasilkan portofolio saham optimal.

Kata kunci: algoritme genetika terdistribusi, portofolio saham, *single index model*.

ABSTRACT

In conducting stock investment, it is necessary to verify or spread the investment in order to form a stock portfolio with the proportion or optimal weight of every stock, good profit, and risk that can be borne by investors. Therefore, a system that can determine the optimal stock portfolio must be made by implementing distributed genetic algorithm.

Distributed genetic algorithm generate chromosomes randomly at the certain interval as the representation of the stocks proportion. Then reproduction, evaluation and selection can be done based on the largest fitness derived from the calculation with single index model to find the return and risk. Distributed genetic algorithm has migration mechanism that able to maintain a diversity of individual variation. It is necessary to find out a broader solution which can produce a diverse and optimal stock portfolio.

From the test result, distributed genetic algorithms can be applied properly and produce an optimal stock portfolio. The best parameter of the popsize test result is 80, the number of generations 150, 0.8 crossover rate, and 0.2 mutation rate and the number of sub-optimal population of 10 to produce an optimal stock portfolio.

Keywords: distributed genetic algorithm, stock portfolio, single index model.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dewasa ini banyak masyarakat yang melakukan investasi untuk mendapatkan keuntungan. Investasi adalah komitmen sejumlah uang atau sumber daya lain yang dilakukan pada saat ini agar mendapatkan keuntungan di kemudian hari (Tandelilin, Portofolio dan Investasi; Teori dan Aplikasi, 2010). Terdapat berbagai bentuk investasi yang bisa dilakukan seperti investasi pada aset *real* yaitu menginvestasikan dana untuk membeli tanah, emas, bangunan dan mesin. Selain itu, investasi juga bisa dilakukan pada aset finansial seperti deposito, saham dan obligasi. Salah satu bentuk investasi yang banyak dilakukan adalah investasi pada saham. Saham merupakan surat bukti kepemilikan dan aset dari perusahaan yang menerbitkan saham (Tandelilin, Portofolio dan Investasi; Teori dan Aplikasi, 2010). Banyaknya investor yang memilih investasi saham disebabkan karena saham sangat menjanjikan keuntungan yang lebih tinggi dibandingkan melakukan investasi pada sektor *real.* Meskipun menjanjikan untuk mendapatkan *return* atau pengembalian yang tinggi, investasi pada saham juga memiliki resiko tinggi. Untuk mengatasi permasalahan tersebut, para investor melakukan diversifikasi atau mengombinasikan berbagai saham pada portofolio saham.

Portofolio saham berisi berbagai jenis saham dengan proporsi yang optimal dengan *return* terbaik dengan resiko yang tidak membebani investor sehingga investor dapat mengurangi resiko kerugian karena melakukan diversifikasi saham yang bertujuan untuk menyebar investasi saham ke berbagai jenis saham. Hal ini dilakukan untuk mengatasi permasalahan jika salah satu saham mengalami kerugian, investor masih memiliki saham di bidang lain yang berpeluang untuk memberikan *return* (Samsul, 2015). Untuk menentukan proporsi saham pada portofolio saham bukan hal yang mudah karena investor harus memilih beberapa saham dari sekian banyak saham yang berpeluang untuk menghasilkan *return* yang tinggi dengan resiko yang masih dalam batas wajar.

Penelitian tentang portofolio saham sebelumnya telah banyak dilakukan, seperti yang sudah dilakukan yaitu optimasi portofolio saham dengan algoritme genetika *multi-objective* dengan memperhitungkan biaya transaksi (Sofariah, Saepudin, & Umbara, 2016) dengan menggunakan metode *Markowitz* untuk meminimumkan resiko, sedangkan *multi-objective* digunakan untuk menentukan portofolio optimal karena pada penelitian tersebut memperhitungkan variabel lain yaitu biaya transaksi. Hasil dari penelitian tersebut sudah cukup baik pada generasi ke 100 dengan nilai *crossover rate (cr)* sebesar 0,09 dan *mutation rate (mr)* sebesar 0,1, 0,2, dan 0,3 dengan *intermediate crossover* dan *swap mutation*.

Penelitian berikutnya yang dilakukan oleh (Wahyuni, Mahmudy, & Setiawan, 2017) yaitu menentukan portofolio saham optimal dengan algoritme

genetika meenggunakan data *alpha, beta*, dan kesalahan residu pada saham indeks LQ45 dan representasi kromosom secara *real-code*. Pada penelitian tersebut, penulis menentukan proporsi saham serta *return* dan resiko dari saham yang dipilih dengan menggunakan metode *single index model* untuk menghitung *return* dan resiko. Dari hasil pengujian dengan *extended intermediate crossover* dan *reciprocal exchange mutation* didapatkan hasil populasi optimal 100 dengan *fitness* 0,349660178 dan nilai *cr* sebesar 0,7 dan *mr* 0,3. Namun, pada penelitian ini titik optimal pada uji coba populasi, uji coba generasi, dan uji coba kombinasi *cr* dan *mr* terjebak dalam varian individu yang sama.

Algortime genetika terdistribusi mampu menjaga keragaman variasi individu dengan mekanisme migrasi yaitu memindahkan individu terbaik ke sub populasi yang lain sehingga mampu menghasilkan solusi yang lebih optimal atau lebih baik dibandingkan dengan algoritme genetika biasa yang sering terjebak pada variasi yang sama. Dengan menerapkan algoritme genetika terdistribusi, permasalahan penentuan portofolio saham optimal diharapkan dapat memberikan solusi yang baik.

Berdasarkan permasalahan tersebut serta dari penelitian-penelitian sebelumnya, penulis mencoba menerapkan sistem untuk optimasi portofolio saham menggunakan algoritme genetika terdistribusi dengan harapan algoritme genetika terdistribusi dapat memberikan hasil portofolio saham optimal yang lebih baik. Penelitian ini meggunakan data saham dari indeks LQ45, data IHSG sebagai acuan harga pasar, metode *single index model* untuk mengetahui *return* serta resiko, sedangkan representasi kromosom menggunakan *real-code*.

1.2 Rumusan Masalah

Berdasarkan penjelasan dari latar belakang maka dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana pengaruh perubahan paramater algoritme genetika terdistribusi untuk mendapatkan hasil optimal pada portofolio saham?
2. Berapa nilai parameter algoritme genetika terdistribusi yang digunakan agar mendapatkan hasil optimal?

1.3 Tujuan

Tujuan yang akan dicapai pada penelitian ini adalah:

1. Mengetahui pengaruh perubahan parameter algoritme genetika terdistribusi untuk mendapatkan hasil optimal pada portofolio saham.
2. Mengetahui berapa nilai parameter algoritme genetika terdistribusi yang digunakan agar mendapatkan hasil optimal.

1.4 Manfaat

Penelitian ini diharapkan memberikan manfaat yang berguna bagi pembaca dan penulis

1. Pialang saham atau investor dalam menentukan portofolio saham yang optimal.
2. Pialang saham atau investor dapat mengetahui *return* dan resiko dari portofolio saham yang dipilih.

1.5 Batasan Masalah

Pembahasan dalam penelitian ini perlu diberikan batasan agar leboh terarah dan menghindari pembahasan yang tidak sesuai topik dan terlalu kompleks, maka dibuatlah batasan masalah sebagai berikut:

1. Data saham yang digunakan pada penelitian ini yaitu saham yang terdaftar pada indeks LQ45 Bursa Efek Indonesia pada periode Januari 2013-Januari 2018 sejumlah 21 indeks saham.
2. Data yang digunakan pada penelitian adalah data yang sudah diolah bukan data *real-time*.
3. Data historis bulanan saham indeks LQ45 merupakan faktor fundamental yang digunakan untuk menentukan portofolio optimal.
4. Metode *single index model* digunakan untuk menghitung *return* dan resiko.
5. Metode *crossover* yang digunakan adalah *extended intermediate crossover*, dan *one-cut point crossover* sedangakan metode mutasi yang digunakan adalah *reciprocal exchange mutation* dengan metode *elitism* sebagai metode seleksi.
6. Pada penelitian ini pengujian akan dilakukan untuk menguji ukuran populasi, jumlah generasi, *crossover rate, mutation rate* dan jumlah sub populasi.

1.6 Sistematika Pembahasan

Sistematika pembahasan pada penulisan skripsi adalah sebagai berikut:

BAB I PENDAHULUAN

Bab I berisi latar belakang masalah, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab II berisi tentang kajian pustaka dan teori-teori yang berkaitan dengan topik penelitian dari penulis yang dijadikan acuan untuk menerapkan algoritme genetika terdistribusi pada penentuan komposisi optimal untuk portofolio saham. Hal tersebut meliputi tentang saham, portofolio saham, *single index model*, algoritme genetika, dan algoritme genetika terdistribusi.

BAB III METODOLOGI

Bab III menjelaskan tahapan-tahapan yang akan dilakukan untuk menerapkan algoritme genetika terdistribusi dalam menentukan komposisi optimal pada portofolio saham.

BAB IV PERANCANGAN

Bab IV menjelaskan tentang bagaimana perancangan sistem, perancangan pengujian serta perhitungan manual untuk penentuan komposisi optimal pada portofolio saham dengan algoritme genetika terdistribusi.

BAB V IMPLEMENTASI

Bab V berisi tentang bagaimana implementasi yang dilakukan untuk menentukan portofolio saham optimal menggunakan algoritme genetika terdistribusi.

BAB VI PENGUJIAN DAN ANALISIS

Bab VI berisi pengujian dan analisis dari hasil sistem untuk penentuan portofolio saham optimal dengan algoritme genetika terdistribusi.

BAB VII PENUTUP

Bab VII menjelaskan bagaimana kesimpulan dari penelitian ini yang meliputi hasil, kelebihan dan kekurangan dari penentuan komposisi optimal dengan algoritme genetika terdistribusi, tingkat akurasi. Selain itu bab ini juga berisi saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi penjabaran tentang penelitian-penelitian sebelumnya mengenai penentuan portofolio saham, serta teori-teori yang berhubungan dengan penelitian ini yaitu saham, portofolio saham, *single index model,* algoritme genetika, serta algoritme genetika terdistribusi.

2.1 Kajian Pustaka

Penelitian tentang sistem rekomendasi portofolio investasi berbasis algoritme genetika bertujuan untuk menentukan portofolio investasi yang optimal sehingga mampu membantu investor untuk mengambil keputusan portofolio saham dengan pengembalian yang baik dan resiko yang siap diambil (Fiarni & Bastiyan, 2013). Penelitian tersebut menggunakan data historis harian saham khususnya data penutupan saham pada indeks LQ45 tahun 2010 hingga tahun 2012. Sedangkan data tingkat suku bunga harian menggunakan data dari Bank Indonesia, lalu data tingkat pengembalian pasar menggunakan data dari IHSG. Penelitian yang dilakukan (Fiarni & Bastiyan, 2013) menggunakan metode CAPM atau *Capital Asset Pricing Model* untuk mengetahui bagaimana resiko dan *return* yang diharapkan. Representasi kromosom menggunakan biner dengan gen sepanjang 8 bit dengan bilangan biner pertama untuk representasi apakah saham tersebut akan dipilih dan 7 bilangan biner lainnya sebagai representasi komposisi saham dalam portofolio. Setelah representasi kromosom dilakukan, tahapan berikutnya adalah melakukan *crossover* dengan mengganti komposisi portofolio dengan komposisi portofolio lainnya, lalu dilanjutkan dengan proses mutasi dan menghitung *fitness* dengan rumus $\frac{1}{resiko\ portofolio}$ dilanjutkan dengan proses seleksi dengan metode *elitisim.* Hasil dari penelitian tersebut algoritme genetika dapat meberikan nilai *return* lebih besar 32,28% dibandingkan dengan metode CAPM. Hasil portofolio dari algoritme genetika memberikan nilai resiko lebih kecil 1,05883 dibandingkan rata-rata resiko portofolio dari CAPM. Tingkat akurasi dari sistem yang dibuat mencapai 67%. Tingkat akurasi pada penelitian ini masih bisa disempurnakan.

Optimasi portofolio saham dengan algoritme genetika yang dilakukan oleh (Putri, Saepuddin, & Setiawan, 2014) menggunakan prinsip Markowitz untuk menentukan *return* dan resiko portofolio saham, sedangkan algoritme genetika digunakan untuk mencari bobot optimal sehingga dapat menghasilkan portofolio saham optimal. Data yang digunakan adalah data saham indeks LQ45. Representasi kromosom menggunakan biner dengan menentukan jumlah gen dari awal saat proses inisialisasi individu. Setelah itu dilakukan *decode* kromosom dengan menjumlahkan individu yang bernilai 1 pada setiap kromosom selanjutnya dibagi dengan *m*, dimana *n* adalah jumlah gen bernilai 0 dan m jumlah gen bernilai 1 sehingga bentuk *real* dari individu jika dijumlahkan hasilnya 1. *Shrinking crossover* digunakan untuk mencari keturunan, dilanjutkan proses mutasi dengan permutasi yaitu menentukan titik mutasi secara *random* kemudian ditukar, lalu mengitung *fitness* untuk seluruh individu. Pengujian yang

dilakukan yaitu dengan menghitung bobot optimal dengan inisialisasi nilai α, dimana nilai α merepresentasikan portofolio tersebut hanya mementingkan resiko dan mengabaikan *expected return* atau memperhatikan keduanya. Hasil dari pengujian menunjukkan bahwa semakin kecil nilai α berbanding lurus dengan *expected return*.

Penelitian selanjutnya tentang penentuan portofolio saham optimal menggunakan algoritme genetika menggunakan data saham indeks LQ45 beserta data *alpha*, *beta*, dan kesalahan residu saham (Wahyuni, Mahmudy, & Setiawan, 2017) dengan representasi kromosom secara *real-code* yang berisi saham dan proporsi yang ditentukan secara acak, kemudian menerapkan *single index model* sebagai metode yang digunakan untuk menghitung resiko beserta *return* pada saham. Data *alpha, beta*, dan kesalahan residu saham dibutuhkan pada perhitungan *return* dan resiko pada *single index model*. Tujuan untuk membentuk portofolio adalah mendapatkan keuntungan yang optimal dengan resiko yang wajar maka rumus *fitness* yang digunakan adalah $\frac{return\ ekspektasi}{resiko\ portofolio}$. Proses *crossover* menggunakan *extended intermediate crossover* yang mengasilkan keturunan dari dua induk, banyaknya keturunan didapatkan dari *cr* x Ukuran populas*i*. Pada penelitian tersebut dilakukan perbaikan *offspring* karena *offspring* yang dihasilkan tidak sesuai dengan *constraint*. Setelah itu, dilakukan mutasi dengan *reciprocal exchange mutation* yaitu memilih dua titik kromosom dari individu yang dipilih untuk kemudian ditukar (Mahmudy, 2015). Banyaknya kromosom yang dibutuhkan didapatkan dari rumus *mr* x Ukuran populasi. Hasil dari penelitian yang dilakukan oleh (Wahyuni, Mahmudy, & Setiawan, 2017) dapat menentukan proporsi saham beserta resiko dan *return* sehingga dapat dihasilkan portofolio saham yang optimal dengan ukutan populasi optimal yaitu 100 dengan rata-rata fitness 0,349770178 dengan nilai *cr* 0,7 dan *mr* 0,3. Namun, pada penelitian ini pengujian populasi, generasi, *cr* dan *mr* terjebak dalam kisaran yang sama.

Penerapan algoritme genetika terdistribusi atau bisa disebut dengan *parallel genetic algorithm* (PGA) pernah digunakan untuk optimasi penyusunan bahan makanan untuk keluarga penderita penyait *hiperkolesterolemia*, dimana penyakit tersebut dapat menyababkan serangan jantung atau stroke sehingga diperlukan sistem yang dapat memberikan sususan bahan makanan bagi keluarga karena penderita *hiperkolesterolemia* harus menjaga pola makan dan asupan gizinya (Sasmito, Cholissodin, & Sutrisno, 2018). *Parallel genetic algortihm* terbentuk dari sub populasi. Setiap sub populasi menggunakan operator genetika yang berbeda, sehingga tiap sub populasi berjalan secara paralel (Mahmudy, 2015). Inisialisasi kromosom pada penelitian tersebut dibuat dalam bentuk *integer* yang berisi komposisi gizi makanan keluarga untuk makan pagi, siang, dan malam. Untuk nilai kromosom pada sub populasi digunakan nilai permutasi *integer* yang merepresntasikan indeks dari bahan makanan, hasil dari tahap ini selanjutnya digunakan untuk proses reproduksi. *Extended intermediate crossover* digunakan untuk mencari *offspring* dengan menyilangkan dua individu yang dipilih secara random, sedangkan *random mutation* adalah mutasi yang digunakan pada

penelitian tersebut. Dilanjutkan dengan menghitung *fitness* gizi, *fitness* harga serta variasi bahan makanan. Tahap selanjutnya adalah seleksi dengan *elitism*, dilanjutkan dengan melakukan migrasi. Migrasi adalah proses menukar individu antar sub populasi agar menambah keragaman pada setiap sub populasi (Mahmudy, 2015), pada penelitian proses migrasi dilakukan pada generasi tertentu. Penelitian tersebut menggunakan populasi sejumlah 65 dengan sub populasi sebanyak 5, dengan jumlah generasi 60, nilai *cr* 0,4 dan mr *0,6*. Dengan batas permutasi sebesar 145 sistem menghasilkan solusi pada keluarga 1 zat gizinya tidak melebihi batas toleransi dari pakar yaitu kolesterol dan keluarga 2 berada di lemak. Pada penelitian tersebut, tidak ada kondisi untuk mengendalikan apakah bahan makanan yang terpilih sesuai dengan batas ambang toleransi atau tidak. Hal ini terjadi karena pembangkitan individu dilakukan secara acak (Sasmito, Cholissodin, & Sutrisno, 2018).

Penelitian untuk menentukan aturan perdagangan pasar saham yang menerapkan metode *parallel genetic algorithm* (PGA) atau algoritme genetika terdistribusi dibuat untuk mengatasi permasalahan dalam menganalisis data pasar, memberikan rekomendasi dan melakukan transaksi dari sejumlah data yang besar dan dalam waktu yang relatif cepat (Staflburg, Martel, & Alexandrov, 2012). Pengunaan algoritme genetika dapat mengatasi permasalahan tersebut, namun hasilnya masih kurang optimal karena terjebak pada variasi yang sama. Maka penelitian ini menggunakan PGA untuk mengatasi permasalahan tersebut, dengan menggunakan histori dari harga saham dan volume saham sebagai data masukan sekaligus sebagai *rule-based* yang kemudian digunakan bagi investor untuk mengambil keputusan. Dengan porbabilitas *crossover* sebesar 0,6%, probabilitas mutasi 0,005% dan iterasi maksimum sebanyak 200 dan populaasi sebesar 120 didapatkan hasil yang optimal dengan waktu eksekusi yang lebih cepat dan variasi hasil yang beragam. Penulis merangkum kajian pustaka dalam sebuah tabel yaitu pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No.	Judul	Objek (*Input*)	Metode (proses)	Hasil
1	Sistem Rekomendasi Portofolio Investasi Berbasis Algoritme Genetika (Fiarni & Bastiyan, 2013)	Data historis harian penutupan saham pada indeks LQ45 tahun 2010 hingga tahun 2012, data tingkat suku bunga harian menggunakan data dari Bank Indonesia, data tingkat pengembalian pasar menggunakan data dari IHSG.	- Metode CAPM atau *Capital Asset Pricing Model* digunakan untuk menghitung resiko dan *return*. - Algoritme genetika dengan representasi kromosom menggunakan biner, *crossover* dan mutasi, lalu seleksi menggunakan *elitism.*	Hasil dari penelitian yaitu nilai resiko lebih kecil 1,05883 dibandingan rata-rata resiko portofolio dari CAPM dengan akurasi 67%.
2	Optimasi Portofolio Saham Menggunakan Algoritme Genetika (Putri, Saepuddin, & Setiawan, 2014)	Data saham pada indeks LQ45	- Model Markowitz untuk mencari *return* dan resiko saham. - Algoritme genetika untuk menghitung bobot optimal pada portofolio saham - Representasi kromosom menggunakan biner, melakukan *crossover* dengan *shrinking*	Berdasarkan pengujian t-test dapat disimpulkan bahwa nilai α sangat berpengaruh karena jika α semakin kecil, maka semakin kecil pula *expected return* yang dihasilkan.

No.	Judul	Objek (*Input*)	Metode (proses)	Hasil
			crossover, mutasi dengan permutasi, lalu mengitung fitness untuk seluruh individu.	
3	Penentuan Portofolio Saham Optimal Menggunakan Algoritme Genetika (Wahyuni, Mahmudy, & Setiawan, 2017)	Data saham LQ45, data *alpha* dan beta saham, kesalahan residu saham.	- Model *single index model* untuk menghitung *return* dan resiko. - Representasi kromosom secara *real-code* yaitu saham dan proporsi saham secara *random*. - menggunakan *extended intermediate* untuk *crossover* dan *reciprocal exchange* untuk mutasi, lalu menghitung *fitness* untuk setiap individu.	Penelitian ini dapat menghasilkan proporsi dengan *return* dan resiko optimal dengan populasi 100, rata-rata fitness 0,349770178 dengan nilai *cr* 0,7 dan *mr* 0,3
4	Penerapan *Parallel Genetic Algorithm* untuk Optimasi Penyusunan Bahan Makanan Keluarga Penderita	Jenis kelamin, berat badan, tinggi badan, usia, aktivitas fisik	- Menghitung Angka Metabolisme Basal (AMB) dan aktivitas fisik. - Menghitung penalti	Hasil penelitian menunjukkan hasil optimal yaitu populasi sejumlah 65 dengan sub populasi sebanyak 5, dengan jumlah generasi 60, nilai *cr* 0,4 dan *mr* 0,6.

No.	Judul	Objek (*Input*)	Metode (proses)	Hasil
	Hiperkolesterolemia (Sasmito, Cholissodin, & Sutrisno, 2018)		gizi - Inisialisasi kromosom yang merepresentasikan komposisi gizi untuk makan pagi, siang, dan malam. *Generate* populasi awal pada setiap sub populasi, lalu melakukan *crossover* dan mutasi. Seleksi dengan *elitism,* lalu proses migrasi.	
5	*Parallel Genetic Algorithms for Stock Market Trading Rules* (Staflburg, Martel, & Alexandrov, 2012)	Data histori dari harga saham dan volume saham	- Menggunakan MA (*moving average)* sebagai *rule* untuk menentukan apakah saham akan dibeli, dijual atau *hold*. - Algoritme genetika untuk memberikan solusi terbaik pada invzestor dengan representasi kromosom secar *random*, evaluasi performa tiap kandidat	Dari pengujian yang dilakukan didapatkan hasil yang optimal yaitu porbabilitas *crossover* sebesar 0,6%, probabilitas mutasi 0,005% dengan iterasi maksimum 200 dan populaasi 120 dengan waktu eksekusi yang lebih cepat dan variasi hasil yang beragam.

No.	Judul	Objek (*Input*)	Metode (proses)	Hasil
			pada kromosom, memilih kandidat untuk rekombinasi, melakukan proses *crossover* dan mutasi, evaluasi kandidat dan dilakukan iterasi sampai batas waktu yang ditentukan.	

Pada Tabel 2.1 berisi rangkuman dari kajian pustaka yang digunakan pada penelitian ini, terdapat lima kajian pustaka utama yang digunakan. kajian pustaka yang dipilih adalah penelitian-penelitian terdahulu yang berkaitan dengan penelitian ini seperti portofolio saham dan penggunakan algoritme genetika terdistribusi sebagai solusi pada permasalahan-permasalahan tersebut.

2.2 Saham

Saham termasuk dalam instrumen pasar modal yang ditunjukkan dengan kepemilikan atau sekuritas (Kasmir, 2014). Sekuritas disebut juga sebagai surat berharga pada aset finansial yang menyatakan klaim keuangan (Tandelilin, Portofolio dan Investasi, 2010). Maka apabila seorang investor memiliki saham pada suatu perusahaan, investor tersebut turut memiliki perusahaan. Semakin besar saham yang dimiliki, semakin besar kepemilikannya terhadap suatu perusahaan. Sarana untuk jual dan beli sekuritas dalam hal ini saham disebut dengan bursa efek, di Indonesia terdapat satu bursa efek yaitu Bursa Efek Indonesia (Tandelilin, Portofolio dan Investasi, 2010). Pada Bursa Efek Indonesia atau BEI dijelaskan bahwa satu lot berarti 500 saham yang merupakan minimal pembelian saham yang harus dilakukan. Sebagai contoh, harga saham dari Perusahaan A sebesar 1000 rupiah maka biaya minimal yang dibutuhkan untuk satu lot saham sebesar 500.000 rupiah (Bursa Efek Indonesia, 2008). Jenis-jenis saham dapat diketahui dari segi kepemilikannya (Tandelilin, Portofolio dan Investasi, 2010)

a. Saham biasa

Saham biasa atau *common stocks* merupakan surat berharga yang menunjukkan kepemilikan. Saham biasa tidak memiliki waktu jatuh tempo dan bernilai nominal, sehingga perusahaan dapat menentukan memberikan nilai nominal atau tidak pada saham perusahaannya. Jika investor memiliki 1 juta lembar saham biasa dari 100 juta lembar saham, maka kepemilikan investor sebesar 1%. Pemilik saham biasa memiliki hak suara pada setiap keputusan penting dari perusahaan seperti saat diadakan Rapat Umum Pemegang Saham (RUPS). Selain itu, sebagai pemilik saham biasa, investor memiliki hak klaim pada pendapatan dan aktiva perusahaan. Jika perusahaan mendapatkan keuntungan, maka sebagian dari keuntungan atau laba tersebut dapat dibagikan kepada pemilik saham yang kemudian disebut dengan pemegang saham sebagai *dividen* (Tandelilin, Portofolio dan Investasi, 2010).

b. Saham bonus

Saham yang akan diberikan ke pemegang saham dari perusahaan berasal dari keuntungan perusahaan sehingga investor atau pemegang saham memiliki tambahan jumlah saham (Tandelilin, 2010).

c. Saham preferen

Saham preferen atau dapat disebut juga sebagai *preferred stock* adalah jenis sekuritas ekuitas yang memiliki perbedaan dengan saham biasa dan saham bonus dalam segi pembagian laba kepada pemegang saham atau dalam *dividen*.

Pada saham preferen, *dividen* yang diberikan kepada pemegang saham jumlahnya selalu sama dan tidak berubah dari waktu ke waktu. Disebut *preferred* atau dilebihkan maka pemegang saham preferen akan mendapatkan hasil *dividen* terlebih dahulu dibandingkan dengan pemegang saham biasa.

Saham dapat juga diidentifikasi dari cara peralihan dan hak tagih (Darmadji & Fakhruddin, 2012). Jenis saham berdasarkan hak tagih ada dua, yaitu saham biasa dan saham preferen. Sedangkan untuk jenis saham berdasarkan cara peralihannya, yaitu:

a. Saham atas rujuk

Saham atas rujuk adalah saham yang tidak memiliki nama pemilik, sehingga saham jenis ini mudah untuk di perjual belikan.

b. Saham atas nama

Saham atas nama adalah kebalikan dari saham atas rujuk, saham atas nama berarti saham yang memiliki nama pemilik sehingga jika saham ini akan dijual atau di beli dan dialihkan maka membutuhkan persetujuan terlebih dahulu dari pemilik saham (Darmadji & Fakhruddin, 2012).Saham-saham yang memiliki persyaratan untuk masuk ke dalam indeks saham yaitu indeks LQ45 yang akan digunakan untuk penelitian ini.

2.2.1 Indeks LQ45

Pergerakan dan intensitas transaksi pada setiap saham berbeda-beda, ada saham yang memiliki intensitas transaksi tinggi dan aktif namun ada juga saham yang tingkat transaksinya rendah dan cenderung pasif. Karena hal ini menyebabkan tingkat IHSG atau indeks harga saham gabungan yang merupakan gabungan dari seluruh saham sebagai dasar perhitungan indeks harga yang mencerminkan kondisi nyata yang terjadi pada bursa efek, maka dibentuklah indeks LQ45 (Tandelilin, 2010).

Indeks LQ45 adalah indeks saham yang ada di BEI terdiri dari 45 saham dengan kriteria perusahaan yang memiliki likuiditas dan kapitallsasi pasar yang tinggi serta memenuhi beberapa kriteria yang lainnya (Tandelilin, 2010). Indeks LQ45 akan selalu di evaluasi setiap 6 bulan sekali, dari hasil evaluasi ini bisa jadi saham yang sebelumnya termasuk dalam indeks LQ45 harus dikeluarkan dari indeks karena tidak sesuai dengan kriteria dan saham baru yang memenuhi kriteria bisa menjadi bagian dari indeks saham LQ45. Kriteria saham pada indeks LQ45 adalah:

1. Saham termasuk 60 besar ditinjau dari total transaksi sahan tersebut selama 12 bulan terakhir.
2. Urutan saham di urutkan berdasarkan tingkat kapitalisasi pasar.
3. Saham sudah tercatat atau terdaftar minimal 3 bulan di Bursa Efek Indonesia (BEI) .

4. Prospek pertumbuhan keuangan, keadaan keuangan perusahaan serta frekuensi dan jumlah hari perdagangan transaksi saham di bursa pasar.

Saham-saham yang termasuk sebagai indeks LQ45 pada periode yang sudah ditentukan pada penelitian ini yang akan digunakan sebagai data saham untuk penentuan portofolio saham optimal.

2.3 Portofolio Saham

Dalam melakukan investasi, investor akan memilih indeks saham tertentu dengan dasar keputusan investasi yang melibatkan dua hal yaitu *return* dan resiko (Tandelilin, 2001). Terdapat dua jenis *return* yaitu *actual return* dan *return* ekspektasi. *Actual return* merupakan return saham yang sesungguhnya adalah tingkat *return* yang benar-benar sudah didapatkan oleh investor sedangkan *return* ekspektasi adalah *return* yang diharapkan dari investasi yang berasal dari biaya kesempatan dan resiko daya beli. Perbedaan antara *actual return* dengan *return* ekspektasi adalah adanya resiko yang juga harus menjadi pertimbangan dari investor (Tandelilin, 2010).

Selain *return* investor juga harus memperhitungkan resiko yang akan ditanggung. Resiko dipahami sebagai kemungkinan perbedaan antara *return* yang sesungguhnya dengan *return* yang diharapkan. Resiko yang tinggi memberikan *return* ekspektasi yang juga tinggi. Namun, keputusan ini tergantung dari sifat investor itu sendiri. Apakah investor memang ingin melakukan investasi dengan resiko tinggi atau investor memilih investasi dengan *return* yang cukup baik dengan resiko yang siap ditanggung (Tandelilin, 2010).

Investor melakukan penyebaran dana atau diversifikasi pada dana yang akan diinvestasikan. Hal ini dilakukan untuk mengurangi resiko dan meningkatkan peluang mendapatkan keuntungan atau *return* (Samsul, 2015). Portofolio saham terdiri dari berbagai saham perusahaan yang bervariasi, tujuan diversifikasi pada saham antara lain dapat meminimalkan resiko jika investasi saham dilakukan di banyak perusahaan dan meningkatkan keuntungan karena jika investor tidak bergantung pada satu saham perusahaan, saat terjadi kenaikan *return* maka investor berpeluang mendapatkan keuntungan yang besar (Zubir, 2011).

Investor yang akan melakukan diversifikasi pada saham sehingga membentuk portofolio saham harus memperhatikan resiko yang harus ditanggung selain menghitung *return* yang diharapkan (Tandelilin, 2001). Apabila investor ingin melakukan diversifikasi, investor harus teliti dalam menentukan apakah akan melakukan investasi pada saham yang berpeluang memberikan *return* sangat tinggi dengan tingkat resiko tertentu atau melakukan investasi pada saham yang nilai *return* nya tidak terlalu tinggi dengan resiko yang rendah. Hal ini disebut dengan portofolio efisien (Tandelilin, 2001).

Portofolio efisien dibentuk dari asumsi perilaku investor. Salah satu asumsi yaitu semua investor tidak menyukasi resiko. Sehingga, apabila investor tersebut diberikan pilihan pada investasi yang menawarkan nilai *return* yang

sama dengan nilai resiko yang berbeda. Maka, investor akan memilih investasi dengan nilai resiko yang terendah. Portofolio yang dipilih dari beberapa portofolio efisien disebut dengan portofolio optimal. Portofolio yang dipilih oleh investor untuk menjadi portofolio optimal adalah investasi yang memiliki *return* yang diharapkan dan resiko yang dapat ditanggung (Tandelilin, 2001).

Menentukan saham yang akan dimasukkan ke dalam portofolio membutuhkan beberapa pertimbangan selain *return* dan resiko. Salah satu pertimbangan yaitu proporsi atau bobot pada saham yang dipilh. Proporsi ini akan mempengaruhi besaran *return* dan resikonya (Samsul, 2015).

2.3.1 Model Analisis Saham

Terdapat beberapa model analisis saham yang digunakan untuk menghitung kelayakan saham sehingga investor dapat menentukan saham mana yang akan dipilih untuk investasi. Setiap model analisis saham memiliki kelebihan dan kekurangannya, berikut penjelasan mengenai empat model analisis saham

1. *Markowitz*

Pengurangan resiko portofolio adalah manfaat dari penggunaan model *Markowitz*. Namun, model ini belum bisa memberikan hasil diversifikasi yang optimal karena *Markowitz* hanya didasari tiga asumsi yaitu periode investasi tunggal (1 tahun), tidak memperhitungkan biaya transaksi, dan preferensi investor hanya berdasarkan resiko dan *return* yang diharapkan (Tandelilin, 2010).

2. *Captital Asset Pricing Model* (CAPM)

Pada dasarnya, model CAPM mengadopsi teori portofolio dari *Markowitz*. CAPM menggambarkan hubungan *return* dan resiko melalui beberapa asumsi. CAPM mengembangkan asumi Markowitz menjadi beberapa asumi yang sebenarnya sesuai untuk diterapkan hanya untuk investor secara pribadi, tetapi pada kenyatannya asumsi-asumi dari CAPM tidak sesuai dengan kenyataan (Tandelilin, 2010).

3. *Arbitage Pricing Theory* (APT)

Perbedaan model APT dengan CAPM adalah asumsi dan prosedur yang berbeda, model APT tidak hanya mempertimbangkan resiko dari indeks pasar melainkan dari resiko-resiko lain yang mungkin akan mempengaruhi kepurusan investor (Tandelilin, 2010).

4. *Single Index Model*

Markowitz memiliki perhitungan kovarians yang kompleks, sehingga muncul model baru yaitu *Single Index Model* yang merupakan hasil dari pengembangan model Markowitz (Tandelilin, 2001). Model ini menggunakan *alpha* dan beta untuk menghitung resiko dan *return* dari setiap sekuritas, model ini berasumsi bahwa *return* saham berkolerasi dengan perubahan nilai pasar (Samsul, 2015).

Pemilihan model yang akan digunakan untuk menghitung dan mengetahui kelayakan portofolio saham tergantung dari preferensi investor. Pada penelitian ini, model yang paling sesuai adalah *single index* model karena model ini hanya menggunakan satu prediktor yang paling mempengaruhi *return* dan resiko saham.

2.4 Single Index Model

Pada penelitian ini, model yang digunakan untuk menghitung *return* dan resiko adalah *single index model* yang dikembangkan oleh William Sharpe. *Single index model* menghubungkan perhitungan *return* pada setiap aset dengan *return* indeks pasar. Rumus *single index model* seperti yang tertera pada persamaan 2.1.

Tahapan pertama yaitu menghitung *return* historis atau *actual return* (R_i) yang merupakan *return* yang sesungguhnya pada setiap emiten saham dengan menggunakan rumus yang tertera pada Persamaan 2.1. Rumus ini digunakan untuk menghitung *return* pada setiap saham dan indeks pasar (IHSG)

$$R_i = \frac{(P_t - (P_{t-1}))}{P_{t-1}} \quad (2.1)$$

Keterangan:

P_t = harga *close* saham pada periode *t*

P_{t-1} = harga *close* saham pada periode *t-1* atau periode sebelumnya

Setelah menghitung *actual return* atau R_i pada setiap emiten saham dan indeks pasar IHSG, selanjutnya menghitung *expected return* atau *return* investasi yang diharapkan pada masa mendatang dengan cara menghitung rata-rata dari *actual return* pada setiap saham dan IHSG, *expected return* merupakan estimasi *return* yang akan diperoleh dari suatu investasi (Tandelilin, 2001). Rumus *expected return* tertera pada Persamaan 2.2.

$$E(R) = \Sigma\frac{R_i}{n} \quad (2.2)$$

Keterangan:

E(R) = *expected return*

R_i = *actual return* pada setiap emiten saham *i*

n = jumlah periode yang digunakan

Menghitung resiko berkaitan dengan *return* yang diharapkan atau *expected return* dari setiap saham, maka untuk menghitung resiko dapat dilakukan dengan menghitung varians pada setiap saham (Tandelilin, 2001). Varians menunjukkan besarnya penyebaran variabel *rendom* maka semakin besar penyebarannya, semakin besar pula variansya (Tandelilin, 2001).

$$\sigma_i{}^2 = \Sigma_{n-1} \frac{\left(R_i - E(R_i)\right)^2}{n-1} \quad (2.3)$$

Keterangan:

$\sigma_i^{\ 2}$ = varians return dari saham *i*

R_i = *return* saham *i*

E(R) = *expected return*

n = jumlah periode *actual return* saham *i*

Nilai *alpha* (α) adalah titik potong dari hubungan linear dari *actual return* saham i dengan *actual return* dari pasar, dalam penelitian ini menggunakan IHSG sebagai acuan harga pasar. *Alpha* digunakan untuk menghitung *variance error* atau kesalahan residual (e_i). Rumus menghitung *alpha* (α) dijabarkan pada Persamaan 2.4

$$\alpha_i = R_i - \beta_i \times R_m \quad (2.4)$$

Keterangan:

α_i = *alpha* saham *i*

R_i = *return* saham *i*

β_i = *beta* saham *i*

R_m = *return* pasar atau *return* market yaitu *return* dari harga IHSG

Beta (β) digunakan untuk mengukur resiko pada portofolio karena *beta* merupakan resiko yang sistematis. Rumus menghitung *beta* dijabarkan pada Persamaan 2.5

$$\beta_i = \frac{\sigma_{im}}{\sigma_m} \quad (2.5)$$

Keterangan:

β_i = *beta* saham *i*

σ_{im} = kovarian atau standar deviasi saham *i* dan pasar

σ_m = kovarian atau standar deviasi pasar

Tahapan selanjutnya adalah menghitung *variance error residual* atau disebut dengan resiko tidak sistematis pada saham dengan menggunakan rumus yang dijabarkan pada Persamaan 2.6

$$\sigma_{ei}^{\ 2} = \frac{\Sigma_{i=1}^{n} \left(R_i - E(R_i)\right)^2}{n-1} \quad (2.6)$$

Keterangan:

σ_{ei}^2 = varians *error residual* atau resiko tidak sistematis saham

R_i = *return* saham *i*

α_i = *alpha* saham *i*

β_i = *beta* saham *i*

R_m = *return market*

e_i = *variance error*

Sedangkan untuk perhitungan *return* portofolio pada *single index model* menggunakan dua komponen yaitu *alpha* (α) yang merupakan komponen *return* yang menggambarkan keunikan perusahaan, komponen kedua yaitu *beta* (β) yang berkaitan dengan pasar (Tandelilin, 2001). Sebelum menghitung *return* portofolio dilakukan perhitungan *alpha* portofolio dan *beta* portofolio dengan rumus pada Persamaan 2.7

$$\alpha_p = \sum_{i=1}^{N} W_i \alpha_i \quad (2.7)$$

Keterangan:

α_p = *alpha* portofolio

W_i = bobot pada saham *i*

α_i = nilai *alpha* pada saham *i*

N = jumlah saham

Selanjutnya adalah menghitung *beta* (β) portofolio dengan rumus yang dijabarkan pada persamaan 2.8

$$\beta_p = \sum_{i=1}^{N} W_i \beta_i \quad (2.8)$$

Keterangan:

β_p = *beta* portofolio

W_i = bobot pada saham i

β_i = nilai *beta* pada saham i

N = jumlah saham

Setelah mengetahui *alpha* portofolio dan *beta* portofolio pada saham, selanjutnya dilakukan proses perhitungan ekspektasi *return* portofolio dengan rumus yang tertera pada persamaan 2.9

$$(E)R_p = \alpha_p + \beta_p R_M \quad (2.9)$$

Keterangan:

$(E)R_p$ = resiko portofolio

α_p = *alpha* portofolio

β_p = *beta* portofolio

R_M = *return market* yang didapatkan dari harga IHSG

Sebelum menghitung resiko portofolio, dilakukan perhitungan kesalahan residu atau *variance error* portofolio dengan rumus yang tertera pada Persamaan (2.10)

$$\sigma_{ep}{}^2 = \sigma_{ei}{}^2 \times W_i^2 \quad (2.10)$$

Keterangan:

$\sigma_{ep}{}^2$ = kesalahan residu portofolio

σ_{ei} = kesalahan residu saham atau *variance error residual*

W_i = bobot pada saham i

Setelah menghitung kesalahan residu portofolio, langkah berikutnya adalah menghitung resiko portofolio. Resiko portofolio dapat didapatkan dari menjumlahkan hasil perkalian dari beta dan varians pasar dengan varians residu portofolio. Rumus menghitung resiko terdapat pada Persamaan 2.11757

$$\sigma_p^2 = \beta_p^2 \sigma m^2 + \sum_{i=1}^{N} W_i^2 \sigma_{ei}^2 \quad (2.11)$$

Keterangan:

σ_p^2 = resiko portofolio

β_p^2 = beta portofolio

σm^2 = varians market

W_i^2 = bobot saham

σ_{ei}^2 = kesalahan residu saham

2.5 Algoritme Genetika

Algoritme genetika merupakan salah satu tipe dari algoritme evolusi yang paling banyak digunakan. Algoritme genetika dapat menyelesaikan masalah yang kompleks, permasalahan yang sulit diselesaikan dapat diselesaikan dengan algortima genetika untuk mencari solusi terbaik atau solusi optimal (Mahmudy, 2015). Algoritme genetika melakukan proses pencarian dan optimasi dengan cara yang berbeda dibandingkan dengan proses pencarian dan optimasi biasa (Micalewicz, 1996).

Algortime genetika terdiri dari individu yang merupakan kumpulan dari kromosom, kromosom terdiri dari kumpulan gen dimana gen adalah satuan dasar yang memiliki arti tertentu sehingga kumpulan dari gen dapat disebut sebagai individu atau kromosom. Kumpulan dari individu atau kromosom disebut dengan populasi, sedangkan generasi adalah satuan siklus yang dialami oleh algoritme genetika (Mahmudy, 2015).

Tahapan dari algoritme genetika dimulai dengan inisialisasi yang bertujuan untuk membangkitkan himpunan solusi baru secara acak yang terdiri dari sejumlah kromosom dalam sebuah populasi. Selanjutnya dilanjutkan dengan proses reproduksi, reproduksi dilakukan untuk menghasilkan *offspring* atau keturunan dari individu yang ada di populasi. Terdapat dua jenis reproduksi yang dilakukan yaitu *crossover* dan mutasi, *crossover* adalah proses tukar silang yang harus ditentukan terlebih dahulu dengan cr atau *crossover rate* untuk mengetahui berapa jumlah keturunan yang harus dihasilkan pada proses reproduksi dengan *crossover*. Selanjutnya adalah reproduksi mutasi yang sebelum proses reproduksi, harus ditentukan terlebih dahulu nilai mr atau *mutation rate* agar dapat diketahui berapa keturunan yang harus dihasilkan pada mutasi. Tahap evaluasi yaitu menggabungkan seluruh individu *parent* dan

child yang dihasilkan, untuk kemudian dilakukan proses seleksi yaitu memilih individu terbaik untuk menjadi himpunan individu baru (Mahmudy, 2015).

2.5.1 Algoritme Genetika Terdistribusi

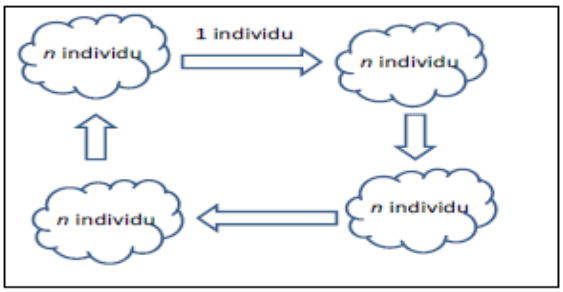
Algoritme genetika terdistribusi atau DGA sering juga disebut dengan *Parallel Genetic Algorithm.* Algoritme ini dibutuhkan karena jika hanya menggunakan algoritme genetika, solusinya sering terjebak dalam daerah yang sama atau pada daerah optimum lokal. Hal ini terjadi karena variasi atau keragaman individu dalam populasi masih kurang (Mahmudy, 2015).

Algoritme genetika terdistribusi bekerja secara paralel sehingga dapat memberikan solusi yang lebih baik dibandingan dengan algoritme genetika biasa. Algoritme genetika terdistribusi memungkinkan keanekaragaman yang terjaga dan bervariasi, karena pekerjaan algoritme ini dapat dilakukan secara paralel maka dapat dilakukan paralelisasi perangkat keras sehingga dapat menjaga lebih banyak ketersediaan memori dan CPU tambahan (Alba & Troya, 1999). Meskipun penerapan DGA memungkinkan untuk melakukan paralelisasi pada perangkat keras, namun faktanya meski hanya menggunakan prosesor tunggal DGA tetap mampu memberikan hasil yang lebih baik daripada algoritme genetika biasa.

Solusi yang digunakan pada algoritme ini untuk mengatasi permasalahan tersebut ialah dengan meletakkan individu pada beberapa sub populasi, dimana pada setiap sub populasi akan diterapkan proses *crossover,* mutasi dan seleksi dengan operasi yang berbeda (Mahmudy, 2015). Untuk memindahkan satu atau beberapa individu dari suatu sub populasi ke dalam sub populasi lain disebut dengan operator migrasi.

Terdapat beberapa strategi pada mekanisme migrasi untuk memilih kromosom apa yang akan dipilih untuk di migrasi (Yussof & Razali, 2012). Migrasi dapat dilakukan dengan memilih kromosom terbaik dari sebuah sub populasi untuk ditukar dengan kromosom terburuk dari sub populasi yang lain, kromosom terbaik di migrasi dengan kromosom lain secara acak, kromosom acak dari sebuah sub populasi di migrasi dengan kromom terburuk dari sub populasi lain dan kromosom acak migrasi dengan kromosom acak dari sub populasi yang lain (Yussof & Razali, 2012).

Pada penelitian ini, mekanisme migrasi yang dipilih adalah menukar satu kromosom terbaik dengan kromosom terbaik lainnya (Mahmudy, 2015) untuk menjaga keragaman pada sub populasi. Mekansime migrasi dapat diilustrasikan seperti pada Gambar 2.1.



Gambar 2.1 Mekanisme Migrasi

Sumber: (Mahmudy, 2015)

Gambar 2.1 menunjukkan mekanisme migrasi dimana terdapat empat sub populasi dan pada setiap generasi tertentu setiap satu individu terbaik dari suatu sub populasi dipindahkan ke sub populasi yang lain. Migrasi adalah mekanisme penting dalam algoritme genetika terdistribusi karena tanpa adanya migrasi tidak akan ada interaksi antar sub populasi sehingga tidak ada perbedaan dengan algoritme gentika biasa, dengan adanya migrasi tercipta pula interaksi antar sub populasi sehingga dapat menjaga keragaman pada sub populasi (Yussof & Razali, 2012).

2.5.1.1 Representasi kromosom

Kromosom dibentuk untuk merepresentasikan saham, serta proporsi pada setiap saham dengan bilangan acak. Pada tahap ini pula akan ditentukan nilai ukuran populasi untuk menentukan jumlah individu dalam sebuah populasi. Pada DGA terdapat lebih dari 1 sub populasi sehingga akan di definiskan pula jumlah sub populasi yang akan digunakan.

Setiap individu akan diberi nilai secara acak yang merepresentasikan nilai proporsi saham dari saham yang dipilih. Setelah itu, akan dilanjutkan proses reproduksi dari hasil *generate* populasi awal. Contoh representasi kromosom atau *generate* populasi awal dengan dua sub populasi dapat dilihat pada Gambar 2.2 dan Gambar 2.3

Parent	**Kromosom**			
P_1	0,3	0,4	0,2	0,1
P_2	0,6	0,2	0,1	0,1

Gambar 2.2 Inisialisasi sub populasi 1

Pada Gambar 2.2 dapat diketahui pada sub populasi yang pertama memiliki dua induk dengan kromosom yang merepresentasikan proporsi saham dari 4 indeks saham yang dipilih. Proporsi saham dibangkitkan secara acak pada interval [0,1].

Parent	Kromosom			
P_1	0,2	0,5	0,2	0,1
P_2	0,4	0,1	0,3	0,2

Gambar 2.3 Inisialisasi sub populasi 2

Sub populasi kedua memiliki jumlah *parent* atau induk dan kromosom yang sama dengan sub populasi satu, hal ini dapat dilihat dari Gambar 2.3 dimana kromosom merepresentasikan proporsi saham dari 4 indeks saham yang dipilih, sehingga setiap saham memiliki proporsi yang berbeda-beda dengan nilai proporsi saham yang dibangkitkan secara acak.

2.5.1.2 Crossover

Dalam *crossover* harus ditentukan nilai cr atau *crossover rate* yang merupakan rasio *offspring* yang dihasilkan pada proses *crossover* terhadap ukuran populasi sehingga dihasilkan keturunan sebanyak cr x *Ukuran populasi*. Misalnya ditentukan nilai cr 0,5 dengan *Ukuran populasi* 4 maka akan dihasilkan 0,5 x 4 = 2, yang berarti akan dihasilkan dua *offspring*.

Terdapat beberapa metode *crossover* yang dapat digunakan pada algoritme genetika seperti *one-cut point crossover, extended intermediate crossover, order crossover* (OX), *cycle crossover* (CX), *partial mapped crossover* (MRX), *heuristic crossover* dan *order-based crossover* (Gen & Cheng, 1997 dalam (Mahmudy, 2015),

Jenis reproduksi yang akan digunakan pada penelitian ini adalah *extended intermediate crossover* untuk menghasilkan keturunan atau *offspring* dari kombinasi dua induk yang dipilih secara acak. Misalnya induk P_1 dan P_2 dipilih sebagai *parent* untuk menghasilkan *offspring*, menentukan *offspring* C_1 dan C_2 didapatkan dengan rumus sebagai berikut

$$C_1 = P_1 + \alpha(P_2 - P_1)$$

$$C_2 = P_2 + \alpha(P_1 - P_2) \qquad (2.12)$$

Pada persamaan (2.7) nilai α dibangkitkan secara acak pada interval yang sudah ditentukan sebelumnya (Mahmudy, 2015). Umumnya, nilai α dibangkitkan pada interval [0,1]. Contoh *extended intermediate crossover* dapat dilihat pada Gambar 2.4

α	0,71	0,14	0,25	0,87
P_1	0,3	0,4	0,2	0,1
P_2	0,7	0,2	0,6	0,5

C_1	0,3	0,228	0,125	0,226

C_2	0,3	0,372	0,175	0,374

Gambar 2.4 Contoh *crossover*

Berdasarkan Gambar 2.4 induk yang dipilih sebagai *parent* adalah P1 dan P2 dengan nilai α yang dibangkitkan secara acak pada interval [0,1] sehingga dapat menghasilkan 2 *offspring* yaitu C_1 dan C_2 dengan menggunakan rumus pada Persamaan (2.7).

Sedangkan *one cut point crossover* dilakukan dengan cara memilih *cut point* atau titik potong secara acak kemudian menukar nilai gen untuk menghasilkan *offspirng*. Satu kali proses *one cut point* menghasilkan 2 *child* dari 2 indvidu *parent* yang dipilih secara acak. Contoh *one cut point crossover* dapat dilihat pada Gambar 2.5.

P1	0,3	0,4	0,2	0,1
P2	0,7	0,2	0,6	0,5

C1	0,3	0,4	0,6	0,5
C2	0,7	0,2	0,2	0,1

Gambar 2.5 Contoh *one cut point crossover*

Dari Gambar 2.5 dapat dilihat bahwa titik potong terletak pada gen kedua sehingga child yang akan dihasilkan adalah menukar gen ketiga dan keempat pada P1 ke P2 begitu juga sebaliknya.

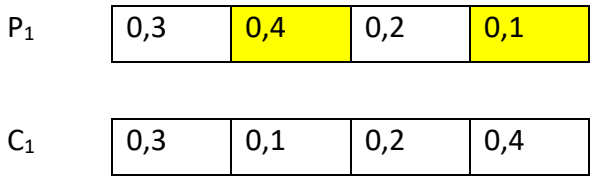
Pada penelitian ini akan digunakan dua jenis *crossover* yang berbeda untuk masing-masing sub populasi yaitu *extended intermediate crossover* dan *one-cut point crossover*.

2.5.1.3 Mutasi

Proses reproduksi yang akan digunakan selain *crossover* adalah mutasi, mutasi dilakukan dengan cara menukar, merubah atau menggeser kromosom pada suatu individu yang dipilih (Mahmudy, 2015). Mutasi dilakukan agar keragaman populasi terjaga, karena mutasi dapat melahirkan individu baru dari modifikasi satu gen atau lebih pada individu yang sama. Pada mutasi, dibutuhkan *mutation rate* atau mr untuk menentukan berapa banyak *offspring* yang akan dihasilkan dengan cara mr x *Ukuran populasi*. Misalnya mr = 0,2 dengan *Ukuran populasi* 5 sehingga akan dihasilkan 1 *offspring* pada proses mutasi.

Terdapat beberapa jenis mutasi yaitu *insertion mutation,* dan *reciprocal exchange mutation*. *Insertion mutation* yaitu memilih satu posisi secara acak lalu

mengambil dan menyisipkan nilainya pada posisi lain yang ditentukan juga secara acak, mutasi ini digunakan pada representasi permutasi (Mahmudy, 2015). Jenis mutasi yang akan digunakan pada penelitian ini adalah *reciprocal exchange mutation,* contoh reproduksi dengan mutasi terdapat pada Gambar 2.6.



Gambar 2.6 Contoh mutasi

Pada Gambar 2.5 dilakukan proses mutasi yaitu *reciprocal exchange mutation.* Metode mutasi ini menghasilkan *offspring* dengan cara memilih dua posisi sebagai *exchange point* secara acak kemudian nilai pada gen tersebut ditukar (Mahmudy, 2015). Misalnya, dipilih induk P_1 dengan memilih 0,4 dan 0,1 yang akan dilakukan proses penukaran. Sehingga hasil keturunan atau *offspring*nya gen kedua yaitu 0,4 bertukar posisi dengan gen keempat yaitu 0,1, maka *offspring* yang dihasilkan menjadi gen kedua bernilai 0,1 dan gen keempat bernilai 0,4.

Reciprocal exchange mutation adalah operator mutasi yang akan digunakan pada penelitian ini untuk setiap sub populasi.

2.5.1.4 Menghitung *fitness*

Fungsi *fitness* bertujuan untuk mengukur seberapa baik sebuah individu sehingga individu terbaik merupakan solusi terbaik yang dapat diperoleh. Semakin besar nilai *fitness* yang dihasilkan maka semakin besar peluang individu tersebut menjadi calon solusi terbaik (Mahmudy, 2015). Rumus *fitness* disesuaikan pada kasus penentuan portofolio saham, sehingga dihasilkan rumus *fitness* pada Persamaan 2.8 untuk nilai *return* lebih besar dari nilai resiko.

$$Fitness = (\frac{Return\ ekspektasi}{Resiko\ portofolio})/1000)) \quad (2.13)$$

Dari persamaan (2.8), nilai *fitness* pada setiap individu dapat ditentukan dari *return* ekspektasi dibagi resiko portofolio. Formula ini dipilih karena semakin kecil pembagi atau resiko portofolio maka nilai *fitness* akan semakin besar (Wahyuni, Mahmudy, & Setiawan, 2017), hal ini didasari dengan sifat investor yang mencari portofolio saham dengan resiko terkecil. Namun jika nilai *return* lebih kecil daripada resiko maka *fitness* dikali (-1) menjadi seperti pada Persamaan (2.9).

$$Fitness = (\frac{Return\ ekspektasi}{Resiko\ portofolio})/1000)) \ x\ (-1) \quad (2.14)$$

2.5.1.5 Evaluasi

Untuk mengetahui bagaimana kualitas dari setiap kromosom yang ada maka dilakukan proses evaluasi. Evaluasi direpresentasikan dalam *fitness*, nilai

fitness yang akan menjadi pertimbangan untuk menentukan solusi. Semakin besar nilai *fitness* maka kemungkinan dipilihnya suatu individu tersebut juga semakin besar (Mahmudy, 2015).

2.5.1.6 Seleksi

Seleksi dilakukan pada setiap individu mulai dari *parent* hingga *child* atau *offspring*, seleksi ini akan menghasilkan individu mana yang memiliki *fitness* terbaik sehingga dapat lolos ke tahap berikutnya (Gen & Cheng, 1997 dalam (Mahmudy, 2015)).

Terdapat beberapa metode seleksi yang dapat digunakan pada algoritme genetika seperti *roulette wheel* yang menghitung nilai probabilitas pada setiap individu berdasarkan *fitness*nya. Sehingga semakin besar *fitness* suatu individu semakin besar peluang individu tersebut untuk dipilih, namun individu terbaik tidak terjamin dapat selalu terpilih karena seleksi ini menggunakan peluang (Mahmudy, 2015).

Metode selanjutnya yaitu *replacement selection* yang menjamin individu terbaik akan selalu lolos pada generasi berikutnya tetapi tidak menutup kemungkinan individu dengan nilai *fitness* yang lebih rendah untuk lolos ke generasi berikutnya (Mahmudy, 2015).

Elitism selection adalah metode seleksi yang akan digunakan pada penelitian ini, seleksi ini memilih individu-individu terbaik berdasarkan pada nilai *fitness*nya untuk lolos menjadi generasi berikutnya. Metode ini akan mengumpulkan seluruh individu yaitu *parent* dan *offspring* pada satu wadah dimana individu terbaik pada wadah tersebut akan dipilih untuk masuk menjadi generasi berikutnya. Metode ini menjamin bahwa individu terbaik yaitu individu dengan nilai *fitness* tertinggi akan selalu lolos (Mahmudy, 2015). Penelitian menunjukkan bahwa *elitism* menghasilkan solusi yang lebih optimal dibandingkan dengan metode seleksi yang lainnya (Sharma & Mehta, 2013).

2.5.1.7 Migrasi

Migrasi adalah proses menukar individu antar sub populasi dengan tujuan untuk menambah keragaman pada setiap sub populasi yang ada. Migrasi merupakan mekanisme yang dapat menjamin dan menjaga keragaman variasi individu karena individu terbaik pada suatu sub populasi akan ditukar pada sub populasi yang lain, hal ini algoritme genetika terdistribusi dapat menghasilkan solusi yang lebih optimal dengan keragaman individu yang variatif (Mahmudy, 2015). Pada penelitian ini, proses migrasi dilakukan setiap iterasi, sehingga diaharapkan dapat menjaga keragaman individu pada agar tidak mengalami konvergen dini.

BAB 3 METODOLOGI

Bab ini akan dijelaskan bagaimana tahapan-tahapan penelitian yang akan dilakukan, dimana setiap tahapan akan dijeabarkan sehingga dapat membentuk kerangka penelitian yang sistematis.

3.1 Tahapan Penelitian

Tahapan-tahapan yang jelas diperlukan pada penelitian ini karena penelitian yang sistematis dengan tahapan yang jelas daat memudahkan dalam melaksanakan penelitian. Tahapan yang akan dilakukan pada penelitian ini adalah studi literatir, pengumpulan data, analisis dan perancangan sistem, implementasi, pengujian, dan penarikan kesimpulan. Tahapan ini dapat digambarkan seperti pada Diagram 3.1 dibawah ini

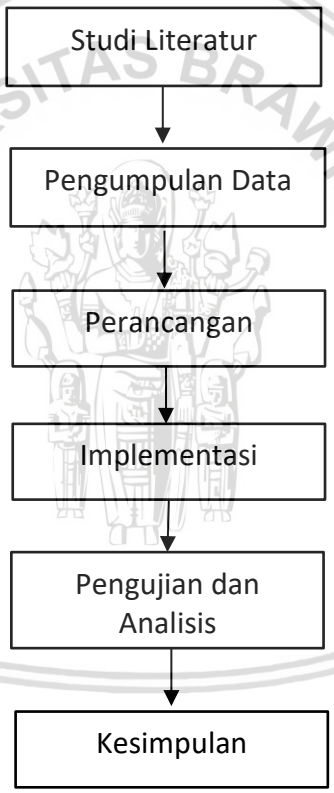


Diagram 3.1 Tahapan Penelitian

Diagram 3.1 menunjukkan bagaimana alur atau urutan yang akan dikerjakan dalam penelitian ini untuk menentukan portofolio saham optimal dengan algoritme genetika terdistribusi. Diawali dengan tahapan studi literatur, kemudian dilanjutkan dengan pengumpulan data untuk menunjang penelitian, setelah itu dilakukan proses analisis kebutuhan, setelah itu peneliti akan melakukan perancangan sistem, setelah sistem sudah dirancang dan dibuat maka akan dilakukan proses pengujian sistem supaya dapat menghasilkan kesimpulan.

3.2 Studi Literatur

Studi literatur merupakan proses pengumpulan dan pengkajian teori-teori yang mendukung penelitian. Teori-teori yang akan menjadi dasar acuan dalam melakukan penelitian sehingga menghasilkan penelitian yang dapat dipertanggungjawabkan karena didukung oleh teori yang ilmiah. Studi literatur yang dijadikan dasar pada penelitian ini adalah:

1. Saham
2. Portofolio saham
3. Perhitungan dengan *single index model*
4. Metode algoritme genetika terdistribusi

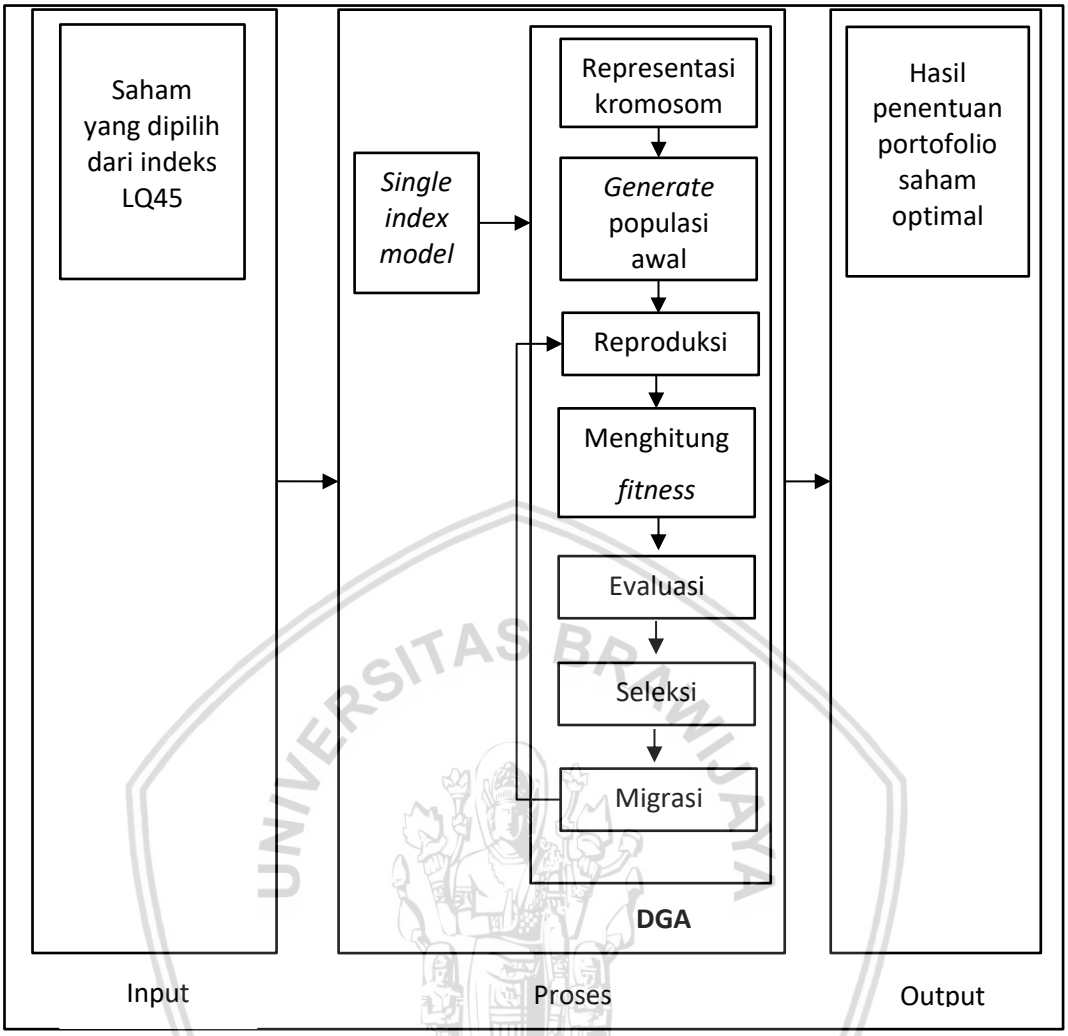
3.3 Pengumpulan data

Pengumpulan data yang dibutuhkan pada penelitian ini didapatkan melalui website resmi *yahoo.finance.com.* Data yang dibutuhkan antara lain:

1. Data historis harga penutupan saham indeks LQ45 pada periode Januari 2013-Januari 2018 dengan 21 indeks saham yang dipilih karena memiliki data harga penutupan saham yang lengkap pada periode yang sudah ditentukan.
2. Data diolah kedalam bentuk tabel yang berisi 21 indeks saham dengan harga penutupan pada periode Januari 2013-Januari 2018, serta data harga penutupan IHSG sebagai acuan harga pasar pada periode Januari 2013-Januari 2018.
3. Pada setiap data indeks saham dilakukan proses perhitungan dengan *single index model* sehingga menghasilkan data *alpha, beta* dan kesalahan residu saham.

3.4 Perancangan Sistem

Pada perancangan sistem akan dijabarkan tentang langkah kerja yang akan dilakukan dalam sistem. Langkah-langkah ini disesuaikan dengan tahapan *single index model* yang kemudian akan dilanjutkan dengan metode algoritme genetika terdistribusi. Pada Gambar 3.2 akan digambarkan bagaimana rancangan sistem pada penelitian ini yang meliputi masukan saham, menghitung *return* dan resiko dengan *single index model*, representasi kromosom dan membetuk sub populasi, *generate* populasi, reproduksi dengan *crossover* dan mutasi, menghitung nilai *fitness*, proses seleksi, dan yang terakhir adalah proses migrasi. Setelah itu sistem akan memberikan hasil keluaran berupa portofolio saham optimal.



Gambar 3.1 Perancangan Sistem

Perancangan sistem yang dilakukan pada penelitian ini dijabarkan pada Gambar 3.1, dimana terdapat tiga blok proses yaitu *input*, proses, dan *output*. Input dari sistem adalah data saham dari indeks LQ45, yang kemudian di proses dengan *single index model* untuk mencari *return* dan resiko dari saham yang dipilih. Setelah itu akan dilakukan proses penentuan portofolio saham yang optimal dengan algoritme genetika terdistribusi yaitu representasi kromosom yang berisi saham yang dipilih dengan proporsi saham secara acak, dilanjutkan dengan proses *generate* populasi awal menjadi beberapa sub populasi. Setelah itu dilakukan proses reproduksi pada setiap sub populasi dengan *crossover* dan mutasi. Tahap selanjutnya adalah menghitung *fitness* untuk kemudian dilakukan evaluasi. Selanjutnya dilakukan proses seleksi dengan *elitism selection* sehingga menghasilkan individu terpilih dengan *fitness* terbesar. Proses terakhir adalah migrasi yaitu menukar individu terbaik pada setiap sub populasi sehingga dapat meningkatkan hasil pencarian dan menjaga keragaman variasi individu (Mahmudy, 2015). Tahap terakhir yaitu *output*, *output* yang diharapkan adalah sistem mampu memberikan hasil portofolio saham optimal kepada investor.

3.5 Implementasi Sistem

Sistem diimplementasikan sesuai dengan perancangan sistem yang dibuat dengan didasari oleh studi literatur yang dijadikan dasar acuan dalam membangun sistem pada penelitian ini. Sistem akan dibangun dengan menggunakan bahasa pemrograman Java, dengan menggunakan MySQL sebagai database. Penggunaan *Microsoft Excel* juga diterapkan untuk proses manualisasi.

3.6 Pengujian dan Analisis

Pengujian sistem diperlukan untuk mengetahui apakah sistem berjalan dengan baik dan dapat memberikan hasil optimasi porotofolio saham sesuai yang diharapkan. Pengujian yang akan dilakukan antara lain pengujian ukuran populasi, pengujian generasi, pengujian kombinasi cr dan mr serta pengujian jumlah sub populasi.

Pengujian dilakukan dengan melakukan data masukan yang sama dengan 10 kali percobaan untuk setiap pengujian parameter. Diawali dengan menguji jumlah ukuran populasi, hasil populasi optimal menjadi masukan untuk pengujian selanjutnya yaitu menguji generasi. Setelah didapatkan populasi dan generasi optimal, dilakukan pengujian *crossover rate* (cr) dan *mutation rate* (mr) menggunakan populasi dan generasi optimal. Pengujian yang terakhir adalah pengujian jumlah sub populasi menggunakan populasi, generasi kombinasi cr dan mr yang optimal dan kemudian dilakukan analisis dari hasil pengujian.

3.7 Penarikan Kesimpulan

Kesimpulan didapatkan dari hasil analisa pengujian yang kemudian akan menjawab pertanyaan yang ada pada rumusan masalah. Setelah melakukan proses penarikan kesimpulan maka akan dituliskan saran yang dapat digunakan sebagai masukan bagi penelitian selanjutnya.

BAB 4 PERANCANGAN

Permasalahan pada penelitian ini adalah bagaimana menentukan portofolio saham optimal yang memiliki keuntungan yang besar dan resiko yang dapat ditanggung oleh investor. Portofolio saham merupakan sesuatu yang penting untuk investor agar dapat memetakan investasinya pada emiten sahan yang tepat, namun pemilihan portofolio saham tidak mudah karena harga saham yang fluktuatif serta harus melihat bagaimana *track record* saham-saham tersebut selama suatu periode tertentu.

Algoritme Genetika Terdistribusi digunakan sebagai solusi untuk permasalahan penentuan portofollio saham yang optimal. Metode ini dapat memberikan hasil yang lebih variatif dibandingkan metode algoritme genetika biasa karena proses reproduksi yang berbeda pada setiap sub populasi serta terdapat mekanisme migrasi yang dapat menjaga variasi individu sehingga dapat memeberikan hasil yang lebih optimal.

Pada bab ini akan dijelaskan bagaimana perancangan algortitme yang berupa *flowchart* pada setiap proses yang akan dilakukan dalam menentukan portofolio saham optimal. *Flowchart* yang ada pada bab ini adalah *flowchart* untuk proses *crossover* yaitu *extended intermediate crossover* dan *one cut point crossover,* proses mutasi, proses menghitung *fitness* yang terdiri dari proses perhitungan *alpha, beta,* dan residu portofolio, kemudian menghitung *return* ekspektasi dan resiko portofolio untuk akhirnya menghitung *fitness,* selanjutnya adalah proses evaluasi. Dilanjutkan dengan *flowchart* untuk proses seleksi dan yang terakhir adalah *flowchart* untuk mekanisme migrasi.

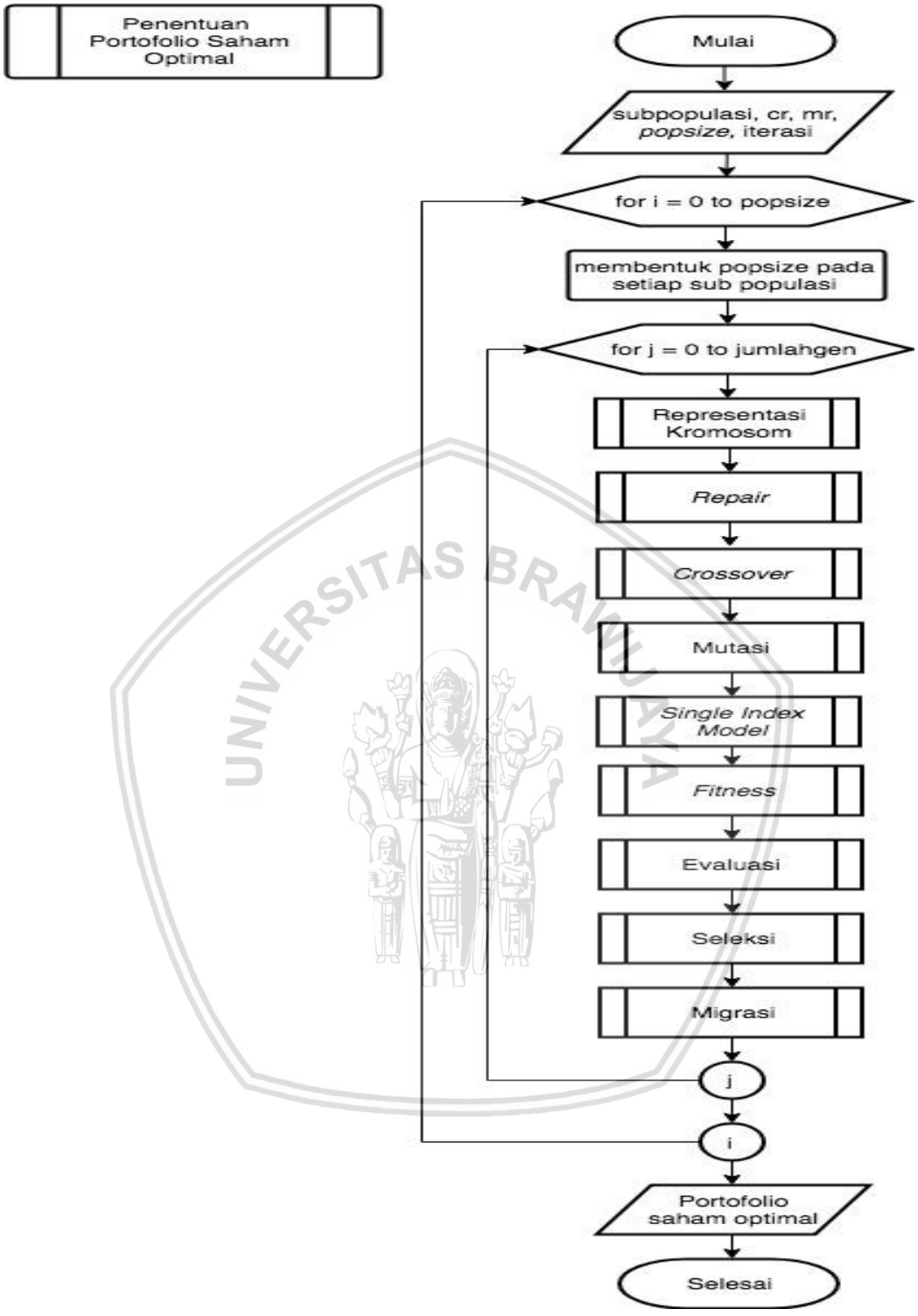
Proses perhitungan manual juga akan dijelaskan pada bab ini, perhitungan manual dibutuhkan untuk menunjukkan bagaimana proses penerapan algoritme genetika terdistribusi ini dalam menentukan portofolio saham optimal.

Perancangan antarmuka akan dijelaskan pada bab ini sebagai gambaran bagaimana antarmuka sistem penentuan portofolio saham optimal menggunakan algoritme genetika terdistribusi.

Selain perancangan algortitme, perhitungan manual dan perancangan antarmuka, bab ini juga akan menjelaskan tentang perancangan pengujian yang akan dilakukan. Perancangan pengujian ini dibutuhkan untuk mengetahui bagaimana pengaruh parameter yang digunakan pada algoritme genetika terdistribusi dalam menentukan portofolio saham yang optimal. Perancangan pengujian terdiri dari perancangan pengujian ukuran populasi, perancangan pengujian jumlah generasi, dan perancangan pengujian kombinasi cr dan mr.

4.1 Perancangan Algoritme

Pada sub bab ini akan dijelaskan mengenai bagaimana *flowchart* atau proses yang digunakan untuk menentukan portofolio saham optimal dengan algoritme genetika terdistribusi. *Flowchart* pertama dapat dilihat pada Gambar 4.1.



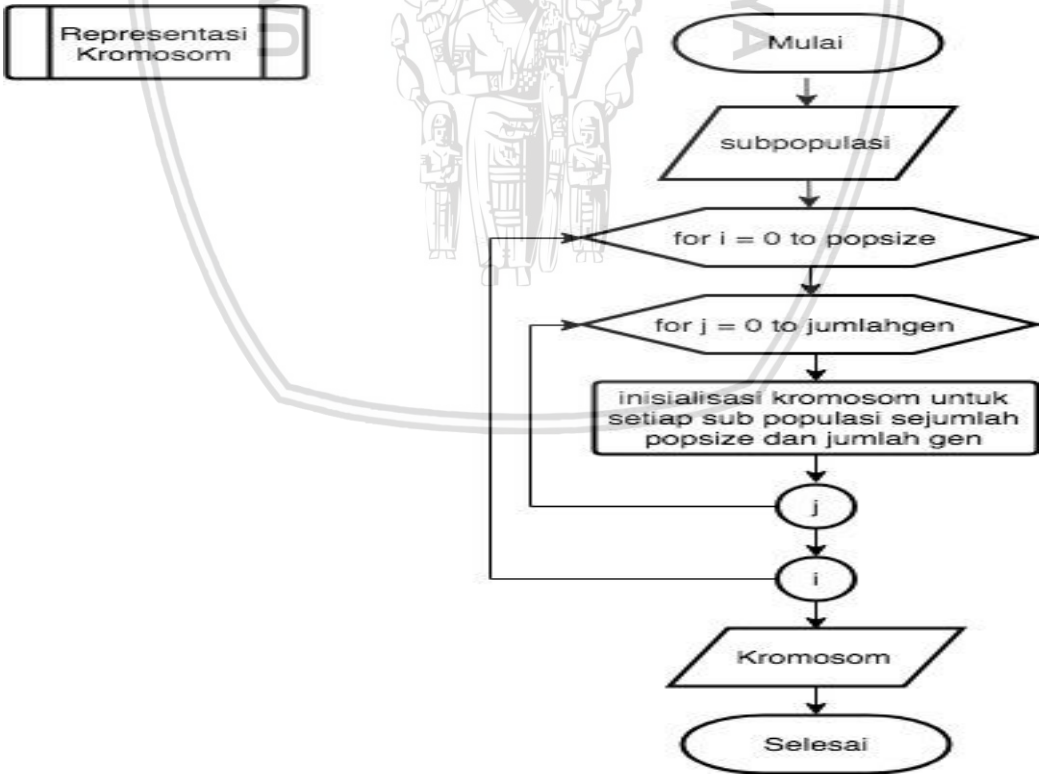
Gambar 4.1 Diagram Alir Penentuan Portofolio Saham Optimal

Tahap pertama yaitu memilih saham apa saja yang diinginkan, kemudian mengisi nilai cr, mr, dan jumlah generasi. Jumlah sub populasi sudah ditentukan oleh program sehingga tidak perlu ada inisialisasi lagi, ukuran popsize sama dengan jumlah saham yang dipilih. Selanjutnya akan dilakukan proses inisialisasi nilai gen pada setiap kromosom untuk setiap sub populasi secara random, sub

populasi yang sudah terisi akan masuk ke proses reproduksi yaitu *crossover* dan proses mutasi untuk menghasilkan keturunan. Terdapat dua proses *crossover* yaitu *extended intermediate crossover* dan *one cut point crossover* yang dipilih secara acak sehingga pada setiap sub populasi akan melakukan proses *crossover* yang berbeda. Setelah itu dilakukan proses evaluasi yaitu menggabungkan *parent* dengan *child* dalam setiap sub populasi tersebut yang kemudian akan dilakukan perhitungan *fitness*. Rumus *fitness* didaptkan melalui perhitungan *single index model*. Setelah nilai *fitness* pada setiap individu diketahui, dilakukan proses seleksi dengan *elitism selection* sehingga menghasilkan individu terbaik sejumlah dengan *popsize* yang sudah ditentukan. Proses terakhir yaitu migrasi, dengan menukar satu individu terbaik pada sebuah sub populasi ke sub populasi yang lain begitu juga sebaliknya.

4.1.1 Representasi Kromosom

Representasi kromosom berfungsi sebagai populasi awal yang dibentuk, kromosom dibentuk berdasarkan jumlah *popsize* dan jumlah gen untuk setiap sub populasi. Representasi kromosom dibangkitkan secara acak pada rentang 0-1 dengan kondisi total jumlah kromosom pada satu individu tidak lebih dari 1 karena kromosom merepresentasikan proporsi saham yang tidak boleh lebih dari 100% yang akan dilakukan oleh fungsi *repair*. Proses representasi kromosom dijabarkan pada Gambar 4.2 dan proses *repair* pada Gambar 4.3.

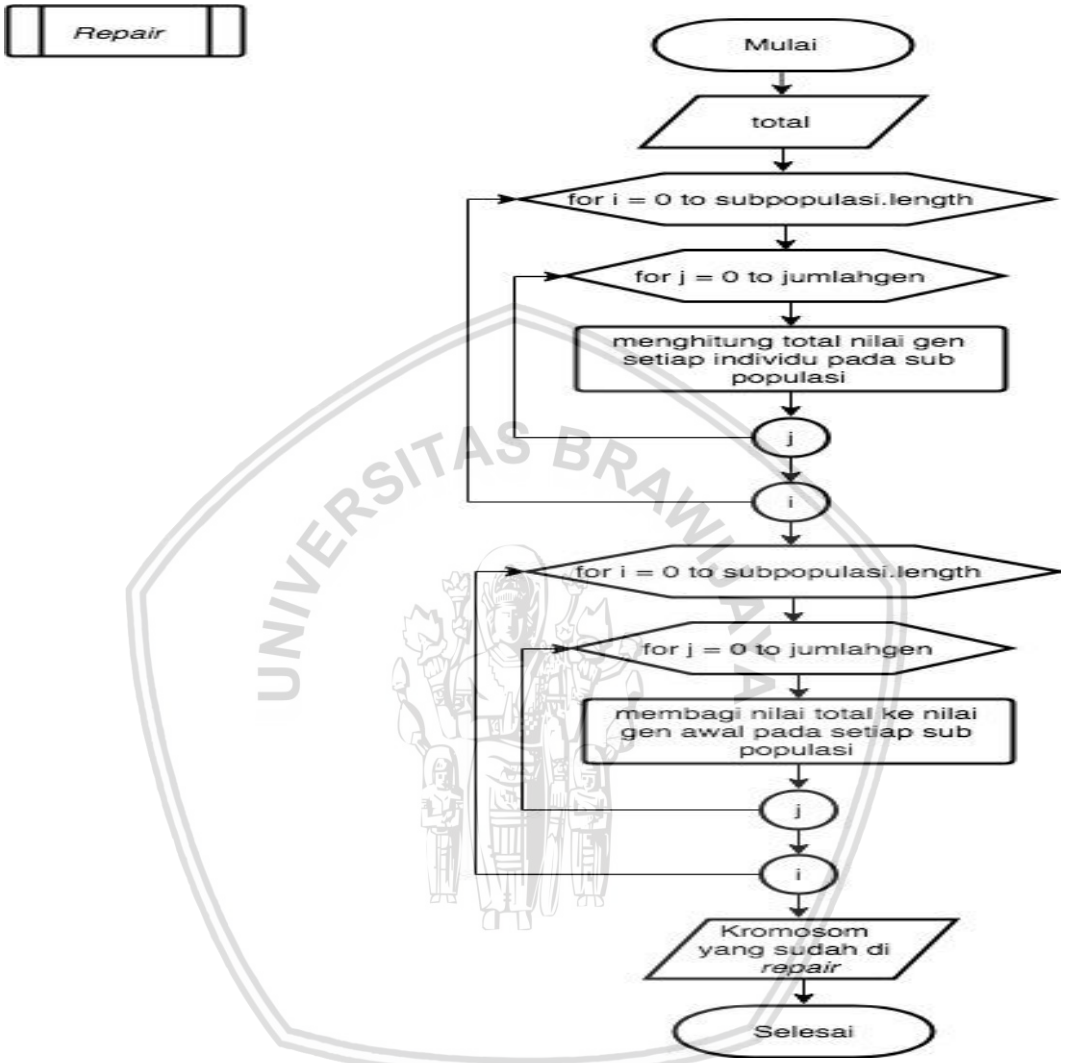


Gambar 4.2 Diagram Alir Representasi Kromosom

Dari Gambar 4.2 dapat diketahui bagaimana alur proses representasi kromosom pada penelitian ini. Diawali dengan inisialisasi jumlah subpopulasi,

lalu masuk ke dalam perulangan sebanyak *popsize* dan jumlah gen kedua perulangan ini berfungsi untuk membangkitkan nilai kromosom secara acak untuk seluruh *popsize* sejumlah dengan jumlah gen pada setiap sub populasi.

4.1.1.1 Repair



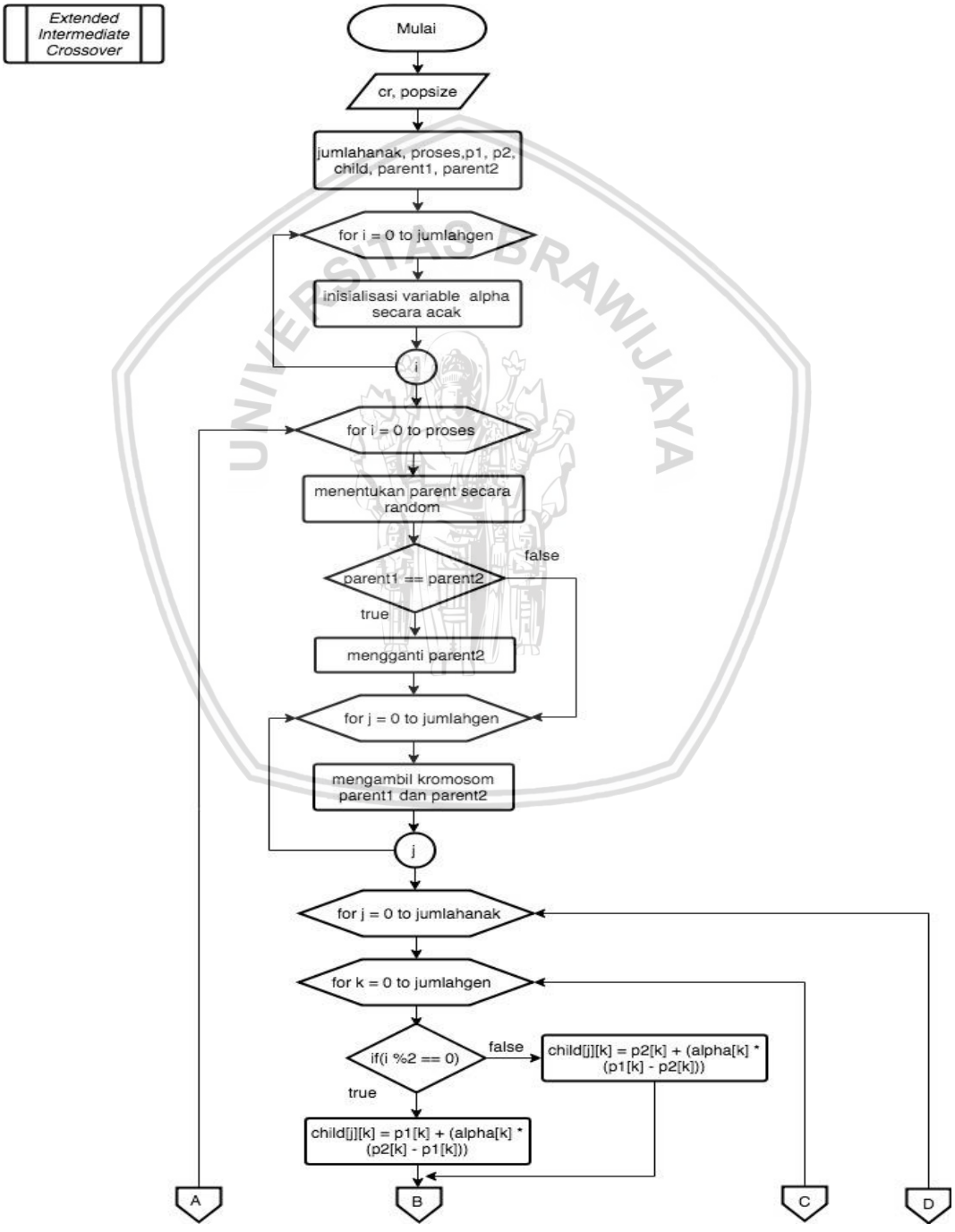
Gambar 4.3 Diagram Alir *Repair*

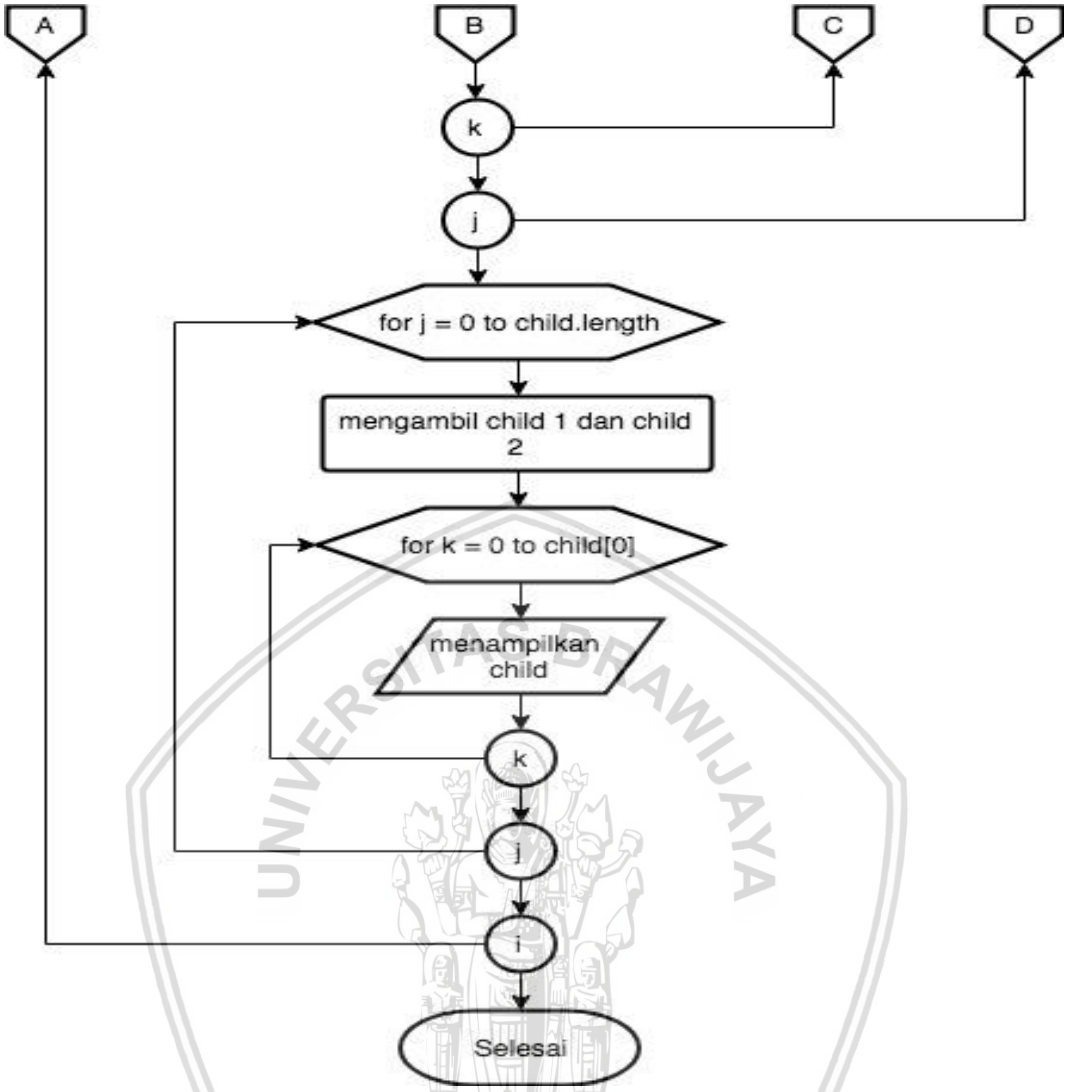
Pada Gambar 4.3 dapat diketahui bagaimana proses *repair* atau normalisasi dilakukan. Normalisasi ini diperlukan karena proporsi pada portofolio saham tidak boleh lebih dari 100%. Dalam penelitian ini berarti nilai gen pada tiap kromosom yang merupakan representasi dari proporsi saham tidak boleh lebih dari 1 atau 100%. Jadi, proses ini dilakukan untuk memperbaiki nilai gen yang jika dijumlahkan lebih dari 1 menjadi tidak lebih atau sama dengan 1 agar tetap memenuhi syarat proporsi saham pada portofolio saham. *Flowchart* mulai dari membuat variabel total untuk menyimpan hasil kromosom yang sudah di *repair*. Dua perulangan pertama berfungsi untuk menghitung total nilai gen pada setiap individu lalu pada dua perulangan berikutnya dilakukan proses *repair* dengan membagi total tersebut ke setiap nilai gen awal sehingga hasilnya akan memenuhi syarat yaitu tidak lebih dari 1.

4.1.2 Crossover

Crossover merupakan bentuk reproduksi dalam algoritme genetika yang digunakan untuk menghasilkan keturunan. Dalam penelitian ini terdapat dua jenis *crossover* yang digunakan karena penelitian ini menggunakan algoritme genetika terdistribusi yang menggunakan jenis *crossover* yang beberbeda pada sub populasi yang berbeda. Proses *crossover* dijabarkan pada Gambar 4.4 untuk *extended intermediate crossover* dan Gambar 4.5 untuk *one cut point crossover.*

4.1.2.1 Extended Intermediate Crossover



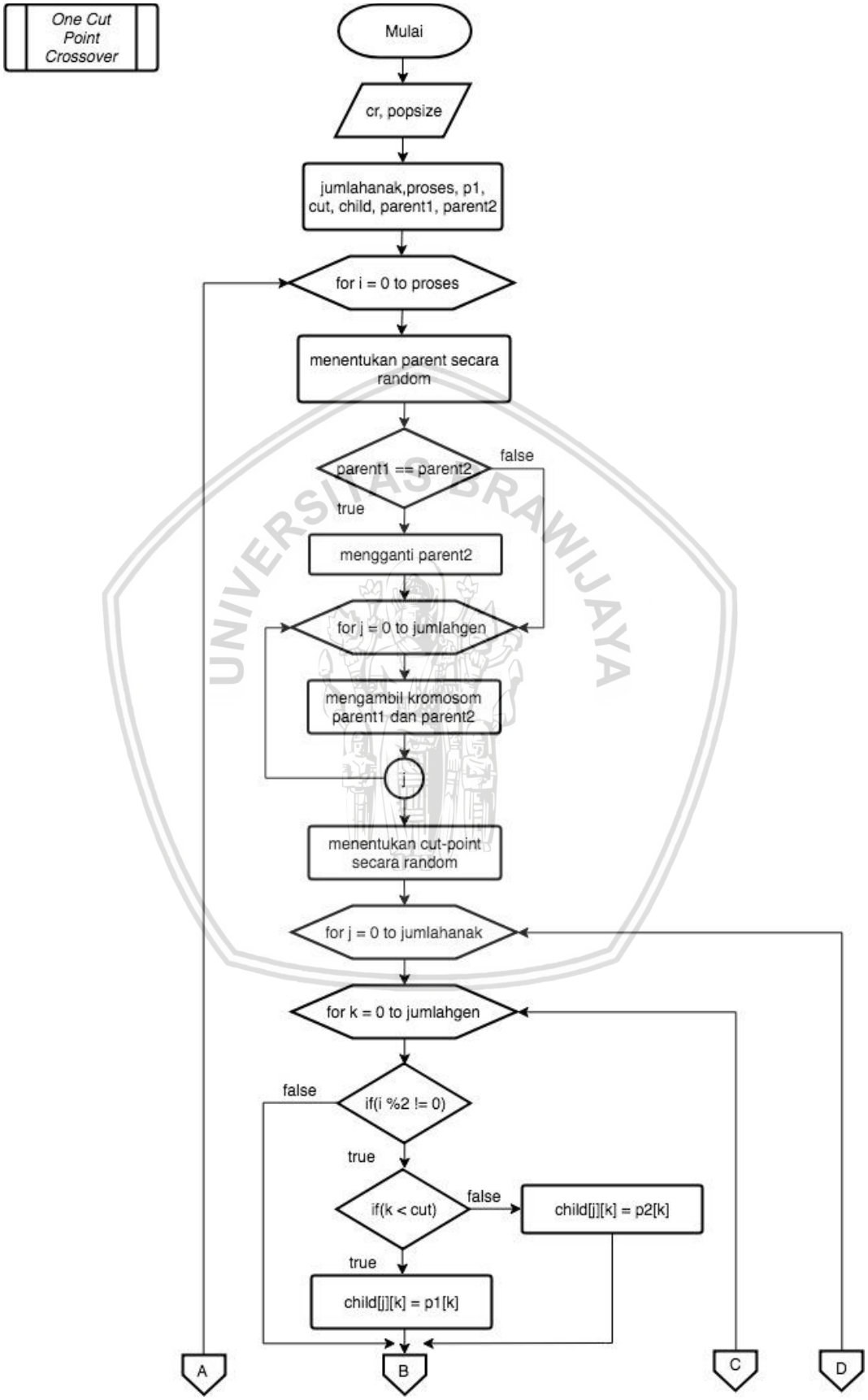


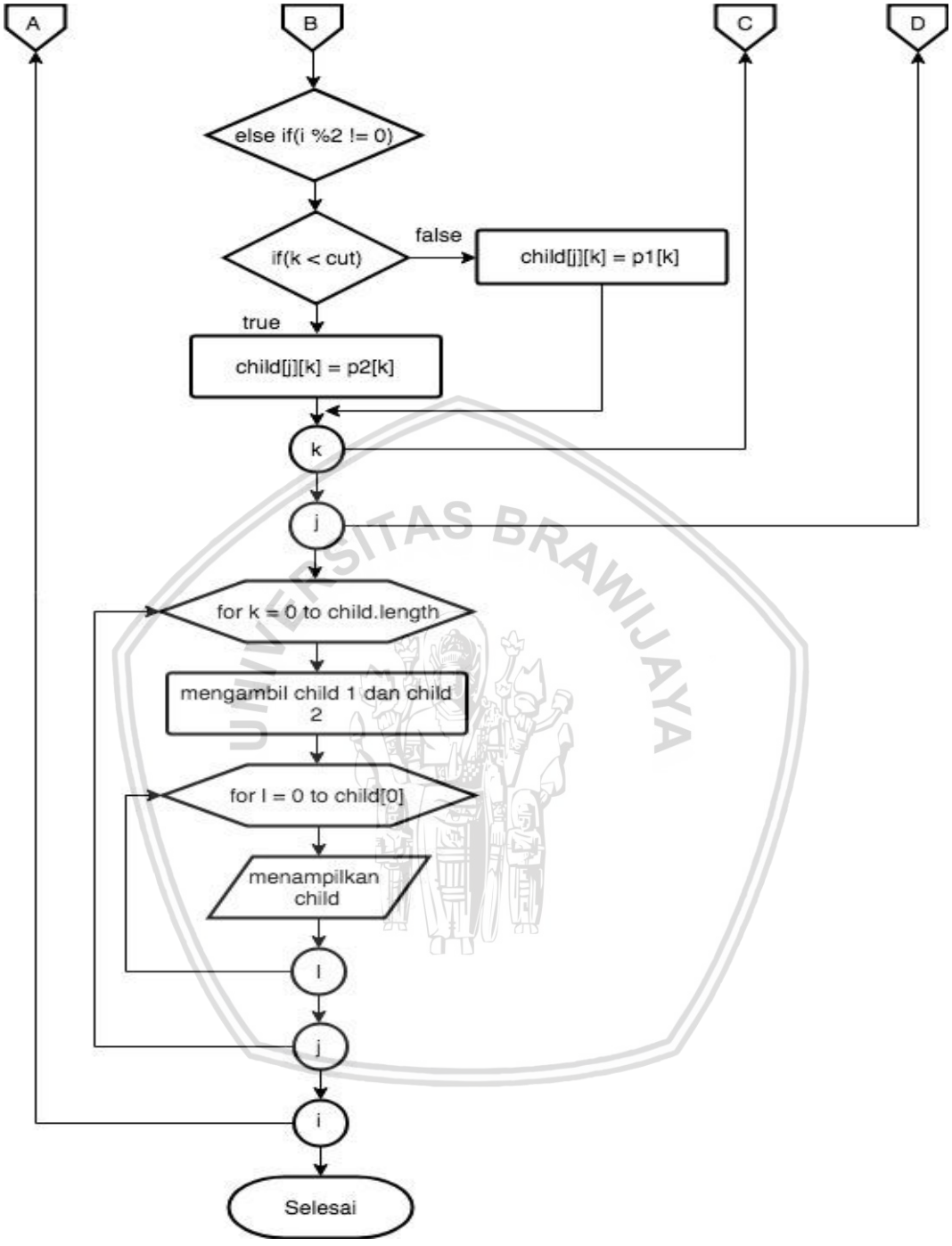
Gambar 4.4 Diagram Alir *Extended Intermediate Crossover*

Pada Gambar 4.4 ditunjukkan bagaimana alur proses *crossover* dengan metode *extended intermediate.* Untuk menentukan berapa banyak jumlah anak yang harus dihasilkan dibutuhkan nilai cr yang merupakan data masukan. Pada *extended intermediate crossover* dilakukan pemilihan 2 *parent* secara acak. Untuk mengetahui hasil *child* maka dilakukan perhitungan dengan rumus yang sudah ditentukan dengan menggunakan nilai *alpha* yang dibangkitkan secara acak pada interval 0 sampai kurang dari 1 untuk setiap *child* dari P1 dan P2 yang dibedakan dengan sebuah kondisi. Setelah proses selesai maka hasil *child* dari proses ini akan ditampilkan.

Proses *crossover* yang lain yang juga digunakan pada penelitian ini adalah *one cut point crossover*. Proses *crossover* ini dilakukan dengan memilih dua *parent* secara acak kemudian dilakukan penentuan titik potong secara acak juga. Hasil *child* pada *crossover* ini berasal dari hasil penukaran gen berdasarkan batas titik potong yang sudah ditentukan. Untuk proses *crossover* dengan *one cut point* dijelaskan pada Gambar 4.5.

4.1.2.2 One-Cut Point Crossover



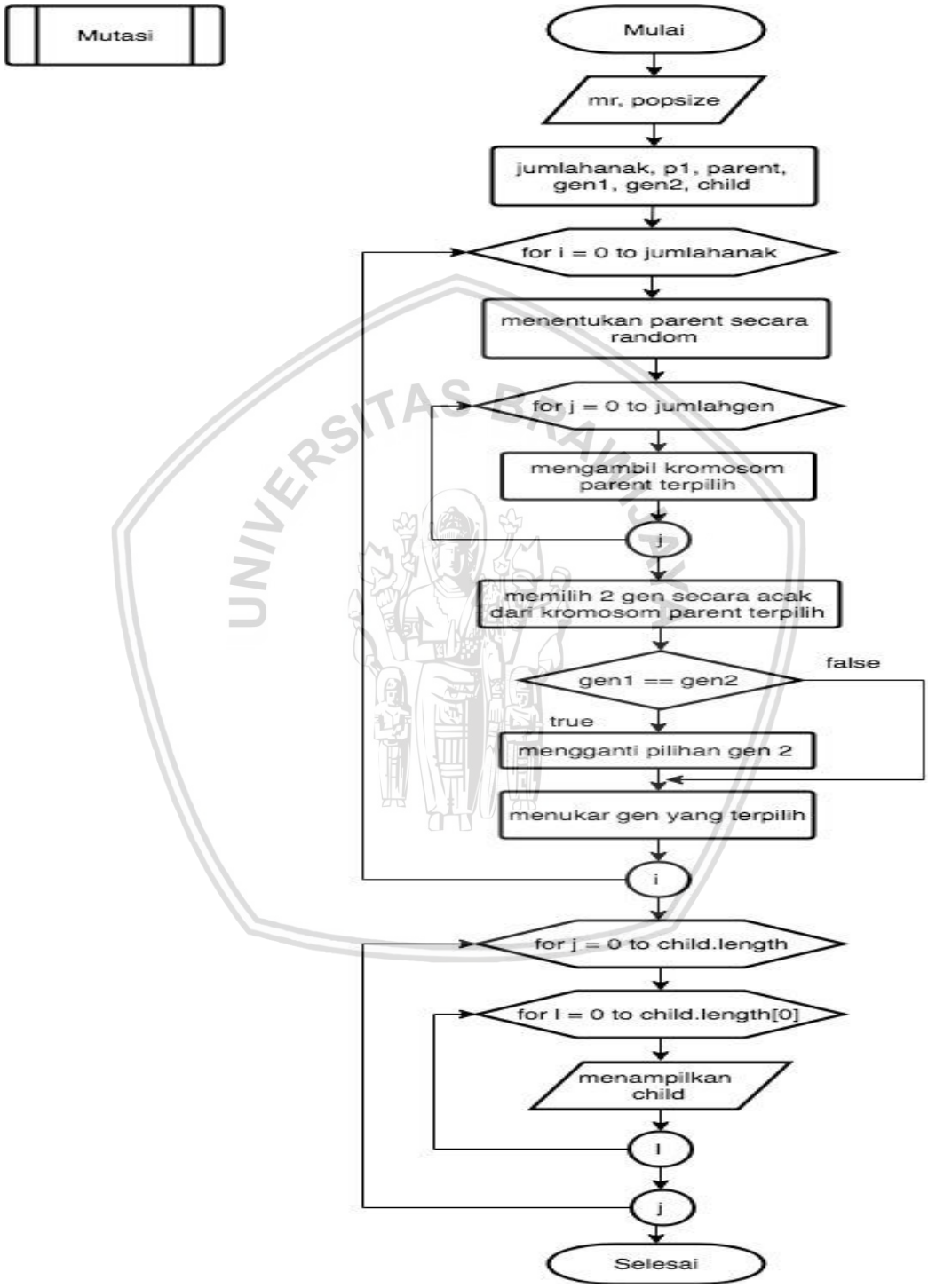


Gambar 4.5 Diagram Alir *One-Cut Point Crossover*

Pada Gambar 4.5 memiliki proses awal yang hampir sama dengan *extended intermediate crossover*, perbedaannya adalah ketika sudah mendapatkan *parent* secara random selanjutnya akan ada proses menentukan titik potong secara acak. Dari titik potong atau *cut point* ini dilakukan proses *one cut point crossover* dengan adanya kondisi untuk menentukan hasil tersebut masuk ke C1 atau C2. Setelah hasil didapatkan, selanjutnya hasil *child* akan ditampilkan.

4.1.3 Mutasi

Mutasi merupakan proses kedua dalam reproduksi, yaitu pembentukan *offspring* yang didapatkan dari parent dalam populasi. Proses mutasi pada penelitian ini dapat dilihat pada Gambar 4.6.



Gambar 4.6 Diagram Alir Mutasi

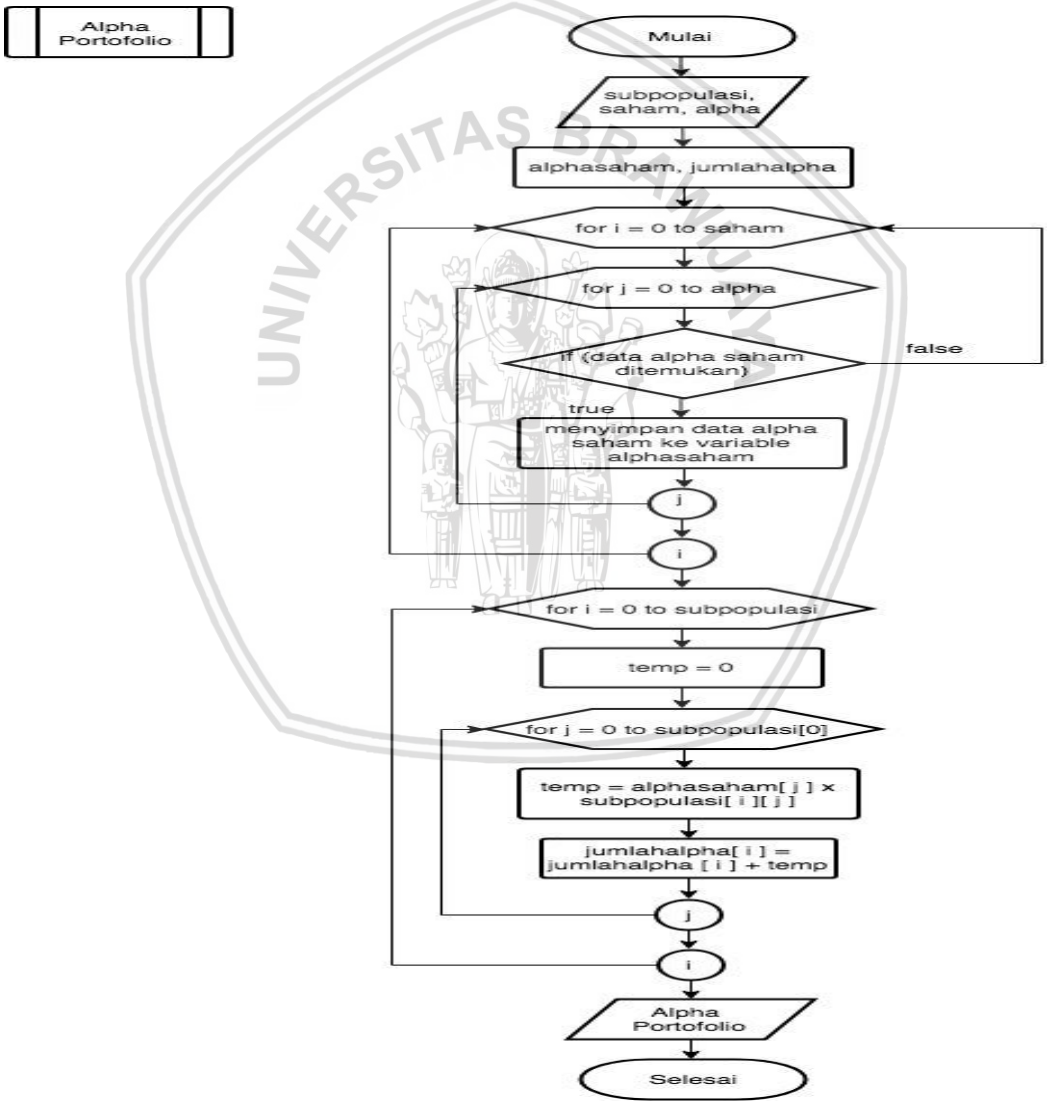
Pada Gambar 4.6 dapat dilihat bagaimana alur proses untuk mendapatkan *offspring* atau keturunan dari proses mutasi yang dilakukan pada

setiap sub populasi. Proses mutasi yang dilakukan yaitu *reciprocal exchange mutation*. Diawali dengan menghitung jumlah anak melalui mr dan popsize, kemudian mencari *parent* secara acak lalu menentukan *parent* yang akan dilakukan proses mutasi. Menentukan gen yang dipilih untuk ditukar, kemudian dilakukan proses penukaran sehingga menghasilkan *offspring*.

4.1.4 Menghitung Fitness

Proses untuk menghitung *fitness* memiliki beberapa tahap pada *single index model* yaitu menghitung *alpha* portofolio, beta portofolio, kesalahan residu portofolio, *return* ekspektasi, dan resiko portofolio. Alur proses menghitung *fitness* dijabarkan pada Gambar 4.7 sampai Gambar 4.12.

4.1.4.1 Alpha Portofolio

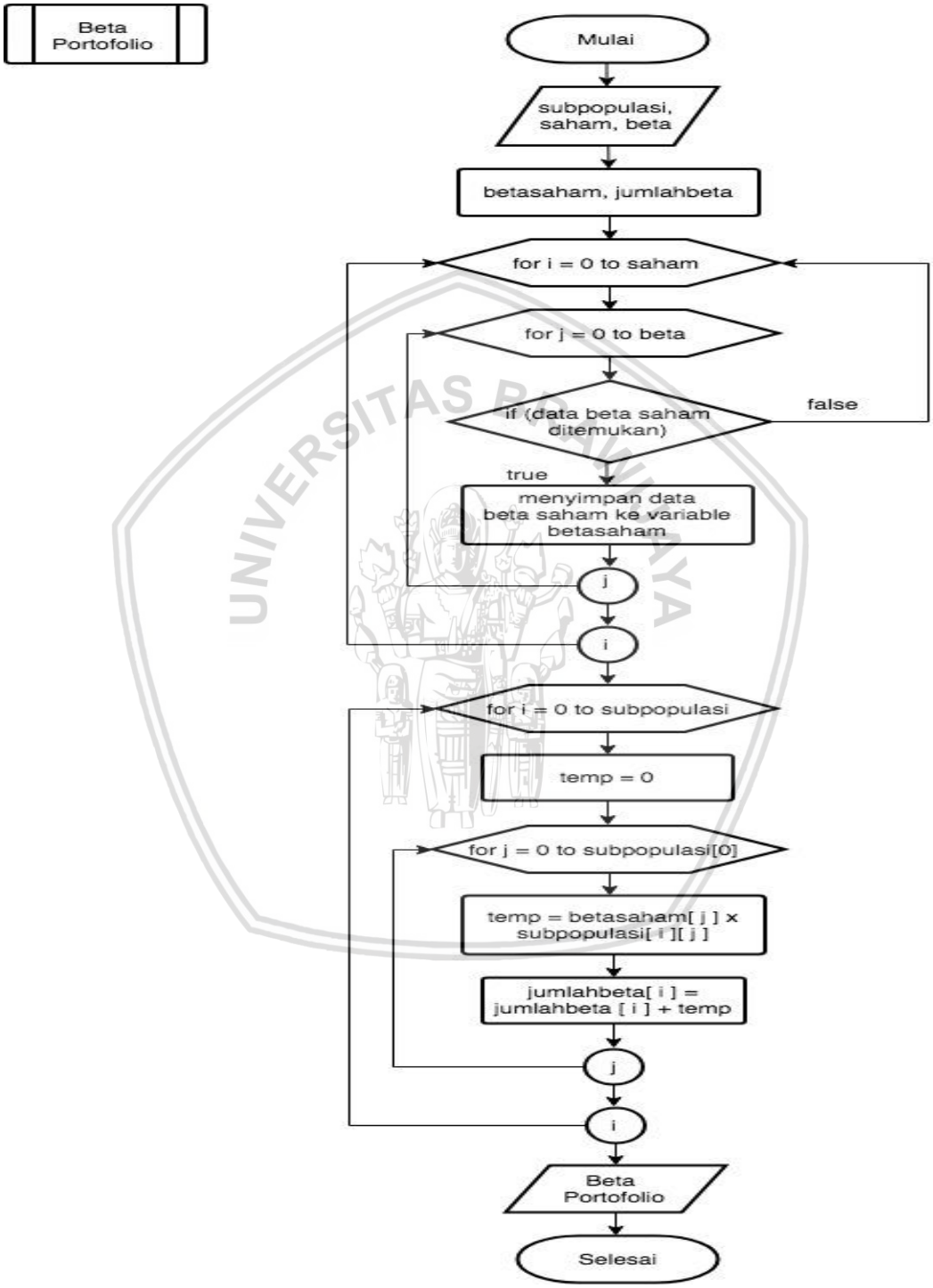


Gambar 4.7 Diagram Alir Alpha Portofolio

Dari Gambar 4.7 dijelaskan bahwa untuk menghitung *alpha* portofolio saham dibutuhkan data *alpha* pada setiap emiten saham yang sudah disimpan di

dalam *database*. Sehingga untuk menghitung dibutuhkan data *alpha* saham yang sesuai dengan saham apa saja yang dipilih kemudian dilakukan perhitungan *alpha* portofolio dengan rumus seperti pada persamaan (2.7).

4.1.4.2 Beta Portofolio

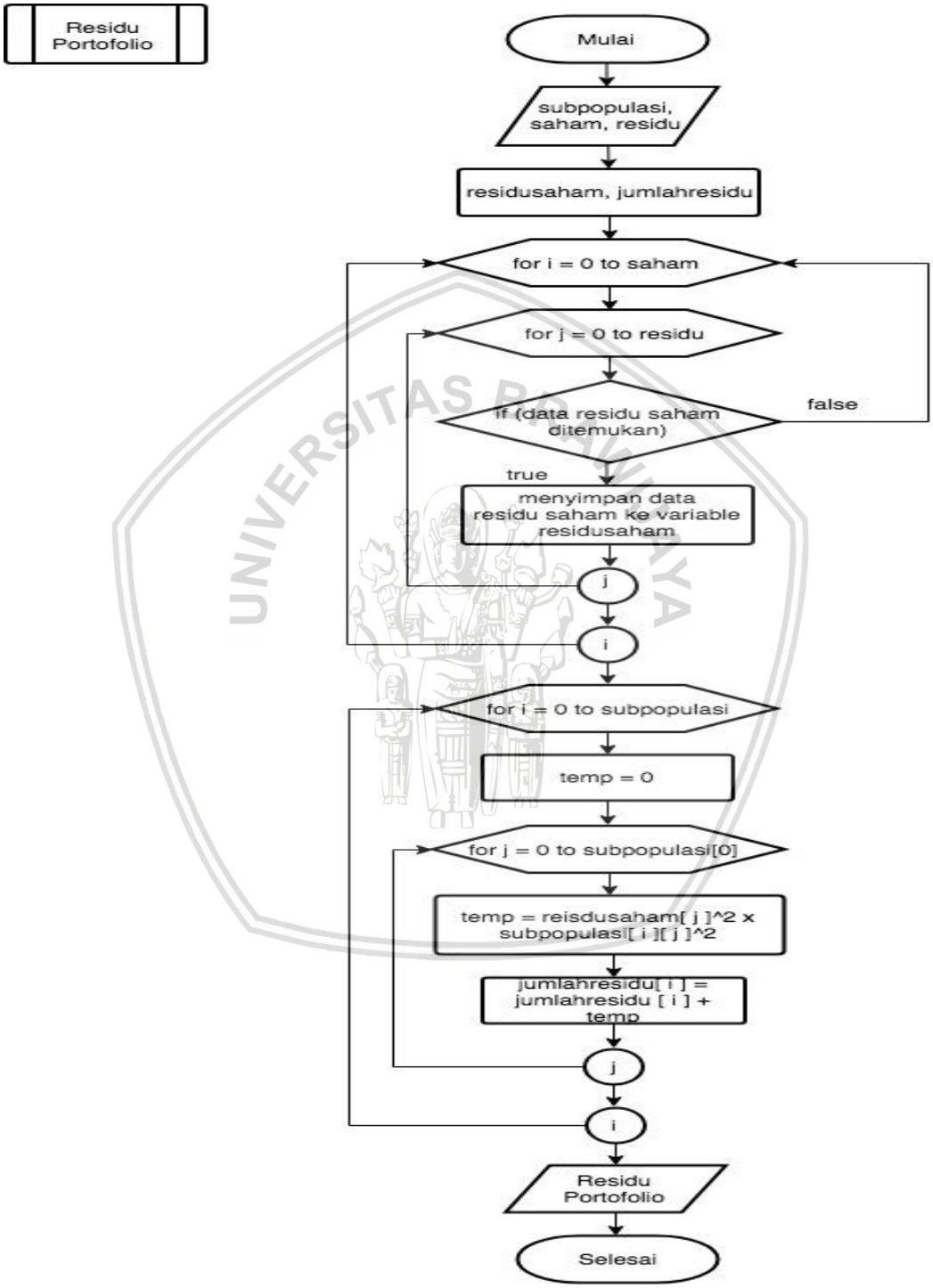


Gambar 4.8 Diagram Alir Beta Portofolio

Setelah menghitung *alpha* portofolio, selanjutnya adalah menghitung beta portofolio seperti yang ada pada Gambar 4.8 Prosesnya sama dengan

proses menghitung *alpha*, namun data yang diambil dari *database* adalah data beta pada setiap emiten saham untuk kemudian dihitung dengan persamaan (2.8).

4.1.4.3 Kesalahan Residu Portofolio

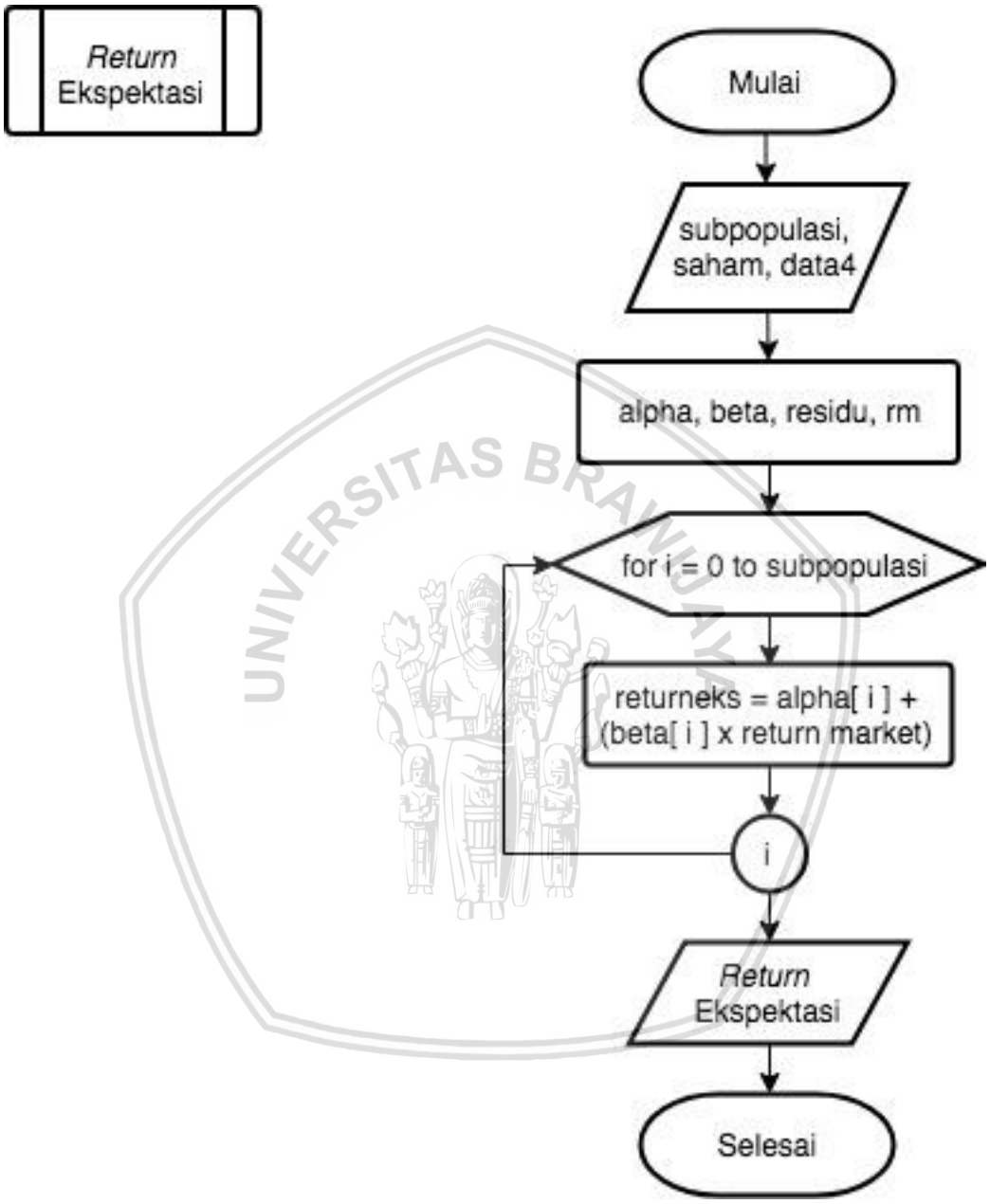


Gambar 4.9 Diagram Alir Kesalahan Residu Portofolio

Kesalahan residu portofolio dihitung dengan rumus yang terdapat pada persamaan (2.9), dengan cara mengambil data residu pada setiap emiten saham

yang dipilih lalu dilakukan perhitungan untuk mendapatkan hasil kesalahan residu portofolio.

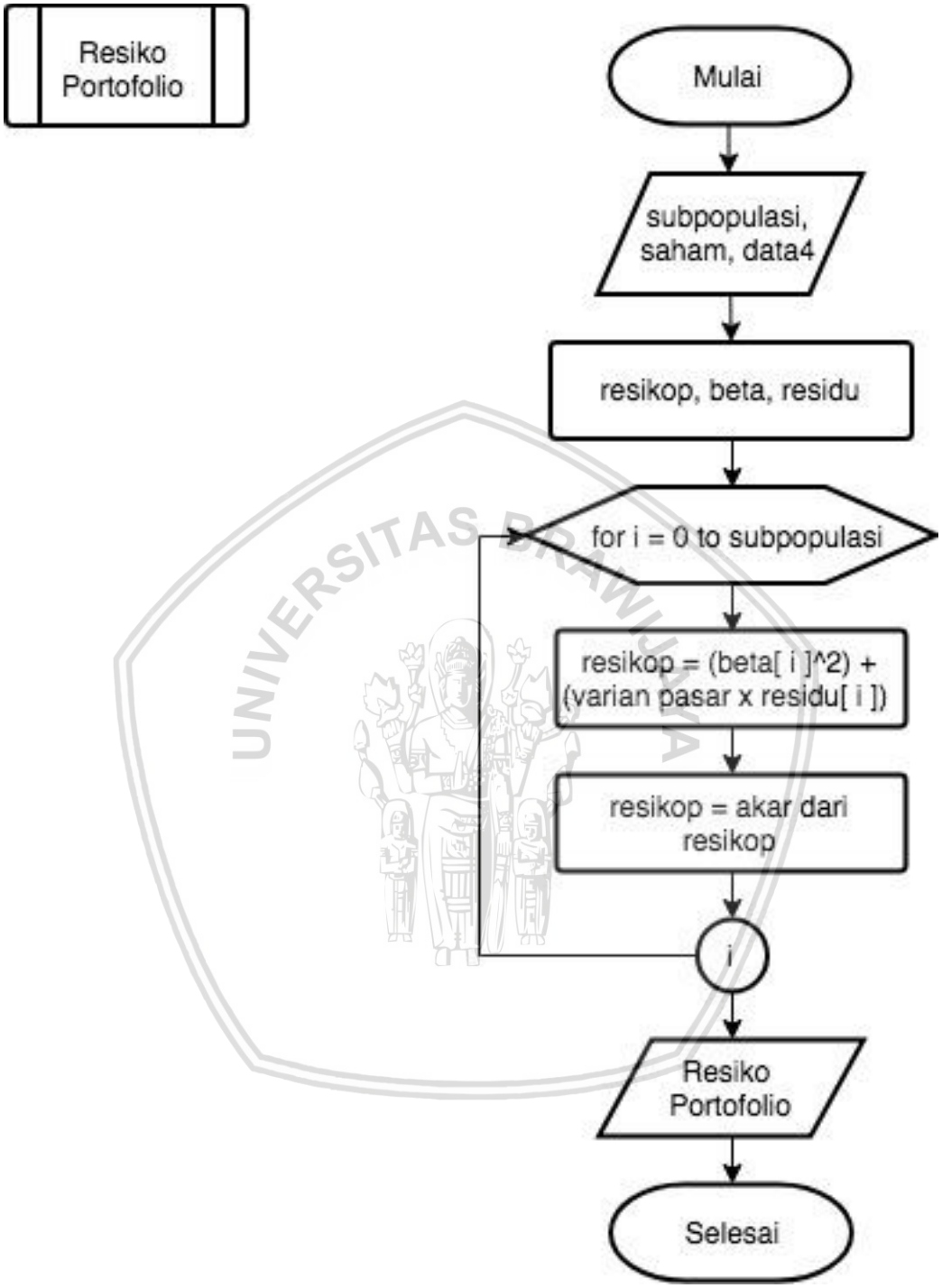
4.1.4.4 *Return* Ekspektasi



Gambar 4.10 Diagram Alir *Return* Ekspektasi

Setelah selesai menghitung *alpha*, beta, dan kesalahan residu portofolio selanjutnya adalah menghitung *return* ekspektasi. Seperti pada Gambar 4.10, akan dilakukan pemanggilan *method* alpha, dan beta serta variabel rm yang merupakan *return market* yang nilainya sudah diketahui kemudian dilakukan perhitungan seperti pada persamaan (2.10) untuk setiap individu pada sub populasi.

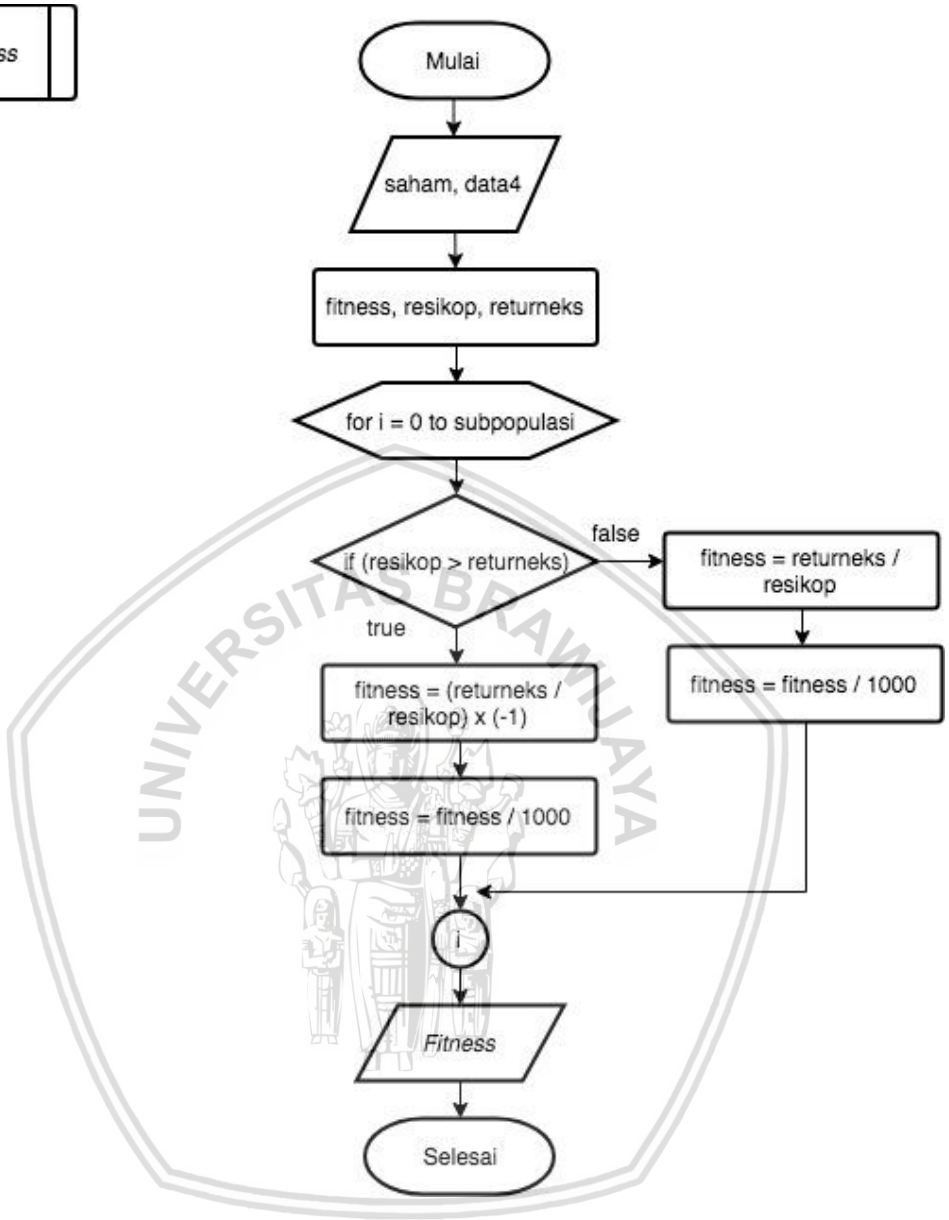
4.1.4.5 Resiko Portofolio



Gambar 4.11 Diagram Alir Resiko Portofolio

Tahapan berikutnya adalah menghitung resiko portofolio seperti proses yang terdapat pada Gambar 4.11 dengan memanggil *method* beta, dan residu serta rp sebagai varians resiko portofolio yang sudah diketahui nilainya. Kemudian dilakukan proses perhitungan seperti pada persamaan (2.11) untuk setiap individu pada setiap sub populasi.

4.1.4.6 Fitness

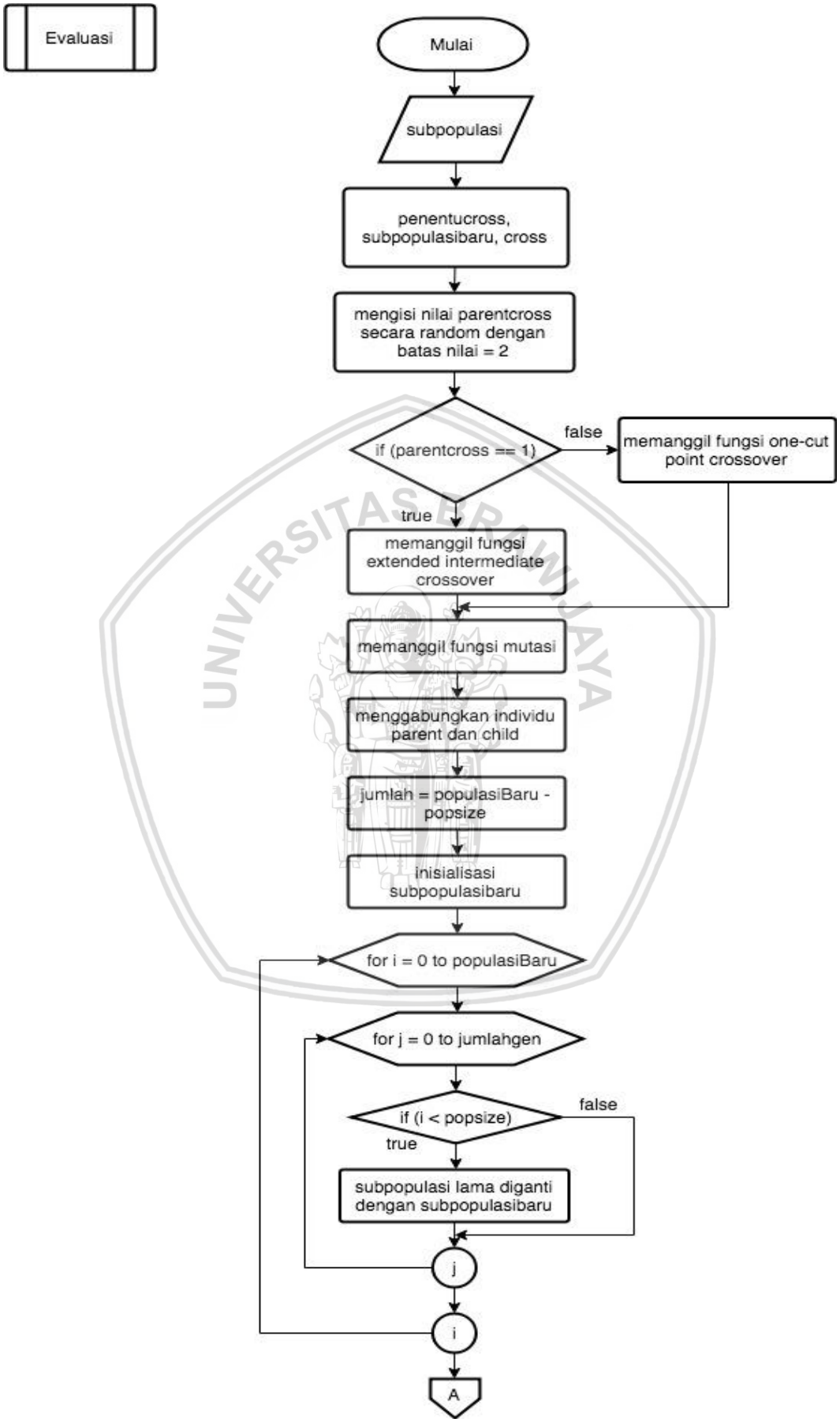


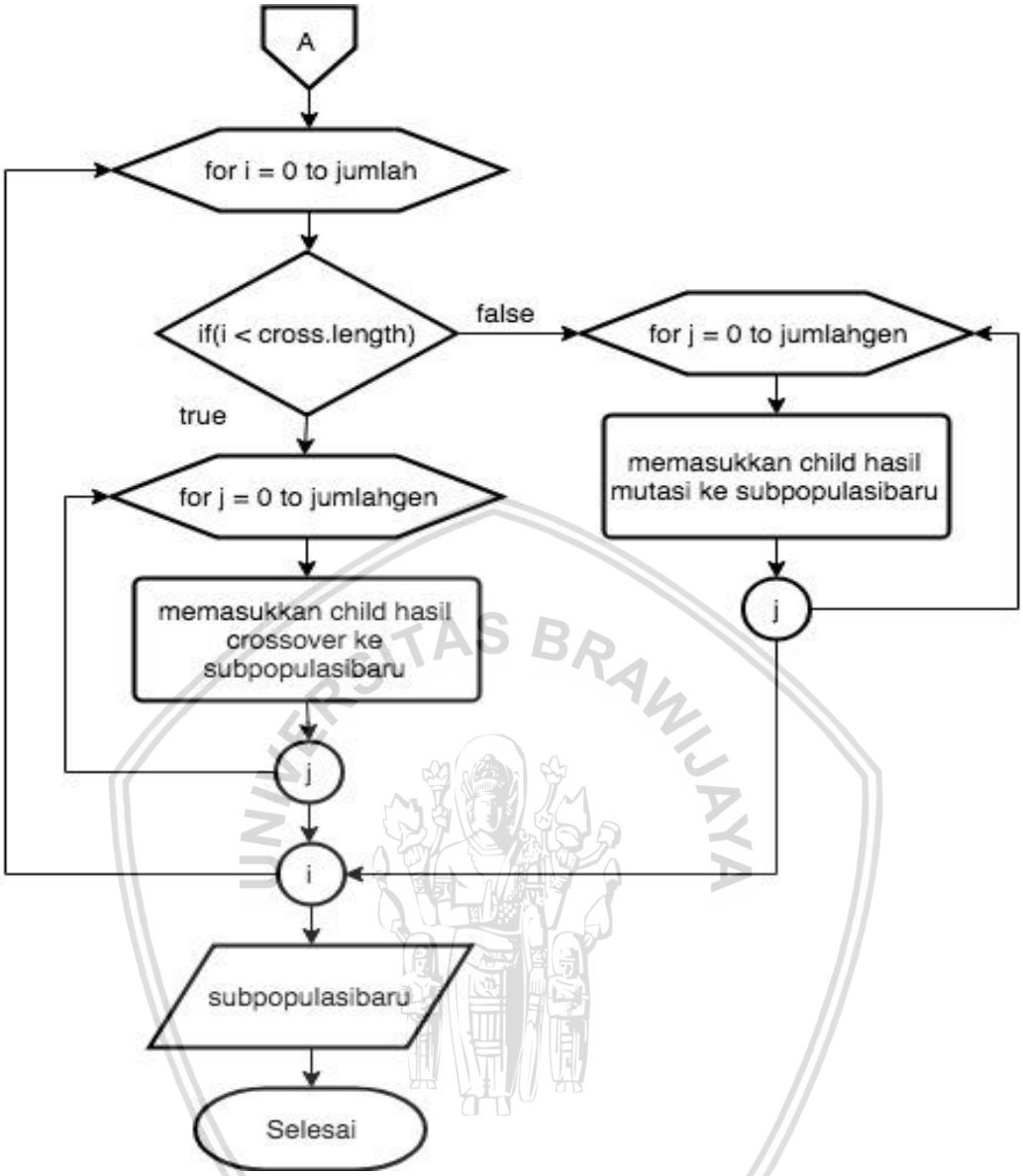
Gambar 4.12 Diagram Alir Fitness

Gambar 4.12 menenunjukkan bagaimana alur dalam menghitung *fitness* setelah seluruh proses pada *single index model* selesai, langkah selanjutnya adalah menghitung *fitness* dengan memanggil *method return* ekspektasi dan resiko portofolio, selanjutnya dilakukan perhitungan *fitness* dengan cara *return* ekspektasi dibagi dengan resiko portofolio. Perhitungan ini dilakukan secara berulang untuk seluruh individu pada setiap sub populasi.

4.1.5 Evaluasi

Proses selanjutnya yaitu evaluasi yang akan menggabungkan seluruh individu *parent* dan *child* atau *offspring* yang dihasilkan ke sub populasi masing-masing. Evaluasi akan dijabarkan pada Gambar 4.13.


Evaluasi
Mulai
subpopulasi
penentucross, subpopulasibaru, cross
mengisi nilai parentcross secara random dengan batas nilai = 2
if (parentcross == 1)
false
memanggil fungsi one-cut point crossover
true
memanggil fungsi extended intermediate crossover
memanggil fungsi mutasi
menggabungkan individu parent dan child
jumlah = populasiBaru - popsize
inisialisasi subpopulasibaru
for i = 0 to populasiBaru
for j = 0 to jumlahgen
if (i < popsize)
false
true
subpopulasi lama diganti dengan subpopulasibaru
j
i
A

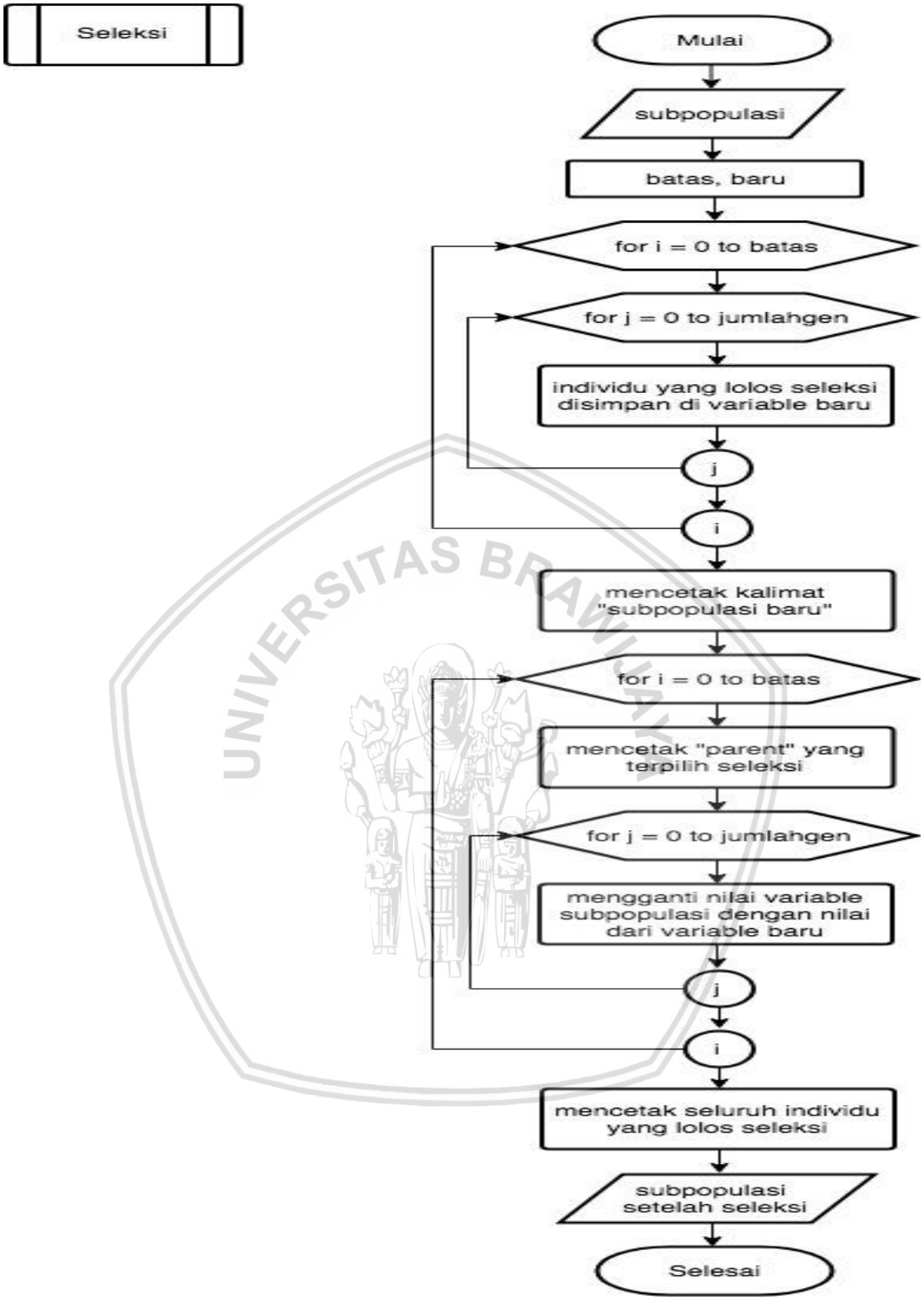


Gambar 4.13 Diagram Alir Evaluasi

Dari Gambar 4.13 dapat diketahui bagaimana alur dari evaluasi. Diawali dengan adanya proses untuk menentukan *crossover* apa yang akan digunakan kemudian akan memanggil salah satu fungsi *crossover* dan fungsi mutasi. Selanjutnya adalah memanggil seluruh subpopulasi yang ada menjadi subpopulasibaru, kemudian menggabungkan hasil dari *offspring* yang dihasilkan masing-masing subpopulasi dari *crossover* dan mutasi untuk disimpan menjadi satu subpopulasi baru.

4.1.6 Seleksi

Seleksi yang digunakan pada penelitian ini adalah *elistism selection* yang berarti hanya mencari individu dengan *fitness* terbesar sebanyak *popsize* awal. Digram alir seleksi dijabarkan pada Gambar 4.14.

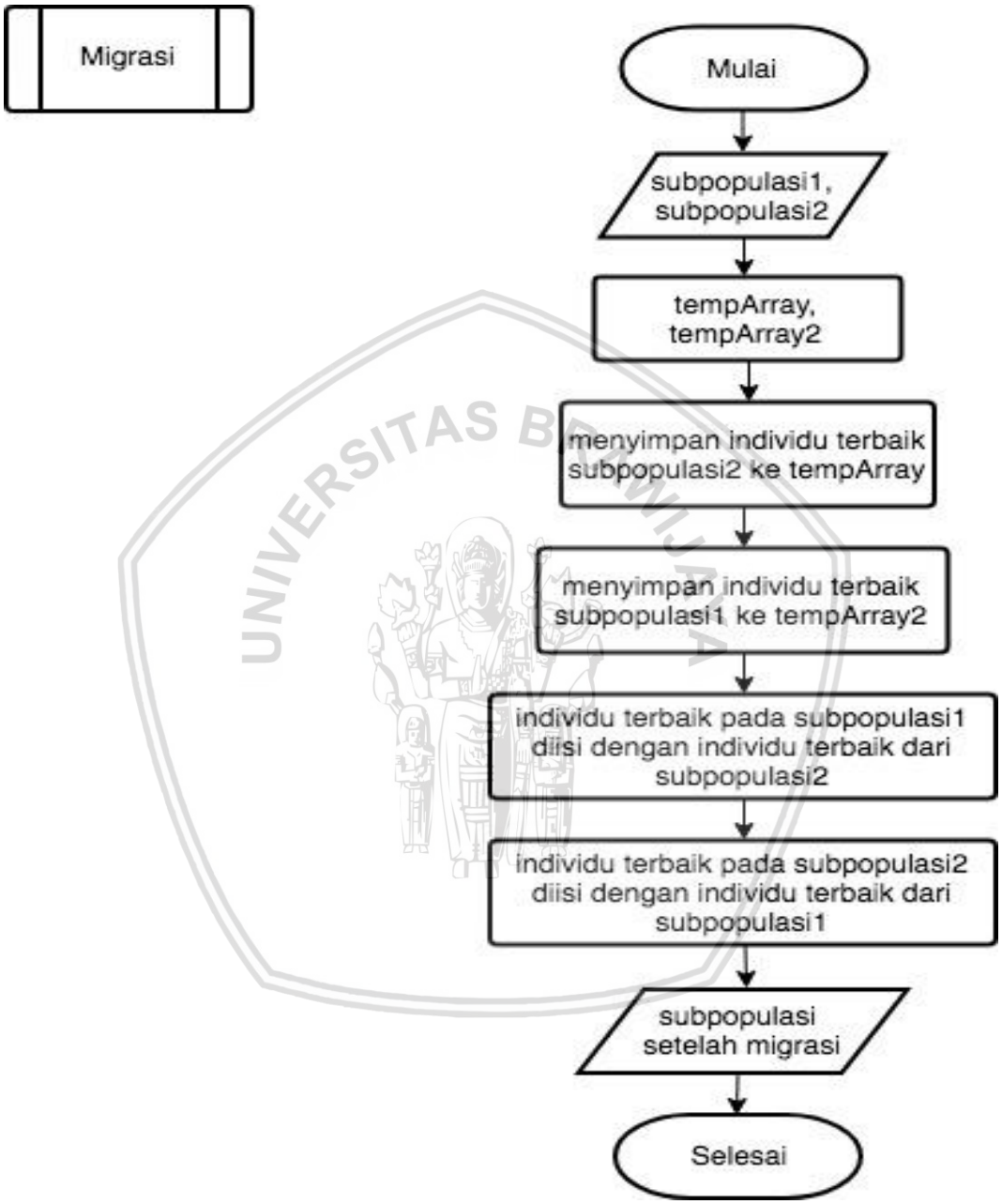


Gambar 4.14 Seleksi

Seleksi yang digunakan adalah *elitism* sehingga pada tahap awal dilakukan inisialisasi batas yang merupakan batas untuk mencari individu terbaik sejumlah *popsize* awal. Kemudian dilakukan pencarian individu terbaik pada setiap sub populasi, setelah ditemukan maka akan ditampilkan individu terbaik yang terpilih pada setiap sub populasi.

4.1.7 Migrasi

Algoritme genetika terdistribusi memiliki mekanisme migrasi yaitu menukar 1 individu terbaik ke sub populasi yang lainnya. Alur proses migrasi dijabarkan pada Gambar 4.15.



Gambar 4.15 Migrasi

Proses terakhir yaitu migrasi, dibuat variabel baru yaitu subpopulasibaru1 dan subpopulasibaru2 yang berfungsi untuk menyimpan sub populasi yang sudah diperbarui karena proses migrasi. Jika satu individu terbaik pada setiap sub populasi ditemukan, maka akan dilakukan proses migrasi kemudian hasil migrasi akan ditampilkan dan menjadi sub populasi yang akan digunakan pada iterasi berikutnya.

4.2 Perhitungan Manual

Perhitungan manual disini berupa contoh penerapan algoritme genetika terdistribusi untuk menentukan portofolio saham optimal. Pada perhitungan manual ini, akan digunakan 4 data saham dari periode Februari 2017 sampai Juli 2017. Data yang dibutuhkan ialah data *alpha,* beta, dan kesalahan residu pada setiap emiten saham yang dipilih yang akan disajikan pada Tabel 4.1

Kode Saham	AIpha	Beta	Kesalahan Residu
AALI	-0,000564032	0,014730914	1,25166E-05
ADHI	0,011458977	0,141946487	0,009045129
AKRA	0,010388746	0,134638114	0,007485772
BBCA	0,012898181	0,449368521	0,014764953

Tabel 4.1 Data Saham

Pada Tabel 4.1 berisi kode saham, *alpha*, beta, dan kesalhan residu saham yang akan menjadi data masukan pada perhitungan manual ini. Dari 4 saham yang dipilih, akan dilakukan proses penerapan algoritme genetika terdistribusi. Setiap sub populasi akan memiliki jumlah gen yang sama.

4.2.1 Membentuk Kromosom

Membentuk *popsize* dapat dilakukan secara random, untuk perhitungan manual ini dibuat 3 *popsize* dengan jumlah gen masing-masing kromosom adalah 4 yang merupakan data saham yang dipilih. Nilai kromosom dibangkitkan secara acak pada rentang 0 sampai 1. Sub populasi pada perhitungan manual ini ada 2, jadi setiap sub populasi memiliki *popsize* dan banyak gen yang sama. Contoh bentuk kromosom tertera pada Gambar 4.16 dan Gambar 4.17.

Sub populasi 1

	AALI	ADHI	AKRA	BBCA
P1	0,3	0,2	0,1	0,4
P2	0,1	0,6	0,1	0,2
P3	0,3	0,4	0,2	0,1

Gambar 4.16 Kromosom Sub Populasi 1

Sub populasi 2

	AALI	ADHI	AKRA	BBCA
P1	0,1	0,1	0,6	0,2
P2	0,5	0,2	0,1	0,2

P3	0,1	0,4	0,2	0,3

Gambar 4.17 Kromosom Sub Populasi 2

4.2.2 Reproduksi

Setelah inisialisasi kromosom, tahapan selanjutnya adalah reproduksi dengan *crossover* dan mutasi. *Crossover* yang digunakan adalah *extended intermediate crossover* untuk sub populasi 1 dan *one-cut point crossover* untuk sub populasi 2. Sedangkan untuk mutasi, setiap sub populasi menggunakan *reciprocal exchange mutation*.

4.2.2.1 Crossover

Pada *crossover* perlu diketahui berapa jumlah anak yang harus dihasilkan dengan rumus *cr* x *PopSize*. Dimana *cr* atau *crossover rate* menjadi data masukan. Setelah jumlah *offspring* yang harus dihasilkan sudah diketahui maka dilanjutkan dengan proses *crossover* pada masing-masing sub populasi. Untuk *crossover* pada sub populasi 1 akan dijelaskan pada Gambar 4.18.

Child = *cr* x popsize

= 0,6 x 3

= 1,8 dibulatkan menjadi 2

Sub populasi 1

α	0,71	0,14	0,25	0,87
P1	0,3	0,2	0,1	0,4
P3	0,3	0,4	0,2	0,1

Gambar 4.18 *Crossover* Sub Populasi 1

Pada proses *extended intermediate crossover,* pemilihan *parent* dilakukan secara acak .Nilai *alpha* atau α dibangkitkan secara acak pada rentang (0 sampai < 1). Untuk menghasilkan 2 *child* maka hanya perlu dilakukan satu kali proses *crossover* dengan rumus pada Persamaan (2.12).

Maka:

C_1 = 0,3 + 0,71 (0,3 – 0,3) = 0,3

= 0,2 + 0,14 (0,4 – 0,2) = 0,228

= .....

C_2 = 0,3 + 0,71 (0,3 – 0,3) = 0,3

= 0,4 + 0,14 (0,2 – 0,4) = 0,372

= ...

Proses *crossover* dilakukan pada setiap gen sehingga perhitungannya akan menghasilkan *child* seperti pada Gambar 4.19.

***Child* Sub Populasi 1**

C1	0,3	0,228	0,125	0,139
C2	0,3	0,372	0,175	0,361

Gambar 4.19 *Child* Sub Populasi 1

Selanjutnya akan dilakukan proses *crossover* dengan *one-cut point* untuk sub populasi 2. Pemilihan *parent* juga dilakukan secara acak seperti pada Gambar 4.20.

Sub Populasi 2

P2	0,5	0,2	0,1	0,2
P3	0,1	0,4	0,2	0,3

Gambar 4.20 *Parent* terpilih untuk *crossover*

Setelah *parent* ditentukan, langkah selanjutnya adalah menentukan titik potong secara acak untuk kemudian dilakukan *crossover* agar menghasilkan 2 *child* seperti pada Gambar 4.21.

P2	0,5	0,2	0,1	0,2
P3	0,1	0,4	0,2	0,3

Gambar 4.21 *Cut Point*

Cut-point berada di antara gen ketiga dan gen keempat sehingga akan dilakukan proses menukar gen pada gen keempat saja, hasil dari *one-cut point crossover* tertera pada Gambar 4.2

***Child* Sub Populasi 2**

C1	0,5	0,2	0,1	0,3
C2	0,1	0,4	0,6	0,2

Gambar 4.22 *Child* Sub Populasi 2

4.2.2.2 Mutasi

Reproduksi yang kedua adalah mutasi, setiap sub populasi memiliki proses mutasi yang sama yaitu *reciprocal exchange mutation*. Sebelum menerapkan proses mutasi akan dilakukan perhitungan *child* dengan menggunakan *mr* atau *mutation rate* dikali dengan *popsize*, nilai *mr* merupakan data masukan sehingga penerapan mutasi adalah sebagai berikut:

Child = *mr* x popsize

= 0,4 x 3

= 1,2 dibulatkan menjadi 1

Maka pada proses mutasi akan dihasilkan 1 keturunan atau *child* pada setiap sub populasi seperti pada Gambar 4.23.

Sub Populasi 1

P2	0,1	0,6	0,1	0,2

Gambar 4.23 Mutasi

Pada mutasi pemilihan *parent* juga dilakukan secara acak, setelah itu memilih gen mana yang akan ditukar. Pada sub populasi 1 gen yang dipilih dengan diarsir pada Gambar 4.23 adalah gen kedua 0,6 dan gen ketiga 0,1 kemudian ditukar, sehingga akan menghasilkan *child* seperti pada Gambar 4.24.

Child Sub Populasi 1

C3	0,1	0,1	0,6	0,2

Gambar 4.24 *Child* mutasi

Proses mutasi ini dilakukan juga pada sub populasi 2 dengan cara yang sama yaitu menentukan *parent* secara acak lalu menentukan gen yang akan ditukar secara acak kemudian dilakukan proses penukaran.

4.2.3 Evaluasi

Proses evaluasi berarti menggabungkan seluruh individu yaitu *parent* dan *child* ke dalam sub populasinya masing-masing, sehingga akan terbentuk sub populasi yang dijabarkan pada Gambar 4.25 dan Gambar 4.26.

Sub Populasi 1

Individu	**Kromosom**			
P1	0,3	0,2	0,1	0,4
P2	0,1	0,6	0,1	0,2
P3	0,3	0,4	0,2	0,1
C1	0,3	0,228	0,125	0,226
C2	0,3	0,372	0,175	0,374
C3	0,1	0,1	0,6	0,2

Gambar 4.25 Evaluasi Sub Populasi 1

Sub Populasi 2

Individu	Kromosom			
P1	0,1	0,1	0,6	0,2
P2	0,5	0,2	0,1	0,2
P3	0,1	0,4	0,2	0,3
C1	0,5	0,2	0,1	0,2
C2	0,1	0,4	0,6	0,2
C3	0,6	0,1	0,1	0,2

Gambar 4.26 Evaluasi Sub Populasi 2

4.2.4 Menghitung *Fitness*

Dalam menghitung *fitness* perlu dilakukan perhitungan dengan *single index model* untuk mendapatkan nilai *return* ekspektasi dan resiko portofolio yang akan digunakan dalam menghitung *fitness*. Terdapat beberapa tahapan sebelum menghitung *fitness,* tahapan pertama adalah menghitung *alpha* portofolio, lalu *beta* portofolio, residu portofolio, *return* ekspektasi dan resiko portofolio baru kemudian menghitung *fitness*.

4.2.4.1 *Alpha* Portofolio

Menghitung *alpha* portofolio dilakukan dengan cara mengambil data *alpha* tiap emiten saham kemudian dikali dengan bobot saham, dalam hal ini dikalikan dengan setiap kromosom dari setiap individu pada masing-masing sub populasi. Perhitungan *alpha* portofolio dijelaskan pada Gambar 4.27.

Sub populasi 1 (P1)

Saham	αi	Wi	αi*Wi
AALI	-0,000564032	0,3	-0,00016921
ADHI	0,011458977	0,2	0,002291795
AKRA	0,010388746	0,1	0,001038875
BBCA	0,012898181	0,4	0,005159272
αp			0,008320733

Gambar 4.27 *Alpha* Portofolio

Dimana αi adalah alpha masing-masing saham dan Wi adalah kromosom, sedangkan αp adalah hasil total dari Alpha Portofolio. Perhitungan ini dilakukan untuk setiap individu *parent* dan *child*.

4.2.4.2 Beta Portofolio

Menghitung beta portofolio dilakukan dengan cara mengambil data beta tiap emiten saham kemudian dikali dengan bobot saham, dalam hal ini dikalikan dengan setiap kromosom dari setiap individu pada masing-masing sub populasi. Perhitungan beta portofolio dijelaskan pada Gambar 4.28.

Sub populasi 1 (P1)

Saham	βi	Wi	βiWi
AALI	0,014730914	0,3	0,004419274
ADHI	0,141946487	0,2	0,028389297
AKRA	0,134638114	0,1	0,013463811
BBCA	0,449368521	0,4	0,179747408
βp			0,226019791

Gambar 4.28 Beta Portofolio

Dimana βi adalah alpha masing-masing saham dan Wi adalah kromosom, sedangkan βp adalah hasil total dari beta portofolio. Perhitungan ini dilakukan untuk setiap individu *parent* dan *child*.

4.2.4.3 Kesalahan Residu Portofolio

Menghitung residu portofolio dilakukan dengan cara mengambil data residu tiap emiten saham kemudian dikali dengan bobot saham, dalam hal ini dikalikan dengan setiap kromosom dari setiap individu pada masing-masing sub populasi. Perhitungan residu portofolio dijelaskan pada Gambar 4.29.

Sub populasi 1 (P1)

Saham	σ_{ei}	Wi	σ_{ei}^2 * Wi^2
AALI	1,25166E-05	0,3	1,41E-11
ADHI	0,009045129	0,2	3,27257E-06
AKRA	0,007485772	0,1	5,60368E-07
BBCA	0,014764953	0,4	3,48806E-05
σ_{ep}^2			3,87136E-05

Gambar 4.29 Kesalahan Residu Portofolio

Dimana $\boldsymbol{\sigma_{ei}}$ adalah residu masing-masing saham dan Wi adalah kromosom, sedangkan σ_{ep}^2 adalah hasil total dari residu portofolio. Perhitungan ini dilakukan untuk setiap individu *parent* dan *child*.

4.2.4.4 *Return* Ekspektasi

Tahapan selanjutnya adalah menghitung *return* ekspektasi pada setiap individu *parent* dan *child* dengan rumus yang tertera pada Persamaan (2.9).

Sub populasi 1 (P1)

(E)Rp = 0,008320733 + (0,226019791 x 0,006500723)
= 0,009790025

= 0,009790025 X 100%

= 0,979002483%

Perhitungan *return* ekspektasi dilakukan ke seluruh individu dalam sub populasi.

4.2.4.5 Resiko Portofolio

Tahapan terakhir dalam *single index model* adalah menghitung resiko portofolio pada setiap individu *parent* dan *child* dengan rumus yang tertera pada Persamaan (2.10).

Sub populasi 1 (P1)

σ^2_p = 0,226019791^2 x 0,002451749 + 3,87136E-05

= √0,000163961

= 0,012804727 x 100%

= 1,280472715%

Perhitungan *return* ekspektasi dilakukan ke seluruh individu dalam sub populasi.

4.2.4.6 Fitness

Setelah nilai *return* ekspektasi dan resiko portofolio pada setiap individu sudah diketahui, selanjutnya adalah menghitung *fitness* untuk setiap individu *parent* dan *child* di setiap sub populasi menggunakan rumus sesuai dengan kondisi nilai *return* dan resiko seperti pada Persamaan (2.13) dan Persamaan (2.14).

Sub populasi 1 (P1)

P1 = $(\frac{0{,}009790025}{0{,}012804727})/1000)) \; x \; (-1)$

= -0,000764563

4.2.4.7 Evaluasi

Setelah menghitung nilai *fitness*, dilakukan proses evaluasi yaitu menggabungkan *parent* dan *child* pada setiap sub populasi. Hasil evaluasi dapat dilihat pada Gambar 4.30 dan Gambar 4.31.

Sub Populasi 1

Individu	Kromosom				Fitness
P1	0,3	0,2	0,1	0,4	-0,000764563
P2	0,1	0,4	0,1	0,4	0,001034863
P3	0,3	0,4	0,2	0,1	0,001107842
C1	0,3	0,228	0,125	0,139	-0,000966189

C2	0,3	0,372	0,175	0,361	-0,000890149
C3	0,1	0,1	0,4	0,4	-0,000856712

Gambar 4.30 *Fitness* Sub Populasi 1

Sub Populasi 2

Individu	Kromosom				Fitness
P1	0,1	0,1	0,6	0,2	0,001037125
P2	0,5	0,2	0,1	0,2	-0,000843437
P3	0,1	0,4	0,6	0,4	0,001010062
C1	0,5	0,2	0,1	0,4	-0,000749772
C2	0,1	0,4	0,6	0,2	0,001135755
C3	0,6	0,1	0,1	0,2	-0,000747721

Gambar 4.31 *Fitness* Sub Populasi 2

4.2.5 Seleksi

Setelah perhitungan *fitness* tahap selanjutnya adalah melakukan proses seleksi dengan *elitism selection* dengan nilai *fitness* terbesar, hasil dari seleksi ini adalah 3 individu sesuai dengan jumlah *popsize* awal yang memiliki *fitness* terbesar seperti pada Gambar 4.32 dan Gambar 4.33.

Sub Populasi 1

Individu	Kromosom				Fitness
P3	0,3	0,4	0,2	0,1	0,001107842
P2	0,1	0,4	0,1	0,4	0,001034863
P1	0,3	0,2	0,1	0,4	-0,000764563

Gambar 4.32 Seleksi Sub Populasi 1

Sub Populasi 2

Individu	Kromosom				Fitness
C2	0,1	0,4	0,6	0,2	0,001135755
P1	0,1	0,1	0,6	0,2	0,001037125
P3	0,1	0,4	0,6	0,4	0,001010062

Gambar 4.33 Seleksi Sub Populasi 2

4.2.6 Migrasi

Tahap akhir dari proses algortime genetika terdistribusi adalah migrasi, migrasi berarti menukar satu individu terbaik ke sub populasi yang lain. Sehingga dari hasil seleksi tersebut, akan dilakukan proses migrasi. Hasil dari proses migrasi dijabarkan pada Gambar 4.34 dan Gambar 4.35.

Sub Populasi 1

Individu	Kromosom				Fitness
C2	0,1	0,4	0,6	0,2	0,001135755
P2	0,1	0,4	0,1	0,4	0,001034863
P1	0,3	0,2	0,1	0,4	-0,000764563

Gambar 4.34 Migrasi Sub Populasi 1

Sub Populasi 2

Individu	Kromosom				Fitness
P3	0,3	0,4	0,2	0,1	0,001107842
P1	0,1	0,1	0,6	0,2	0,001037125
P3	0,1	0,4	0,6	0,4	0,001010062

Gambar 4.35 Migrasi Sub Populasi 2

Dari hasil migrasi dapat dilihat bahwa nilai P1 sebagai individu terbaik di sub populasi 1 ditukar menjadi bagian dari sub populasi 2, sedangkan C1 sebagai individu terbaik pada sub populasi 2 menjadi bagian dari sub populasi 1.

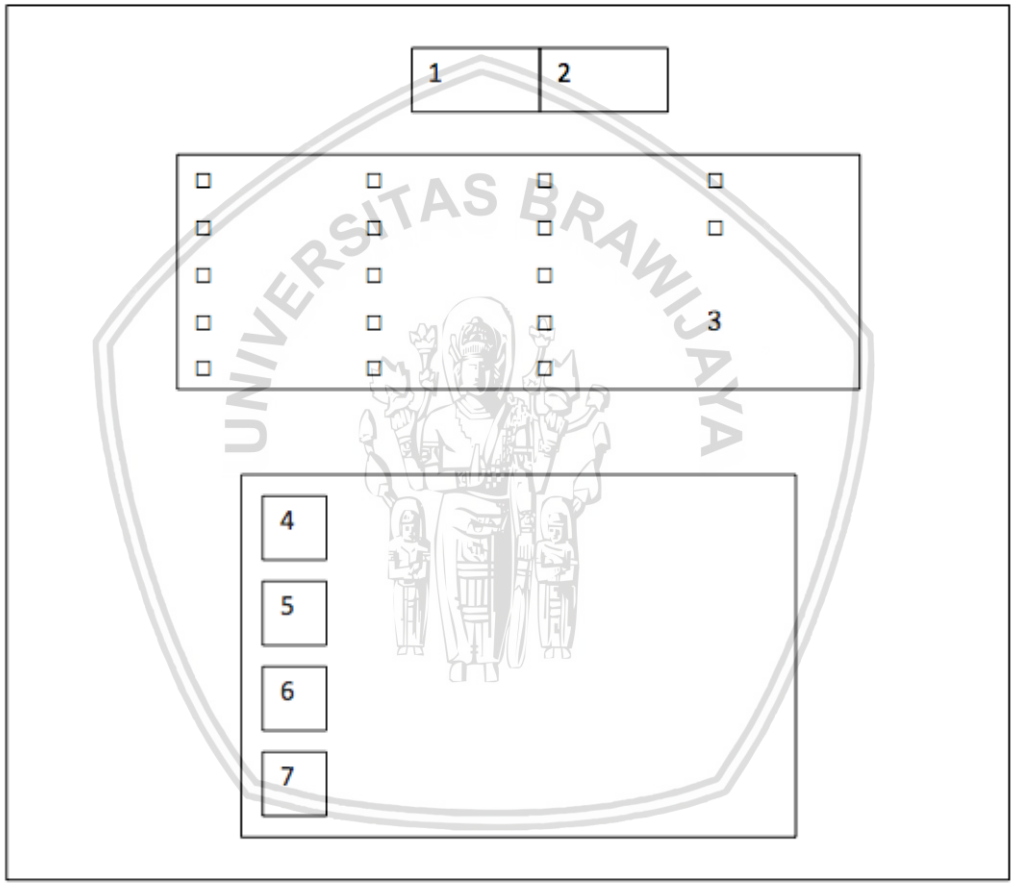
Terbentuk dua portofolio saham dari sub populasi 1 dan sub populasi 2, hasil terbaik dilihat dari nilai *fitness* terbesar pada setiap sub populasi. Misalnya pada sub populasi 1 terdapat pada individu C2 dengan proporsi untuk masing-masing saham 0,1 untuk AALI, 0,4 untuk ADHI, 0,6 untuk AKRA dan 0,2 untuk BBCA dengan nilai *fitness* sebesar 0,001135755 dengan nilai *return* ekspektasi sebesar 0,014828141% dan resiko portofolio sebesar 0,01305576%. Hasil dari migrasi akan digunakan sebagai kromosom *parent* pada iterasi berikutnya

4.3 Perancangan Antarmuka

Antarmuka pada sistem ini terdiri dari dua bagian yaitu halaman untuk *input* dan juga halaman yang menampilkan hasil. Perancangan antarmuka berfungsi sebagai gambaran bagaimana antarmuka atau tampilan sistem ini.

4.3.1 Perancangan Antaramuka Halaman *Input*

Perancangan antarmuka untuk halaman *input* dapat membuat pengguna untuk memilih saham apa saja yang diinginkan untuk dibentuk menjadi porotofolio saham optimal dengan juga memasukkan parameter algoritme genetika terdistribusi seperti pada Gambar 4.36.



Gambar 4.36 Perancangan Antarmuka Halaman *Input*

Perancangan antarmuka untuk halaman *input* yang tertera pada Gambar 4.36 akan dijelaskan makna dari setiap nomor sebagai berikut:

1. Kotak nomor 1 merupakan tab menu halaman *input.*
2. Kotak nomor 2 merupakan tab menu halaman hasil.
3. Nomor 3 merupakan *chekcbox* saham yang dapat dipilih oleh pengguna.
4. Nomor 4-7 berfungsi untuk memasukkan jumlah ukuran populasi, jumlah generasi, nilai cr dan nilai mr.

4.3.2 Perancangan Antaramuka Halaman Hasil

Perancangan antarmuka untuk halaman hasil akan muncul saat proses penentuan portofolio saham optimal sudah selesai. Halaman hasil akan menampilkan hasil akhir proporsi saham yang terbentuk dari setiap sub populasi, *return* ekspektasi, resiko portofolio serta nilai *fitness.* Perancangan antarmuka halaman hasil tertera pada Gambar 4.37.

Gambar 4.37 Perancangan Antarmuka Halaman Hasil

Penjelasan terkait perancangan antarmuka halaman hasil dari Gambar 4.37 adalah:

1. Kotak nomor 1 merupakan tab menu halaman *input*.
2. Kotak nomor 2 merupakan tab menu halaman hasil.
3. Kotak nomor 3 akan menampilkan hasil akhir dari setiap sub populasi meliputi proporsi saham, *return* ekspektasi, resiko portofolio, dan *fitness*.
4. Kotak nomor 4 akan menampilkan *fitness* terbesar.

4.4 Perancangan Pengujian

Perancangan pengujian dibuat untuk mengetahui bagaimana pengaruh parameter algoritme genetika terdistribusi dalam penentuan portofolio saham optimal. Jenis pengujian yang akan dilakukan pada penelitian ini antara lain:

1. Pengujian jumlah ukuran populasi.
2. Pengujian jumlah generasi.
3. Pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr)*.
4. Pengujian jumlah sub populasi.

4.4.1 Perancangan Pengujian Jumlah Ukuran Populasi

Pengujian jumlah ukuran populasi diperlukan agar dapat mengetahui pada populasi keberapa algortme genetika terdistribusi dapat memberikan hasil yang optimal. Parameter yang digunakan untuk pengujian populasi adalah jumlah ukuran populasi, jumlah generasi, nilai *cr* dan nilai *mr*.

1. Jumlah ukuran populasi: 20
2. Jumlah generasi: 300
3. *Cr* : 0,6
4. *Mr :* 0,4

Tabel 4.2 Perancangan Pengujian Jumlah ukuran populasi

Jumlah ukuran populasi	Nilai *fitness*										Rata - rata *fitness*
	Percobaan jumlah ukuran populasi ke -										
	1	2	3	4	5	6	7	8	9	10	
20											
40											
60											
80											
100											

4.4.2 Perancangan Pengujian Jumlah Generasi

Dari pengujian jumlah ukuran populasi yang dilakukan sebelumnya, pengujian selanjutnya adalah pengujian jumlah generasi karena dari pengujian generasi dapat diketahui pada generasi keberapa algoritme genetika terdistribusi dapat mencapai optimal. Parameter yang digunakan untuk pengujian jumlah generasi adalah jumlah ukuran populasi, jumlah generasi, nilai *cr* dan nilai *mr*.

1. Jumlah ukuran populasi: menggunakan hasil terbaik dari pengujian jumlah ukuran populasi
2. Jumlah generasi: 50 sampai 300
3. *Cr* : 0,1
4. *Mr* : 0,9

Tabel 4.3 Perancangan Pengujian Jumlah Generasi

Jumlah Generasi	Nilai *fitness*										Rata - rata *fitness*
	Percobaan jumlah generasi ke -										
	1	2	3	4	5	6	7	8	9	10	
50											
150											
200											
250											
300											
500											

4.4.3 Perancangan Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Rancangan pengujian ketiga adalah menguji kombinasi nilai *cr* dan *mr*, pengujian ini dilakukan untuk mengetahui kombinasi nilai *cr* dan *mr* yang paling optimal. Untuk nilai *cr* dan *mr* dibatasi pada interval 0 sampai 1. Parameter yang digunakan untuk pengujian kombinasi *cr* dan *mr* adalah jumlah ukuran populasi, jumlah generasi, dan kombinasi nilai *cr* dan *mr*.

1. Jumlah ukuran populasi: menggunakan hasil terbaik dari pengujian jumlah ukuran populasi
2. Jumlah Generasi: menggunakan hasil terbaik dari pengujian jumlah generasi
3. *Cr* : interval 0 sampai 1
4. *Mr* : interval 0 sampai 1

Tabel 4.4 Perancangan Pengujian Kombinasi Cr dan Mr

Kombinasi		Nilai Fitness Percobaan kombinasi cr dan mr ke-										Rata - rata fitness
cr	mr	1	2	3	4	5	6	7	8	9	10	
1	0											
0,9	0,1											
0,8	0,2											
0,7	0,3											
0,6	0,4											
0,5	0,5											
0,4	0,6											
0,3	0,7											
0,2	0,8											
0,1	0,9											
0	1											

4.4.4 Perancangan Pengujian Jumlah Sub Populasi

Rancangan pengujian yang terakhir adalah menguji jumlah sub populasi. Pengujian ini dilakukan untuk mengetahui bagaimana pengaruh jumlah sub populasi terhadap hasil penentuan portofolio saham optimal. Parameter yang digunakan untuk pengujian jumlah sub populasi adalah jumlah ukuran populasi, jumlah generasi, *cr, mr*, dan jumlah sub populasi.

1. Jumlah ukuran populasi: menggunakan hasil terbaik dari pengujian jumlah ukuran populasi.
2. Jumlah Generasi: menggunakan hasil terbaik dari pengujian jumlah generasi.
3. *Cr* : menggunakan hasil terbaik dari pengujian kombinasi *cr* dan *mr*.
4. *Mr* : menggunakan hasil terbaik dari pengujian kombinasi *cr* dan *mr*.

Tabel 4.5 Perancangan Pengujian Jumlah Sub Populasi

Jumlah Sub Populasi	Nilai *fitness*										Rata - rata *fitness*
	Percobaan jumlah sub populasi ke -										
	1	2	3	4	5	6	7	8	9	10	
2											
4											
6											
8											
10											
12											
14											

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan bagaimana implementasi dari perancangan algoritme dan implementasi dari perancangan antarmuka yang sudah dilakukan pada Bab 4. Implementasi dilakukukan untuk mengetahui bagaimana penerapan algoritme genetika terdistribusi untuk menentukan portofolio saham optimal dalam bentuk program sehingga dapat memberikan hasil atau *output* seperti yang diharapkan dan ditampilkan sesuai dengan perancangan antarmuka yang sudah dilakukan sebelumnya.

5.1 Implementasi Algoritme

Berdasarkan perancangan algoritme yang dilakukan sebelumnya, maka untuk implementasi algoritme terdiri dari implementasi representasi kromosom, reproduksi yaitu *crossover (extended intermediate crossover, one cut point crossover)* dan mutasi, menghitung *fitness*, evaluasi, seleksi dan migrasi. Implementasi dibuat dengan bahasa pemrograman Java dengan *database* MySQL.

5.1.1 Implementasi Algoritme Representasi Kromosom

Representasi kromosom dibangkitkan secara acak pada interval (0-1) dengan batasan jumlah nilai kromosom pada setiap individu tidak lebih dari 1 karena kromosom merpresentasikan proporsi saham, maka pada implementasinya dibuat sebuah fungsi bernama *repair* yang bertujuan untuk melakukan normalisasi agar setiap jumlah kromosom pada setiap individu tidak lebih dari 1. Implementasi *source code* representasi kromosom seperti pada Gambar 5.1, sedangkan untuk implementasi *repair* dijelaskan pada Gambar 5.2.

```
1   public void setKromosom() {
2           subpopulasi = new double[popsize][jumlahgen];
3           System.out.println("popsize " + popsize);
4           for (int i = 0; i < popsize; i++) {
5               for (int j = 0; j < jumlahgen; j++) {
6                   subpopulasi[i][j] = r.nextDouble();
7               }
8           }
9           System.out.println(" +" + subpopulasi.length);
10      }
```

Gambar 5.1 *Source Code* Representasi Kromosom

Penjelasan *source code* Gambar 5.1 adalah:

1. Baris 4-6 inisialisasi kromosom secara acak sejumlah *popsize* dan jumlah gen yang merupakan jumlah saham yang dipilih.

```
1   public double[][] repair() {
2           double[] total = new double[subpopulasi.length];
3           for (int i = 0; i < subpopulasi.length; i++) {
4               for (int j = 0; j < jumlahgen; j++) {
5                   total[i] = total[i] + subpopulasi[i][j];
6               }
```

```
7           }
8           for (int i = 0; i < subpopulasi.length; i++) {
9               for (int j = 0; j < jumlahgen; j++) {
10                  subpopulasi[i][j] = subpopulasi[i][j] /
11  total[i];
12              }
13          }
14          return subpopulasi;
15      }
```

Gambar 5.2 *Source Code Repair*

Penjelasan *source code* Gambar 5.2 adalah:

1. Baris 4-7 menjumlahkan nilai gen pada setiap individu di sub populasi.
2. Baris 10-13 setelah diketahui total nilai gen, dilakukan proses *repair* dengan cara membagi total nilai ke nilai gen sehingga nilai berubah dan memenuhi syarat. Maka hasil *repair* ini yang akan digunakan sebagai kromosom.

5.1.2 Implementasi Algoritme *Extended Intermediate Crossover*

Pada penelitian ini digunakan dua jenis *crossover* yang berbeda. Untuk *extended intermediate crossover* maka proses *crossover* akan dilakukan dengan menggunakan nilai *alpha* yang dibangkitkan secara acak pada interval (0-1) untuk menghasilkan keturunan dengan cara memilih dua *parent* secara acak untuk dilakukan proses *extended intermediate crossover* seperti pada Gambar 5.3.

```
1   public double[][] ExtendedCross(double[][] subpopulasi) {
2           int jumlahanak = (int) Math.round(popsize * cr);
3           int proses = (int) Math.ceil((double) jumlahanak /
4   2);
5           double[] p1 = new double[jumlahgen];
6           double[] p2 = new double[jumlahgen];
7           double[][] child = new
8   double[jumlahanak][jumlahgen];
9           int parent1;
10          int parent2;
11          double[] alpha = new double[jumlahgen];
12          for (int i = 0; i < jumlahgen; i++) {
13              alpha[i] = r.nextDouble();
14          }
15          System.out.println("\t proses Crossover");
16
17          for (int i = 0; i < proses; i++) {
18              parent1 = r.nextInt(popsize);
19              System.out.println("\tP1: " + parent1);
20              parent2 = r.nextInt(popsize);
21
22              while (parent1 == parent2) {
23                  parent2 = r.nextInt(popsize);
24              }
25              System.out.println("\tP2: " + parent2);
26              for (int j = 0; j < jumlahgen; j++) {
27                  p1[j] = subpopulasi[parent1][j];
28                  p2[j] = subpopulasi[parent2][j];
29
```

```
30                  System.out.print(alpha[j] + "\t");
31              }
32              System.out.println("Alpha: ");
33              for (int j = 0; j < jumlahanak; j++) {
34                  for (int k = 0; k < jumlahgen; k++) {
35  if (j % 2 != 0) {
36                          child[j][k] = p1[k] + (alpha[k] *
37  (p2[k]) - p1[k]);
38
39                      } else {
40                          child[j][k] = p2[k] + (alpha[k] *
41  (p1[k]) - p2[k]);
42                      }
43
44                  }
45                  System.out.println();
46              }
47
48              for (int j = 0; j < child.length; j++) {
49                  System.out.print("\tChild CrossOver  :  ");
50                  for (int k = 0; k < child[0].length; k++) {
51                      System.out.print(df.format(child[j][k])
52  + "\t");
53
54                  }
55                  System.out.println("");
56              }
57
58          }
59          return child;
60      }
```

Gambar 5.3 *Source Code Extended Intermediate Crossover*

Penjelasan *source code* pada Gambar 5.3 adalah:

1. Baris 12-14 digunakan untuk membangkitkan nilai *alpha* secara acak pada interval (0,1) sejumlah dengan kromosom.
2. Baris 17-31 berfungsi untuk memilih dua *parent* secara acak kemudian mengambil nilai kromosom dari masing-masing *parent* terpilih.
3. Baris 33-46 melakukan proses *extended intermediate crossover* dengan cara menghitung untuk masing-masing *child* yang dihasilkan dari dua *parent* yang terpilih. Baris 36-37 untuk menghasilkan *child* pertama dan baris 39-41 untuk menghasilkan *child* kedua.
4. Baris 48-59 berfungsi untuk menampilkan hasil *child* yang didapat dari proses *extended intermediate crossover*.

5.1.3 Implementasi Algoritme *One Cut Point Crossover*

Proses *crossover* yang kedua adalah *one cut point* yang berarti akan dilakukan pemilihan *parent* secara acak kemudian akan ditentukan *cut point* atau titik potong yang juga ditentukan secara acak untuk menghasilkan *child* seperti pada Gambar 5.4.

```
1   public double[][] OneCutCross(double[][] subpopulasi) {
```

```
2
3           int jumlahanak = (int) Math.round(popsize * cr);
4           int proses = (int) Math.ceil((double) jumlahanak /
5   2);
6           double[] p1 = new double[jumlahgen];
7           double[] p2 = new double[jumlahgen];
8           double[][] child = new
9   double[jumlahanak][jumlahgen];
10          double cut;
11          int parent1;
12          int parent2;
13          System.out.println("\t proses Crossover");
14          System.out.println("jml anak " + jumlahanak);
15          System.out.println("proses " + proses);
16          for (int i = 0; i < proses; i++) {
17              parent1 = r.nextInt(popsize);
18              System.out.println("\tP1: " + parent1);
19              parent2 = r.nextInt(popsize);
20              while (parent1 == parent2) {
21                  parent2 = r.nextInt(popsize);
22              }
23              System.out.println("\tP2: " + parent2);
24              for (int j = 0; j < jumlahgen; j++) {
25                  p1[j] = subpopulasi[parent1][j];
26                  p2[j] = subpopulasi[parent2][j];
27
28              }
29
30              cut = r.nextInt(jumlahgen - 1) + 1;
31              System.out.println("\tcut point: " + cut);
32              for (int j = 0; j < jumlahanak; j++) {
33                  for (int k = 0; k < jumlahgen; k++) {
34                      if (j % 2 != 0) {
35                          if (k < cut) {
36                              child[j][k] = p1[k];
37
38                          } else {
39                              child[j][k] = p2[k];
40
41                          }
42                      } else if (j % 2 == 0) {
43                          if (k < cut) {
44                              child[j][k] = p2[k];
45
46                          } else {
47                              child[j][k] = p1[k];
48
49                          }
50
51                      }
52
53                  }
54              }
55              for (int j = 0; j < child.length; j++) {
56                  System.out.print("\tChild CrossOver  :  ");
57                  for (int l = 0; l < child[0].length; l++) {
58                      System.out.print(df.format(child[j][l])
59  + "\t");
60
```

```
61                      }
62                      System.out.println("");
63                  }
64              }
65              return child;
66      }
```

Gambar 5.4 *Source Code One Cut Point Crossover*

Penjelasan *source code* pada Gambar 5.4 adalah:

1. Baris 16-28 menentukan dua *parent* secara acak kemudian mengambil nilai kromosom dari kedua *parent* yang sudah dipilih.
2. Baris 30-54 menentukan *cut point* secara acak, kemudian dilakukan proses *one cut point crossover.* Baris 34-40 jika masuk ke kondisi saat di mod dengan 2 tidak sama dengan 0 dan jika gen terletak sebelum *cut point* maka nilai *child* tidak berubah tapi jika tidak maka nilai *child* berasal dari *parent 2* sedangkan baris 42-47 jika lebih dari *cut point* maka *child* berasal dari *parent* 1.
3. Baris 55-65 digunakan untuk mencetak hasil *child.*

5.1.4 Implementasi Algoritme Mutasi

Setelah proses *crossover*, tahapan reproduksi selanjutnya adalah reproduksi dengan mutasi. Mutasi yang digunakan pada penelitian ini adalah *reciprocal exchange mutation* yang berarti memilih satu *parent* secara acak kemudian memilih dua gen untuk ditukar, satu kali proses mutasi menghasilkan satu *child*. Implementasi mutasi dijelaskan pada Gambar 5.5.

```
1   public double[][] mutasi(double[][] subpopulasi) {
2           int jumlahanak = (int) Math.round(popsize * mr);
3           double[] p1 = new double[jumlahgen];
4           int parent;
5           int gen1;
6           int gen2;
7           double[][] child = new
8   double[jumlahanak][jumlahgen];
9           System.out.println("\n\tPROSES MUTASI");
10          for (int i = 0; i < jumlahanak; i++) {
11              parent = r.nextInt(popsize);
12              System.out.println("\tparent: " + parent);
13              for (int j = 0; j < jumlahgen; j++) {
14                  p1[j] = subpopulasi[parent][j];
15              }
16              gen1 = r.nextInt(jumlahgen);
17              gen2 = r.nextInt(jumlahgen);
18              while (gen1 == gen2) {
19                  gen2 = r.nextInt(jumlahgen);
20              }
21              System.out.println("\tgen terpilih: " + gen1 +
22  "dan" + gen2);
23              double temp;
24              child[i] = p1;
25              temp = child[i][gen1];
26              child[i][gen1] = child[i][gen2];
27              child[i][gen2] = temp;
```

```
28
29          }
30          for (int j = 0; j < child.length; j++) {
31              System.out.print("\tChild Mutasi  :  ");
32              for (int l = 0; l < child[0].length; l++) {
33                  System.out.print(df.format(child[j][l]) +
34  "\t");
35
36              }
37              System.out.println("");
38          }
39          return child;
40  }
```

Gambar 5.5 *Source Code* Mutasi

Penjelasan *source code* pad Gambar 5.5 adalah:

1. Baris 10-15 menentukan *parent* secara acak.
2. Baris 17-30 menentukan gen yang dipilih secara acak kemudian jika gen tidak terpilih maka nilai gen tetap tapi jika tidak maka nilai gen akan ditukar sehingga menghasilkan *child*.

5.1.5 Implementasi Algoritme *Fitness*

Proses menghitung *fitness* melalui beberapa tahapan yaitu melalui proses perhitungan dengan *single index model,* diawali dengan menghitung *alpha* portofolio, beta portofolio, kesalahan residu portofolio, *return* eksepektasi, resiko portofolio dan yang terakhir adalah menghitung *fitness*.

5.1.5.1 *Alpha* Portofolio

Tahapan pertama pada *single index model* adalah menghitung *alpha* portofolio seperti yang tertera pada perancangan algoritme. Implementasi algoritme *alpha* portofolio dijelaskan pada Gambar 5.6.

```
1   public double[] AlphaPortofolio(double[][] subpopulasi,
2   int[] saham, Object[][] alpha) {
3           double[] alphasaham = new double[saham.length];
4           double[] jumlahalpha = new
5   double[subpopulasi.length];
6           for (int i = 0; i < saham.length; i++) {
7               for (int j = 0; j < alpha.length; j++) {
8                   if (saham[i] == (int) alpha[j][0]) {
9                       alphasaham[i] = (double) alpha[j][3];
10
11                  }
12              }
13
14          }
15          for (int i = 0; i < subpopulasi.length; i++) {
16              double temp = 0;
17              for (int j = 0; j < subpopulasi[0].length; j++){
18                  temp = alphasaham[j] * subpopulasi[i][j];
19                  jumlahalpha[i] = jumlahalpha[i] + temp;
20              }
21          }
```

```
22          return jumlahalpha;
23      }
```

Gambar 5.6 *Source Code Alpha* Portofolio

Penjelasan *source code* pada Gambar 5.6 adalah:

1. Baris 6-12 mengambil data *alpha* pada setiap emiten saham yang dipilih.
2. Baris 15-22 melakukan proses perhitungan *alpha* portofolio dengan cara mengalikan *alpha* setiap saham ke nilai kromosom yang ada pada sub populasi kemudian menjumlahkan hasilnya sehingga didapatkan *alpha* portofolio.

5.1.5.2 Beta Portofolio

Setelah *alpha* portofolio ditemukan, selanjutnya adalah menghitung beta portofolio seperti yang tertera pada perancangan algoritme. Implementasi algoritme beta portofolio dijelaskan pada Gambar 5.7.

```
1   public double[] BetaPortofolio(double[][] subpopulasi, int[]
2   saham, Object[][] beta) {
3           double[] betasaham = new double[saham.length];
4           double[] jumlahbeta = new
5   double[subpopulasi.length];
6
7           for (int i = 0; i < saham.length; i++) {
8               for (int j = 0; j < beta.length; j++) {
9                   if (saham[i] == (int) beta[j][0]) {
10                      betasaham[i] = (double) beta[j][2];
11                  }
12              }
13
14          }
15          for (int i = 0; i < subpopulasi.length; i++) {
16              double temp = 0;
17              for (int j = 0; j < subpopulasi[0].length; j++)
18  {
19                  temp = betasaham[j] * subpopulasi[i][j];
20                  jumlahbeta[i] = jumlahbeta[i] + temp;
21              }
22          }
23          return jumlahbeta;
24      }
```

Gambar 5.7 *Source Code* Beta Portofolio

Penjelasan *source code* pada Gambar 5.7 adalah:

1. Baris 7-12 mengambil data beta pada setiap emiten saham yang dipilih.
2. Baris 15-23 melakukan proses perhitungan beta portofolio dengan cara mengalikan beta setiap saham ke nilai kromosom yang ada pada sub populasi kemudian menjumlahkan hasilnya sehingga didapatkan beta portofolio.

5.1.5.3 Kesalahan Residu Portofolio

Setelah *alpha* dan beta portofolio ditemukan, tahapan selanjutnya adalah menghitung kesalahan residu portofolio seperti yang tertera pada perancangan algoritme. Implementasi algoritme kesalahan residu portofolio dijelaskan pada Gambar 5.8.

```
1   public double[] ResiduPortofolio(double[][] subpopulasi,
2   int[] saham, Object[][] residu) {
3           double[] residusaham = new double[saham.length];
4           double[] jumlahresidu = new
5   double[subpopulasi.length];
6           for (int i = 0; i < saham.length; i++) {
7               for (int j = 0; j < residu.length; j++) {
8                   if (saham[i] == (int) residu[j][0]) {
9                       residusaham[i] = (double) residu[j][4];
10                  }
11              }
12
13          }
14          for (int i = 0; i < subpopulasi.length; i++) {
15              double temp = 0;
16              for (int j = 0; j < subpopulasi[0].length; j++)
17  {
18                  temp = Math.pow(residusaham[j] *
19  subpopulasi[i][j], 2);
20                  jumlahresidu[i] = jumlahresidu[i] + temp;
21              }
22          }
23          return jumlahresidu;
24      }
```

Gambar 5.8 *Source Code* Kesalahan Residu Portofolio

Penjelasan *source code* pada Gambar 5.8 adalah:

1. Baris 7-11 mengambil data kesalahan residu pada setiap emiten saham yang dipilih.
2. Baris 16-23 melakukan proses perhitungan kesalahan residu portofolio dengan cara mengalikan kesalahan residu setiap saham ke nilai kromosom yang ada pada sub populasi, lalu menjumlahkan hasilnya sehingga didapatkan kesalahan residu portofolio.

5.1.5.4 *Return* Ekspektasi

Alpha, beta, dan kesalahan residu portofolio sudah dihitung maka tahapan selanjutnya adalah menghitung *return* ekspektasi seperti yang tertera pada perancangan algoritme. Implementasi algoritme *return* ekspektasi dijelaskan pada Gambar 5.9.

```
1   public double[] ReturnEkspektasi(double[][] subpopulasi,
2   int[] saham, Object[][] data4) {
3           double[] returneks = new double[subpopulasi.length];
4
5           double[] alpha = AlphaPortofolio(subpopulasi, saham,
```

```
6       data4);
7               double[] beta = BetaPortofolio(subpopulasi, saham,
8       data4);
9
10              for (int i = 0; i < subpopulasi.length; i++) {
11                  returneks[i] = alpha[i] + (beta[i] * rm);
12              }
13              System.out.println("Return Ekspektasi Portofolio:
14      "+Arrays.toString(returneks));
15              return returneks;
16          }
```

Gambar 5.9 *Source Code Return* Ekspektasi

Penjelasan *source code* pada Gambar 5.9 adalah:

1. Baris 6-9 memanggil fungsi *alpha* portofolio dan beta portofolio.
2. Baris 11-15 melakukan proses perhitungan *return* ekspektasi dengan cara *alpha* portofolio ditambah dengan hasil perkalian dari beta portofolio dikali dengan *return market* lalu pada baris 14-15 hasil *return* ekspektasi ditampilkan.

5.1.5.5 Resiko Portofolio

Tahapan terakhir pada *single index model* sebelum menghitung *fitness* adalah menghitung resiko portofolio seperti yang tertera pada perancangan algoritme. Implementasi algoritme resiko portofolio dijelaskan pada Gambar 5.10,

```
1       public double[] ResikoPortofolio(double[][] subpopulasi,
2       int[] saham, Object[][] data4) {
3               double[] resikop = new double[subpopulasi.length];
4
5               double[] beta = BetaPortofolio(subpopulasi, saham,
6       data4);
7               double[] residu = ResiduPortofolio(subpopulasi,
8       saham, data4);
9
10              for (int i = 0; i < subpopulasi.length; i++) {
11                  resikop[i] = Math.pow(beta[i], 2) + (rp *
12      residu[i]);
13                  resikop[i] = Math.sqrt(resikop[i]);
14              }
15              System.out.println("Resiko Portofolio:
16      "+Arrays.toString(resikop));
17              return resikop;
18          }
```

Gambar 5.10 *Source Code* Resiko Portofolio

Penjelasan *source code* pada Gambar 5.10 adalah:

1. Baris 6-9 memanggil fungsi beta portofolio dan kesalahan residu portofolio.
2. Baris 11-16 melakukan proses perhitungan resiko portofolio dengan cara beta portofolio dikuadrat lalu ditambah dengan hasil perkalian dari

variabel rp dengan kesalahan residu portofolio kemudian hasilnya diakar. Pada baris 16-17 hasil resiko portofolio ditampilkan.

5.1.5.6 Fitness

Seluruh tahapan *single index model* telah selesai maka tahapan selanjutnya adalah menghitung *fitness* yang tertera pada perancangan algoritme. Implementasi algoritme *fitness* dijelaskan pada Gambar 5.11.

```
1   public double[] HitungFitness(int[] saham, Object[][] data4)
2   {
3           double[] fitness = new double[subpopulasi.length];
4
5           double[] resikop = ResikoPortofolio(subpopulasi,
6   saham, data4);
7           double[] returneks = ReturnEkspektasi(subpopulasi,
8   saham, data4);
9
10          for (int i = 0; i < subpopulasi.length; i++) {
11              if (resikop[i] > returneks[i]) {
12                  fitness[i] = (returneks[i] / resikop[i]) *
13  (-1);
14                  fitness[i] = -(fitness[i] / 1000);
15              } else {
16                  fitness[i] = returneks[i] / resikop[i];
17                  fitness[i] = fitness[i] / 1000;
18              }
19          }
20          System.out.println("fitness: " +
21  Arrays.toString(fitness));
22          return fitness;
23      }
```

Gambar 5.11 *Source Code Fitness*

Penjelasan *source code* pada Gambar 5.11 adalah:

1. Baris 6-9 memanggil fungsi *return* ekspektasu dan resiko portofolio.
2. Baris 12-14 jika nilai resiko lebih besar dari *return* maka proses perhitungan *fitness* dengan cara *return* ekspektasi dibagi resiko portofolio lalu dikali (-1) lalu hasilnya dibagi dengan 1000, Jika tidak memenuhi kondisi *if*, maka *fitness* dilakukan dengan *return* dibagi resiko lalu hasilnya dibagi dengan 1000,

5.1.6 Implementasi Algoritme Evaluasi

Setelah proses mutasi selesai selanjutnya adalah melakukan proses evaluasi atau menggabungkan seluruh *parent* dan *child* menjadi satu sub populasi, implementasi evaluasi seperti pada Gambar 5.12.

```
1   public double[][] Evaluasi(double[][] subpopulasi) {
2           int penentucross = 0;
3           double[][] subpopulasibaru;
4           double[][] cross;
5           penentucross = r.nextInt(2);
6           if (penentucross == 1) {
7               System.out.println("Extended");
```

```
8               cross = ExtendedCross(subpopulasi);
9               penentucross++;
10          } else {
11              System.out.println("One cut");
12              cross = OneCutCross(subpopulasi);
13          }
14          double[][] mutasi = mutasi(subpopulasi);
15
16          populasiBaru = popsize + cross.length +
17  mutasi.length;
18          int jumlah = populasiBaru - popsize;
19          subpopulasibaru = new
20  double[populasiBaru][jumlahgen];
21          for (int i = 0; i < populasiBaru; i++) {
22              for (int j = 0; j < jumlahgen; j++) {
23                  if (i < popsize) {
24                      subpopulasibaru[i][j] =
25  subpopulasi[i][j];
26                  }
27              }
28          }
29          for (int i = 0; i < jumlah; i++) {
30              if (i < cross.length) {
31                  for (int j = 0; j < jumlahgen; j++) {
32                      subpopulasibaru[i + popsize][j] =
33  cross[i][j];
34                  }
35              } else {
36                  for (int j = 0; j < jumlahgen; j++) {
37                      subpopulasibaru[i + popsize][j] =
38  mutasi[i - cross.length][j];
39                  }
40              }
41
42          }
43
44          this.subpopulasi = subpopulasibaru;
45
46          return subpopulasibaru;
47      }
```

Gambar 5.12 *Source Code* Evaluasi

Penjelasan *source code* pada Gambar 5.12 adalah:

1. Baris 6-14 menentukan sub populasi tersebut akan menggunakan *extended intermediate* atau *one cut point crossover* secara acak, lalu memanggil fungsi mutasi.
2. Baris 16-20 inisialisasi variabel populasiBaru dengan *parent* dan *child* dari hasil *crossover* dan mutasi.

Baris 21-46 perulangan yang berguna untuk mengambil individu pada sub populasi yang tersimpan di variabel subpopulasi disimpan di variabel subpopulasibaru. Baris 29-38 memanggil *child* yang didapatkan dari proses *crossover* dan mutasi untuk kemudian disimpan ke dalam variabel subpopulasibaru.

5.1.7 Implementasi Algoritme Seleksi

Setelah diketahui nilai *fitness* dari setiap individu yang terdiri dari *parent* dan *child* maka tahapan selanjutnya adalah melakukan proses seleksi dengan *elitism selection* yaitu memilih individu terbaik dengan nilai *fitness* terbesar sejumlah *popsize* awal. Implementasi algoritme seleksi dijelaskan pada Gambar 5.13.

```
1   public double[][] SeleksiElitism(double[][] subpopulasi) {
2           int batas = popsize;
3           baru = new double[batas][jumlahgen];
4           for (int i = 0; i < batas; i++) {
5               for (int j = 0; j < jumlahgen; j++) {
6                   baru[i][j] = subpopulasi[(int) ind[i]][j];
7               }
8           }
9           System.out.println("sub populasi baru");
10          for (int i = 0; i < batas; i++) {
11              System.out.print("parent " + (i) + ": ");
12              for (int j = 0; j < jumlahgen; j++) {
13                  System.out.print(baru[i][j] + " ");
14              }
15              System.out.println("");
16          }
17          subpopulasi = baru;
18          return baru;
19      }
```

Gambar 5.13 *Source Code* Seleksi

Penjelasan *source code* pada Gambar 5.13 adalah:

1. Baris 5-10 mengambil individu terbaik sejumlah *popsize*.
2. Baris 11-18 mencetak individu-individu terbaik yang lolos seleksi sejumlah *popsize*.

5.1.8 Implementasi Algoritme Migrasi

Migrasi adalah proses terakhir dari algoritme genetika terdistribusi yang berfungsi untuk menukar satu individu terbaik dari sebuah sub populasi ke sub populasi yang lain. Implementasi algoritme seleksi dijelaskan pada Gambar 5.14.

```
1   public void Migrasi(double[][] subpopulasi1, double[][]
2   subpopulasi2, double[][] subpopulasi3, double[][]
3   subpopulasi4, double[][] subpopulasi5) {
4
5           double[] tempArray, tempArray2, tempArray3,
6   tempArray4, tempArray5;
7           tempArray = subpopulasi5[0];
8           tempArray2 = subpopulasi1[0];
9           tempArray3 = subpopulasi2[0];
10          tempArray4 = subpopulasi3[0];
11          tempArray5 = subpopulasi4[0];
12
13          subpopulasi1[0] = tempArray;
14          subpopulasi2[0] = tempArray2;
15          subpopulasi3[0] = tempArray3;
16          subpopulasi4[0] = tempArray4;
```

```
17          subpopulasi5[0] = tempArray5;
18
19          setSubpopulasi1(subpopulasi1);
20          setSubpopulasi2(subpopulasi2);
21          setSubpopulasi3(subpopulasi3);
22          setSubpopulasi4(subpopulasi4);
23          setSubpopulasi5(subpopulasi5);
24
25      }
```

Gambar 5.14 *Source Code* Migrasi

Penjelasan *source code* pada Gambar 5.14 adalah:

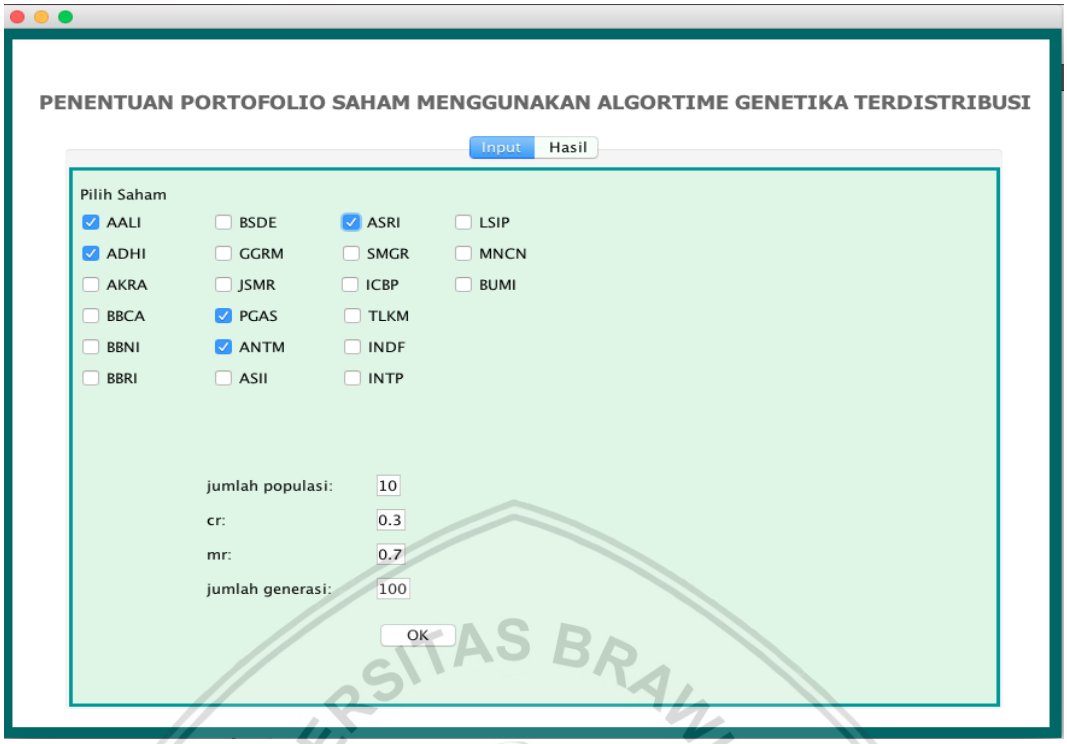
1. Baris 7-11 melakukan proses penukaran satu individu terbaik dari sebuah sub populasi ke sub populasi yang lain.
2. Baris 13-17 menyimpan hasil penukaran sub populasi ke variabel tempArray.
3. Bari 19-23 menyimpan hasil migrasi untuk dijadikan nilai sub populasi pada iterasi berikutnya.

5.2 Implementasi Antarmuka

Berdasarkan perancangan antarmuka yang dibuat sebelumnya maka pada bab ini akan dijelaskan bagaimana implementasi antarmuka yang dilakukan sesuai dengan perancangan yang sebelumnya.

5.2.1 Implementasi Antarmuka Halaman *Input*

Antarmuka halaman *input* dapat dilihat pada Gambar 5.15, antarmuka halaman *input* dibuat sesuai dengan perancangan yang sudah dilakukan. Halaman *input* dapat menerima data masukan dari pengguna untuk kemudian di proses dengan algoritme genetika terdistribusi.

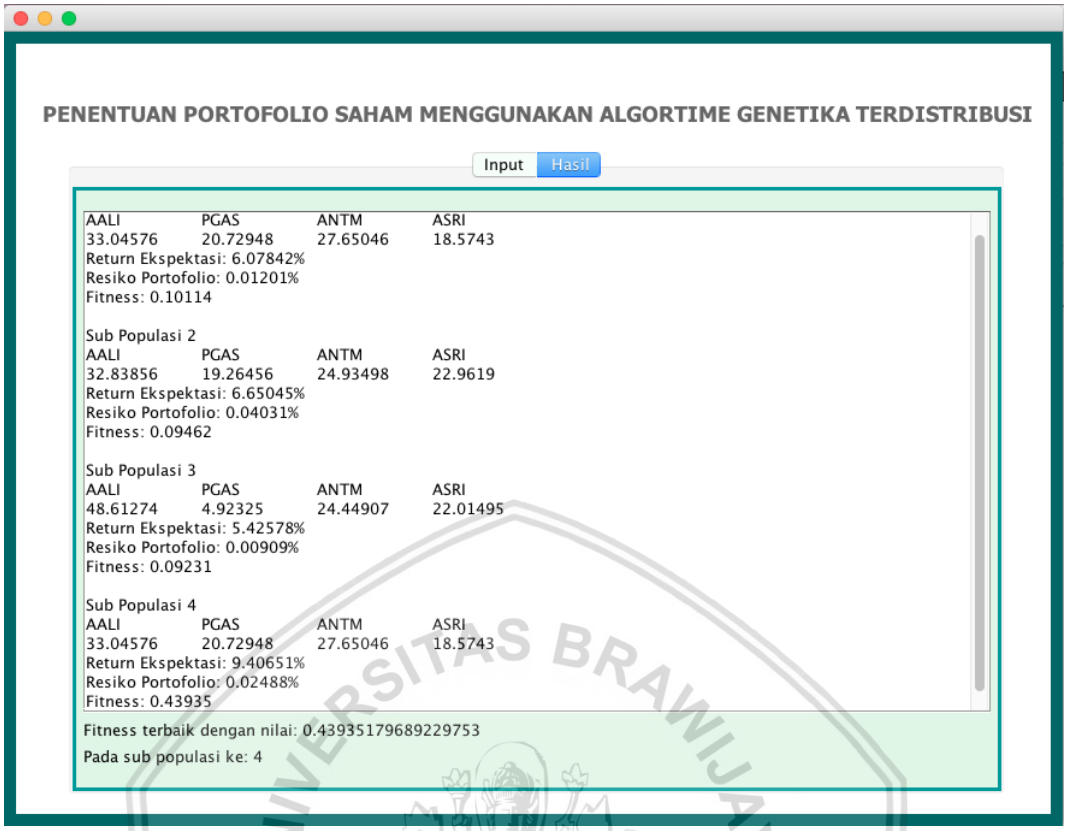


Gambar 5.15 Implementasi Antarmuka Halaman *Input*

Pada Gambar 5.15 dapat dilihat bagaimana implementasi *input* yang dibuat sesuai dengan perancangan. Pengguna dapat memilih saham yang diinginkan dan memasukkan data parameter algoritme genetika terdistribusi ke dalam program.

5.2.2 Implementasi Antarmuka Halaman Hasil

Antarmuka halaman hasil merupakan antarmuka yang akan ditampilkan jika proses penentuan portofolio saham optimal sudah selesai. Antarmuka halaman hasil dapat dilihat pada Gambar 5.16.



Gambar 5.16 Antarmuka Halaman Hasil

Halaman hasil menunjukkan portofolio saham yang terbentuk pada setiap sub populasi beserta proporsi setiap saham, *return* ekspektasi, resiko portofolio, dan *fitness*. Dari antarmuka halaman hasil dapat diketahui nilai *fitness* terbaik terdapat pada sub populasi keberapa.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan bagaimana pengujian yang dilakukan sesuai dengan perancangan pengujian pada Bab 4 serta menganalisis hasil dari setiap pengujian yang sudah dilakukan. Pengujian parameter algoritme genetika terdistribusi yang dilakukan adalah pengujian jumlah ukuran populasi, pengujian jumlah generasi, pengujian kombinasi *crossover rate* (*cr*) serta *mutation rate* (*mr*), dan pengujian jumlah sub populasi.

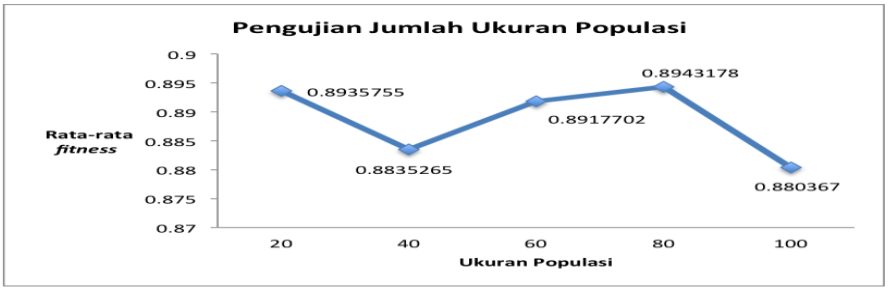
6.1 Pengujian dan Analisis Hasil Pengujian Jumlah Ukuran Populasi

Berdasarkan perancangan pengujain jumlah ukuran populasi yang dilakukan sebelumnya, maka untuk pengujian jumlah ukuran populasi diawali dengan jumlah ukuran populasi sebesar 20 dan kelipatan 20 hingga 100 sesuai dengan perancangan pengujian jumlah ukuran populasi. Generasi yang digunakan yaitu 300 generasi, dengan nilai *cr* sebesar 0,6 dan nilai *mr* sebesar 0,4. Sedangkan jumlah sub populasi yang digunakan berjumlah 4 sub populasi. Pengujian untuk setiap jumlah ukuran populasi diuji sebanyak 10 kali. Hasil dari pengujian yang dilakukan pada setiap jumlah ukuran populasi yang dilakukan tertera pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Jumlah Ukuran Populasi

Jumlah ukuran populasi	Nilai *Fitness* Percobaan Populasi ke-										Rata - rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
20	0,873	0,905	0,880	0,903	0,902	0,889	0,890	0,902	0,893	0,889	0,8935755
40	0,888	0,879	0,889	0,880	0,889	0,896	0,872	0,881	0,882	0,873	0,8835265
60	0,898	0,899	0,890	0,894	0,896	0,879	0,902	0,893	0,884	0,878	0,8917702
80	0,882	0,904	0,903	0,893	0,900	0,900	0,900	0,900	0,877	0,879	0,8943178
100	0,886	0,883	0,889	0,891	0,851	0,879	0,889	0,853	0,881	0,897	0,880367

Dari Tabel 6.1 dibuat sebuah grafik yang menampilkan hasil pengujian jumlah ukuran populasi. Dari grafik dapat dengan mudah diketahui populasi keberapa yang optimal. Grafik hasil pengujian jumlah ukuran populasi terdapat pada Gambar 6.1



Gambar 6.1 Grafik Hasil Pengujian Jumlah ukuran populasi

Pada Gambar 6.1 dapat dilihat pada jumlah ukuran populasi 20 rata-rata *fitness* yang dihasilkan cukup besar yaitu 0,8935755 tetapi kemudian mengalami

penurunan yang siginifikan pada jumlah ukuran populasi 40 dengan rata-rata *fitness* sebesar 0,8835265. Pada populasi sejumlah 60 rata-rata *fitness* kembali naik dengan nilai 0,8917702 kemudian kembali mengalami kenaikan nilai rata-rata *fitness* pada jumlah ukuran populasi 80 yaitu 0,8943178. Pada populasi ke 100 rata-rata *fitness* mengalami penurunan yang signifikan dengan nilai 0,880367, semakin besar jumlah ukuran populasi semakin lama waktu komputasinya. Dari pengujian jumlah ukuran populasi yang dilakukan, dapat diketahui bahwa jumlah ukuran populasi sebanyak 80 memiliki nilai rata-rata *fitness* terbesar dan jumlah ukuran populasi 100 menunjukkan nilai rata-rata *fitness* terkecil. Maka, populasi sejumlah 80 merupakan jumlah ukuran populasi optimal, populasi optimal ini akan digunakan sebagai parameter berikutnya untuk pengujian jumlah generasi.

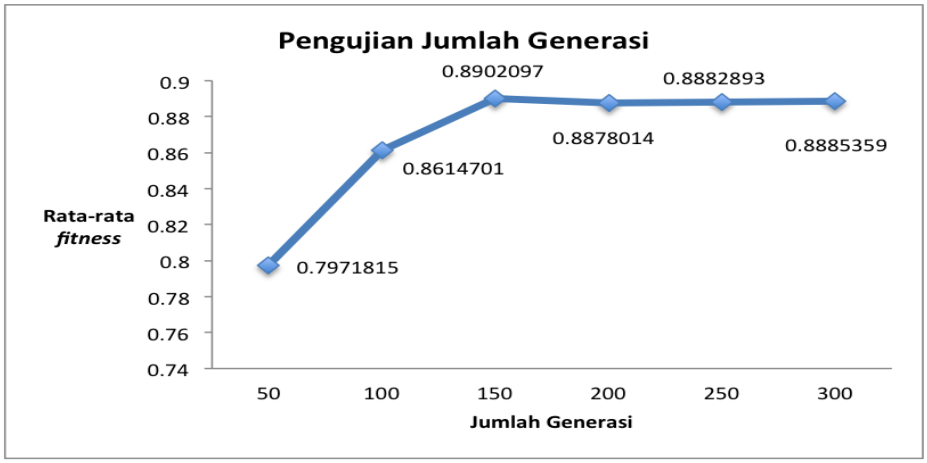
6.2 Pengujian dan Analisis Hasil Pengujian Jumlah Generasi

Pengujian kedua yang dilakukan adalah pengujian jumlah generasi. Pengujian ini dilakukan untuk mengetahui berapa jumlah generasi optimal pada penelitian ini. Jumlah generasi yang akan diuji sesuai dengan jumlah dari perancangan pengujian jumlah generasi, sedangkan jumlah ukuran populasi yang digunakan berasal dari hasil pengujian jumlah ukuran populasi yaitu 80, sedangkan untuk kombinasi nilai *cr* dan *mr* serta jumlah sub populasi berjumlah sama dengan yang digunakan untuk menguji jumlah generasi. Hasil pengujian jumlah generasi tertera pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Jumlah Generasi

Jumlah Generasi	Nilai *Fitness* Percobaan Generasi ke-										Rata - rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
50	0,887	0,844	0,751	0,694	0,874	0,641	0,879	0,628	0,898	0,871	0,7971815
100	0,866	0,877	0,902	0,732	0,878	0,902	0,886	0,896	0,776	0,894	0,8614701
150	0,895	0,879	0,903	0,890	0,894	0,893	0,880	0,894	0,881	0,888	0,8902097
200	0,893	0,880	0,878	0,885	0,880	0,903	0,906	0,885	0,874	0,888	0,8878014
250	0,902	0,898	0,894	0,887	0,875	0,887	0,885	0,879	0,885	0,886	0,8882893
300	0,886	0,879	0,903	0,891	0,879	0,894	0,890	0,885	0,893	0,881	0,8885359

Dari Tabel 6.2 yang merupakan hasil pengujian jumlah generasi dibentuk sebuah grafik yang akan memudahkan untuk mengetahui berapa jumlah generasi yang optimal pada penelitian ini berdasarkan pengujian yang sudah dilakukan. Grafik hasil pengujian jumlah generasi terdapat pada Gambar 6.2.



Gambar 6.2 Grafik Hasil Pengujian Jumlah Generasi

Pada Gambar 6.2 dapat diketahui bahwa rata-rata *fitness* meningkat secara signifikan dari 50 generasi ke 100 generasi dengan rata-rata *fitness* sebesar 0,861471, selanjutnya masih terjasi peningkatan rata-rata nilai *fitness* yang cukup besar pada generasi ke 150 dengan rata-rata *fitness* 0,8902097. Kemudian terjadi penurunan pada generasi ke 200, rata-rata *fitness* pada generasi 200 hingga generasi 300 tidak mengalami perubahan yang cukup signifikan hal ini berarti generasi ke 200 sampai generasi ke 300 sudah mengalami konvergen dan memakan waktu komputasi yang cukup lama sehingga pengujian generasi cukup dilakukan hingga 300 generasi. Berdasarkan hasil pengujian jumlah generasi tersebut, jumlah generasi yang optimal adalah 150 generasi.

6.3 Pengujian dan Analisis Hasil Pengujian Kombinasi Cr dan Mr

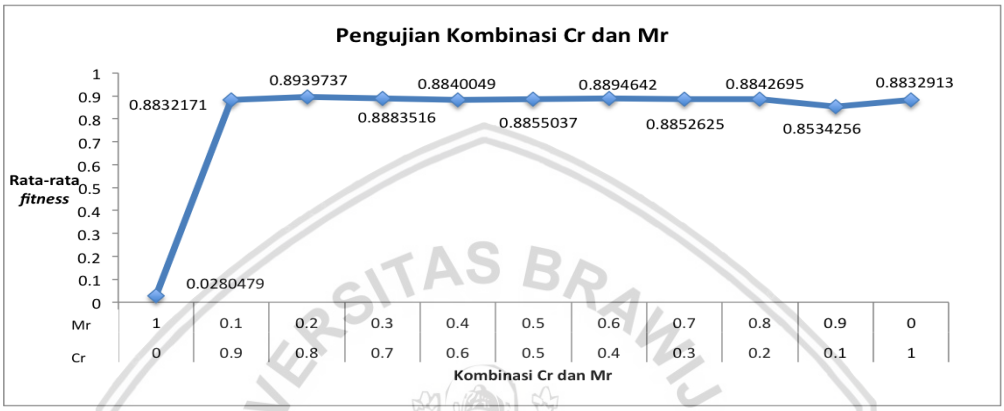
Pengujian yang ketiga adalah menguji kombinasi cr dan mr atau *crossover rate* dan *mutation rate*. Pengujian ini menggunakan jumlah ukuran populasi optimal yaitu 80, jumlah generasi optimal yaitu 150 dengan jumlah sub populasi sebanyak 4. Kombinasi *cr* dan *mr* yang akan diuji berdasarkan perancangan pengujian kombinasi *cr* dan *mr*. Tabel hasil pengujian dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Kombinasi Cr dan Mr

Kombinasi		Nilai Fitness Percobaan kombinasi cr dan mr ke-										Rata fitness
cr	mr	1	2	3	4	5	6	7	8	9	10	
0	1	0,003	0,033	0,030	0,030	0,028	0,031	0,030	0,029	0,029	0,032	0,0280479
0,9	0,1	0,881	0,889	0,871	0,878	0,882	0,890	0,873	0,885	0,885	0,894	0,8832171
0,8	0,2	0,904	0,903	0,894	0,883	0,901	0,882	0,881	0,889	0,900	0,898	0,8939737
0,7	0,3	0,893	0,849	0,901	0,903	0,902	0,884	0,901	0,873	0,902	0,871	0,8883516
0,6	0,4	0,891	0,895	0,870	0,888	0,884	0,880	0,874	0,884	0,900	0,870	0,8840049
0,5	0,5	0,882	0,895	0,876	0,877	0,899	0,889	0,898	0,887	0,873	0,874	0,8855037
0,4	0,6	0,883	0,903	0,890	0,895	0,886	0,884	0,886	0,864	0,902	0,896	0,8894642

0,3	0,7	0,894	0,883	0,900	0,874	0,899	0,872	0,873	0,901	0,883	0,870	0,8852625
0,2	0,8	0,896	0,882	0,877	0,899	0,905	0,904	0,805	0,895	0,884	0,892	0,8842695
0,1	0,9	0,883	0,898	0,855	0,726	0,878	0,831	0,894	0,792	0,879	0,893	0,8534256
1	0	0,872	0,872	0,879	0,893	0,857	0,892	0,899	0,878	0,903	0,882	0,8832913

Pada Tabel 6.3 yang merupakan hasil pengujian kombinasi *cr* dan *mr* dibuat sebuah grafik untuk memudahkan dalam mengetahui kombinasi *cr* dan *mr* yang mana yang merupakan kombinasi paling optimal berdasarkan pengujian yang sudah dilakukan. Grafik hasil pengujian kombinasi *cr* dan *mr* terdapat pada Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Kombinasi Cr dan Mr

Dari Gambar 6.3 dapat dilihat bagaimana rata-rata *fitness* dari setiap kombinasi *cr* dan *mr*. Pada kombinasi *cr* 0 dan *mr* 0,1 rata-rata *fitness* hanya sebesar 0,0280479 kemudian terjadi peningkatan yang signifikan pada *cr* 0,9 dan *mr* 0,1 dengan rata-rata nilai *fitness* sebesar 0,8832171. Kemudian terjadi kenaikan rata-rata *fitness* pada kombinasi *cr* 0,8 dan *mr* 0,2 dengan rata-rata *fitness* 0,8939737. Pada kombinasi *cr* dan *mr* selanjutnya tidak mengalami kenaikan atau penurunan yang signifikan, kombinasi *cr* 0,8 *mr* 0,2 sampai *cr* 0,2 dan *mr* 0,8 rata-rata *fitness* tidak mengalami perubahan yang berarti sehingga dapat dikatakan konvergen. Terjadi sedikit penurunan rata-rata *fitness* pada *cr* 0,1 dan *mr* 0,9 dengan nilai 0,8534256 kemudian kembali naik pada *cr* 1 dan *mr* 0 dengan rata-rata *fitness* sebesar 0,8832913 yang dapat dilihat tidak lebih besar dibandingkan *cr* 0,8 dan *m mr* 0,2. Dari pengujian kombinasi *cr* dan *mr* didapatkan *cr* 0,8 dan *mr* 0,2 sebagai kombinasi *cr* dan *mr* yang optimal.

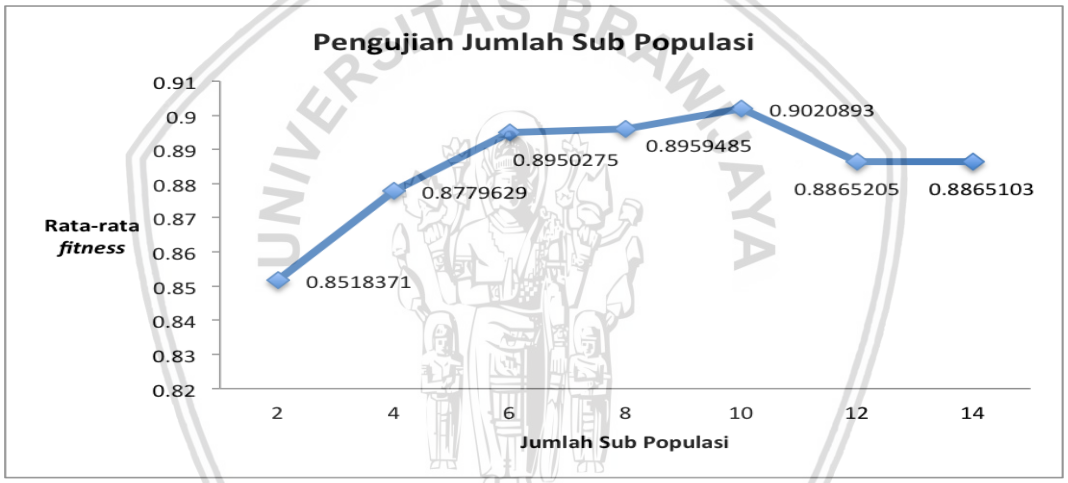
6.4 Pengujian dan Analisis Hasil Pengujian Jumlah Sub Populasi

Pengujian terakhir adalah menguji jumlah sub populasi. Pengujian ini dilakukan untuk mengetahui apakah jumlah sub populasi mempengaruhi hasil dari penentuan portofolio saham optimal. Jumlah ukuran populasi yang digunakan adalah 80, jumlah generasi 150 dan *cr* 0,8 *mr* 0,2. Hasil pengujian jumlah sub populasi tertera pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian Jumlah Sub Populasi

Jumlah Sub Populasi	Nilai *Fitness* Percobaan Populasi ke-										Rata - rata *fitness*
	1	2	3	4	5	6	7	8	9	10	
2	0,884	0,893	0,890	0,864	0,884	0,887	0,816	0,902	0,613	0,881	0,8518371
4	0,891	0,896	0,890	0,859	0,867	0,872	0,882	0,889	0,869	0,861	0,8779629
6	0,885	0,903	0,905	0,899	0,903	0,879	0,881	0,899	0,886	0,903	0,8950275
8	0,885	0,894	0,901	0,886	0,903	0,903	0,904	0,882	0,896	0,901	0,8959485
10	0,903	0,899	0,903	0,890	0,902	0,905	0,906	0,902	0,902	0,903	0,9020893
12	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,8865205
14	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,886	0,8865103

Dari Tabel 6.4 selanjutnya dibuat sebuah grafik hasil pengujian jumlah sub populasi yang dapat memudahkan untuk mengetahui pada sub populasi keberapa yang merupakan sub populasi optimal. Grafik hasil pengujian jumlah sub populasi terdapat pada Gambar 6.4.



Gambar 6.4 Hasil Pengujian Jumlah Sub Populasi

Pada Gambar 6.4 rata-rata *fitness* cenderung terus mengalami kenaikan, diawali dengan jumlah sub populasi sebanyak 2 dengan rata-rata *fitness* 0,8518371 kemudian rata-rata *fitness* kembali mengalami kenaikan pada 4 sub populasi dengan rata-rata *fitness* sebesar 0,8779629 dan masih mengalami kenaikan rata-rata *fitness* pada sub populasi berikutnya. Saat sub populasi berjumlah 6 rata-rata *fitness*nya adalah 0,8950275 dan tidak mengalami perubahan yang signifikan pada 8 sub populasi dengan rata-rata *fitness* 0,8959485. Sub populasi terakhir yang diuji berjumlah 10 memiliki rata-rata *fitness* terbesar yaitu 0,9020893. Rata-rata *fitness* mengalami penurunan saat jumlah sub populasi 12 dengan rata-rata *fitness* 0,8865205. Pada 14 sub populasi rata-rata *fitness* tidak mengalami banyak perubahan dengan rata-rata *fitness* 0,8865103. Berdasarkan hasil tersebut dapat diketahui bahwa 10 sub populasi adalah jumlah sub populasi yang optimal.

6.5 Analisis Hasil Pengujian Parameter

Berdasarkan pengujian yang sudah dilakukan yaitu pengujian jumlah ukuran populasi, pengujian jumlah generasi, pengujian kombinasi *crossover rate* dan *mutation rate* serta pengujian jumlah sub populasi didapatkan hasil optimal mengacu pada besarnya rata-rata *fitness* pada setiap pengujian parameter tersebut.

Pada pengujian jumlah ukuran populasi didapatkan hasil optimal yaitu 80 populasi, kemudian hasil ini digunakan sebagai parameter dalam menguji generasi sehingga didapatkan hasil optimal yaitu 150 generasi. Setelah populasi dan generasi optimal didapatkan maka dilakukan pengujian kombinasi *crossover rate* dan *mutation rate* dengan menggunakan parameter jumlah ukuran populasi dan jumlah generasi optimal dan didapatkan kombinasi *crossover rate* dan *mutation rate* yang optimal adalah 0,8 untuk *crossover rate* dan 0,2 untuk *mutation rate*. Setelah tiga parameter optimal didapatkan, pengujian selanjutnya adalah menguji jumlah sub populasi menggunakan jumlah ukuran populasi, jumlah generasi, serta kombinasi *crossover rate* dan *mutation rate* yang optimal dan didapatkan hasil jumlah sub populasi optimal sebesar 10.

Parameter yang digunakan pada algoritme genetika terdistribusi mempengaruhi bagaimana hasil penentuan portofolio saham. Jumlah sub populasi adalah parameter yang paling mempengaruhi kemampuan eksplorasi dan variasi dari hasil penerapan algoritme genetika terdistribusi. Jumlah sub populasi yang optimal menjadikan kemampuan eksplorasi lebih baik dan keragaman variasi indvidiu tetap terjaga.

BAB 7 KESIMPULAN DAN SARAN

Bab ini menjelaskan bagaimana kesimpulan dari penelitian yang sudah dilakukan untuk menentukan portofolio saham optimal menggunakan algortime genetika terdistribusi serta saran untuk penelitian berikutnya.

7.1 Kesimpulan

Berdasarkan pengujian yang sudah dilakukan, penulis dapat menyimpulkan bagaimana pengaruh parameter algoritme genetika terdistribusi dan berapa nilai parameter optimal untuk penentuan portofolio saham optimal sebagai berikut:

1. Parameter yang paling berpengaruh pada penerapan algoritme genetika terdistribusi untuk penentuan portofolio saham optimal adalah jumlah sub populasi karena terjadinya interaksi antar sub populasi yang dapat menjaga keragaman variasi individu.
2. Jumlah ukuran populasi dan generasi berpengaruh terhadap area pencarian solusi. Semakin banyak populasi atau generasi semakin luas area pencarian solusi namun waktu komputasi yang dibutuhkan semakin lama. Dari hasil pengujian, jumlah ukuran populasi optimal sebanyak 80 populasi dengan rata-rata *fitness* sebesar 0,8943178 sedangkan jumlah generasi optimal yaitu 150 generasi dengan rata-rata *fitness* sebesar 0,8902097.
3. Kombinasi *cr* dan *mr* yang optimal adalah 0,8 untuk *cr* dan 0,2 untuk *mr* dengan rata-rata nilai *fitness* sebesar 0,8939737. Sedangkan jumlah sub populasi optimal adalah 10 dengan rata-rata *fitness* 0,9020893.
4. Nilai parameter optimal dari hasil pengujian untuk jumlah ukuran populasi sebesar 80, jumlah generasi sebesar 150, *crossover rate* 0,8 *mutation rate* 0,2 dan jumlah sub populasi 10.

7.2 Saran

Berdasarkan hasil dan kesimpulan, algoritme genetika terdistribusi sudah cukup baik dalam menentukan portofolio saham optimal namun, untuk penelitian selanjutnya dapat dikembangkan dengan menggunakan operator *crossover*, mutasi dan seleksi yang berbeda. Serta menggunakan mekanisme migrasi dan strategi migrasi yang lain untuk mendapatkan hasil yang lebih baik lagi.

DAFTAR PUSTAKA

Tandelilin, E. 2010. *Portofolio dan Investasi; Teori dan Aplikasi* ed. 1. Yogyakarta: Kanisius.

Samsul, M. 2015. *Pasar Modal dan Manajemen Portofolio* ed. 2. Erlangga.

Sofariah, A., Saepudin, D., & Umbara, R. F. 2016. Optimasi Portofolio Saham Dengan Memperhitungkan Biaya Transaksi Menggunakan Algoritma Genetika Multi-Objective. *e-Proceeding of Engineering , 3* (1), 1156.

Wahyuni, R., Mahmudy, W. F., & Setiawan, B. D. 2017. Penentuan Portofolio Saham Optimal Menggunakan Algoritma Genetika. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer , 1* (1), 63-68.

Fiarni, C., & Bastiyan. 2013. Sistem Rekomendasi Portofolio Investasi Algoritma Genetika. *Seminar Nasional Sistem Informasi, 2-4 Desember* .

Putri, I. I., Saepuddin, D. D., & Setiawan, E. B. 2014. Optimasi Portofolio Saham Menggunakan Algortima Genetika. *Universitas Telkom* .

Mahmudy, W. F. 2015. *Dasar-Dasar Algoritma Evolusi.* Univeritas Brawijaya, Malang: Program Teknologi Informasi dan Ilmu Komputer.

Sasmito, A. A., Cholissodin, I., & Sutrisno. 2018. Penerapan Parallel Genetic Algorithm untuk Optimasi Penyusunan Bahan Makanan Keluarga Penderita Hiperkolesterolemia . *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer , 2* (8), 2343-2351.

Kasmir. 2014. *Bank Dan Lembaga Keuangan Lainnya* ed. 14. Jakarta: Raja Grafindo Persada.

Tandelilin, E. 2010. *Portofolio dan Investasi* ed. 1. Yogyakarta: Kanisius.

Bursa Efek Indonesia. 2008. *Panduan Pemodal.* Jakarta.

Fahmi, I. 2012. *Manajemen Investasi: Teori dan Soal Tanya Jawab.* Jakarta: Salemba Empat.

Zubir, Z. 2011. *Manajemen Portofolio Penerapannya Dalam Investasi Saham.* Jakarta: Salemba Empat.

Tandelilin, E. 2001. *Analisis Investasi dan Manajemen Portofolio* ed. 1. Yogyakarta: BPFE.

Staflburg, J., Martel, C. G., & Alexandrov, V. 2012. Parallel Genetic Algorithms for Stock Market Trading Rules. *Procedia Computer Science , 9*, 1306-1313.

Micalewicz, Z. 1996. *Genetic Algorithms + Data Structure = Evolution Programs.* Heidelberg: Springer-Verlag.

Gen, M., & Cheng, R. 1997. *Genetic Algorithms and Engineering Design.* New York: John Wiley & Sons, Inc.

Alba, E., & Troya, J. M. 1999. A Survey of Parallel Distributed Genetic Algorithms. *Complexity , 4* (4), 31-52.

Sharma, A., & Mehta, A. 2013. Review Paper of Various Selection Methods in Genetic Algorithm. *International Journal of Advanced Research in Computer Science adn Software Engineering , 3* (7), 1476-1479.

Darmadji, T., & Fakhruddin, H. M. 2012. *Pasar Modal di Indonesia* ed. 3. Jakarta: Salemba Empat.

Yussof, S., & Razali, R. A. 2012. An Investigation on the Effect of Migration Strategy on Parallel GA-Based Shortest Path Routing Algorithm. *Communication and Network* (4), 93-100.