PENJAJARAN SEKUEN JAMAK PROTEIN DENGAN ALGORITMA GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ceerinda Macro Jingga
NIM: 125150100111024

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PENJAJARAN SEKUEN JAMAK PROTEIN DENGAN ALGORITMA GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ceerinda Macro Jingga
NIM: 125150100111024

Skripsi ini telah diuji dan dinyatakan lulus pada
24 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Lailil Muflikhah, S.Kom, M.Sc	Marji, Drs., M.T
NIP: 19741113 200501 2 001	NIP: 19670801 199203 1 001

Mengetahui
Ketua Jurusan Informatika

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

PERNYATAAN ORISINILITAS

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, 17 Juni 2016

Ceerinda Macro Jingga

NIM: 125150100111024

KATA PENGANTAR

Rasa syukur penulis panjatkan kepada Allah SWT karena rahmat dan hidayahNya penulis dapat melaksanakan dan menyusun skripsi, untuk memenuhi sebagian persyaratan guna memperoleh gelar Sarjana Komputer dalam program studi Teknik Informatika.

Dalam Penyusunan skripsi ini penulis telah berusaha segenap kemampuan serta penyusunan skripsi ini tidak lepas dari adanya kerjasama dan bantuan dari berbagai pihak. Oleh karena itu pada kesempatan ini perkenankan penulis untuk mengucapkan terima kasih kepada yang terhormat :

1. Lailil Muflikhah, S.Kom., M.Sc. selaku dosen pebimbing skripsi pertama yang penuh kesabaran membimbing dan mengarahkan penulis untuk menyelesaikan penyusunan skripsi ini.
2. Marji, Drs., M.T. selaku dosen pembimbing skripsi kedua yang telah meluangkan waktu dan mengarahkan penulis dalam menyelesaikan penyusunan skripsi ini.
3. Agus Wahyu Widodo, S.T., M.Sc. selaku Ketua Prodi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Kepala Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Papa, Mama, Nila, Eldo, Mae, Mbah, keluarga Penulis & keluarga WB yang selalu memberikan doa, dukungan moral & materi kepada penulis.
7. Zaenab yang banyak membantu, meluangkan waktu, memberikan masukan, dan berdiskusi terhadap skripsi ini. Sahabat Penulis Gumilang, Safira, Yunita, Ganis, Kusuma, Adisa, Scelen, Dewi, Ari, Hamzah, Ida, Laras, Rifan, Antok, Upik, Wika, John, Ayip, Afa, Fasol, Mbah Rosyidin, Denny, Cakdi serta teman Kelas K dan teman kuliah penulis yang selalu memberikan informasi dan dukungan penuh tulus ikhlas kepada penulis.

Penulis menyadari bahwa skripsi ini belum sempurna, masih ada berbagai kekurangan yang luput dari perhatian saat mengerjakannya. Karena itu, penulis berharap adanya kritik dan saran yang akan sangat membantu perbaikan skripsi ini. Penulis juga berharap skripsi ini dapat berguna dan bermanfaat untuk penulis sendiri dan juga untuk pembaca.

Malang, 14 Maret 2016

Penulis

ceerinda@gmail.com

ABSTRAK

Penjajaran sekuen adalah proses dimana sekuen dibandingkan dengan mencari pola karakter yang paling umum dan berhubungan antar sekuen. Penjajaran sekuen jamak adalah proses penjajaran lebih dari dua sekuen yang merupakan dasar dari pencarian kemiripan pada database dan penjajaran banyak sekuen (*multiple sequence alignment*). Penjajaran sekuen memiliki masalah dalam hal waktu karena semakin panjang sebuah sekuen maka dibutuhkan waktu yang semakin banyak, selain itu sulit mengoptimasi atau sering disebut dengan masalah kombinatorial. Salah satu metode yang dapat menyelesaikan masalah ini adalah algoritma genetika. Algoritma genetika membentuk populasi dari solusi acak lalu menggunakan konsep seleksi alami, *crossover* dan mutasi yang digunakan untuk mengembangkan solusi tersebut. Algoritma genetika berhasil menghasilkan suatu solusi untuk masalah optimasi yang sulit. Keuntungan yang ditawarkan oleh algoritma ini adalah mengoperasikan beberapa solusi dengan mengkombinasikan solusi terakhir. Pada penelitan sebelumnya jika menggunakan algoritma genetika, menghasilkan hasil yang optimal dan lebih cepat dibandingkan dengan metode pe*mr*ogaman dinamis serta algoritma genetika ini inheren parallel sehingga dapat diimplementasikan sangat efisien pada komputer secara parallel. Pada kasus penjajaran sekuen tanpa *gap*s dan adanya *gap*s algortima genetika ini menghasilkan solusi yang baik. Bioinformatika adalah bidang ilmu biologi dan ilmu komputer. Data molekuler di dalam ilmu biologi sangat besar dan terus berkembang termasuk database pada protein. Protein adalah salah satu biomolekul yang sangat penting pada manusia yang berperan dalam proses seluler, fungsi enzim, antibody, *hormone* dan transport molekul. Pada penelitian ini, kami menampilkan bagaimana algoritma genetika dapat membantu dalam menyelesaikan penjajaran sekuen jamak protein. Data sekuen yang digunakan pada pengujian sebanyak 4 dan panjang sekuen maksimal 36. Penghitungan nilai *fitness* yang digunakan adalah *sum of pairs* pada *blosum62* serta skema affine yang terdiri dari *gap* open bernilai -11 dan *gap* extension bernilai -1. Pada penelitian ini algoritma genetika mampu menghasilkan hasil optimasi yang baik dari permasalahan yang komplek dari penjajaran sekuen jamak protein.

Kata kunci: affine, algoritma genetika, bioinformatika, *blosum62*, penjajaran sekuen jamak, protein

ABSTRACT

Sequence Alignment is the process whereby a sequence compared by looking for patterns of the most common character and inter-related sequences. Multiple sequence Alignment is the process of more than two sequences alignment that the basis of similarity search on the database. Multiple Sequence Alignment have a problem in terms of time because the longer a sequence so it takes more and more, but it is difficult to optimize or often called combinatorial problems. One method that can solve this problem is a genetic algorithm. Genetic algorithms form the population of random solutions and using the concept of natural selection, crossover and mutation are used to develop the solution. Genetic algorithms managed to produce a solution to a difficult optimization problems. The benefits offered by this algorithm is operating several solutions by combining the final solution. In the previous research has been generated if using a genetic algorithm, produce optimal results and faster than the dynamic programming method and the genetic algorithm is inherently parallel so that it can be implemented very efficiently on parallel computers. In the case of sequence alignment without gaps and the gaps genetic algorithms produce good solutions. Bioinformatics is a field the sciences and computer science. Molecular data in the science of biology very large and continued to grow including a database in proteins. Protein is one of biomolekul very important for human being who participate in the process cellular, function an enzyme, antibody, hormone and transport molecules. In this study, we show how a genetic algorithm can help in solving the multiple sequence alignment of protein. Data sequence used for testing is 4 and long sequence maximum 36 .The calculation of value fitness used is sum of pairs with blosum62 and the scheme of affine transformations consisting of gap open -11 and gap extension -1 .This research prove that algorithm genetics capable of produce results good optimize from trouble a complex of the multiple sequence alignment of protein .

Keywords: *affine, genetics algorithm, bioinformatics, blosum62, multiple sequence alignment, protein*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bagian utama skripsi terdiri dari beberapa komponen atau bab yang tersusun dengan alur yang logis. Pendahuluan merupakan komponen/bab pertama yang harus menjelaskan apa yang akan dikerjakan dalam skripsi dan mengapa ini perlu dikerjakan.

1.1 Latar belakang

Bidang ilmu biologi dan ilmu komputer bisa disebut dengan istilah bioinformatika (Xiong, 2006). Pada bidang ilmu ini membahas tentang pengolahan, analisis, prediksi, penyimpanan dan pencarian data biologi molekuler seperti DNA, RNA dan protein menggunakan computer. Data molekuler di dalam bidang ilmu biologi sangat besar dan terus berkembang seiring berjalannya waktu, seperti pada 5 Mei 2015, database protein di genbank mencapai 66.314.864 sekuen protein (Anon., 2015). Sekuen protein terdiri dari gen dan *encode* dari kode genetic (Sharma, 2009). Proses untuk mensejajarkan dua atau lebih sekuen untuk mencari kemiripan disebut dengan penjajaran sekuen. Jika hanya dua sekuen yang disejajarkan disebut penjajaran *pairwise* , dan kalau lebih dari dua sekuen disebut dengan penjajaran sekuen jamak. Penjajaran sekuen jamak dilakukan untuk mengetahui motif dari sekuen tersebut dan diasumsikan memiliki hubungan evolusi dimana diturunkan dari sekuen terdahulu sehingga bisa disimpulkan memiliki fungsi yang sama (Sharma, 2009).

Penjajaran sekuen jamak membutuhkan metode yang canggih dibandingkan dengan penjajaran sekuen *pairwise* karena penjajaran tiga sekuen lebih sulit, memakan banyak waktu dan komputasi yang komplek. Penjajaran sekuen jamak menggunakan metode heuristik. Algoritma komputasi yang efisien dapat digunakan untuk menghasilkan hasil yang lebih baik (Agarwal, 2013). Penjajaran sekuen jamak mengalami masalah juga dalam hal optimasi atau sering disebut dengan masalah kombinatorial. Salah satu metode yang dapat menyelesaikan permasalahan diatas dengan algoritma genetika.

Algoritma genetika membentuk populasi dari solusi acak lalu menggunakan konsep seleksi alami, *crossover* dan mutasi serta fungsi *fitness* dengan perhitungan matrik menggunakan tool PAM-250 dan BLOSUM-45 yang digunakan untuk mengembangkan solusi tersebut. *PAM (Percent Accepted Mutations)* adalah matrik yang digunakan untuk memberikan nilai penjajaran sekuen protein dengan kekerabatan terdekat atau urutan terdekat. Sedangkan *BLOSUM (Block Subtitution Matrix)* adalah matrik yang digunakan untuk memberikan nilai penjajaran sekuen protein dengan evolusi berbeda (Sharma, 2009).

Algoritma genetika berhasil menghasilkan suatu solusi untuk masalah optimasi yang sulit. Keuntungan yang ditawarkan oleh algoritma ini adalah mengoperasikan beberapa solusi dengan mengkombinasikan solusi terakhir. Pada penelitan sebelumnya jika menggunakan algoritma genetika, menghasilkan hasil yang optimal dan lebih cepat dibandingkan dengan metode pemrogaman dinamis serta

algoritma genetika ini inheren parallel sehingga dapat diimplementasikan sangat efisien pada komputer secara parallel.Pada kasus penjajaran sekuen tanpa *gap*s dan adanya *gap*s algortima genetika ini menghasilkan solusi yang baik dengan hanya menggunakan sedikit sumber daya *computer* (Agarwal, 2013). Berdasarkan kebutuhan akan optimasi dari penjajaran sekuen jamak protein menggunakan metode algoritma genetika diperlukan suatu perangkat lunak bantu yang akan dikembangkan dalam skripsi ini.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah diuraikan, dapat dirumuskan permasalahan yang akan dibahas pada penelitian ini. Rumusan permasalahan tersebut antara lain:

1. Bagaimana menerapkan metode algoritma genetika ke dalam sistem penjajaran sekuen jamak protein ?
2. Bagaimana hasil solusi optimal dari penjajaran sekuen jamak protein menggunakan metode algoritma genetika ?

1.3 Tujuan

Tujuan dari penelitian ini disusun untuk memberikan arah pada capaian penelitian. Berdasarkan rumusan masalah yang disusun tujuan penelitian, antara lain:

1. Menerapkan metode algoritma genetika ke dalam sistem penjajaran sekuen jamak protein.
2. Mengetahui hasil optimasi penjajaran sekuen jamak protein menggunakan metode algoritma genetika.
3. Mengetahui sekuen terbaik dari penjajaran sekuen jamak protein menggunakan metode algoritma genetika.

1.4 Manfaat

Penelitian ini diharapkan mempunyai manfaat baik bagi penulis maupun pembaca. Penulis dan pembaca dapat menjadikan penelitian ini sebagai media pembelajaran atau pengimplementasian ilmu pengetahuan dalam bidang bioinformatika khususnya algoritma genetika. Selain itu penulis juga mendapatkan pengetahuan bagaimana cara kerja algoritma genetika dalam pencarian optimasi penjajaran sekuen jamak protein.

Manfaat lain dengan dilakukannya penelitian ini, dapat dijadikan salah satu acuan dalam penelitian berikutnya terhadap penggunaan metode algoritma genetika dalam pencarian optimasi penjajaran sekuen jamak protein. Serta dari sisi pengguna aplikasi akan lebih mudah dalam pencarian optimasi penjajaran sekuen jamak protein secara otomatis.

1.5 Batasan masalah

Batasan masalah pada penulisan laporan skripsi ini diharapkan menjadi pembatas bahasan dan kajian masalah agar tidak terlalu luas serta sesuai dengan tujuan penelitian. Adapun batasan masalah terdiri dari :

1. Untuk pengujian sekuen dibatasi sejumlah 4 data sekuen dengan panjang sekuen terpanjang 36.
2. Data yang digunakan diperoleh dari NCBI dengan memilih sekuen protein pada manusia.

1.6 Sistematika pembahasan

Pembuatan laporan penelitian ini dilakukan dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Berisi tentang latar belakang, rumusan masalah, pembatasan masalah, tujuan dan manfaat penelitian, serta sistematika penulisan dalam penelitian skripsi yang berjudul penjajaran sekuen protein jamak protein dengan algoritma genetika.

BAB II LANDASAN KEPUSTAKAAN

Berisi penjelasan tentang penelitian yang sudah dilakukan sebelumnya dan berbagai teori yang diperlukan untuk merancang sistem penjajaran sekuen protein jamak dengan algoritma genetika.

BAB III METODOLOGI

Dalam bab ini membahas serangkaian langkah yang dilakukan peneliti untuk menyelesaikan permasalahan dalam pembuatan sistem penjajaran sekuen protein dengan algoritma genetika.

BAB IV PERANCANGAN

Bab ini menjelaskan tentang perancangan sistem penjajaran sekuen jamak protein dengan algoritma genetika mulai dari rancangan model, rancangan antarmuka sistem dan rancangan pengujian.

BAB V IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem yang telah dirancang pada bab sebelumnya mulai dari implementasi database sampai dengan implementasi antarmuka sistem penjajaran sekuen jamak protein dengan algoritma genetika.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan tentang proses dan hasil pengujian terhadap sistem penjajaran sekuen jamak protein dengan algoritma genetika yang telah diimplementasikan serta analisis dari pengujian tersebut.

BAB VII PENUTUP

Bab ini berisi kesimpulan dari pengembangan dan pengujian serta saran untuk penegmbangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan penjajaran sekuen protein dengan algoritma genetika, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian.

2.1 Kajian Pustaka

Pada penelitian Pertama dengan judul *"Alignment of Multiple DNA Sequences by Using Improved GA operators"* telah dilakukan oleh Manish Kumar. Pada penelitian ini telah dibuktikan bahwa panjang sekuen DNA mempengaruhi tingkat akurasi dari penjajaran. Panjang sekuen DNA meningkatkan kualitas sekuen namun persentase turun. Implementasi dari penelitian ini menggunakan *crossover rate* 0.8 dan *mutation rate* 0.2. Dilakukan penghitungan *fitness* 3 generasi dengan ukuran populasi yang semakin meningkat menghasilkan *success rate* yang semakin menurun yaitu saat jumlah populasi 10 *success rate* 22% dan saat jumlah populasi 22 *success rate* 8% serta saat jumlah populasi 10 *score fitness* generasi pertama, kedua dan ketiga berturut-turut -494, -437, -401. Bisa disimpulkan bahwa nilai *fitness* meningkat dengan peningkatan jumlah iterasi namun dapat diketahui pula bahwa peningkatan jumlah ukuran populasi tidak membuat *success rate* juga meningkat (Kumar, 2015).

Pada penelitian kedua dengan judul *"Alignment of Multiple Sequence using GA method"* telah dilakukan oleh Pankaj Agarwal. Penelitian ini mengusulkan beberapa operator genetik untuk memperoleh penjajaran sekuen jamak yang lebih baik. Dataset yang digunakan untuk analisis adalah *DNA families* dari *Canis_familiaris.* Perangkat lunak ini menjamin bahwa populasi dari generasi berikutnya akan menghasilkan *fitness* yang lebih baik. Pada fungsi *fitness* untuk penghitungan matrik digunakan *tools* PAM-250 dan BLOSUM-45. Metode evaluasi *fitness* berperan penting dalam kinerja algoritma evolusi dan st*rate*gi yang paling sering digunakan adalah *Sum-Of-Pair*. Pada metode ini salah satu dari ketiga kejadian ini yaitu *match*, *mismatch*, atau *gap* akan terjadi pada setiap lokasi pada penjajaran sekuen. Pada penelitian ini menghasilkan setiap penjajaran cenderung membaik, hal ini ditunjukkan dengan meningkatnya nilai *fitness* dengan peningkatan jumlah iterasi (Agarwal, 2013).

Pada penelitian pertama dapat disimpulkan bahwa berdasarkan fakta penelitan di masa depan diharapkan lebih fokus untuk meningkatkan *fitness success rate* sesuai dengan meningkatnya ukuran jumlah populasi sehingga metode yang diusulkan dapat digunakan untuk sekuen yang panjang. Sedangkan untuk penelitian yang kedua menunjukkan peningkatan *fitness* dengan peningkatan jumlah iterasi dengan panjang sekuen yang berbeda-beda, maka dapat dikatakan bahwa pada penelitian kedua algoritma genetika dapat

menghasilkan hasil yang baik dengan populasi yang berbeda dan panjang sekuen yang berbeda-beda pula.

2.2 Bioinformatika

Bioinformatika terdiri dari algoritma, representasi sekuen, komputasi dan metode model matematika untuk menganalisi data biologi. Hal ini termasuk mempelajari tentang struktur dan fungsi dan evolusi dari gen, protein, dan semua genom. Dapat dikatakan bahwa bioinfomatika ini merupakan perkawinan antara teknologi dan biologi molekuler. Dua molekul penting dalam bioinformatika adalah protein dan *nucleic acid* (Sharma, 2009).

Bioinformatika adalah cabang ilmu pengetahuan antara ilmu computer dan ilmu biologi atau bisa dikatakan pula dengan gabungan antara biologi dan in*for*matika. Bioinformatika termasuk dalam teknologi yang menggunakan computer untuk penyimpanan, pengambilan, manipulasi dan distribusi informasi ang berhubungan dengan biologi makromolekul seperti DNA, RNA dan protein. Untuk menganalisis data genom secara matematis dan komplek maka diperlukan adanya komputer. Komputer juga berguna untuk menggali informasi dan membangun ilmu pengetahuan (Xiong, 2006).

Tujuan dari bioinformatika ini adalah untuk memahami kehidupan sel dan apa fungsinya pada level molekul dengan menganalisis sekuen molekul mentah dan struktur data. Dengan menganalisis data sekuen fungsi dari sebuah sel dapat mudah dipahami karena alur dari informasi genetika ditentukan oleh "central dogma" dari biologi dimana DNA ditranskripsi ke RNA yang diterjemahkan ke protein. Fungsi sel sebagian besar dilakukan oleh protein yang ditentukan oleh sekuen. Oleh Karena itu dengan menyelesaikan masalah fungsi menggunakan sekuen dan struktur adalah cara yang terbaik (Xiong, 2006).

Ruang lingkup bioinformatika adalah analisis struktur, analisis sekuens dan analisis fungsi. Untuk analisis struktur terdiri dari prediksi struktur asam amino, prediksi struktur protein, klasifikasi struktur protein, dan perbandingan struktur protein. Untuk analisis sekuens terdiri dari perbandingan genom, filogeni, prediksi gen & promoter, pencarian motif, pencarian sekuens database, dan penjajaran sekuens. Untuk analisis fungsi terdiri dari pemodelan jalan metabolisme, profil ekspresi gen, prediksi interaksi protein, dan prediksi lokalisasi protein subseluler (Xiong, 2006).

Sarana utama dalam bidang bioinformatika adalah berbagai perangkat lunak yang didukung oleh basis data yang tersedia pada *world wide web*. Bioinformatika merupakan disiplin ilmu yang sedang berkembang dengan pesat dan para ahli terus menyempurnakan program-program perangkat lunak yang kompleks untuk mencari, memilah-milah, menganalisis, memprediksi, dan menyimpan data-data biologi molekuler yang terus bertambah jumlahnya (Wargasetia, 2006).

2.3 Algoritma Genetika

Algoritma genetika adalah populasi berdasarkan algoritma berdasarkan dari konsep evolusi biologi dan genetika biologi. Algoritma genetika menggunakan kromosom untuk merepresentasikan sebuah solusi dan mulai dengan set populasi acak yang merepresentasikan gen di dalam kromosom. Populasi ini menghasilkan generasi lain mengikuti cara seleksi alam. Hal ini termasuk *crossover,*mutasi selama beberapa generasi dan seleksi alam. Biasanya nilai *fitness* dapat dievaluasi dengan beberapa ukuran utilitas atau manfaat yang sesuai. Individu dengan nilai *survival rate* tertinggi memiliki nilai keturunan yang relatif tinggi juga dan peningkatan jumlah individu di setiap populasi dapat dihasilkan dengan gen dari individu yang sangat fir. Perbedaan *ancestors* dapat memproduksi keturunan yang super (Wu, et al., 2008).

Algoritma genetika adalah teknik optimasi komputasi yang diadaptasi dari fenomena evolusi alam spesies untuk menyelesaikan masalah optimasi. Masalah data dapat diketahui pada kode kromosom dan populasi kromosom tersebut diproses oleh operator genetic seperti *crossover* dan mutasi dengan tujuan evolusi. Operator genetik mengirimkan fitur penting untuk generasi berikutnya selama proses evolusi (Bilolikar, et al., 2012).

Algoritma genetika dengan seleksi baru dan skema *crossover* membantu untuk menghasilkan populasi terbaik sehingga penjajaran yang leih baik dapat ditemukan. Alur proses algoritma genetika yakni :

2.3.1 Inisialisasi

Proses inisialisasi menciptakan populasi baru secara acak yang terdiri dari sejumlah *string* kromosom. Pada kasus penjajaran sekuen jamak ini inisialisasi dilakukan dengan memasukkan *gap* pada sekuen. Dalam tahap ini harus ditentukan ukuran populasi (*popsize*), nilai ini menyatakan banyaknya kromosom atau individu. Panjang setiap *string* kromosom dihitung pada tahap ini yang digunakan untuk proses selanjutnya yaitu representasi kromosom (Kumar, 2015).

2.3.2 Representasi Kromosom

Kromosom seharusnya memiliki informasi tentang solusi yang diwakilinya. Cara yang paling sering digunakan dalam pengkodean adalah pengkodean ke dalam bentuk bilangan biner. Kemudian kromosom bisa terlihat seperti ini: setiap kromosom memiliki satu bilangan biner. Setiap bit pada bilangan ini dapat mewakili beberapa karakter dari solusi atau seluruh bilangan dapat diwakili oleh nomor. Ada banyak cara untuk pengkodean, hal ini tergantung pada masalah yang dihadapi. Seperti contohnya, satu dapat langsung di *encode* menjadi integer, kadang-kadang cara ini berguna untuk *encode* permutasi. Pada penelitian penjajaran sekuen jamak protein dengan algoritma genetika representasi kromosom diawali dengan mencari *maxlength. Maxlength* adalah panjang maksimal penjajaran sekuen dengan adanyan *gap* yang disisipkan. Untuk mencari *maxlength* adalah dengan mangalikan 1,2 dengan sekuen terpanjang dari penjajaran yang dilakukan (Agarwal, 2013).

2.3.3 *Crossover*

Reproduksi atau perkawinan silang atau bisa dikenal dengan sebutan *crossover* adalah proses dimana dua buah kromosom induk terpilih menghasilkan keturunan yang mana keturunan atau kromosom anak tersebut mengandung kombinasi gen-gen dari dua buah kromosom induk. Nilai tingkat *crossover* atau biasa disebut dengan *crossover rate* adalah nilai rasio *offspring* dari proses *crossover* terhadap ukuran populasi sehingga hasil *offspring* didapat sebanyak *crossover rate* x populasi (*popsize*). Proses ini adalah satu dari operator genetika yang paling penting. Salah satu metode dari *crossover* adalah *one-cut-point.* Sebagai contoh :

P_1 [0 1 1 0]
P_2 [1 1 0 0]
Dilakukan one-cut-point menjadi :
C_1 [0 1 0 0]
C_2 [1 1 1 0]

2.3.4 Mutasi

Proses mutasi dilakukan setelah proses *crossover* atau reproduksi. Proses ini dilakukan dengan cara mengubah satu gen atau lebih dari sebuah kromosom. Mutasi ini berguna agar individu yang ada di dalam populasi menjadi lebih bervariasi dan sangat berperan jika populasi awal hanya menghasilkan solusi yang sedikit. Dalam hal ini mutasi bisa dikatakan berfungsi untuk mempertahankan keanekaragaman individu dalam populasi (Kumar, 2015).

Nilai tingkat mutasi atau biasa disebut dengan *mutation rate* adalah nilai rasio *offspring* dari proses mutasi terhadap ukuran populasi sehingga hasil *offspring* didapat sebanyak *mutation rate* x populasi (*popsize*). Metode mutasi yang digunakan pada penelitian ini adalah *reciprocal exchange mutation*. *Reciprocal exchange mutation* adalah metode yang menghasilkan *offspring* atau anak dari hasil penukaran posisi gen *parent* (*exchange point* / XP) yang dipilih secara *random* (Mahmudy, 2013). Contoh sebagai berikut :

P_1 [1 3 4 2 5 7 6]

C_1 [1 5 4 2 3 7 6]

2.3.5 Fungsi *fitness*

Fungsi *fitness* digunakan untuk mengukur tingkat *fitness* suatu kromosom dalam populasi. Semakin besar nilai *fitness* pada suatu individu maka semakin besar kesmpatan kromosom tersebut terpilih untuk melakukan reproduksi lagi pada generasi berikutnya (Mahmudy, 2013).

Pada penelitian ini menggunakan penjajaran sekuen jamak dan fungsi penilaian menggunakan *sum of pairss* (SP). Sesuai dengan namanya *sum* yang artinya jumlah sehingga fungsi ini menjumlahkan semua kemungkinan pasangan dari sekuen pada penjajaran jamak berdasarkan matrik *blosum62*. *Blosum62* adalah matrik dua dimensi dengan nilai atau score yang mendeskripsikan

kemungkinan asam amino atau *nucleotide* tergantikan dengan yang lainnya dalam evolusi sekuen (Xiong, 2006). Contoh dari fungsi *sum of pairs* adalah :

Sekuen 1	G	K	N
Sekuen 2	T	R	N
Sekuen 3	S	H	E
Sum of pairs	-1 +	1 +	6 = 6

Cara penghitungan *sum of pairs* dengan melihat *blosum62* diatas adalah :

1. GT + GS + TS = (-2) + 0 + 1 = (-1)
2. KR + KH + RH = 2 + (-1) + 0 = 1
3. NN + NE +NE = 6 + 0 + 0 = 6 (Perlu diketahui disini bahwa NN tidak dijumlahkan lagi karena pada *sum of pairs* ini gabungan huruf yang sama hanya dihitung satu kali)

Pada pemberian *score* jika huruf atau karakter bertemu dengan *gap* maka nilai tidak dilihat pada matrik *blosum62* namun sudah ditentukan dengan nilai yang disebut dengan skema *affine* yaitu jika *gap* open adalah -11 dan *gap extension* -1. *Gap open* adalah *gap* pertama yang ditemukan pada kromosom atau *gap* pertama yang ditemukan setelah sebelumnya terdapat karakter atau huruf. *Gap extension* adalah *gap* yang ditemukan setelah *gap* pertama. *Gap* affine ini sering digunakan karena menghasilkan hasil yang optimal (Xiong, 2006).

2.3.6 Seleksi

Pada proses seleksi individu ini yang pertama kali dilakukan adalah dengan pencarian nilai *fitness* dan kemudian setiap dua individu yang terbaik dipilih yang selanjutnya akan dilakukan proses *crossover* atau reproduksi. Tujuan dari seleksi adalah memberikan kesempatan untuk melakukan reproduksi yang lebih besar bagi anggota populasi yang memiliki nilai *fitness* terbaik (Mawaddah & Mahmudy, 2006). Metode seleksi ada beberapa yaitu *roulette wheel*, *binary tournament*, dan *elitism*.

Metode seleksi *elitism* adalah metode yang digunakan dalam penelitian ini. Metode ini memilih *fitness* terbaik sejumlah *popsize* dari kumpulan individu di populasi (*parent*) dan anak (*offspring*). Individu terbaik sejumlah *popsize* terpilih dan akan masuk ke dalam generasi berikutnya. Pada metode ini menjamin individu terbaik lolos ke dalam generasi berikutnya (Mahmudy, 2013).

Metode *elitism* ini memiliki kelemahan yaitu tidak memberikan kesempatan individu dengan *fitness* terendah untuk melakukan generasi, karena ada beberapa kasus solusi optimum didapat dari hasil reproduksi individu dengan *fitness* rendah.

2.4 Protein

Protein adalah biomolekul yang sangat penting pada makhluk hidup. Protein melaksanakan sebagian besar proses seluler dan bertindak sebagai konstituen struktur, agen katalis, molekul sinyal dan mesin molekul di setiap sistem biologis (Kadam, et al., 2016). Komponen dari membran sel adalah protein sehingga dapat melakukan fungsi enzim, antibody, *hormone* dan transport molekul. Protein terdiri dari rantai asam amino yang mengandung unsur karbon, hidrogen, oksigen, dan nitrogen yang terikat satu sama lain dalam ikatan peptide. Unsur utama dalam protein adalah nitrogen karena terdapat di dalam semua protein namun tidak terdapat di dalam karbohidrat dan lemak. Unsur protein adalah 16% dari keseluruhan berat protein.

Protein adalah dasar dari komponen struktural utama dari jaringan hewan dan manusia. Percobaan kimia yang luas telah menunjukkan bahwa fungsi protein ditentukan oleh struktur. Molekul protein memiliki keanekaragaman asam amino yang membentuknya seperti mengandung fosfor, belerang dan ada juga yang mengandung besi dan tembaga. Maka dari itu struktur protein lebih kompleks dari pada struktur karbohidrat dan lemak atau dari struktur kimia organik. Protein terdiri dari 20 alfabet dari molekul yang lebih kecil dari asam amino. Jumlah molekul yang banyak penyusun protein menyebabkan banyak sekali kemungkinan susunan 3 dimensi (Primasoni, 2011).

Ada 20 asam amino yang terjadi pada protein. Nama-nama asam amino tersebut disingkat menjadi 3 kode huruf atau 1 kode huruf. Asam amino dan singkatan kodenya dijelaskan dalam table berikut.

Table 2.1 Asam Amino

Glycine	Gly	G	Tyrosine	Try	Y
Alanine	Ala	A	Methionine	Mer	M
Serine	Ser	S	Tryptophan	Trp	T
Threonine	Thr	T	Asparagine	Asn	A
Cysteine	Cys	C	Glutamine	Gln	G
Valine	Val	V	Histidine	His	H
Isoleucine	Ile	I	Aspartic Acid	Asp	A
Leucine	Leu	L	Glutamic Acid	Glu	G
Proline	Pro	P	Lycine	Lys	L
Phenylalanine	Phe	P	Arginine	Arg	A

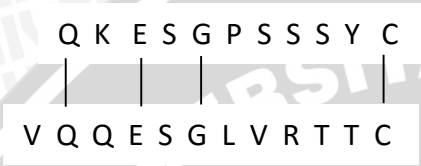
Sumber: (Xiong, 2006)

2.5 Penjajaran Sekuen Jamak

Proses penjajaran dua sekuen atau lebih untuk mencari kemiripan diantara kedua sekuen tersebut disebut penjajaran sekuen. Penjajaran dua sekuen disebut dengan penjajaran *pairwise*, sedangkan untuk lebih dari dua sekuen disebut

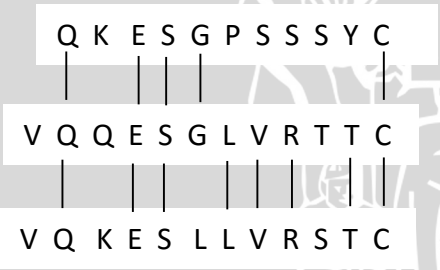
dengan penjajaran sekuen jamak (Sharma, 2009). Penjajaran dua sekuen yang memiliki panjang yang berbeda bisa saja terjadi. Pendekatan ini lebih tepat untuk menyelaraskan sekuen yang berbeda yang hanya berisi modul yang mirip yang disebut dengan domain atau motif. Perpanjangan alami dari penjajaran *pairwise* adalah penjajaran sekuen jamak. Keuntungan yang unik dari penjajaran sekuen jamak adalah dapat memperlihatkan informasi biologi yang lebih banyak dibandingkan dengan penjajaran *pairwise* karena penjajaran sekuen jamak ini mengidentifikasi pola dan motif sekuen pada semua keluarga sekuen yang jelas tidak hanya membandingkan dua sekuen saja (Xiong, 2006).

Contoh penjajaran sekuen *pairwise* :



Pada contoh diatas diketahui bahwa sekuen disejajarkan sehingga dapat diketahui bahwa huruf pada sekuen tersebut memiliki huruf yang sama.

Contoh penjajaran sekuen jamak :



Penjajaran sekuen jamak adalah implemen penting dalam analisis biologi. Penjajaran sekuen jamak merupakan implemen *preprocessing* untuk analisis keluarga protein berikutnya. Kemiripan diantara sekuen-sekuen mungkin merupakan evolusi, struktur, atau hubungan fungsional diantara sekuen tersebut. Kemiripan yang dikaitkan dengan keturunan dari nenek moyang yang sama disebut dengan homologi. Ketika dua sekuen atau lebih disejajarkan dan berhubungan dengan nenek moyang, dan ketika mismatch ditemukan dalam penjajaran, maka mismatch dapat dideteksi sebagai titik mutasi. *Gap* di dalam sekuen dapat dilihat sebagai indels (Sharma, 2009).

BAB 3 METODOLOGI

Bab ini akan menjelaskan tentang metode yang digunakan beserta tahap-tahap Penjajaran Sekuen Jamak Protein Dengan Algoritma Genetika. Tahapan metodologi penelitian diilustrasikan pada diagram blok pada Gambar 3.1

Gambar 3.1 Diagram Blok Metodologi Penelitian

3.1 Studi literatur

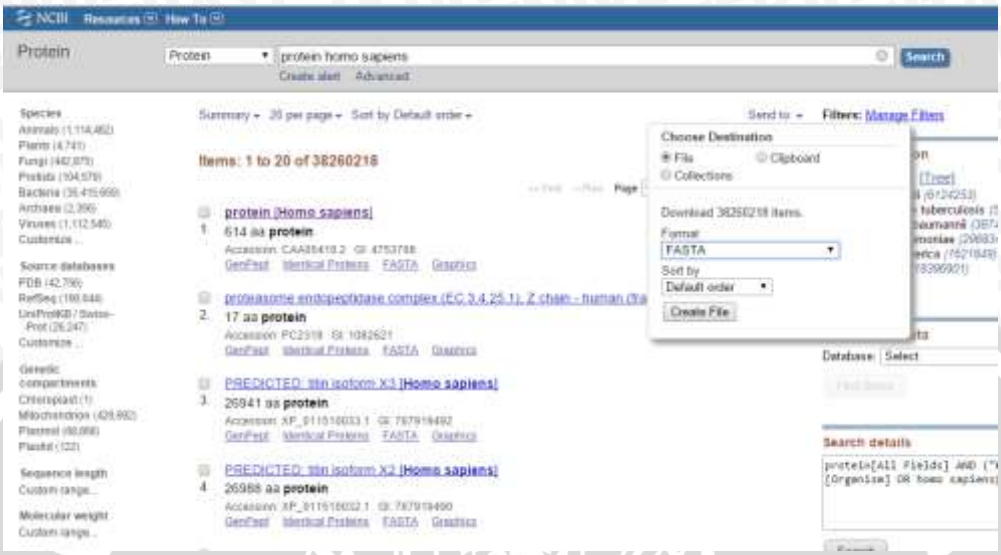
Studi literatur pada penelitian ini yaitu mempelajari literatur dari berbagai ilmu, informasi dan pustaka yang berhubungan dengan pembuatan perancangan sistem untuk penjajaran sekuen jamak protein dengan metode algoritma genetika. Buku, jurnal e-book, penelitian sebelumnya, internet dan bimbingan dari dosen pembimbing adalah sumber yang digunakan untuk literatur. Literatur tersebut diantaranya:

a. Algoritma Genetika
b. Bioinformatika
c. Protein
d. Pemrogaman dinamis
e. Penjajaran sekuen jamak

3.2 Pengumpulan Data

Data diperoleh dari situs resmi NCBI (National Center of Biotechnology Information) dengan alamat website http://www.ncbi.nlm.nih.gov. Pada kategori pencarian *all database*, kemudian masukkan “protein homo sapiens” pada kolom pencarian, kemudian klik *search*. Proses ini mencari sekuen protein. Akan muncul

sejumlah 38260218 data sekuen protein pada manusia. Kemudian klik *send to* akan muncul *choose destination* pilih *file,* pada isian fomat pilih FASTA lalu klik *create file,* maka otomatis akan ter-*download* 38260218 data sekuen protein pada manusia bernama *sequence.fasta*. Data dapat dibuka pada file editor seperti Notepad++.



Gambar 3.2 Cara Memperoleh Database dari NCBI

Sumber: (Anon., 2015)

3.3 Analisis Kebutuhan

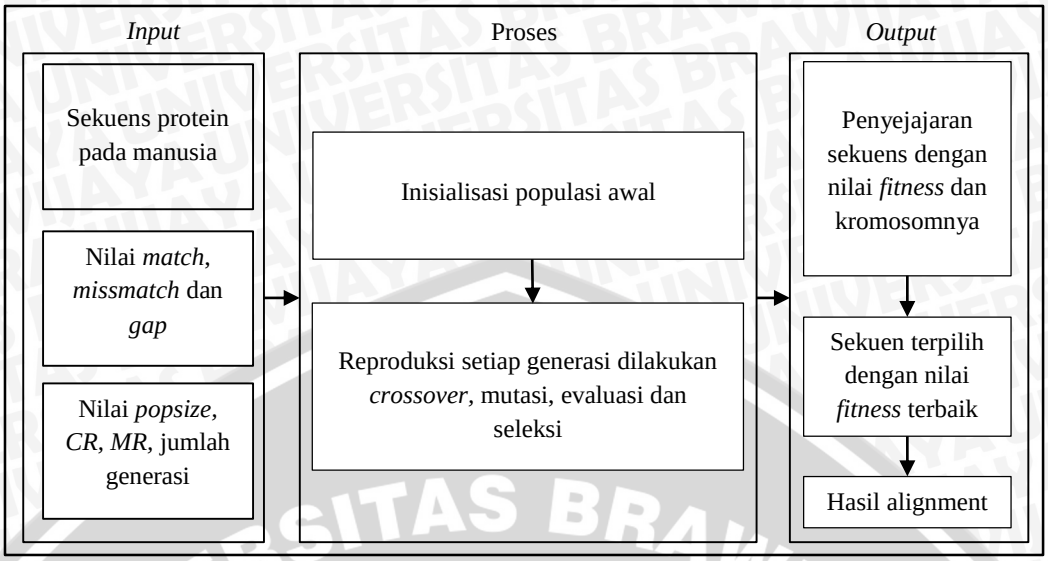
Kebutuhan apa saja yang dianalisis untuk membangun sistem penjajaran sekuen jamak protein dengan algoritma genetika dinamakan analisis kebutuhan. Tahap ini digunakan agar sistem dapat berjalan dengan optimal. Analisis kebutuhan terdiri dari deskripsi sistem, data yang digunakan disini data protein yang digunakan serta spesifikasi kebutuhan perangkat yang digunakan untuk membangun sistem.

3.4 Perancangan Sistem

Perancangan sistem dibuat untuk rancangan langkah kerja dari sistem secara menyeluruh guna mempermudah implementasi dan pengujian. Langkah kerja dalam sistem disesuaikan dengan arsitektur algoritma genetika.

3.4.1 Model Perancangan Sistem

Model perancangan sistem menjelaskan mengenai cara kerja sistem secara terstruktur mulai dari *input* yang dimasukkan hingga mendapatkan hasil. Diagram model perancangan sistem dapat dilihat pada Gambar 3.3.



Gambar 3.3 Model Perancangan Sistem

Pada Gambar 3.3 terdapat tiga proses yaitu :

- Input

  Pada proses ini inputan berisi beberapa sekuens protein pada manusia, nilai match, missmatch, dan *gap*. Untuk proses algoritma genetika kita membutuhkan inputan nilai *popsize*, *crossover rate*, *mutation rate*, dan jumlah generasi.

- Proses

  Pada proses ini terdapat proses perhitungan dengan menggunakan algoritma genetika yang menghasilkan penjajaran sekuen jamak protein pada manusia. Langkah-langkah dalam perhitungan ini adalah :

  1. Inisialisasi populasi awal : Menciptakan populasi awal secara acak yang berisi kumpulan kromosom yaitu pembangkitan populasi awal yang dilakukan dengan pengkodean permutasi berdasarkan sekuen protein.
     - *Preprocessing* : Menentukan *maxlength* kemudian membentuk kromosom dengan panjang *maxlength* dengan mengisikan gen dengan *gap*. Langkah terakhir sekuen di*encode* dari index array yang berisi *gap* dan dibatasi dengan *maxlength* untuk inisialisasi populasi.
  2. Reproduksi : Proses untuk menghasilkan keturunan (*offspring*) yang berasal dari induk kromosom yang dpilih secara acak.
     - Tukar Silang atau *Crossover* : Metode *one cut point* yaitu melibatkan dua buah individu atau gen dipilih secara acak dengan penentuan *crossover rate* lalu ditukar silang yang menghasilkan *offspring* atau anak.

- Mutasi : Metode Repricoral Exchange yaitu mengubah satu gen atau lebih dengan penentuan *mutation rate* terlebih dahulu lalu menghasilkan *offspring* atau anak.
- Evaluasi : Mengevaluasi nilai *fitness* dari setiap kromosom dengan table *blosum62*.
- Seleksi : Menggunakan metode *elitism* yaitu mempertahankan individu atau sekuen terbaik yang diperoleh dari suatu generasi ke generasi berikutnya.

- Output

  Pada proses ini terdapat hasil berupa penjajaran sekuen terpilih dengan nilai *fitness* terbaik.

3.5 Implementasi Sistem

Implementasi sistem adalah tahap dimana membangun sistem yang mengacu pada perancangan algoritma genetika dan menerapkan hal yang telah didapatkan dalam proses studi literature. Tahap-tahap yang ada dalam implementasi antara lain :

- Implementasi interface, menggunakan JAVA GUI
- Implementasi algoritma, melakukan perhitungan dengan metode algoritma genetika ke dalam bahasa pe*mr*ogaman JAVA
- Implementasi ini akan menghasilkan penjajaran sekuen jamak protein dengan algoritma genetika dengan *best alignment.*

3.6 Uji Coba Sistem

Uji coba sistem dilakukan untuk mengetahui apakah sistem berjalan dengan baik dan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan dapat berjalan dengan baik. Uji coba sistem juga berguna untuk mengetahui parameter algoritma yang optimal agar dapat menghasilkan nilai *fitness* yang optimal. Uji coba sistem dilakukan dengan cara membandingkan ukuran populasi, jumlah generasi, kombinasi *cr* dan *mr* dan yang terakhir adalah *factor* pengali. Dari keempat pengujian akan ditemukan nilai *fitness* yang optimal.

a. Pengujian ukuran populasi

Pada pengujian ini,ukuran populasi yang digunakan yaitu 100 sampai dengan 4000 dengan interval 100 serta masing-masing 10 kali pengujian. Nilai parameter *crossover rate* dan *mutation rate* diambil dari hasil pengujian kombinasi *crossover rate* dan *mutation rate*. Jumlah generasi yang digunakan adalah 100. Hasil dari setiap pengujian akan didapatkan rata-rata *fitness* untuk mengetahui solusi terbaik dari solusi yang optimal.

Tabel 3.1 Pengujian Ukuran Populasi atau *Popsize*

Ukuran Populasi atau *Popsize*	Rata-rata Nilai *Fitness*
100	
200	
300	
400	
500	
600	
700	
800	
900	
1000	
1100	
1200	
1300	
1400	
1500	
1600	
1700	
1800	
1900	
2000	
2100	
2200	
2300	
2400	
2500	
2600	
2700	
2800	
2900	
3000	
3100	
3200	
3300	
3400	
3500	
3600	
3700	
3800	
3900	
4000	

b. Pengujian ukuran generasi

Pada pengujian ini, jumlah generasi yang digunakan dalam pengujian ini yaitu 100 sampai dengan 2000 dengan interval 100 serta masing-masing 10 kali pengujian. Pada pengujian ini nilai parameter *crossover rate* (*cr*) dan *mutation rate* (*mr*) didapatkan dari hasil pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*). Jumlah populasi yang digunakan diambil dari hasil pengujian ukuran populasi.

Tabel 3.2 Pengujian Ukuran Generasi

Ukuran Generasi	Rata-rata Nilai *Fitness*
100	
200	
300	
400	
500	
600	
700	
800	
900	
1000	
1100	
1200	
1300	
1400	
1500	
1600	
1700	
1800	
1900	
2000	

c. Pengujian kombinasi *crossover rate* dan *mutation rate*

Penentuan kombinasi nilai *crossover rate* dan *mutation rate* sangat penting dilakukan untuk mendapatkan solusi yang mendekati optimum. Nilai dari kombinasi *cr* dan *mr* yang digunakan pada pengujian ini yaitu 0.1:0.9, 0.2:0.8, 0.3:0.7, 0.4:0.6, 0,5:0.5, 0.6:0.4, 0.7:0.3, 0.8:0.2 dan 0.9:0.1 dengan masing-masing 10 kali pengujian. Perbandingan nilai *cr* dan *mr* tersebut akan menghasilkan jumlah anak yang sama pada masing-masing parameter, sehingga proses pembandingan masing masing parameter kombinasi *cr* dan *mr* akan seimbang.

Tabel 3.3 Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Crossover Rate	*Mutation Rate*	Rata-rata Nilai *fitness*
0,1	0,9	
0,2	0,8	
0,3	0,7	
0,4	0,6	
0,5	0,5	
0,6	0,4	
0,7	0,3	
0,8	0,2	
0,9	0,1	

d. Pengujian factor pengali

Pada pengujian ini digunakan untuk mengetahui factor pengali berapa-kah dalam perhitungan maxlength yang menghasilkan nilai fitness terbaik. Parameter yang digunakan adalah generasi dan populasi atau popsize 100 serta nilai crossover rate dan mutation rate hasil pengujian. Pengujian ini dilakukan 10 kali dan diambil rata-rata.

Tabel 3.4 Pengujian Faktor Pengali

Faktor Pengali	Nilai Fitness
1	
1,1	
1,2	
1,3	
1,4	
1,5	
1,6	
1,7	
1,8	
1,9	
2	

3.7 Kesimpulan

Setelah semua tahap dari perancangan, implementasi dan pengujian metode yang diterapkan selesai dilakukan maka akan menghasilkan kesimpulan. Kesimpulan diambil berdasarkan pengujian dan analisis metode. Tahap terakhir dari penulisan skripsi ini adalah saran, sehingga jika terjadi kesalahan penulis bisa memperbaiki dan menjadi pertimbangan untuk penelitian selanjutnya.

BAB 4 PERANCANGAN

Bab perancangan sistem ini membahas mengenai analisis kebutuhan perangkat lunak, perancangan sistem, perhitungan manual dan perancangan antarmuka.

4.1 Analisis Kebutuhan

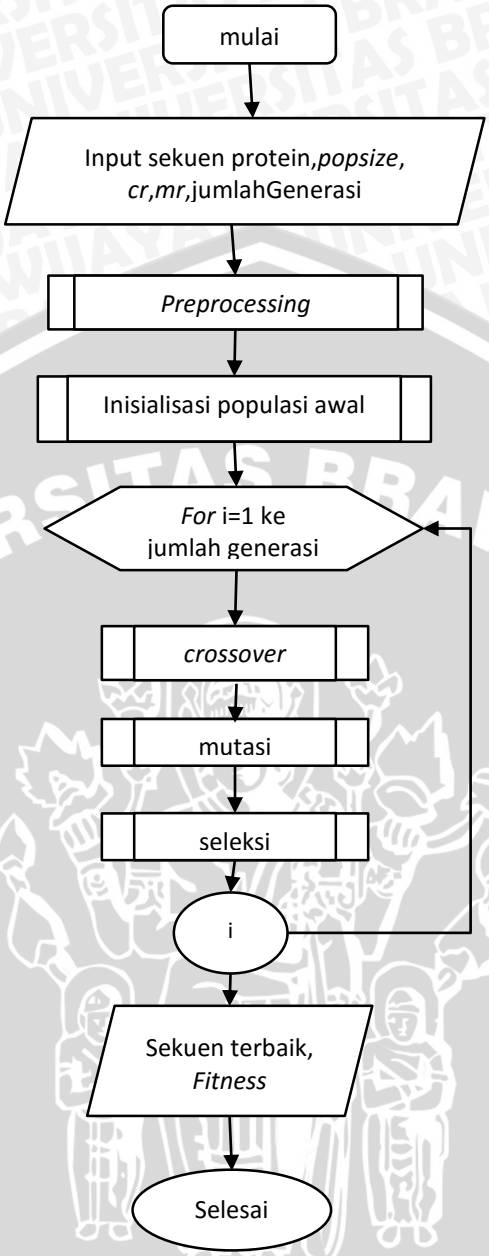
Analisis kebutuhan adalah tahapan yang meliputi analisis spesifikasi perangkat lunak sehingga mempermudah untuk memahami sistem yang telah dibuat.

Tabel 4.1 Model Perancangan Analisis Kebutuhan

Kebutuhan	Aktor	Proses
Sistem menyediakan antarmuka *form* input sekuen, *crossover rate*, *mutation rate*, *popsize*, jumlah generasi	User	Membentuk nilai Densitas
Sistem menyediakan antarmuka untuk melihat hasil kromosom terbaik dengan nilai *fitness* serta deskripsi dari sekuen tersebut	User	Menguji penghitungan sistem

4.2 Perancangan Sistem

Perancangan sistem meliputi flowchart untuk menggambarkan proses dari sistem ini yaitu proses pencarian *fitness* algoritma genetika. Proses dari algoritma genetika ini adalah inisialisasi populasi awal, evaluasi *fitness*, proses reproduksi yang terdiri dari *crossover* dan mutasi serta terakhir adalah proses seleksi. Proses ini ditunjukkan pada Gambar 4.1.



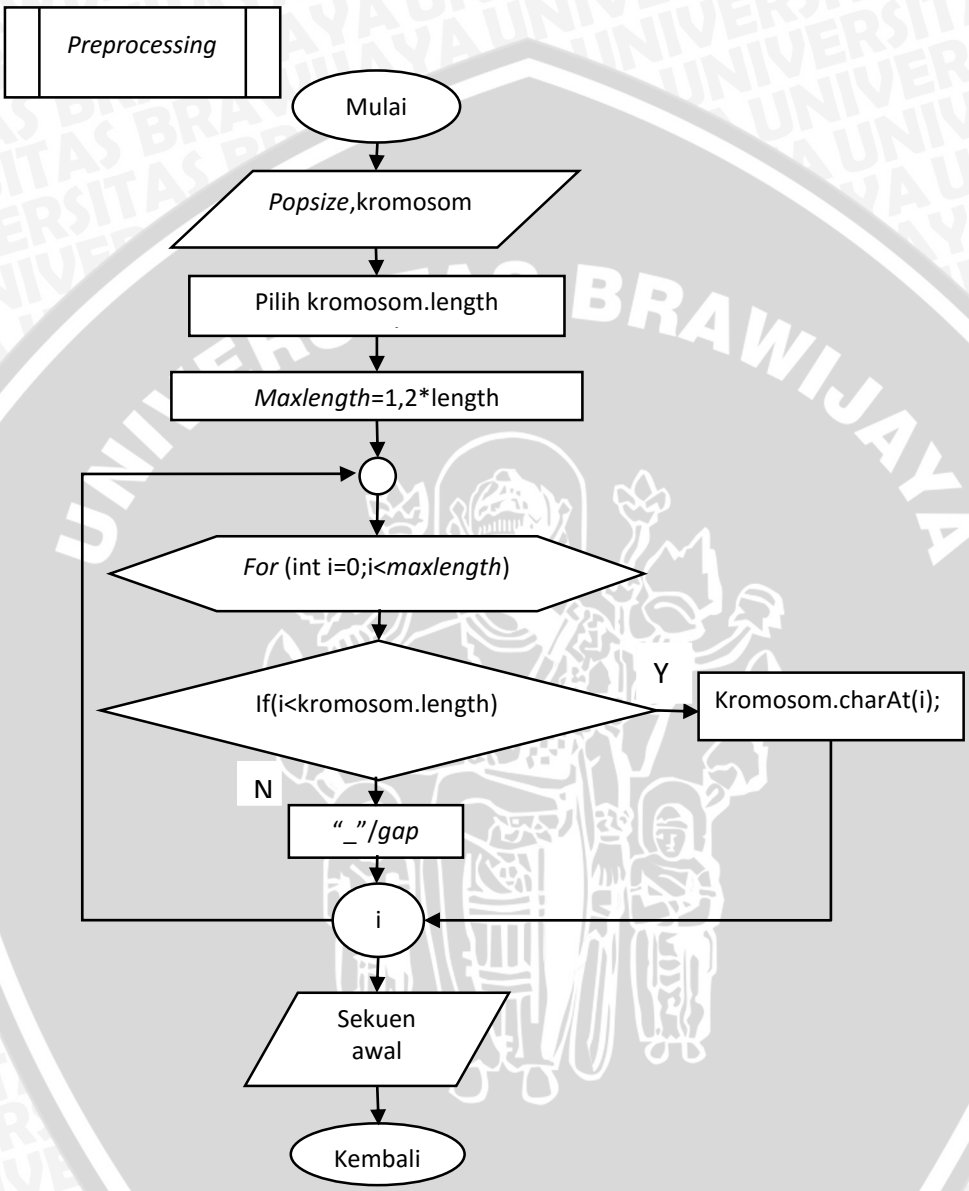
Gambar 4.1 Diagram Alir Perancangan

Awal proses dari sistem ini adalah dengan memasukkan atau menginputkan sekuen protein, *crossover rate, mutation rate, popsize,* dan jumlah generasi. Setelah itu dilakukan inisialisasi atau menciptakan populasi awal secara acak. Selanjutnya dilakukan proses *encode* sehingga bisa dilakukan proses reproduksi yaitu *crossover* dan mutasi. Hasil *crossover* dan mutasi menghasilkan individu baru (*offspring/child*), dan kemudian dilanjutkan proses *decode* atau seleksi individu. Seleksi ini mendapatkan generasi baru yang digunakan untuk mengulang proses dari proses reproduksi.

.

4.2.1 *Preprocessing*

Dalam penjajaran sekuen jamak ini inisialisasi populasi awal yaitu representasi kromosom diawali dengan proses *preprocessing* ini.



Gambar 4.2 Diagram Alir *Preprocessing*

Pada Gambar 4.3 dapat dilihat bahwa proses dimulai dengan inputan *popsize* dan kromosom atau pada kasus ini adalah sekuen. Setelah itu dilakukan proses penghitungan panjang kromosom dari semua sekuen. Dengan ditemukannya panjang sekuen maksimal dari beberapa panjang sekuen hasilnya diolah untuk menghasilkan *maxlength* dengan rumus panjang sekuen maksimal dikali dengan 1,2. Panjang sekuen diisi dengan karakter sisanya diisi dengan *gap*. Gambaran secara umum *preprocessing* dapat dilihat pada Gambar 4.2.

Sekuen 1	Karakter Pada Sekuen	Sisa index array diisi oleh *gap*
Sekuen 2		
Sekuen 3		
Index Array	Panjang index array adalah *maxlength* =1,2 * panjang sekuen maksimal	

Gambar 4.3 Gambaran Umum *Preprocessing*

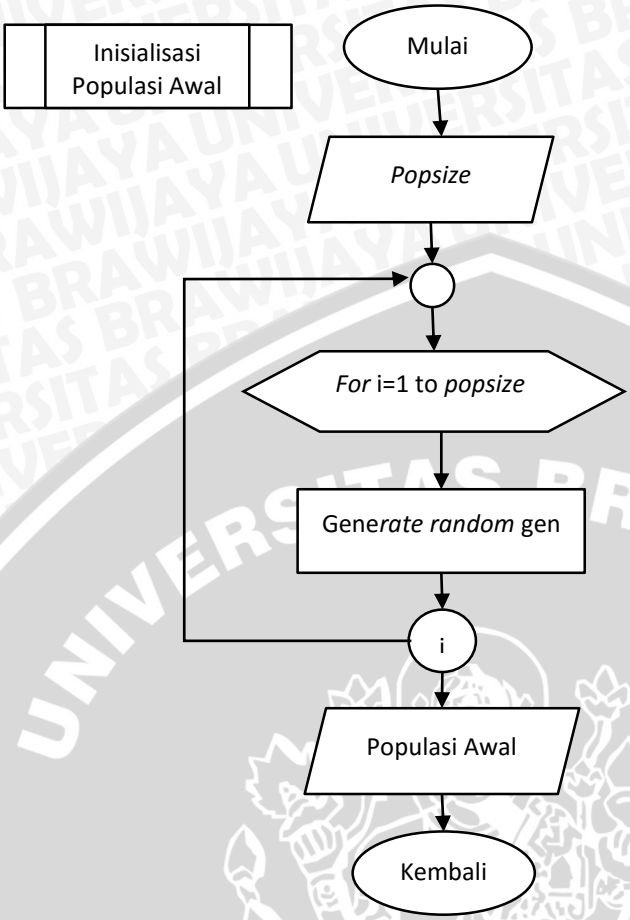
Pada Gambar 4.4 diatas dapat kita lihat bahwa *preprocessing* terdiri dari banyaknya sekuen yang disejajarkan dengan index array sejumlah *maxlength.* Isi dari sekuen-sekuen yang disejajarkan adalah berisi karakter pada sekuen tersebut dan sisanya diisi dengan *gap*. Index array yang berisi *gap* inilah yang akan diproses untuk representasi kromosom sehingga dapat dilanjutkan untuk insisalisasi populasi.

4.2.2 Inisialisasi Populasi

Sekuen di*encode* dari index array yang berisi *gap* dan dibatasi dengan *maxlength* untuk inisialisasi populasi. *Encode* disini merupakan representasi kromosom dari *preprocessing*. Gambaran umum representasi kromosom dapat dilihat pada Gambar 4.5.

P 1	Index array *gap* sekuen 1	*maxlength*	Index array *gap* sekuen 2	*maxlength*	Index array *gap* sekuen 3	*maxlength*
P 2	Index array *gap* acak 1 sekuen 1	*maxlength*	Index array *gap* acak 1 sekuen 2	*maxlength*	Index array *gap* acak 1 sekuen 3	*maxlength*
P 3	Index array *gap* acak 2 sekuen 1	*maxlength*	Index array *gap* acak 2 sekuen 2	*maxlength*	Index array *gap* acak 2 sekuen 3	*maxlength*
P 4	Index array *gap* acak 3 sekuen 1	*maxlength*	Index array *gap* acak 3 sekuen 2	*maxlength*	Index array *gap* acak 3 sekuen 3	*maxlength*
P 5	Index array *gap* acak 4 sekuen 1	*maxlength*	Index array *gap* acak 4 sekuen 2	*maxlength*	Index array *gap* acak 4 sekuen 3	*maxlength*

Gambar 4.4 Gambaran Umum Representasi Kromosom

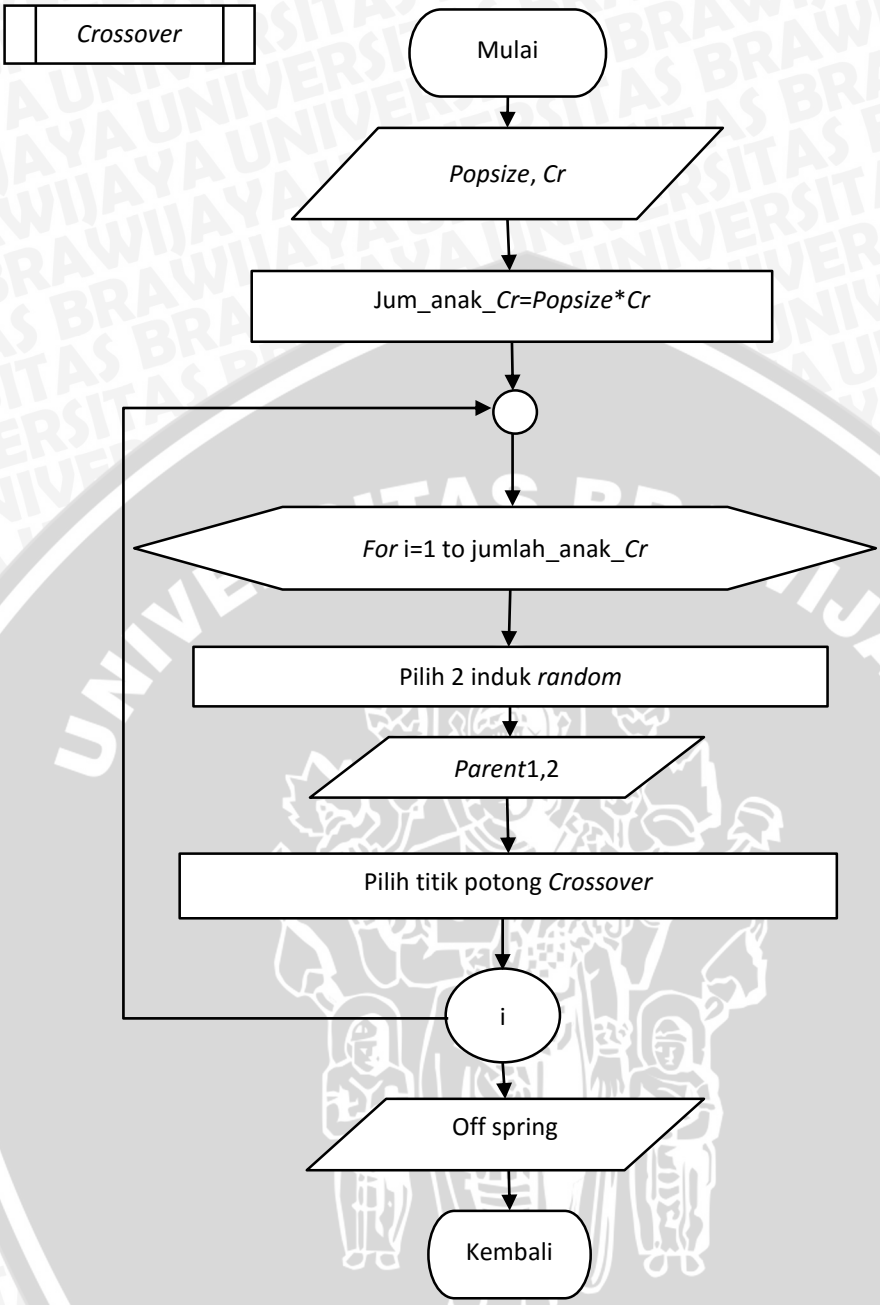


Gambar 4.5 Diagram Alir Inisialisasi Populasi

Pada Gambar 4.6 diatas dapat diketahui bahwa inisialisasi populasi awal dimulai dari inputan jumlah *popsize* yang kemudian dilakukan perulangan sebanyak jumlah *popsize* yaitu mengene*rate random* gen.

4.2.3 Tukar Silang (*Crossover*)

Di dalam proses reproduksi terdapat proses pertama yaitu *crossover* dan metode yang digunakan adalah one-cut point. Proses pada metode one-cut point ditunjukkan pada gambar .



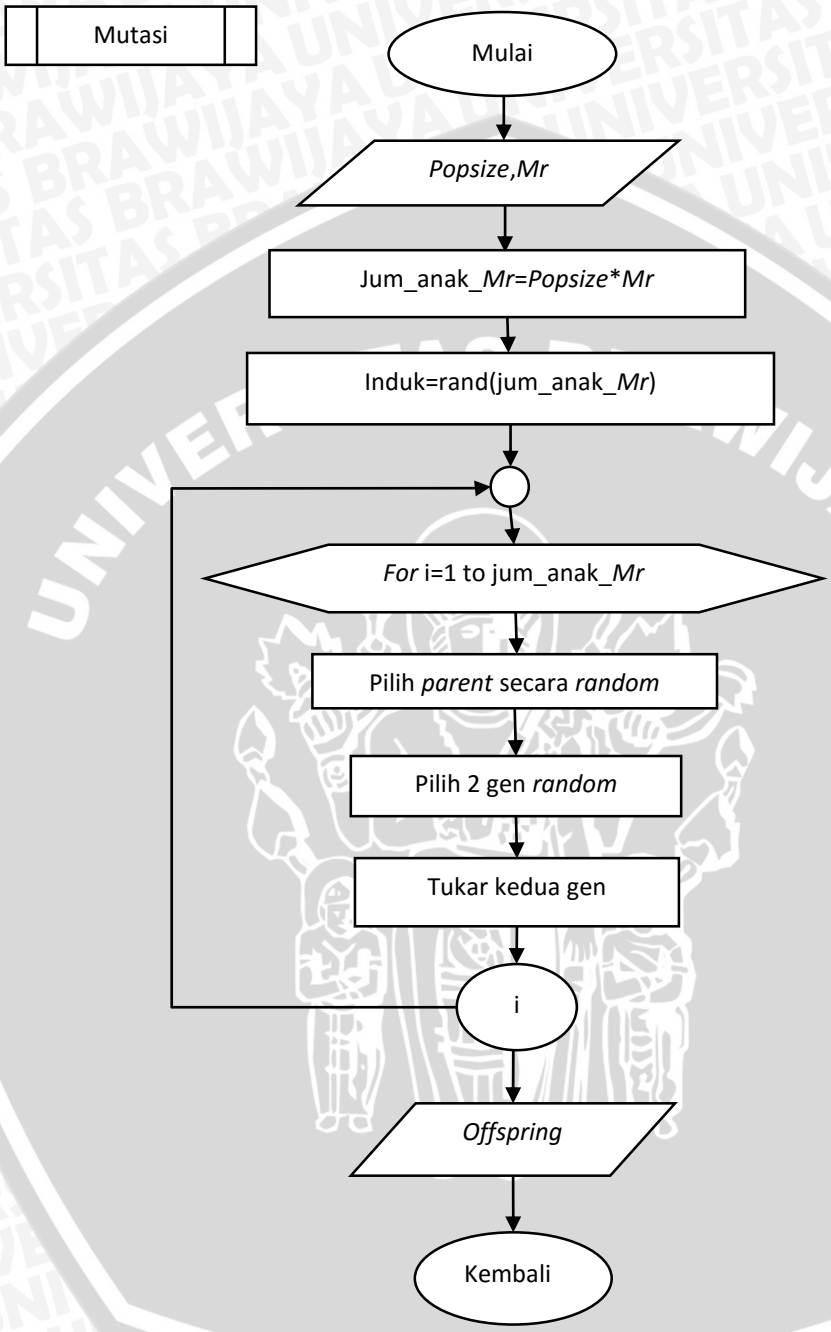
Gambar 4.6 Diagram Alir *Crossover*

Pada Gambar 4.7 diatas dapat diketahui bahwa prosses *crossover* dimulai dengan inputan jumlah *popsize* dan *crossover rate*. Lalu sistem akan mencari jumlah anak dengan rumus *crossover rate* dikali dengan populasi atau *popsize*. Kemudian induk diambil secara acak. Induk atau *parent*s terpilih akan dicari titik potongnya dan kemudian menghasilkan anak.

4.2.4 Mutasi

Pada proses reproduksi kedua adalah mutasi dan dengan metode *Reciprocal exchange mutation*. Metode ini adalah menukar tempat dari dua gen yang terpilih

secara acak. Proses mutasi dengan metode *Reciprocal exchange mutation* ditunjukkan pada gambar .



Gambar 4.7 Diagram Alir Mutasi

Pada Gambar 4.8 Diatas dapat diketahui bahwa proses mutasi diawali dengan inputan *popsize* dan *mutation rate*. Lalu sistem akan mencari jumlah anak dengan rumus *mutation rate* dikali dengan populasi atau *popsize*. Kemudian induk diambil secara acak. Induk atau *parent*s terpilih kemudian akan dicari 2 gen secara acak

yang akan dilakukan penukaran kedua gen tersebut dan kemudian menghasilkan anak.

4.2.5 Evaluasi

Proses ini bisa dinamakan dengan proses *decode* yaitu proses yang terjadi setelah proses *crossover* dan mutasi. Proses ini merupakan evaluasi dari nilai *fitness* setiap kromosom. Pada tahap ini perhitungan nilai *fitness* masing masing sekuen dari table matrik *blosum62.* Gambaran umum evaluasi dapat dilihat pada Gambar 4.9.

P 2	Index array *gap* acak 1 sekuen 1	*maxle ngth*	Index array *gap* acak 1 sekuen 2	*maxle ngth*	Index array *gap* acak 1 sekuen 3	*maxle ngth*

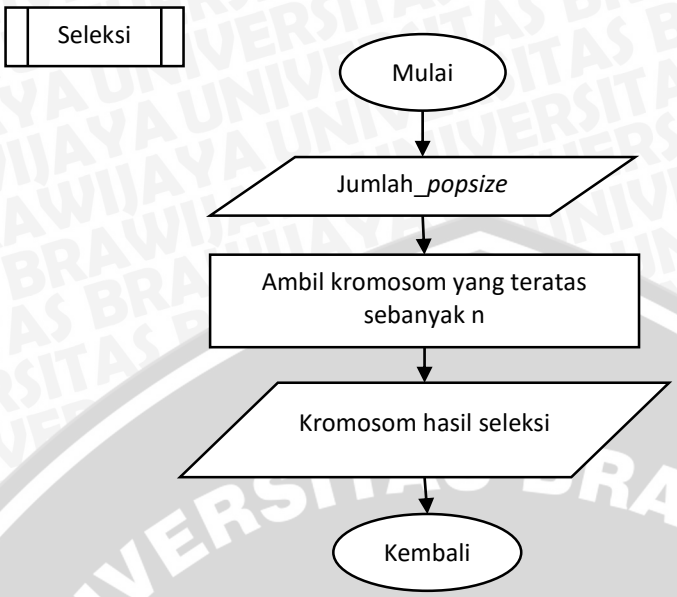
Sekuen 1	*Gap* acak sekuen 1 yang disisipi dengan karakter pada sekuen
Sekuen 2	*Gap* acak sekuen 2 yang disisipi dengan karakter pada sekuen
Sekuen 3	*Gap* acak sekuen 3 yang disisipi dengan karakter pada sekuen
Index Array	sepanjang *maxlength*
Nilai *Fitness*	*sum of pairss blosum62* skema affine

Gambar 4.8 Gambaran Umum Evaluasi

Pada Gambar 4.9 dapat dilihat bahwa saat proses insialisasi populasi terjadi *crossover* dan mutasi sehingga dihasilkan populasi 2. Setelah terjadinya hal tersebut maka populasi 2 di*decode* untuk dilakukan proses perhitungan *fitness*. Satu Populasi jika di*decode* akan dihasilkan 3 sekuen atau sebanyak sekuen yang disejajarkan sehingga akan tampak bahwa sekuen-sekuen tersebut mengalami pergeseran *gap*. Perhitungan nilai *fitness* dilakukan menggunakan *sum of pairss* dengan nilai matrik yang digunakan adalah *blosum62* serta skema yang digunakan adalah skema affine.

4.2.6 Seleksi

Tahap ini digunakan untuk mendapatkan calon induk (*parent*) yang baik. Induk yang baik akan menghasilkan anak yang baik. Semakin tinggi nilai *fitness* semakin tinggi pula kesempatan individu untuk terpilih. Disini menggunakan seleksi dengan metode elitis (*elitism*) berarti usaha untuk mempertahankan individu-individu terbaik yang telah diperoleh di suatu generasi ke generasi selanjutnya (Amelia, 2014). Sehingga dalam generasi berikutnya, individu-individu terbaik tersebut akan tetap ada dan dipertahankan.



Gambar 4.9 Diagram Alir Seleksi

4.3 Perhitungan Manual

Pada tahap ini menjelaskan tentang perhitungan manual pada sistem dan memberikan gambaran umum mengenai bagaimana kerja sistem. Contoh manualisasi ini menggunakan *popsize* 5, *crossover rate* 0.6, *mutation rate* 0.6.

4.3.1 Inisialisasi Populasi awal

Disini bisa dikatakan dengan *preprocessing*.

Terdapat 3 sekuen dengan panjang berbeda :

>MTVK (memiliki panjang sekuen 4)

>MTVKARCILS (memiliki panjang sekuen 10)

>MVKRCI (memiliki panjang sekuen 6)

Pada 3 sekuen di atas sekuen dengan panjang sekuen terpanjang adalah sekuen kedua dengan jumlah 10. Jumlah gen tersebut dikalikan dengan 1,2 berdasarkan landasan kepustakaan pada subbab inisialisasi kromosom untuk menentukan *maxlength* yaitu :

10 x 1.2 = 12

Setelah mendapatkan nilai *maxlength*, sekuen diisi dengan *char* dengan sepanjang *maxlength* dan disisipi oleh *gap.*

Sekuen 1	M	T	V	K	-	-	-	-	-	-	-	-
Sekuen 2	M	T	V	K	A	R	C	I	L	S	-	-
Sekuen 3	M	V	K	R	C	I	-	-	-	-	-	-
Index array	0	1	2	3	4	5	6	7	8	9	10	11

Gambar 4.10 *Preprocessing*

4.3.2 *Encode*

Langkah selanjutnya sekuen di*encode* dari index array yang berisi *gap* dan dibatasi dengan *maxlength* untuk inisialisasi populasi.

Populasi	Index array *gap* sekuen 1	*Maxlength*	Index array *gap* sekuen 2	*maxlength*	Index array *gap* sekuen 3	*maxlength*

P1	4	5	6	7	8	9	10	11	12	10	11	12	6	7	8	9	10	11	12
P2	0	3	5	2	4	10	6	8	12	9	7	12	1	0	6	3	11	9	12
P3	1	4	2	5	7	8	11	10	12	6	8	12	0	1	9	8	6	3	12
P4	5	9	10	3	2	6	8	11	12	5	1	12	2	7	1	0	3	5	12
P5	0	1	2	3	8	4	9	5	12	11	0	12	8	7	5	10	9	4	12

Gambar 4.11 *Encode* Inisialisasi Populasi

Dilakukan acak *random* sebanyak populasi.

4.3.3 *Crossover* dengan Metode One-Cut Point

Persilangan pada algoritma genetika melibatkan dua buah individu yang dipilih secara acak menjadi *parent* yang akan membentuk kromosom baru. Proses ini diawali dengan penentuan tingkat *crossover* (*cr*). Nilai ini menentukan rasio dari keturunan (*offspring*) yang dihasilkan dari proses *crossover*. Hasil *offspring* dari proses *crossover* diperoleh dengan mengalikan *cr* dengan ukuran populasi (*popsize*). Berikut contoh *crossover* :

Jumlah *offspring* = 0.6 * 3 = 1.8 (dibulatkan ke atas 2)

P1	4	5	6	7	8	9	10	11	12	10	11	12	6	7	8	9	10	11	12
P2	0	3	5	2	4	10	6	8	12	9	7	12	1	0	6	3	11	9	12

C1	4	5	6	7	8	9	10	11	12	9	7	12	1	0	6	3	11	9	12
C2	0	3	5	2	4	10	6	8	12	10	11	12	6	7	8	9	10	11	12

Gambar 4.12 *Crossover* Metode *One cut point*

4.3.4 Mutasi dengan Metode Reciprocal Exchange

Memilih acak salah satu *parent* lalu menukar dua posisi gen pada *parent* tersebut sehingga menghasilkan individu baru.

Jumlah *offspring* = 0.3 * 3 = 0.9 (dibulatkan ke atas menjadi 1)

P4	5	9	10	3	2	6	8	11	12	5	1	12	2	7	1	0	3	5	12
C3	5	9	10	3	2	1	8	11	12	5	6	12	2	7	1	0	3	5	12

Gambar 4.13 Mutasi Metode Reciprocal Exchange

4.3.5 Evaluasi

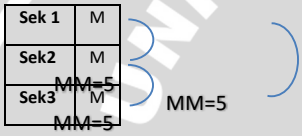
Evaluasi disini adalah *decode* yaitu mengembalikan angka ke dalam bentuk semula dan diisi dengan tanda *gap* dan menghitung nilai *fitness* menggunakan fungsi *sum of pairss.*

P1	4	5	6	7	8	9	10	11	12	10	11	12	6	7	8	9	10	11	12

Sek 1	M	T	V	K	-	-	-	-	-	-	-	-	
Sek2	M	T	V	K	A	R	C	I	L	S	-	-	
Sek3	M	V	K	R	C	I	-	-	-	-	-	-	
Index array	0	1	2	3	4	5	6	7	8	9	10	11	
Bobot *fitness*	5	5	0	7	-11	-4	-6	-6	-6	-6	-13	-3	**-38**

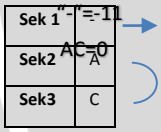
Gambar 4.14 *Decode* Populasi Ke-1

Cara menghitung bobot *fitness* adalah sebagai berikut :



Gambar 4.15 Contoh Menghitung Nilai *Fitness* 1

Dengan melihat matrik *blosum62* nilai M bertemu dengan M adalah 5. Disini yang diambil hanya satu pasang MM karena nilai yang sama dihitung hanya satu.



Gambar 4.16 Contoh Menghitungan Nilai *Fitness* 2

Disini ada *gap* yang terletak setelah adanya karakter sehingga *gap* ini dinamakan *gap* open dengan nilai -11. Dengan melihat matrik *blosum62* nilai A bertemu dengan C adalah 0. Maka jumlah dari -11+0 =-11 nilai *fitness*nya.

P2	0	3	5	2	4	10	6	8	12	9	7	12	1	0	6	3	11	9	12

Sek 1	-	M	-	-	-	-	-	T	-	V	-	K	
Sek2	M	T	V	K	A	R	C	-	I	-	L	S	
Sek3	-	-	M	-	V	K	-	R	C	-	I	-	
Index array	0	1	2	3	4	5	6	7	8	9	10	11	
Bobot *fitness*	-26	-2	-10	-16	-1	-1	-16	-12	-12	-26	-9	-11	**-142**

Gambar 4.17 *Decode* Populasi Ke-2

P4	5	9	10	3	2	6	8	11	12	5	1	12	2	7	1	0	3	5	12

Sek 1	M	T	-	-	V	-	-	K	-	-	-	-
Sek2	M	-	T	V	K	-	A	R	C	I	L	S
Sek3	-	-	-	-	M	-	V	-	K	R	C	I
Index array	0	1	2	3	4	5	6	7	8	9	10	11
Bobot *fitness*	-6	-16	-16	-6	-2	-33	-1	-9	-14	-4	-2	-3	**-112**

Gambar 4.18 *Decode* Populasi Ke-3

C1	4	5	6	7	8	9	10	11	12	9	7	12	1	0	6	3	11	9	12

Sek 1	M	T	V	K	-	-	-	-	-	-	-	-
Sek2	M	T	V	K	A	R	C	I	L	-	-	S
Sek3	-	-	M	-	V	K	-	R	C	-	I	-
Index array	0	1	2	3	4	5	6	7	8	9	10	11
Bobot *fitness*	-6	4	3	-6	-11	1	-16	-4	-2	-23	-6	-16	**-82**

Gambar 4.19 *Decode* Populasi Ke-4

C2	0	3	5	2	4	10	6	8	12	10	11	12	6	7	8	9	10	11	12

Sek 1	-	-	M	-	-	-	-	T	-	V	-	K
Sek2	M	T	V	K	A	R	C	I	L	S	-	-
Sek3	M	V	K	R	C	I	-	-	-	-	-	-
Index array	0	1	2	3	4	5	6	7	8	9	10	11
Bobot *fitness*	-6	-1	-2	-9	-1	-4	-16	-2	-16	-3	-23	-6	**-89**

Gambar 4.20 *Decode* Populasi Ke-5

C3	5	9	10	3	7	1	8	11	12	5	6	12	2	7	1	0	3	5	12

Sek 1	-	M	T	-	V	-	K	-	-	-	-	-	
Sek2	M	T	V	K	A	-	-	R	C	I	L	S	
Sek3	-	-	-	-	M	-	V	-	K	R	C	I	
Index array	0	1	2	3	4	5	6	7	8	9	10	11	
Bobot *fitness*	-26	-2	-1	-16	0	-33	-3	-16	-4	-4	-2	-3	**-110**

Gambar 4.21 *Decode* Populasi Ke-6

Bobot pada *fitness* bisa dilihat dari matrik blossom62 dengan menggunakan fungsi *fitness sum of pairss* .

4.3.6 Seleksi

Pada tahap ini dilakukan seleksi *elitism* dengan melihat induk terbaik jadi akan menghasilkan seperti ini :

Induk	Nilai *Fitness*
P1	-38
P2	-142
P3	-112
C1	-82
C2	-89
C3	-110

Gambar 4.22 Hasil Nilai *Fitness*

Dengan melihat tabel diatas maka yang memiliki nilai *fitness* tertinggi -38 adalah P1 dengan penjajaran sekuen :

> MTVK--------

> MTVKARCILS--

> MTVKRCI-----

Setelah didapatkan hasil *fitness* dan diurutkan maka yang terbaik diambil sejumlah dengan jumlah *popsize*. Disini *popsize* ada 3 maka diambil 3 individu dan dilakukan proses ke generasi berikutnya yaitu P1, C1, dan C2 yang menjadi P1,P2,P3 yang baru.

Induk Lama	Induk Baru	Nilai *Fitness*
P1	>P1	-38
P2	>C1	-82
P3	>C2	-89
C1	>C3	-110
C2	>P3	-112
C3	>P2	-142

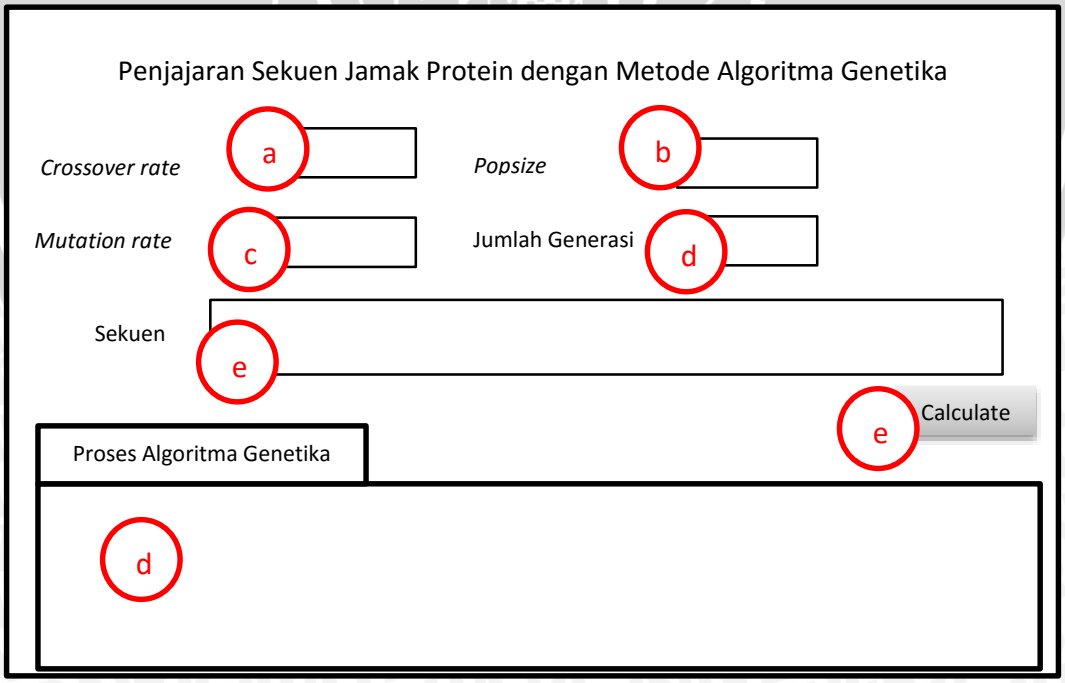
Tinggi ke Rendah

Gambar 4.23 Seleksi *Elitism*

4.4 Perancangan Antarmuka

Perancangan antarmuka atau disebut dengan interface adalah suatu sistem yang dapat dilihat oleh pengguna untuk melihat suatu informasi. Sistem yang dimaksut adalah Penjajaran Sekuen Jamak Protein dengan Metode Algoritma Genetika yang menghasilkan informasi berupa nilai *fitness* terbaik dari beberapa sekuen yang ada.

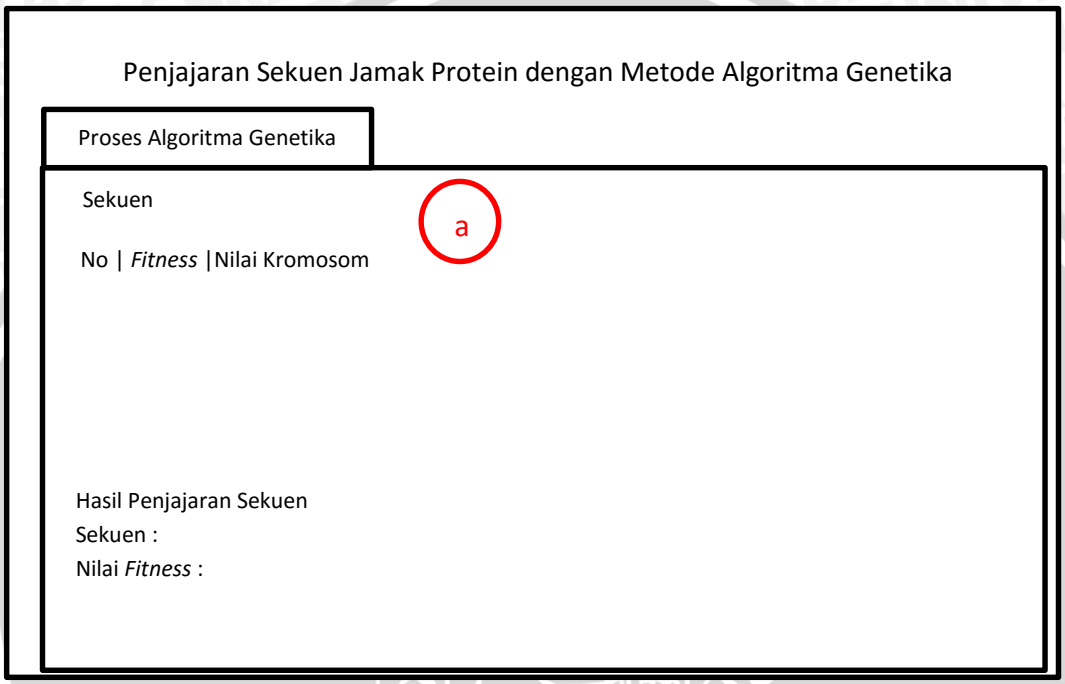
Untuk menghasilkan nilai terbaik atau *fitness* maka dimasukkan dahulu parameter seperti *popsize*, jumlah generasi ,*crossover rate*, *mutation rate*.



Gambar 4.24 Perancangan Antarmuka 1

Deskripsi gambar :

a. Field untuk menginputkan *crossover rate*
b. Field untuk menginputkan *crossover rate*
c. Field untuk menginputkan *crossover rate*
d. Field untuk menginputkan *crossover rate*
e. Field untuk menginputkan *crossover rate*
f. Button *calculate* untuk memproses
g. Field untuk menampilkan proses algoritma genetika



Gambar 4.25 Perancangan Antarmuka 2

Deskripsi gambar diatas :

a. Field untuk menampilkan proses algoritma genetika yang terdiri dari sekuen, nomor, nilai kromosom, *fitness*, sekuen yang terpilih dengan *fitness* , serta deskripsi dari sekuen tersebut.

BAB 5 IMPLEMENTASI

Pada bab ini membahas mengenai hasil yang diperoleh dari analisa kebutuhan dan perancangan yang telah dibuat yaitu implementasi perangkat lunak. Implementasi perangkat lunak yang dimaksud diantaranya penjelasan mengenai spesifikasi sistem, implementasi sistem dan implementasi antarmuka program Penjajaran Sekuen Jamak Protein dengan Algoritma Genetika.

5.1 Spesifikasi Sistem

Bahan yang dijadikan acuan untuk implementasi sistem sehingga dapat berfungsi sesuai dengan kebutuhan adalah analisis kebutuhan dan perancangan yang terdapat pada bab 4. Spesisfikasi sistem ini terdiri dari dua spesifikasi yaitu spesifikasi perangkat keras dan spesifikasi perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras sebuah komputer yang digunakan untuk implementasi sistem dijabarkan pada table 5.1.

Tabel 5. 1 Spesifikasi Perangkat Keras

Komponen	**Spesifikasi**
Processor	Intel(R) Core (TM) i7-3612QM CPU @ 2.10Ghz
Memori (RAM)	4 GB DDR3
Storage	Samsung SSD 840 Series 120GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak sebuah komputer yang digunakan untuk implementasi sistem dijabarkan pada table 5.2.

Tabel 5. 2 Spesifikasi Perangkat Lunak

Komponen	Spesifikasi
Sistem Operasi	Mi*cr*osoft Windows 10 enterprise (64-bit)
Bahasa Pe*mr*ograman	JAVA 1.8
Tools Pe*mr*ogaman	Netbeans IDE 8.0 dan Notepad++

5.2 Implementasi Sistem

Sesuai dengan perancangan yang dibuat dan dijelaskan pada bab 4, maka selanjutnya akan dibahas mengenai implementasi sistem. Pada implementasi sistem ini menggunakan bahasa Pe*mr*ogaman JAVA. Pada sistem Penjajaran

Sekuen Jamak Protein dengan Algoritma Genetika ini memiliki beberapa operasi (method) yang dapat dilihat pada tabel 5.3 .

Tabel 5.3 Daftar Tabel Operasi atau Method dan Fungsi

Operasi atau Method	Fungsi
preprocessing	Fungsinya untuk inisialisasi populasi awal mencari *maxlength*.
encode	Fungsinya untuk *encode* data *gap* dari index array.
decode	Fungsinya untuk *decode* data *gap* dari index array.
fitness	Fungsinya untuk menghitung nilai setiap individu.
crossover	Fungsinya untuk melakukan *crossover* atau tukar silang pada individu.
mutation	Fungsinya untuk melakukan mutasi.
Seleksi*Elitism*	Fungsinya untuk mencari calon induk terbaik.
*random*Individu	Fungsinya untuk melakukan pengacakan pada individu.
printAlignment	Fungsinya untuk menampilkan hasil penjajaran sekuen.
printKromosom	Fungsinya untuk menampilkan kromosom.
getIndex	Fungsinya untuk mengetahui index dari setiap karakter.
getDistance	Fungsinya untuk mengetahui nilai matrik.

5.2.1 Inisialisasi Populasi Awal

Pada tahap ini inisialisasi populasi awal yaitu membentuk populasi awal secara acak yang diikuti dengan tahap *preprocessing*, *encode* dan *decode*.

```
1  private void preprocessing() {
2          int temp = 0;
3          for (int i = 0; i < f.size(); i++) {
4              if (temp < f.getSequence(i).length()) {
5                  temp = f.getSequence(i).length();
6              }
```

```
7           }
8     maxlength = (int) (Math.ceil((double) (temp * k)));
9           for (int i = 0; i < f.size(); i++) {
10              for (int j = 0; j <maxlength; j++) {
11                  if (j == 0) {
12                  SEQS.add(f.getSequence(i));
13                  j = f.getSequence(i).length() - 1;
14                  } else {
15                  SEQS.set(i, SEQS.get(i).concat("-"));
16                  }
17              }
18          }
19      }
```

Keterangan Source Code :

Baris 3-5 : Perulangan yang dilakukan untuk mencari panjang sekuen masing-masing individu.

Baris 8-15 : Mencari nilai *maxlength* dengan mengalikan jumlah gen atau kromosom dengan 1,2 dan hasil dibulatkan ke atas.

```
1   private void encode() {
2           for (int i = 0; i < SEQS.size(); i++) {
3               int index = SEQS.get(i).indexOf("-");
4               while (index != -1) {
5                   kromosom.add(SEQS.get(i).indexOf("-", index));
6                 index = SEQS.get(i).indexOf("-", index + 1);
7               }
8               kromosom.add(maxlength);
9           }
10      }
```

Keterangan Source Code:

Baris 2-8 : Perulangan setiap index array untuk mengisi sisa index dengan *gap* setelah char habis.

Baris 4 : Operasi indexof akan menghasilkan nilai -1 jika char yang dicari tidak ada.

```
1   public void decode() {
2           int temp = 0;
3           for (int i = 0; i < f.size(); i++) {
4               seq.setLength(0);
5               for (int k = 0; k < maxlength; k++) {
6                   seq.append('0');
7               }
```

```
8                 for (int j = temp; j < kromosom.size(); j++) {
9                     if (kromosom.get(j) == maxlength) {
10                        temp = j + 1;
11                        break;
12                    }
13                 seq.setCharAt(kromosom.get(j), '-');
14                }
15                int temp2 = 0;
16                for (int k = 0; k < maxlength; k++) {
17                    if (seq.charAt(k) == '0') {
                          seq.setCharAt(k,
18  SEQS.get(i).charAt(temp2));
19                        temp2++;
20                    }
21                }
22                SEQS_decode.add(seq.toString());
23            }
24        }
```

Keterangan :

Baris 7-13 : Perulangan jika angka dari index 0 sampai *maxlength* akan diisi dengan nilai *gap* pada beberapa sekuen dengan pembatas *maxlength*.

Baris 15-19 : Jika selain dari *gap* maka diisi dengan sekuen secara berurutan.

5.2.2 Implementasi *Fitness*

Pada proses implementasi *fitness* ini setiap individu atau kromosom dihitung menggunakan matrik *blosum62* dengan mencari nilai *maxlength* dan di*encode*.

```
    private int fitness(String SEQ1, String SEQ2) throws
1   Exception {
2           int fitness = 0;
3           for (int i = 0; i < maxlength; i++) {
                if (SEQ1.charAt(i) != '-' && SEQ2.charAt(i) != '-
4   ') {
                    fitness += m.getDistance(SEQ1.charAt(i),
5   SEQ2.charAt(i));
                } else if (SEQ1.charAt(i) == '-' &&
6   SEQ2.charAt(i) != '-') {
7                   if (i == 0) {
8                       fitness += gap_open;
9                   } else {
                        if (SEQ1.charAt(i - 1) == '-' &&
10  SEQ2.charAt(i - 1) != '-') {
11                          fitness += gap_ext;
12                      } else {
13                          fitness += gap_open;
```

```
14                      }
15                  }
              } else if (SEQ1.charAt(i) != '-' &&
16  SEQ2.charAt(i) == '-') {
17                  if (i == 0) {
18                      fitness += gap_open;
19                  } else {
                        if (SEQ2.charAt(i - 1) == '-' &&
20  SEQ1.charAt(i - 1) != '-') {
21                          fitness += gap_ext;
22                      } else {
23                          fitness += gap_open;
24                      }
25                  }
26              }
27          }
28          return fitness;
29      }
```

Keterangan :

Baris 2 : deklarasi variable

Baris 3-5 : Perulangan jika huruf bertemu dengan huruf maka akan dicari nilai *fitness*-nya di matrik *blosum62* dengan method getDistance.

Baris 6-8 : Jika ada huruf bertemu dengan *gap* maka termasuk *gap* open nilai *fitness*nya atau skema affine.

Baris 9-11 : Jika *gap* berada setelah *gap* maka nilainya termasuk *gap* extension.

5.2.3 Implementasi *Crossover*

Pada implementasi *crossover* penelitian ini menggunakan metode *one cut point*. Metode *one cut point* ini memilih acak 2 *parent* sehingga terjadi persilangan dan menghasilkan *offspring* atau anak atau kromosom baru. Penentuan titik silang ditentukan dengan penentuan *crossover rate* (*cr*) dikali dengan populasi (*popsize*).

```
 1  private void crossover() throws Exception {
 2          Collections.shuffle(p);
 3          p1 = p.peek();
 4          p.pop();
 5          p2 = p.peek();
 6          p.pop();
 8          Collections.shuffle(cutPoint);
 9          cp = cutPoint.peek();
11          k1.clear();
12          k2.clear();
13          k1.addAll(pops.get(p1).getKromosom());
14          k2.addAll(pops.get(p2).getKromosom());
```

```
19          p.add(p1);
20          p.add(p2);
22          cutPoint.pop();
23          cutPoint.add(cp);
25          c1.clear();
26          c2.clear();
27          c1a.clear();
28          c2a.clear();
29          c1b.clear();
30          c2b.clear();
32          int maxlength = pops.get(p1).maxlength;
33          int kromosomSize = k1.size();
34          int point = -1;
36          for (int i = 0; i < kromosomSize; i++) {
37              if (k1.get(i) == maxlength) {
38                  point++;
39              }
40              if (point < cp) {
41                  c1a.add(k1.get(i));
42                  c2a.add(k2.get(i));
43              } else {
44                  c1b.add(k1.get(i));
45                  c2b.add(k2.get(i));
46              }
48          }
49          c1.addAll(c1a);
50          c1.addAll(c2b);
51          c2.addAll(c2a);
52          c2.addAll(c1b);
53          Individu C1 = new Individu(c1, fileInput);
54          Individu C2 = new Individu(c2, fileInput);
55          C1.calculateFitness();
56          pops.add(C1);
57          C2.calculateFitness();
58          pops.add(C2);
```

Keterangan :

Baris 2 : Mengacak list untuk *parent*s.

Baris 3-6 : Melihat posisi teratas dari stack lalu diambil untuk dijadikan P1 dan P2 (*Parent*1 dan *Parent*2)

Baris 4 :Mengacak list untuk cut point.

Baris 11-14 : Menghapus stack lalu mengisinya lagi dengan kromosom yang baru.

Baris 19-23 :Menggabungkan *parent*1 dan *parent*2 lalu diambil titik potong atau cut point

Baris 25-30 : Menghapus tumpukan di anak atau individu baru

Baris 32 : Menggabungkan nilai *parent* dengan *maxlength*

Baris 34 : Deklarasi point

Baris 40-42 : Menambahkan untuk bagian kiri

Baris 43-45 : Menambahkan untuk bagian kanan

Baris 49-52 : Menggabungkan 2 array menjadi satu.

Baris 53-54 : Individu baru atau anak.

Baris 55-58 : Menghitung nilai *fitness* individu baru atau anak C1 dan C2.

5.2.4 Implementasi Mutasi

Pada implementasi mutasi penelitian ini menggunakan metode *Reciprocal exchange mutation* yaitu mengubah satu gen atau lebih pada suatu individu yang diawali dengan menentukan *mutation rate* (*mr*) dikali dengan ukuran populasi sehingga pada *parent* acak tersebut dapat diketahui gen yang terpilih terdapat pada posisi ke berapa.

```
1   private void mutation() throws Exception {
2           LinkedList<Integer> gen = new LinkedList<>();
3           for (int i = 0; i < pops.get(0).maxlength; i++) {
4               gen.push(i);
5           }
6           Collections.shuffle(gen);
7           int p = r.nextInt(popsize);
8           int seq = r.nextInt(ind.SEQS.size());
9           List<Integer> k = new ArrayList<>();
10          int seq_num = 0;
11      for (int i = 0; i < pops.get(p).kromosom.size(); i++) {
12              if (seq_num == seq) {
13                  k.add(gen.pop());
14              } else {
15              k.add(pops.get(p).getKromosom().get(i));
16              }
17              if (pops.get(p).getKromosom().get(i) ==
pops.get(p).maxlength) {
18                  seq_num++;
19                  k.remove(i);
20                  k.add(pops.get(p).maxlength);
21              }
22          }
```

Keterangan :

Baris 2 : Untuk meng-gene*rate* gen atau kromosom unik.

Baris 4 : Untuk menambahkan data gen ke dalam stack.

Baris 6 : Mengacak gen.

Baris 7 : Memilih induk atau *parent*s.

Baris 9 : Inisialisasi arraylist baru.

Baris 19 : Menghapus stack.

5.2.5 Implementasi Seleksi

Pada implementasi seleksi penelitian ini menggunakan metode *elitism* yaitu mempertahankan individu terbaik dengan melakukan perhitungan *fitness* tiap generasi.

```
1   private void seleksiElitism() throws Exception {
2   Collections.sort(pops, (c1, c2) -> c2.fitness - c1.fitness);
    List<Individu> newPOP = new ArrayList<>(pops.subList(0,
3   popsize));
4           pops.clear();
5           pops.addAll(newPOP);
6       }
```

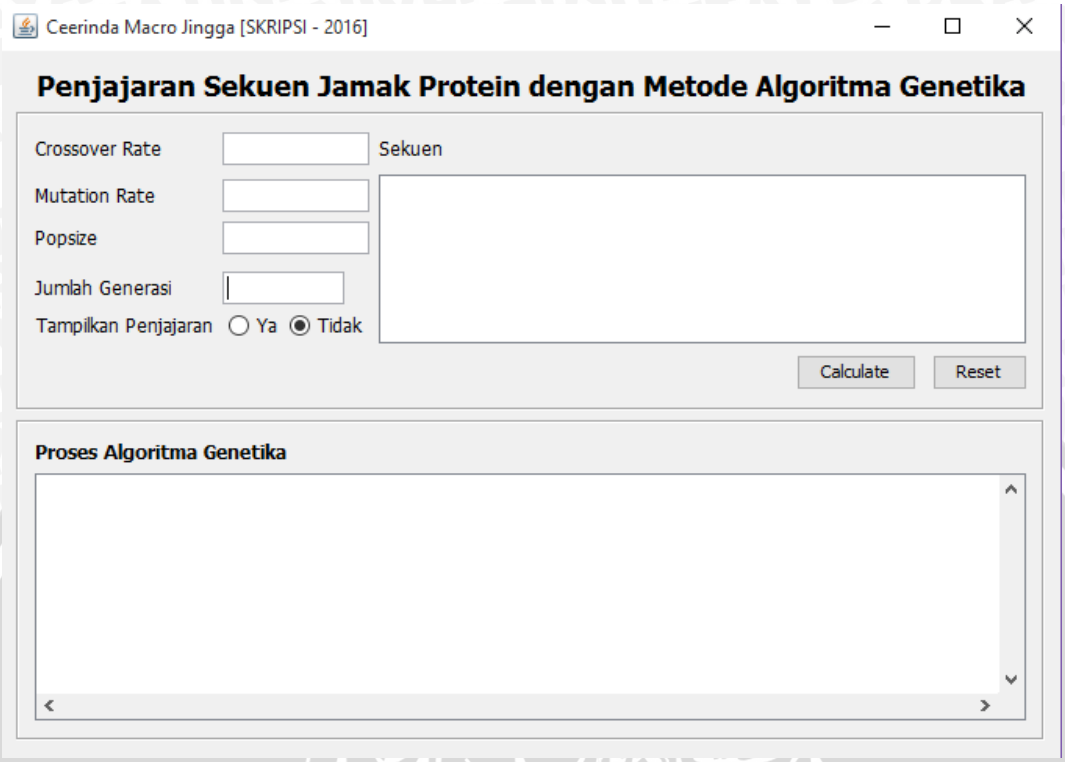
Keterangan Source Code:

Baris 2 : Mengumpulkan semua populasi dan mengurutkannya.

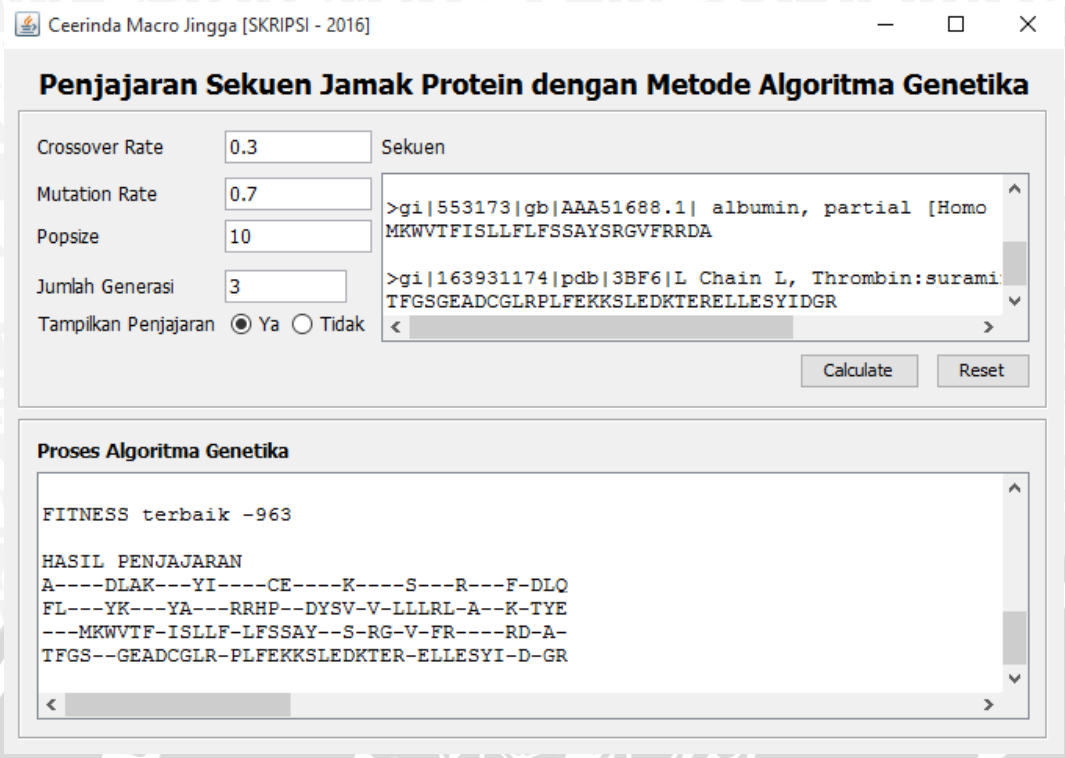
Baris 3 : Inisialisasi arraylist baru.

5.3 Implementasi Antarmuka

Pada implementasi antarmuka penelitian ini terdapat output :



Setelah diisikan oleh parameter yang diinginkan maka tinggal klik calculate dan proses algoritma genetika akan berjalan sehingga menghasilkan output dibawah ini.



Hasil output yang dihasilkan adalah berikut ini menggunakan tampilan penjajaran :

Generasi *Fitness* Kromosom

0 -1019 [2, 36, 15, 38, 3, 37, 12, 16, 29, 40, 0, 11, 32, 5, 17, 4, 34, 23, 25, 30, 27, 33, 24, 19, 10, 22, 18, 28, 44, 22, 37, 16, 5, 9, 4, 13, 40, 0, 35, 8, 21, 3, 15, 27, 29, 38, 10, 14, 44, 42, 19, 6, 18, 7, 40, 14, 37, 23, 4, 24, 22, 43, 39, 36, 5, 30, 11, 44, 14, 23, 7, 10, 0, 18, 11, 16, 44]

-A----DLAK---YI-----CE----K----S---R---F-DLQ

-FL---YK---YA----RRHP--DYSV-V-LLLRL-A--K-TYE

MKWV----TFI-SL-LFL--FS---SAYSR-GVFRR--D--A--

-TFGSGE-AD--CG-L-R-PLFE-KKSLEDKTERELLESYIDGR

1 -1019 [2, 36, 15, 38, 3, 37, 12, 16, 29, 40, 0, 11, 32, 5, 17, 4, 34, 23, 25, 30, 27, 33, 24, 19, 10, 22, 18, 28, 44, 22, 37, 16, 5, 9, 4, 13, 40, 0, 35, 8, 21, 3, 15, 27, 29, 38, 10, 14, 44, 42, 19, 6, 18, 7, 40, 14, 37, 23, 4, 24, 22, 43, 39, 36, 5, 30, 11, 44, 14, 23, 7, 10, 0, 18, 11, 16, 44]

-A----DLAK---YI-----CE----K----S---R---F-DLQ

-FL---YK---YA----RRHP--DYSV-V-LLLRL-A--K-TYE

MKWV----TFI-SL-LFL--FS---SAYSR-GVFRR--D--A--

-TFGSGE-AD--CG-L-R-PLFE-KKSLEDKTERELLESYIDGR

2 -963 [2, 36, 15, 38, 3, 37, 12, 16, 29, 40, 0, 11, 32, 5, 17, 4, 34, 23, 25, 30, 27, 33, 24, 19, 10, 22, 18, 28, 44, 22, 37, 16, 5, 9, 4, 13, 40, 0, 35, 8, 21, 3, 15, 27, 29, 38, 10, 14, 44, 27, 41, 17, 35, 16, 37, 10, 38, 30, 43, 2, 24, 32, 3, 36, 25, 0, 1, 44, 5, 31, 15, 0, 6, 16, 41, 39, 44]

A----DLAK---YI----CE----K----S---R---F-DLQ

FL---YK---YA---RRHP--DYSV-V-LLLRL-A--K-TYE

---MKWVTF-ISLLF-LFSSAY--S-RG-V-FR----RD-A-

TFGS--GEADCGLR-PLFEKKSLEDKTER-ELLESYI-D-GR

FITNESS terbaik -963

HASIL PENJAJARAN

A----DLAK---YI----CE----K----S---R---F-DLQ

FL---YK---YA---RRHP--DYSV-V-LLLRL-A--K-TYE

---MKWVTF-ISLLF-LFSSAY--S-RG-V-FR----RD-A-

TFGS--GEADCGLR-PLFEKKSLEDKTER-ELLESYI-D-GR

Tanpa tampilkan penjajaran

Generasi *Fitness* Kromosom

0 -1035 [27, 3, 38, 13, 36, 17, 40, 39, 18, 21, 31, 0, 26, 29, 25, 6, 16, 34, 12, 43, 14, 28, 5, 22, 10, 9, 7, 15, 44, 13, 22, 10, 18, 7, 41, 6, 38, 21, 5, 9, 16, 34, 17, 2, 36, 33, 19, 31, 44, 17, 43, 35, 10, 9, 27, 38, 1, 3, 26, 7, 31, 33, 2, 18, 14, 0, 41, 44, 27, 24, 40, 13, 36, 43, 35, 23, 44]

1 -1035 [27, 3, 38, 13, 36, 17, 40, 39, 18, 21, 31, 0, 26, 29, 25, 6, 16, 34, 12, 43, 14, 28, 5, 22, 10, 9, 7, 15, 44, 13, 22, 10, 18, 7, 41, 6, 38, 21, 5, 9, 16, 34, 17, 2, 36, 33, 19, 31, 44, 17, 43, 35, 10, 9, 27, 38, 1, 3, 26, 7, 31, 33, 2, 18, 14, 0, 41, 44, 27, 24, 40, 13, 36, 43, 35, 23, 44]

2 -1010 [39, 7, 20, 21, 34, 14, 11, 0, 18, 32, 19, 37, 36, 10, 38, 15, 4, 3, 17, 1, 9, 13, 24, 29, 28, 40, 26, 12, 44, 4, 3, 37, 38, 31, 43, 20, 21, 1, 2, 8, 25, 15, 40, 28, 27, 9, 13, 18, 44, 22, 29, 43, 12, 27, 14, 40, 7, 8, 38, 39, 9, 42, 33, 20, 18, 1, 11, 44, 27, 36, 20, 38, 14, 16, 35, 33, 44]

FITNESS terbaik -1010

HASIL PENJAJARAN

-AD-L---A--K-------YI--CE-----K-SR-F-D--LQ-

FL-YK---Y--AR-RH----P--DYSVVLLL-R--L-AKT-YE

----MKW-V--TFI-SL--LFLFSSA--YSR-G-V-FRRD-A-

TFGSGEADCGLRPL-F-EKK-SLEDKT-ERELL-E--SYIDGR

BAB 6 PENGUJIAN

Pada bab ini akan diuraikan tentang pengujian dan analisis dari sistem Penjajaran Sekuen Jamak Protein dengan Algoritma Genetika. Pengujian pada sistem ini menggunakan parameter *crossover rate (Cr)*, *mutation rate (Mr)*, ukuran populasi atau *popsize* dan juga jumlah generasi pada algoritma genetika sehingga dapat menghasilkan nilai *fitness* yang terbaik.

6.1 Pengujian Parameter Algoritma Genetika

Pengujian parameter algoritma genetika ini meliputi pengujian kombinasi *crossover rate (Cr)* dan *mutation rate (Mr),* ukuran populasi, dan yang terakhir jumlah generasi. Pegujian ini dilakukan selama beberapa kali pengujian sehingga dapat diketahui pengaruh perubahan nilai parameter algoritma genetika ini terhadap nilai *fitness* yang dihasilkan.

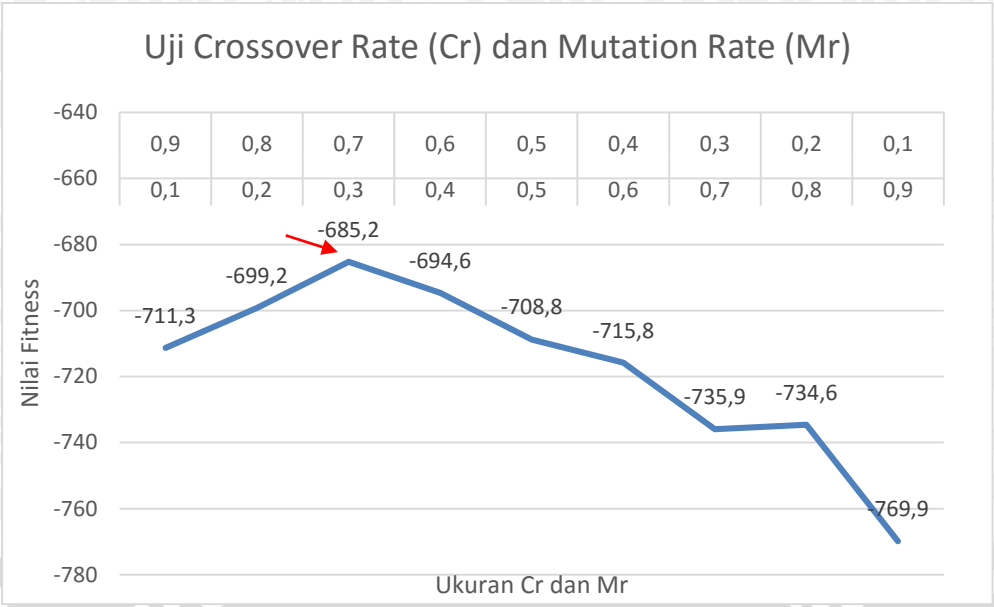
6.1.1 Hasil Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Pada pengujian ini dilakukan kombinasi ukuran *crossover rate* (*Cr*) dan *mutation rate* (*Mr*) dengan tujuan untuk mengetahui pada ukuran *crossover rate* (*Cr*) dan *mutation rate* (*Mr*) keberapa-kah penjajaran sekuen jamak protein dengan algoritma genetika ini menghasilkan nilai *fitness* terbaik. Jumlah populasi dan generasi yang digunakan adalah 100. Hasil pengujian dan ukuran kombinasi *crossover rate* (*Cr*) dan *mutation rate* (*Mr*) ditunjukkan pada tabel 6.2.

Tabel 6.1 Hasil Pengujian Kombinasi *Crossover Rate* dan *Mutation Rate*

Jumlah Populasi atau *popsize*	100	
Jumlah generasi	100	
Ukuran *Crossover Rate*(*Cr*)	**Ukuran *Mutation Rate* (*Mr*)**	**Nilai *Fitness***
0,1	0,9	-711,3
0,2	0,8	-699,2
0,3	**0,7**	**-685,2**
0,4	0,6	-694,6
0,5	0,5	-708,8
0,6	0,4	-715,8
0,7	0,3	-735,9
0,8	0,2	-734,6
0,9	0,1	-769,9

Pada tabel 6.2 diatas dapat dilihat bahwa pada ukuran *crossover rate* (*cr*) dan *mutation rate* (*mr*) (0,3) (0,7) menghasilkan nilai *fitness* yang terbaik yaitu -685,2. Grafik Hasil pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) dapat digambarkan pada Gambar 6.2.



Gambar 6.1 Grafik Hasil Pengujian Kombinasi *Crossover Rate* (*CR*) dan *Mutation Rate* (*MR*)

Pada grafik diatas dapat dilihat bahwa pada pengujian *crossover rate* (*cr*) dan *mutation rate* (*mr*) ke-3 dengan nilai *crossover rate* (*cr*) (0,3) dan *mutation rate* (*mr*) (0,7) menghasilkan nilai *fitness* terbaik yaitu -685,2. Hal ini terjadi karena pada nilai tingkat *crossover* 0,3 dan nilai tingkat mutasi 0,7 yang menghasilkan nilai *fitness* tertinggi, setelah itu nilai *fitness* mengalami penurunan.

6.1.2 Hasil Pengujian Ukuran Populasi atau *Popsize*

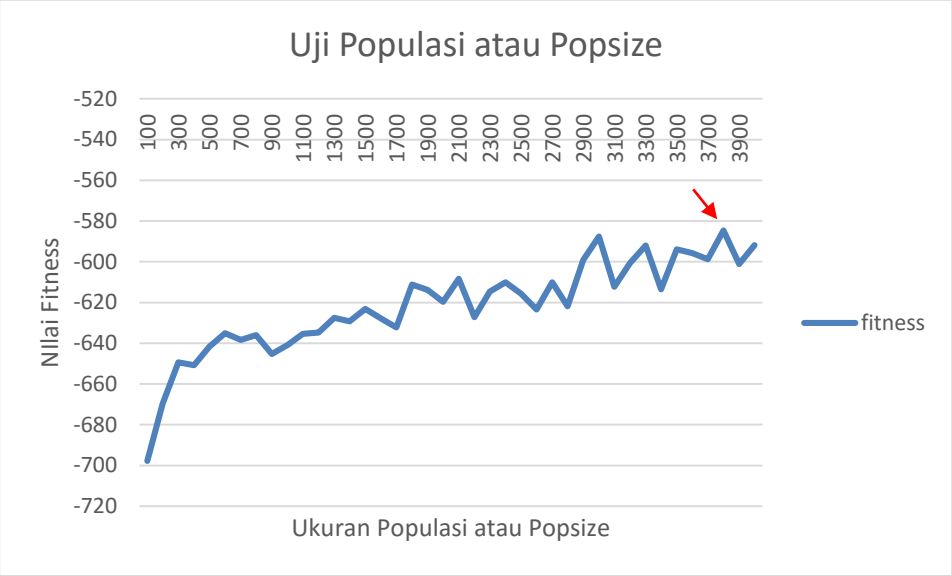
Hasil pengujian ukuran populasi atau *popsize* ini untuk mengetahui pada *popsize* keberapa-kah sistem menghasilkan penjajaran sekuen jamak protein dengan *fitness* terbaik. Ukuran populasi atau *popsize* yang diujikan yaitu dari 100-5100 dengan interval 100 dan masing-masing 10 kali pengujian serta diambil nilai rata-ratanya *fitness* untuk mengetahui solusi terbaik dari solusi optimal. Pengujian ini menggunakan parameter algoritma genetika *crossover rate* (*cr*) 0,3 dan *mutation rate* (*mr*) 0,7. Nilai *crossover rate* (*cr*) dan *mutation rate* (*mr*) 0,3 dan 0,7 karena sebelumnya dilakukan pengujian terhadap kombinasi *crossover rate* (*cr*) dan *mutation Rate* (*mr*) pada angka tersebut sistem menghasilkan *fitness* terbaik. Jumlah generasi yang digunakan adalah 100, karena pada pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) menggunakan jumlah generasi 100. Hasil pengujian terhadap ukuran populasi atau *popsize* dapat ditunjukkan pada tabel 6.1.

Tabel 6.2 Hasil Pengujian Ukuran Populasi (*Popsize*)

Jumlah Generasi	100
***Crossover Rate*(*Cr*) , *Mutation Rate* (*Mr*)**	(0.3) , (0.7)
Ukuran Populasi (*Popsize*)	**Nilai *Fitness***
100	-697,8
200	-669,6
300	-649,3
400	-650,7
500	-641,7
600	-635,1
700	-638,3
800	-636
900	-645,2
1000	-640,8
1100	-635,4
1200	-634,8
1300	-627,5
1400	-629,2
1500	-623,1
1600	-627,8
1700	-632,3
1800	-611,2
1900	-613,8
2000	-619,7
2100	-608,3
2200	-627,2
2300	-614,6
2400	-610,1
2500	-615,7
2600	-623,4
2700	-610,1
2800	-621,8
2900	-599,1
3000	-587,6
3100	-612,2
3200	-600,6
3300	-592
3400	-613,5
3500	-593,8
3600	-595,8
3700	-598,8
3800	-584,6

3900	-601,1
4000	-591,9

Berdasarkan dengan tabel 6.1 pengujian ukuran populasi atau *popsize* diatas dapat diketahui bahwa ukuran populasi 3800 menghasilkan nilai *fitness* terbaik yaitu -584,6 dan pada ukuran populasi selanjutnya terjadi konvergensi. Hasil pengujian populasi juga dapat digambarkan pada grafik Gambar 6.1.



Gambar 6.2 Grafik Hasil Pengujian Ukuran Populasi atau *Popsize*

Pada gambar 6.1 grafik hasil pengujian ukuran populasi atau *popsize* dapat dilihat bahwa pada ukuran populasi awal 100 menghasilkan nilai *fitness* yang sangat rendah karena ukuran populasi masih sedikit sehingga area eksplorasi algoritma genetika tidak luas dan solusi yang dihasilkannya tidak baik. Namun ukuran populasi 100-400 mengalami kenaikan nilai *fitness*. Pada umumnya semakin tinggi nilai populasi maka akan berpengaruh terhadap rata-rata nilai *fitness* namun semakin besar ukuran populasi juga berpengaruh pada waktu pe*mr*osesan sistem algoritma genetika yang juga akan berlangsung semakin lama (Suprayogi & Mahmudy, 2014). Grafik diatas juga menyimpulkan bahwa ukuran populasi yang tinggi tidak menjamin bahwa akan menghasilkan nilai *fitness* yang tinggi pula. Hal ini terjadi karena algoritma genetika menggunakan konsep *random*. Pada pengujian diatas dapat disimpulkan bahwa pada parameter ukuran populasi atau *popsize* 3800 yang menghasil nilai *fitness* optimal. Pada ukuran populasi setelah 3800 menghasilkan nilai *fitness* yang cenderung naik turun tetapi dengan nilai selisih yang tidak signifikan atau dengan kata lain konvergen.

6.1.3 Hasil Pengujian Ukuran Generasi

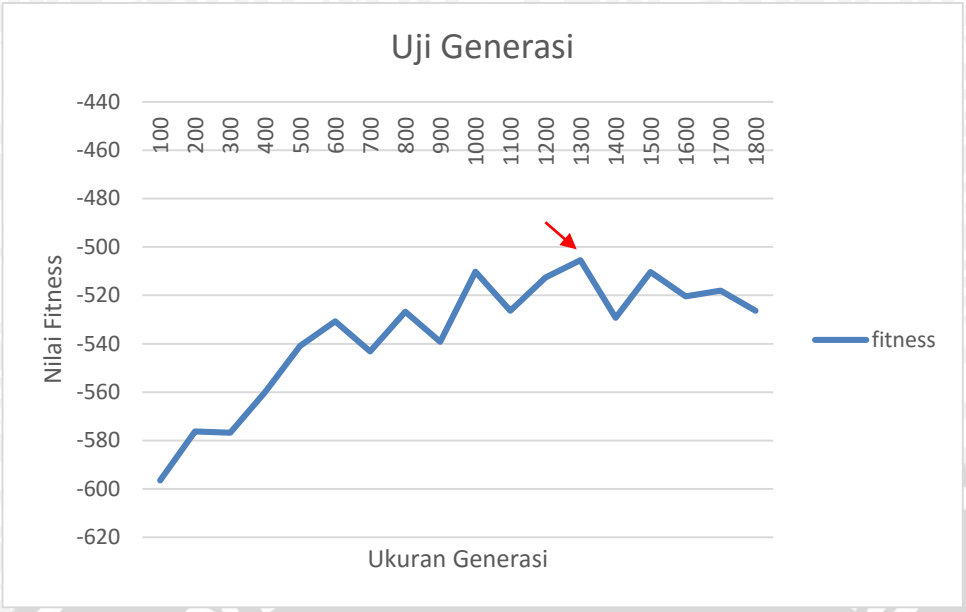
Pengujian ukuran generasi ini untuk mengetahui bahwa pada generasi keberapa-kah sistem penjajaran sekuen jamak protein dengan algoritma genetika ini akan menghasilkan *fitness* terbaik. Dalam pengujian ukuran generasi ini menggunakan parameter jumlah populasi 3800, karena telah kita hitung pada pengujian sebelumnya bahwa pada jumlah populasi tersebut menghasilkan nilai

fitness terbaik. Parameter *crossover rate* (*cr*) dan *mutation rate* (*mr*) yang digunakan adalah 0,3 dan 0,7, karena telah kita ketahui bahwa pada pengujian sebelumnya kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) pada jumlah tersebut menghasilkan nilai *fitness* terbaik. Ukuran generasi yang diuji yaitu dari 100 sampai 1000 dengan interval 100. Pengujian dilakukan 10 kali lalu dihitung nilai rata-rata *fitness* di setiap jumlah generasi. Hasil pengujian ukuran generasi ditunjukkan pada tabel 6.3.

Tabel 6.3 Hasil Pengujian Ukuran Generasi

Jumlah Populasi atau *Popsize*	3800
***Crossover Rate* (*Cr*) dan *Mutation Rate* (*Mr*)**	0.3 dan 0.7
Ukuran Generasi	**Nilai *Fitness***
100	-596,5
200	-576,2
300	-576,8
400	-559,9
500	-540,9
600	-530,7
700	-543,1
800	-526,7
900	-539,2
1000	-510,2
1100	-526,3
1200	-512,6
1300	-505,4
1400	-529,3
1500	-510,4
1600	-520,4
1700	-518
1800	-526,4

Berdasarkan tabel 6.3 diatas dapat diketahui bahwa pada pengujian ukuran generasi 1300 sistem menghasilkan nilai *fitness* terbaik yaitu -505,4 . Grafik hasil pengujian ukuran generasi dapat ditunjukkan pada Gambar 6.3.



Gambar 6.3 Grafik Hasil Pengujian Ukuran Generasi

Pada Grafik diatas dapat diketahui bahwa grafik mengalami kenaikan dari ukuran generasi 100 – 600 namun setelah itu mengalami penurunan nilai *fitness*. Nilai *fitness* terendah terdapat pada ukuran generasi 100 karena untuk memproses algoritma genetika pada generasi tersebut masih kurang atau area pencarian secara optimal masih sempit. Semakin banyak ukuran generasi maka semakin banyak waktu proses yang dibutuhkan atau dengan kata lain semakin lama sistem menjalankan program. Dari grafik gambar 6.3 hasil pengujian ukuran generasi menghasilkan nilai *fitness* terbaik terjadi pada jumlah generasi ke-1300 dengan nilai *fitness* -505,4, karena setelah generasi 1300 nilai *fitness* optimal sulit untuk didapatkan atau dengan kata lain konvergen yaitu nilai *fitness* berdekatan sehingga tidak memungkinkan diproduksi kromosom yang lebih.

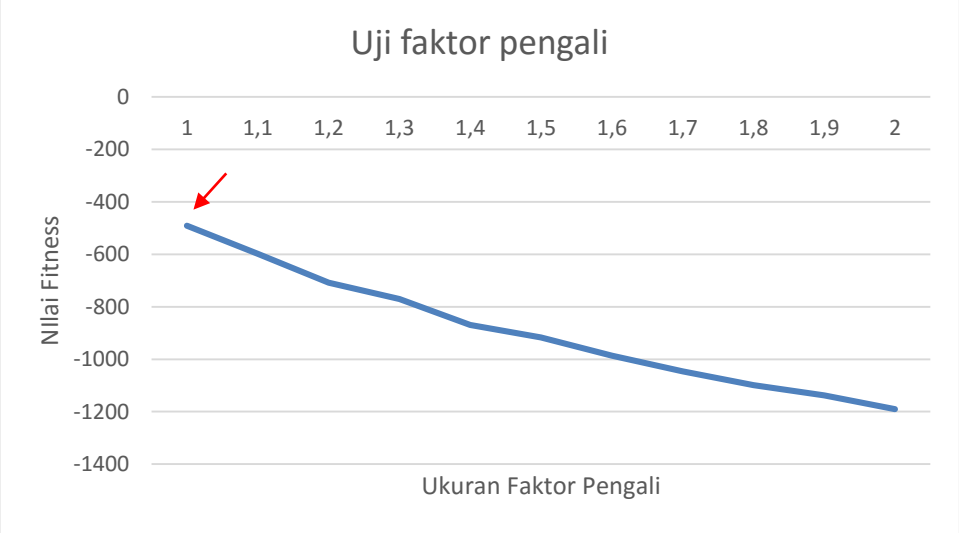
6.1.4 Hasil Pengujian Faktor Pengali

Pengujian faktor pengali ini digunakan untuk mengetahui apakah pada faktor pengali 1,2 sudah menghasilkan nilai fitness tertinggi pada sistem penjajaran sekuen jamak protein dengan algoritma genetika ini. Pada pengujian factor pengali ini menggunakan parameter generasi dan populasi atau popsize 100 serta nilai crossover rate 0,3 dan mutation rate 0,7. Pengujian ini dilakukan 10 kali dan diambil rata-rata. Hasil Pengujian factor pengali ini bisa dilihat pada tabel 6.4.

Tabel 6.4 Hasil Pengujian Faktor Pengali

Jumlah Popsize atau Populasi	100
Jumlah Generasi generasi	100
Crossover rate dan mutation rate	0.3 0.7
Faktor Pengali	**Nilai Fitness**
1	-491,5
1,1	-598,2
1,2	-707,8
1,3	-770,7
1,4	-869,4
1,5	-917,1
1,6	-986,2
1,7	-1046,7
1,8	-1098,6
1,9	-1138
2	-1189,8

Pada Tabel 6.4 di atas dapat diketahui pada nilai factor pengali 1 menghasilkan nilai fitness yang terbaik daripada yang lainnya dengan nilai fitness -491,5. Grafik hasil pengujian factor pengali dapat dilihat pada Gambar 6.4.



Gambar 6.4 Grafik Hasil Pengujian Faktor Pengali

Pada grafik diatas dapat kita lihat bahwa nilai factor pengali 1 menghasilkan nilai fitness terbaik dan setelahnya mengalami penurunan. Hal ini terjadi karena semakin besar factor pengali pada penghitungan maxlength maka semakin banyak gap yang ada dan semakin sedikit kemungkinan penjajaran sekuen jamak sejajar dengan sekuen yang sama.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dalam menerapkan algoritma genetika untuk penjajaran sekuen jamak protein, maka dapat disimpulkan sebagai berikut :

1. Algoritma genetika dapat menghasilkan nilai *fitness* yang optimal dari langkah yang diawali dengan mencari *maxlength* dari sekuen yang disejajarkan untuk insialisasi kromosom, kemudian sekuen dilakukan *encode* untuk inisialisasi populasi dari index array yang berisi *gap* dengan dibatasi oleh *maxlength* dan dilakukan pengacakan index array yang berisi *gap* sebanyak populasi. Setelah itu dilakukan reproduksi yaitu *crossover* metode one-cut point serta mutasi metode reciprocal exchange dengan jumlah *offspring* atau anak dari *crossover rate* dan *mutation rate* yang sudah ditentukan. Tahap evaluasi atau disini peneliti melakukan *decode* yaitu mengembalikan index array berisi *gap* ke dalam bentuk semula dengan diisi *gap* untuk dilakukan penghitungan nilai *fitness* menggunakan *sum of pairs* menggunakan *blosum62* serta skema affine yang berisi *gap* open dan *gap* extension. Tahap terakhir adalah seleksi *elitism* yaitu individu atau kromosom dengan nilai *fitness* terbaik sejumlah populasi atau *popsize* akan dipertahankan untuk dilakukan proses ke generasi selanjutnya.
2. Telah dilakukan beberapa pengujian untuk mengetahui seberapa optimalkah sistem ini berjalan. Pengujian yang dilakukan adalah pengujian pada parameter algoritma genetika yaitu *crossover rate* dan *mutation rate*, populasi atau *popsize* dan generasi. Hasil Pengujian ukuran populasi didapatkan parameter ukuran populasi (*popsize*) yang optimal pada populasi ke-3800 dengan nilai *fitness* -584,6. Pada pengujian jumlah generasi, optimal pada pengujian generasi ke-1300 yaitu sebesar -505,4. Untuk pengujian kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*) optimal pada pengujian *crossover rate* 0,3 dan *mutation rate* 0,7 dengan nilai *fitness* -685,2. Pengujian factor pengali 1 menghasilkan nilai *fitness* yang terbaik dengan nilai -491,5.

7.2 Saran

Saran yang dapat diberikan untuk penelitian berikutnya tentang penjajaran sekuen jamak protein dengan algoritma genetika ini adalah :

1. Pada penelitian ini pengujian hanya bisa dilakukan untuk 4 sekuen dengan panjang sekuen maksimal 36. Sebagai saran diharapkan dapat melakukan pengujian dengan data sekuen lebih dari 4 dan panjang lebih dari 36 sekuen.
2. Pada penelitian selanjutnya dapat dilakukan uji coba menggunakan metode lain sehingga dapat dibandingkan hasilnya dengan sistem penjajaran sekuen jamak protein dengan algoritma genetika ini.

DAFTAR PUSTAKA

Agarwal, P., 2013. Alignment of Multiple Sequence using GA method. *International Journal of Emerging Technologies in Computational and Applied Sciences(IJETCAS),* 4(4), pp. 411-421.

Anon., 2015. *NCBI.* [Online] Available at: http://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome [Accessed 5 Mei 2015].

Bilolikar, V. S., Jain, K. & Sharma, M. R., 2012. An Annealed Genetic Algorithm for Multi Mode Resource Constrained Project Scheduling Problem. *International Journal of Computer Applications,* 60(1), pp. 36-42.

Kadam, K., Sawant, S., Kulkarni-Kale, U. & Jayaraman, V. K., 2016. Prediction of Protein Function based on Machine Learning Methods: An Overview. pp. 1-38.

Kumar, M., 2015. Alignment of Multiple DNA Sequence by Using Improved GA Operators. *International Journal of Pharma Sciences and Research (IJPSR),* Volume 6, pp. 406-410.

Mahmudy, W. F., 2013. *Modul Algoritma Evolusi.* Malang: s.n.

Mawaddah, N. K. & Mahmudy, W. F., 2006. Optimasi Penjadwalan Ujian Menggunakan Algoritma Genetika. 2(2), pp. 1-8.

Primasoni, N., 2011. Manfaat Protein untuk Mendukung Aktifitas Olahraga, Pertumbuhan, dan Perkembangan Anak Usia Dini. pp. 1-11.

Sharma, K. R., 2009. *Bioinfomatics Sequence Alignment and Markov Models.* United States of America: The McGraw-Hill Companies.

Suprayogi, D. A. & Mahmudy, W. F., 2014. Penerapan Algoritma Genetika Traveling Salesman Problem with TimePenerapan Algoritma Genetika Traveling Salesman Problem with Time. 6(2), pp. 121-130.

Wargasetia, T. L., 2006. Peran Bioinformatika dalam Bidang Kedokteran. *Jurnal Kesehatan Masyarakat,* 5(2), pp. 59-72.

Wu, S., Lee, M., Lee, Y. & Gatton, T. M., 2008. Multiple Sequence Alignment using GA and NN. *International Journal of Signal Processing, Image Processing and Pattern Recognition,* pp. 21-30.

Xiong, J., 2006. *Essential Bioinformatics.* New York, United States of America: Cambridge University Press.

LAMPIRAN A CONTOH SEKUEN PROTEIN PADA MANUSIA

Sekuen Protein 1 :

>gi|582045545|pdb|4N0F|I Chain I, Human F*c*r*n* Complexed With Human Serum Albumin
IQRTPKIQVYSRHPAENGKSNFLNCYVSGFHPSDIEVDLLKNGERIEKVEHSDLSFSKDWSFYLLY
YTEFTPTEKDEYA*CR*VNHVTLSQPKIVKWDRDM

Sekuen Protein 2 :

>gi|582045544|pdb|4N0F|H Chain H, Human F*c*r*n* Complexed With Human Serum Albumin
HLSLLYHLTAVSSPAPGTPAFWVSGWLGPQQYLSYNSLRGEAEPCGAWVWENQVSWYWEKETTDLR
IKEKLFLEAFKALGGKGPYTLQGLLGCELGPDNTSVPTAKFALNGEEFMNFDLKQGTWGGDWPEAL
AISQRWQQQDKAANKELTFLLFSCPHRLREHLERGRGNLEWKEPPS*MR*LKARPSSPGFSVLTCSAF
SFYPPELQLRFLRNGLAAGTGQGDFGPNSDGSFHASSSLTVKSGDEHHYCCIVQHAGLAQPLRVEL
ESPAKSS

Sekuen Protein 3 :

>gi|4826643|ref|NP_005130.1| annexin A3 [Homo sapiens]
MASIWVGHRGTVRDYPDFSPSVDAEAIQKAIRGIGTDEKMLISILTERSNAQRQLIVKEYQAAYGK
ELKDDLKGDLSGHFEHLMVALVTPPAVFDAKQLKKSMKGAGTNEDALIEILTTRTSRQMKDISQAY
YTVYKKSLGDDISSETSGDFRKALLTLADGRRDESLKVDEHLAKQDAQILYKAGENRWGTDEDKFT
EILCLRSFPQLKLTFDEYRNISQKDIVDSIKGELSGHFEDLLLAIVNCVRNTPAFLAERLHRALKG
IGTDEFTLNRIMVSRSEIDLLDIRTEFKKHYGYSLYSAIKSDTSGDYEITLLKICGGDD

Sekuen Protein 4 :

>gi|4557653|ref|NP_000191.1| histatin-3 precursor [Homo sapiens]

MKFFVFALILALMLSMTGADSHAKRHHGYKRKFHEKHHSHRGYRSNYLYDN

Sekuen Protein 5 :

>gi|4507431|ref|NP_003207.1| thyrotroph embryonic factor isoform 1 [Homo sapiens]

MSDAGGGKKPPVDPQAGPGPGPGRAAGERGLSGSFPLVLKKLMENPPREARLDKEKGKEKLEEDEA
AAASTMAVSASLMPPIWDKTIPYDGESFHLEYMDLDEFLLENGIPASPTHLAHNLLLPVAELEGKE
SASSSTASPPSSSTAIFQPSETVSSTESSLEKERETPSPIDPNCVEVDVNFNPDPADLVLSSVPGG
ELFNPRKHKFAEEDLKPQPMIKKAKKVFVPDEQKDEKYWTRRKKNNVAAKRSRDARRLKENQITIR
AAFLEKENTALRTEVAELRKEVGKCKTIVSKYETKYGPL

Sekuen Protein 6 :

>gi|4507015|ref|NP_001850.1| high affinity copper uptake protein 1 [Homo sapiens]

MDHSHHMGMSYMDSNSTMQPSHHHPTTSASHSHGGGDSSMMMMPMTFYFGFKNVELLFSGLVINTA
GEMAGAFVAVFLLAMFYEGLKIARESLLRKSQVSIRYNSMPVPGPNGTILMETHKTVGQQMLSFPH
LLQTVLHIIQVVISYFLMLIFMTYNGYLCIAVAAGAGTGYFLFSWKKAVVVDITEHCH

Sekuen Protein 7 :

>gi|4504343|ref|NP_003813.1| nuclear receptor subfamily 5 group A member 2 isoform 2 [Homo sapiens]

MSSNSDTGDLQESLKHGLTPIVSQFKMVNYSYDEDLEELCPVCGDKVSGYHYGLLTCESCKGFFKR
TVQNNKRYTCIENQNCQIDKTQRKRCPY*CR*FQKCLSVGMKLEAVRADR*MR*GGRNKFGPMYKRDRAL
KQQKKALIRANGLKLEAMSQVIQAMPSDLTISSAIQNIHSASKGLPLNHAALPPTDYDRSPFVTSP
ISMTMPPHGSLQGYQTYGHFPSRAIKSEYPDPYTSSPESIMGYSYMDSYQTSSPASIPHLILELLK

CEPDEPQVQAKIMAYLQQEQANRSKHEKLSTFGLMCKMADQTLFSIVEWARSSIFFRELKVDDQMK
LLQNCWSELLILDHIYRQVVHGKEGSIFLVTGQQVDYSIIASQAGATLNNLMSHAQELVAKLRSLQ
FDQREFVCLKFLVLFSLDVKNLENFQLVEGVQEQVNAALLDYTMCNYPQQTEKFGQLLLRLPEIRA
ISMQAEEYLYYKHLNGDVPYNNLLIEMLHAKRA

Sekuen Protein 8 :

>gi|4503079|ref|NP_000750.1| granulocyte colony-stimulating factor isoform a precursor [Homo sapiens]

MAGPATQSPMKLMALQLLLWHSALWTVQEATPLGPASSLPQSFLLKCLEQVRKIQGDGAALQEKLV
SECATYKLCHPEELVLLGHSLGIPWAPLSSCPSQALQLAGCLSQLHSGLFLYQGLLQALEGISPEL
GPTLDTLQLDVADFATTIWQQMEELGMAPALQPTQGAMPAFASAFQRRAGGVLVASHLQSFLEVSY
RVLRHLAQP

Sekuen Protein 9 :

>gi|21730550|pdb|1KW2|B Chain B, *C*rystal Structure Of Uncomplexed Vitamin D-Binding Protein

LERGRDYEKNKVCKEFSHLGKEDFTSLSLVLYSRKFPSGTFEQVSQLVKEVVSLTEACCAEGADPD
CYDTRTSALSAKSCESNSPFPVHPGTAECCTKEGLERKLCMAALKHQPQEFPTYVEPTNDEICEAF
RKDPKEYANQFMWEYSTNYGQAPLSLLVSYTKSYLSMVGSCCTSASPTVCFLKERLQLKHLSLLTT
LSNRVCSQYAAYGEKKSRLSNLIKLAQKVPTADLEDVLPLAEDITNILSKCCESASEDCMAKELPE
HTVKLCDNLSTKNSKFEDCCQEKTAMDVFVCTYFMPAAQLPELPDVELPTNKDVCDPGNTKVMDKY
TFELSRRTHLPEVFLSKVLEPTLKSLGECCDVEDSTTCFNAKGPLLKKELSSFIDKGQELCADYSE
NTFTEYKKKLAERLKAKLPDATPKELAKLVNKRSDFASNCCSINSPPLYCDSEIDAELKNIL

Sekuen Protein 10 :

>gi|21730549|pdb|1KW2|A Chain A, *C*rystal Structure Of Uncomplexed Vitamin D-Binding Protein

LERGRDYEKNKVCKEFSHLGKEDFTSLSLVLYSRKFPSGTFEQVSQLVKEVVSLTEACCAEGADPD
CYDTRTSALSAKSCESNSPFPVHPGTAECCTKEGLERKLCMAALKHQPQEFPTYVEPTNDEICEAF
RKDPKEYANQFMWEYSTNYGQAPLSLLVSYTKSYLSMVGSCCTSASPTVCFLKERLQLKHLSLLTT
LSNRVCSQYAAYGEKKSRLSNLIKLAQKVPTADLEDVLPLAEDITNILSKCCESASEDCMAKELPE
HTVKLCDNLSTKNSKFEDCCQEKTAMDVFVCTYFMPAAQLPELPDVELPTNKDVCDPGNTKVMDKY
TFELSRRTHLPEVFLSKVLEPTLKSLGECCDVEDSTTCFNAKGPLLKKELSSFIDKGQELCADYSE
NTFTEYKKKLAERLKAKLPDATPKELAKLVNKRSDFASNCCSINSPPLYCDSEIDAELKNIL

Sekuen Protein 11 :

>gi|22219267|pdb|1LOT|A Chain A, *C*rystal Structure Of The Complex Of Actin With Vitamin D-Binding Protein

LERGRDYEKNKVCKEFSHLGKEDFTSLSLVLYSRKFPSGTFEQVSQLVKEVVSLTEACCAEGADPD
CYDTRTSALSAKSCESNSPFPVHPGTAECCTKEGLERKLCMAALKHQPQEFPTYVEPTNDEICEAF
RKDPKEYANQFMWEYSTNYGQAPLSLLVSYTKSYLSMVGSCCTSASPTVCFLKERLQLKHLSLLTT
LSNRVCSQYAAYGEKKSRLSNLIKLAQKVPTADLEDVLPLAEDITNILSKCCESASEDCMAKELPE
HTVKLCDNLSTKNSKFEDCCQEKTAMDVFVCTYFMPAAQLPELPDVELPTNKDVCDPGNTKVMDKY
TFELSRRTHLPEVFLSKVLEPTLKSLGECCDVEDSTTCFNAKGPLLKKELSSFIDKGQELCADYSE
NTFTEYKKKLAERLKAKLPDATPTELAKLVNKRSDFASNCCSINSPPLYCDSEIDAELKNIL

Sekuen Protein 12 :

>gi|90108563|pdb|1YW9|A Chain A, H-Metap2 Complexed With A849519

KVQTDPPSVPICDLYPNGVFPKGQECEYPPTQDGRTAAWRTTSEEKKALDQASEEIWNDFREAAEA
HRQVRKYVMSWIKPGMTMIEICEKLEDCSRKLIKENGLNAGLAFPTGCSLNNCAAHYTPNAGDTTV
LQYDDICKIDFGTHISGRIIDCAFTVTFNPKYDTLLKAVKDATNTGIKCAGIDVRLCDVGEAIQEV
MESYEVEIDGKTYQVKPIRNLNGHSIGQYRIHAGKTVPIIKGGEATRMEEGEVYAIETFGSTGKGV
VHDDMECSHYMKNFDVGHVPIRLPRTKHLLNVINENFGTLAF*CR*RWLDRLGESKYLMALKNLCDLG
IVDPYPPLCDIKGSYTAQFEHTILLRPTCKEVVSRGDDY

LAMPIRAN B SEKUEN PROTEIN PADA MANUSIA UNTUK PENGUJIAN

Sekuen Protein 1 :
>gi|546033|gb|AAB30282.1| albumin, partial [Homo sapiens]
ADLAKYICEKSRFDLQ

Sekuen Protein 1 :
>gi|247527|gb|AAB21839.1| albumin [human, Peptide Partial Mutant, 25 aa]
FLYKYARRHPDYSVVLLLRLAKTYE

Sekuen Protein 3 :
>gi|553173|gb|AAA51688.1| albumin, partial [Homo sapiens]
MKWVTFISLLFLFSSAYSRGVFRRDA

Sekuen Protein 4 :
>gi|163931174|pdb|3BF6|L Chain L, Thrombin:suramin Complex
TFGSGEADCGLRPLFEKKSLEDKTERELLESYIDGR

LAMPIRAN C MATRIK *BLOSUM62*

Matrix made by matblas from *blosum62*.iij

* column uses minimum score

BLOSUM Clustered Scoring Matrix in 1/2 Bit Units

Blocks Database = /data/blocks_5.0/blocks.dat

Cluster Percentage: >= 62

Entropy = 0.6979, Expected = -0.5209

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-1	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Catatan :

1. Bernilai Positif = Frekuensi substitusi asam amino ditemukan di urutan homolog himpunan data sekuen lebih baik dari yang terjadi kebetulan secara acak.
2. Bernilai Nol = Frekuensi substitusi asam amino ditemukan homolog himpunan data sekuen yang sama dengan yang terjadi secara kebetulan.

3. Bernilai Negatif = Frekuensi substitusi asam amino ditemukan homolog himpunan data sekuen kurang dari yang terjadi kebetulan secara acak.
4. Bintang pada akhir kolom dan baris adalah nilai yang diabaikan untuk protein yang tidak diketahui (Xiong, 2006).