TUGAS AKHIR - SM141501

KAJIAN PENEMPATAN LIGAN PADA PROTEIN MENGGUNAKAN PENDEKATAN ALGORITMA GENETIKA

HARTANTO SETIAWAN
NRP 1213 100 022

Dosen Pembimbing
Prof. Dr. Mohammad Isa Irawan, MT

DEPARTEMEN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT - SM141501

STUDY OF LIGAN PLACEMENT ON THE PROTEIN USING A GENETIC ALGORITHM APPROACH

HARTANTO SETIAWAN
NRP 1213 100 022

Supervisor
Prof. Dr. Mohammad Isa Irawan, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Sciences
Sepuluh Nopember Institute of Technology
Surabaya 2017

LEMBAR PENGESAHAN

KAJIAN PENEMPATAN LIGAN PADA PROTEIN MENGGUNAKAN PENDEKATAN ALGORITMA GENETIKA

STUDY OF LIGAN PLACEMENT ON THE PROTEIN USING A GENETIC ALGORITHM APPROACH

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Sains
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
Hartanto Setiawan
NRP. 1213 100 022

Menyetujui,

Dosen Pembimbing ,

Prof. DR. Mohammad Isa Irawan, MT
NIP. 19631225 198903 1 001

Mengetahui,
Kepala Departemen Matematika
FMIPA ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, Juni 2017

KAJIAN PENEMPATAN LIGAN PADA PROTEIN MENGGUNAKAN PENDEKATAN ALGORITMA GENETIKA

Nama Mahasiswa	: Hartanto Setiawan
NRP	: 1213 100 022
Departemen	: Matematika
Dosen Pembimbing	: Prof. Dr. M. Isa Irawan, MT

Abstrak

Penempatan ligan pada protein atau *molecule docking* merupakan bidang komputasi yang sedang berkembang. Metode *molecular docking* adalah metode yang bermanfaat untuk mencari kombinasi interaksi protein dan ligan serta menjadi dasar penemuan obat secara simulasi. *Molecular docking* yang digunakan adalah *flexible docking* dan jenis *protein-ligand docking.* Pendekatan algoritma genetika merupakan metode alternatif yang bisa digunakan untuk simulasi *molecular docking*. Hasil dari pendekatan algoritma genetika yaitu berupa penempatan posisi docking yang optimum. Penerapan algoritma genetika dalam docking tidak berlaku untuk semua protein dan ligan. Dalam penerapannya tingkat homologi mempengaruhi keberhasilan dari docking.

Kata Kunci : Algoritma Genetika, Molecular docking, Protein-ligand docking, Flexible docking

STUDY OF LIGAN PLACEMENT ON THE PROTEIN USING A GENETIC ALGORITHM APPROACH

Name of Student	: Hartanto Setiawan
NRP	: 1213 100 022
Department	: Mathematics
Supervisor	: Prof. Dr. M. Isa Irawan, MT

Abstract

The placement of the ligand on a docking protein molecules or liquid computing is being developed. Molecular docking methods are methods that are useful for creating a combination of protein and ligand interactions as well as the basis of drug discovery in the simulation. Molecular docking that is used in is a flexible docking and the type of protein-ligand docking. Genetic algorithm approach to illiquid alternative methods that can be used to define the molecular docking simulations. The result of the genetic algorithm approach applies in the form of an optimal docking position placement. Application of genetic algorithm in the docking does not apply to all the protein and Ligand. In its application-level homology affect the success of the Dock

Keywords : Genetika Algorithm, Molecular docking, Protein-ligand docking, Flexible docking

KATA PENGANTAR

Segala puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan ridlo-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul

"KAJIAN PENEMPATAN LIGAN PADA PROTEIN MENGGUNAKAN PENDEKATAN ALGORITMA GENETIKA"

yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departmen Matematika, Fakultas Matematika dan Ilmu Pengethuan Alam, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal tersebut, penulis ingin mengucapkan terima kasih kepada :

1. Dr. Imam Mukhlash, S.Si, MT selaku Kepala Departemen Matematika ITS.
2. Dra. Nur Asiyah, M.Si selaku Dosen Wali yang telah memberikan arahan akademik selama penulis menempuh pendidikan di Departemen Matematika ITS.
3. Prof. Dr. Mohammad Isa Irawan, MT selaku Dosen Pembimbing yang telah memberikan bimbingan dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini sehingga dapat terselesaikan dengan baik.
4. Dr. Didik Khusnul Arif, S.Si, M.Si selaku Ketua Program Studi S1 Departemen Matematika ITS.
5. Drs. Iis Herisman, M.Si selaku Sekretaris Program Studi S1 Departemen Matematika ITS.
6. Seluruh jajaran dosen dan staf Departemen Matematika ITS.
7. Keluarga tercinta yang senantiasa memberikan dukungan dan do'a yang tak terhingga.
8. Teman-teman angkatan 2013 yang saling mendukung dan memotivasi.

9. Semua pihak yang tak bisa penulis sebutkan satu-persatu, terima kasih telah membantu sampai terselesaikannya Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik dan saran dari pembaca. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Juni 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang yang mendasari penulisan Tugas Akhir ini. Di dalamnya mencakup identifikasi permasalahan pada topik Tugas Akhir kemudian dirumuskan menjadi permasalahan yang diberikan batasan-batasan dalam pembahasan pada Tugas Akhir ini.

1.1 Latar Belakang

Seiring berkembangnya teknologi informasi, komputasi pengolahan data menjadi lebih cepat untuk dilakukan. Salah satu bidang teknologi yang berkembang sekarang ini adalah komputasi biologi. Komputasi biologi adalah bidang ilmu yang berfokus pada penyusunan sebuah model matematika dalam menyelesaikan dan menganalisis masalah sekuen biologi. Selain itu dalam perkembangan komputasi biologi juga digunakan untuk menemukan prinsip-prinsip baru yang mendasar dalam ilmu biologi seperti pencarian obat

Komputasi pada bidang biologi atau dikenal sebagai bioinformatika, pada umumnya merupakan kombinasi biologi dan komputasi dimana menggunakan aplikasi dari alat komputasi dan analisis untuk menangkap dan menginterpretasikan data-data biologi. Salah satu nya seperti perkembangan teknologi DNA rekombinan merupakan suatu pengetahuan baru dalam rekayasa genetika organisme yang dikenal sebagai bioteknologi.

Bidang kajian bioinformatika yang sedang berkembang sekarang ini adalah *molecular docking* (penempatan molekul). *Molecular docking* merupakan metode berbasis genetika yang dapat digunakan untuk mencari pola interaksi yang paling tepat dan melibatkan antara dua molekul, yaitu reseptor dan ligan. Ligan sendiri merupakan molekul sinyal kecil yang terlibat dalam kedua proses anorganik dan biokimia. *Molecular docking* bertujuan meniru peristiwa interaksi suatu

molekul ligan dengan protein yang menjadi targetnya pada uji in-vitro [2]. Molecular docking dapat diklasifikasikan menjadi 3 berdasarkan fleksibilitas molekul yaitu *Rigid Docking* (bersifat rigid/kaku), *semi-fleksible docking* (bersifat semi fleksibel) dan *fleksible docking* (bersifat fleksibel). Tujuan dari docking adalah untuk mencapai konformasi protein dan ligan yang optimal. Docking membantu dalam mempelajari obat / ligan atau interaksi reseptor / protein dengan mengidentifikasi situs aktif yang cocok pada protein, mendapatkan geometri terbaik dari kompleks ligan – reseptor. Docking menjadi dasar untuk penemuan obat secara simulasi komputasi. Langkah pertama dari desain obat dibantu komputer adalah menemukan situs pengikatan ligan protein, yang merupakan kantong atau celah pada permukaan protein yang digunakan untuk mengikat ligan (obat terlarang) [16].

Dengan peningkatan jumlah struktur biologi molekul yang tersedia, pendekatan docking telah menjadi alat yang sangat penting dan berguna dalam penemuan obat rasional berbasis struktur dan desain [2]. Untuk reseptor protein dengan struktur tiga dimensi yang dikenal, masalahnya docking ligan-protein pada dasarnya terdiri dalam memprediksi konformasi terikat ligan molekul dalam situs aktif protein.

Berbagai metode komputasi telah dikembangkan untuk menentukan struktur dan fungsi protein dan mempelajari lebih lanjut tentang protein lipat mekanisme[3]. Efisien komputasi teknik dapat membantu untuk memecahkan masalah struktur protein (yaitu Homologi pemodelan, threading dan ab initio), yang memungkinkan aplikasi yang beragam dalam banyak bidang penelitian scientifik [3]. Pengetahuan tentang struktur 3D protein ini juga sangat penting untuk studi mitra makromolekul lain dengan siapa mereka dapat berinteraksi. Selain itu dengan penerapan komputasi dalam hal ini juga dapat mengurangi biaya yang sangat besar, dan menghemat waktu dalam rekombinasinya, serta mempermudah dalam perhitungan kompleks suatu rekombinasi. Dalam hal ini

masalah utama docking adalah sulit optimasi yang melibatkan banyak derajat kebebasan, dan pengembangan efisien docking algoritma dan metodologi akan menjadi manfaat besar dalam desain obat baru. Dalam hal ini pendekatan algoritma optimasi sangat membantu optimasi untuk mendapatkan desian obat secara simulasi. Kombinasi dari algoritma genetika (GA) [4,5], rotamer Perpustakaan dan dinamika molekular (MD) atau normal mode (NM) [6,7] dapat merupakan pendekatan yang baik untuk melakukan docking studi. Algoritma genetika juga dapat digunakan untuk menyelesaiakan permasalahan CVRP, Penempatan pegawai, penjadwalan, optimasi BTS, dll [17, 18, 19, 20]

Berdasarkan latar belakang diatas, bagaimana mengenai pendekatan penempatan ligan pada protein. Jenis molecular docking yang digunakan adalah fleksible docking. Pendekatan yang digunakan adalah algoritma genetika. Algoritma genetika digunakan untuk rekombinasi dan untuk mencari kombinasi terbaik dari struktur konformasi protein.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat dirumuskan permasalahan dalam Tugas Akhir ini adalah bagaimana penempatan ligan pada protein dengan menggunakan pendekatan algoritma genetika.

1.3 Batasan Masalah

Pada penelitian ini, penulis membuat batasan masalah sebagai berikut :

1. *Molecular docking* yang digunakan adalah *flexible docking*.
2. Protein virus yang digunakan adalah *plasmodium malariae* dan *dengue virus*.
3. Ligan yang digunakan adalah tumbuhan kina dan jambu merah.

1.4 Tujuan

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, tujuan penelitian Tugas Akhir ini adalah untuk mengetahui pendekatan algoritma genetika dalam permasalah penempatan ligan pada protein.

1.5 Manfaat

Manfaat dalam penulisan Tugas Akhir ini adalah mendapatkan performansi penggunaan algoritma genetika pada rekombinasi ligan dengan protein, hasil dari algoritma ini diharapkan dapat memberikan kemudahan dalam rekombinasi protein dan ligan. Dalam uji in vitro untuk pembuatan obat dibutuhkan biaya yang mahal. Dengan docking molekul dapat mengurangi biaya yang mahal dan perhitungan yang kompleks.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika dari penulisan Tugas Akhir ini adalah sebagai berikut :

1. BAB I PENDAHULUAN
   Bab ini menjelaskan tentang gambaran umum dari penulisan Tugas Akhir ini yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan, manfaat penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Bab ini berisi tentang materi-materi yang mendukung Tugas Akhir ini, antara lain penelitian terdahulu, kajian tentang protein dan ligan, gaya intermolekuler, *molecular docking* dan klasifikasi berdasarkan ligan dan fleksibilitas molekul, Algoritma Genetika, dan fungsi evaluasi *flexible docking*.
3. BAB III METODOLOGI PENELITIAN
   Pada bab ini dibahas tentang langkah – langkah dan metode yang digunakan untuk menyelesaikan Tugas Akhir ini.
4. BAB IV PERANCANGAN IMPLEMENTASI

Pada bab ini akan menguraikan bagaimana tahapan tahapan dalam perancangan implementasi. Pembahasan perancangan implementasi dimulai dari pencarian parameter untuk *molecular docking*

5. BAB V IMPLEMENTASI
   Bab ini menjelaskan mengenai implementasi Algoritma Genetika untuk penempatan ligan pada protein . Setelah itu, dilakukan analisis terhadap hasil implementasi. Analisis ini bertujuan untuk melihat apakah Algoritma Genetika dapat diterapkan untuk penempatan ligan pada protein sehingga didapat nilai evaluasi yang rendah.
6. BAB VI PENUTUP
   Bab ini berisi kesimpulan yang diperoleh dari pembahasan masalah sebelunya serta saran yang diberikan untuk pengembangan selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai dasar teori yang digunakan dalam penyusunan Tugas Akhir ini. Dasar teori yang dijelaskan dibagi menjadi beberapa subbab yaitu protein dan ligan, *molecular docking* serta klasifikasinya, Algoritma genetika dan fungsi evaluasi untuk *flexible docking*.

2.1 Penelitian Terdahulu

Penelitian tentang metode komputasi untuk membuat kombinasi protein dan ligan sudah dilakukan. Berbagai metode komputasi telah dikembangkan untuk menentukan struktur dan fungsi protein dan mempelajari lebih lanjut tentang protein lipat mekanisme[3].

de Magalhães telah mengembangkan metode docking semi-flexible yang menggunakan algoritma genetika yang disebut 'steady-state genetika' (SSGA), yang menghasilkan hanya satu kromosom baru untuk setiap generasi untuk mensimulasikan docking, dan ligan flexible dengan kemungkinan untuk bergerak (tranlasi dan rotasi) , simulasi menggunakan GROMOS96 [5].

Spyrakis membandingkan 19 set protein-ligan komplek dengan menggunakan program docking AutoDock, GOLD dan Flex [8]. Thomsen dan Christensen telah mengembangkan algoritma docking semi-flexible yang disebut MolDock di mana mereka digunakan pendekatan heuristic untuk memprediksi rongga dalam protein [9].

Huang dan Zou mengembangkan algoritma docking semi-flexible yang menggunakan beberapa struktur protein untuk memperhitungkan flexibility tulang punggung dalam kaitannya dengan ligan mengikat [10].

Guerler et al. telah mengembangkan metode docking semi-flexible yang disebut Fado yang menganggap beberapa

pra-diproses ligan konformasi untuk memperhitungkan ligan flexibility [11].

Angelica Nakagawa et al. telah mengembangkan metode pendekatan protein-ligand docking dengan menggunaan algoritma genetika dan normal mode [15]. Dimana melakukan 2 percobaan dengan versi pertama menggunakan rigid dan semi-fleksible docking dan veris kedua menggunakan normal mode. Hasil penelitian terdahulu menunjukkan pendekatan algoritma genetika merupakan pendekatan yang baik untuk simulasi docking rekombinasi penempatan protein dan ligan.

2.2 Protein

Protein adalah makromolekul yang paling banyak ditemukan di dalam sel makhluk hidup dan merupakan 50 persen atau lebih dari berat kering sel. Protein memiliki jumlah yang sangat bervariasi yang mulai dari struktur maupun fungsinya. Protein dan gen memiliki hubungan yang sangat dekat dimana kode genetika berupa DNA dienkripsi dalam bentuk kromosom yang selanjutnya kode genetika tersebut ditranslasikan menjadi protein melalui serangkain mekanisme yang melibatkan.Protein tersusun dari peptida-peptida sehingga membentuk suatu polimer yang disebut polipeptida. Setiap monomernya tersusun atas suatu asam amino.

```
SENDVIELDDVANLMFYGEGEVGDNHQKFMLIFDTGSANLWVPSKKCNSIGCSTKHLYDSSKSKSY
EKDGTKVEITYGSGTVRGFFSKDLVTLGYLSLPYKFIEVTDTDDLEPLYTAAEFDGILGLGWKDLS
IGSIDPIVVELKNQNKIDQALFTFYLPVHDKHSGYLTIGGIEEKFYEGELTYEKLNHDLFWQVDLD
VNFGKTSMEKANVIVDSGTSTITAPTSFINKFFKDLNVIKVPFLPFYITTCNNKDMPTLEFKSANN
TYTLEPEYYMEPLLDIDDTLCMLYILPVDIDKNTFILGDPFMRKYFTVFDYDKESIGFAVAKN
```

Gambar 2. 1 Sekuen protein malaria

Asam amino dalam suatu protein memiliki bentuk L, terionisir dalam larutan, dan memiliki bentuk C asimetris kecuali asam amino jenis glisin. Di dalam protein tersusun 20 macam asam amino yang memiliki karakteristik yang bebeda-

beda sehingga dapat dikelompokkan berdasarkan sifat dan ciri rantai sampingnya (gugus R).

No	Asam amino	symbol	nucleotide sekuen	complement
1	methionine	M	ATG	TAC
2	tryptophan	W	TGG	ACC
3	cysteine	C	TGY	ACR
4	aspartic acid	D	GAY	CTR
5	glutamic acid	E	GAR	CTY
6	phenylalanine	F	TTY	AAR
7	histidine	H	CAY	GTR
8	lysine	K	AAR	TTY
9	asparagine	N	AAY	TTR
10	glutamine	Q	CAR	GTY
11	tyrosine	Y	TAY	ATR
12	inosine	I	ATH	TAD
13	alanine	A	GCN	CGN
14	glycine	G	GGN	CCN
15	proline	P	CCN	GGN
16	threonine	T	CAN	TGN
17	valine	V	GTN	CAN
18	leucine	L	YTN	RAN
19	arginine	R	MGN	KCN
20	serine	S	WSN	WSN

Gambar 2. 2 Asam amino

2.3 Ligan

Ligan molekul (netral/ anion) yang dapat menyumbangkan sepasang elektron bebas, guna membentuk ikatan kovalen koordinasi dengan ion logam transisi. Ligan merupakan basa Lewis yang memiliki pasangan elektron bebas (lone pair electron), atau memiliki pasangan electron. Di dalam ligan terdapat atom donor yaitu atom yang memiliki pasangan elektron bebas atau atom yang terikat melalui ikatan. Melalui atom-atom donor tersebut suatu ligan mengadakan ikatan kovalen koordinasi dengan atom atau ion pusat yang ada.

```
NISQHQCVKKQCPQNSGCFRHLDEREECKCLLNYKQEGDKCVENPNPTCNENNGGCDADAKCTEED
SGSNGKKITCECTKPDSYPLFDGIFCSSSN
```

Gambar 2. 3 Sekuen Ligan

2.4 Mekanika Molekul

Mekanika molekul merupakan suatu metode empiris yang digunakan untuk menyatakan energi potensial dari molekul sebagai fungsi dari variabel geometri. Elektron tidak dipertimbangkan secara ekplisit dan fungsi energi potensial bergantung pada posisi inti.

Secara umum medan gaya disusun untuk suatu golongan yang spesifik dari molekul. Medan gaya yang dapat digunakan untuk semua golongan senyawa belum tersedia sampai sekarang. Beberapa contoh medan gaya mekanika molekular antara lain AMBER, CHARMM, GROMOS, MM3 dan lain-lain.

Model mekanika molekul dikembangkan untuk mendiskripsikan struktur dan sifat-sifat molekul sesederhana mungkin. Bidang aplikasi mekanika molekular meliputi:

1. Molekul yang tersusun oleh ribuan atom
2. Molekul organik, oligonukleotida, peptida dan sakarida
3. Molekul dalam lingkungan vakum atau berada dalam pelarut
4. Senyawa dalam keadaan dasar
5. Sifat-sifat termodinamika dan kinetika (melalui dinamika molekul)

2.4.1 Anatomi Mekanika Molekular

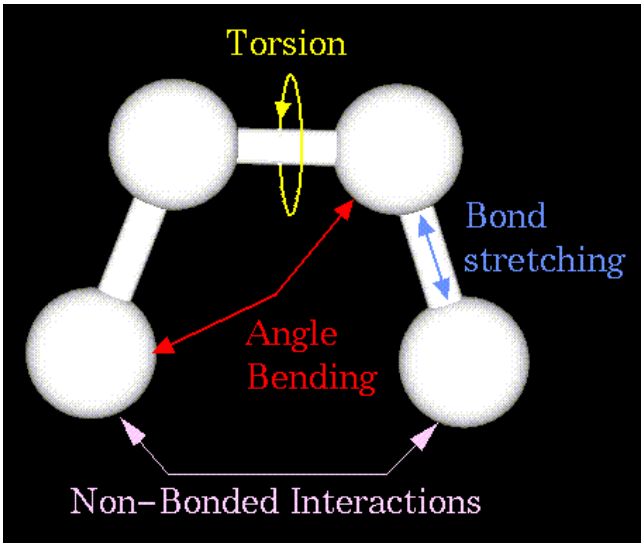
Di dalam model mekanika molekular (MM) atom-atom dipandang sebagai bola pejal dan ikatan antar atom sebagai pegas. Persamaan deformasi pegas dapat digunakan untuk menggambarkan kemampuan ikatan untuk merentang (strech), membengkok (bend) dan memilin (twist).

Model MM juga didasarkan pada energi atom-atom tak berikatan (non-bonded atom) yang berinteraksi melalui tolakan van der Waals dan tolakan elektrostatik. Sifat-sifat tersebut di atas paling mudah untuk digambarkan secara matematis jika

atom-atom dipandang sebagai bola dengan jari-jari yang spesifik.

Pada prinsipnya tujuan dari model MM adalah meramalkan energi berkaitan dengan konformasi tertentu dari molekul. Akan tetapi energi MM tidak memiliki makna sebagai kuantitas mutlak. Persamaan energi MM secara sederhana dapat dinyatakan sebagai :

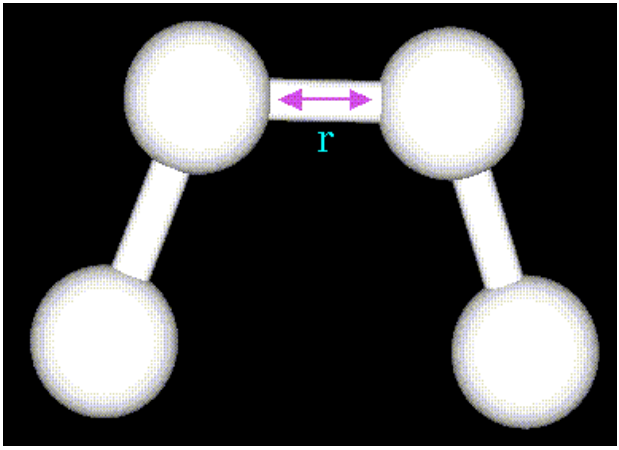
$$\text{Energy} = \text{Stretching energy} + \text{Bending energy} + \text{Torsi energy} + \text{Non} - \text{Bonded Interaction Energy}$$



Gambar 2. 4 Gambar Anatomi Mekanika Molekul

2.4.1.1 Energi Rentangan (Streching Energy)

Persamaan energi rentangan didasarkan atas hukum Hooke. Parameter k_b mengontrol kemiringan dari pegas ikatan, semetara r_0 adalah panjang ikatan dalam keseimbangan. Persamaan ini mengestimasi energi yang berkaitan dengan vibrasi di sekitar panjang ikatan kesetimbangan.



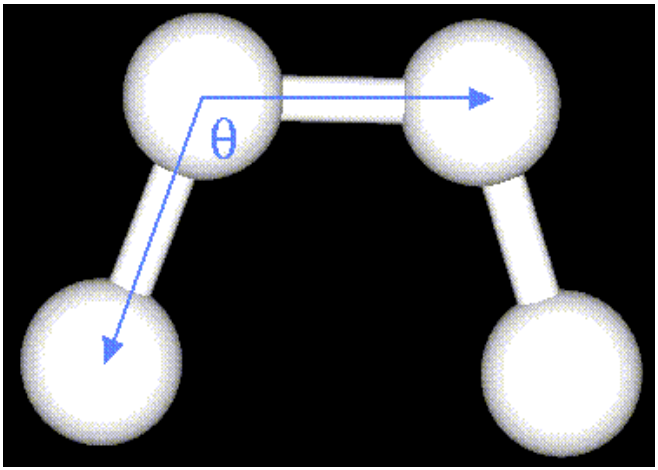
Gambar 2. 5 *Stretching energy*

Untuk perhitungan fungsi energi dari *stretching energy* adalah sebagai berikut :

$$E_s = \sum_{\substack{b=1 \\ Bonds}}^{N} \frac{k_b}{2}(r - r_0)^2 \qquad (1)$$

2.4.1.2 Energi Bengkokan (*Bending energy*)

Persamaan energi bengkokan atau tekukan juga didasarkan pada hukum hooke. Parameter k_θ digunakan untuk mengontrol kemiringan pegas sudut, semetara θ_0 menunjukkan sudut kesetimbangan.



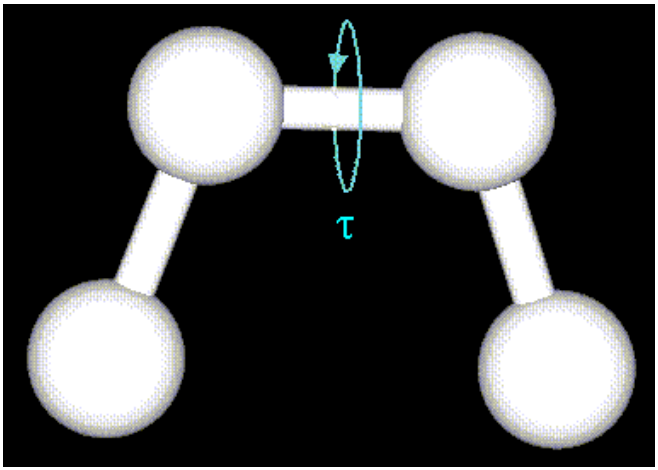
Gambar 2. 6 *Bending energy*

Untuk perhitungan fungsi energi dari *bendiing energy* adalah sebagai berikut :

$$E_b = \sum_{\substack{\theta=1 \\ angles}}^{N} \frac{k_\theta}{2}(\theta - \theta_0)^2 \qquad (2)$$

2.4.1.3 Energi Torsi (*Torsion energy*)

Sudut torsi merupakan sudut yang dibentuk oleh dua bidang yang saling berhubungan. Persamaan energi yang digunakan juga didasarkan pada hukum Hooke. τ merupakan sudut torsi, A adalah amplitude atau ketinggian fungsi, n adalah multiplisitas atau jumlah maksimal fungsi, dan fase adalah faktor fasa yang ditentukan ke arah mana sudut torsi bergeser melalui harga minimumnya.



Gambar 2. 7 *Torsion energy*

Untuk perhitungan dari energi torsi (*Torsion energy*) adalah sebagai berikut :

$$E_{tor} = \sum_{\substack{b=1 \\ torsions}}^{N} \frac{V_b}{2}[1 + \cos(n\tau - \phi)] \qquad (3)$$

2.4.1.4 Energi Tak-Berikatan (*Non-Bonded Energy*)

Energi tak-berikatan mewakili jumlah energi semua interaksi tak-berikatan antara atom i dan j. Energi tak-berikatan memperhitungkan tarikan, tolakan van der Waals dan interaksi elektrostatik. Tolakan van der Waals terjadi pada jarak yang pendek dan hilang jika atom-atom yang berin-teraksi menjauh satu sama lain dalam beberapa angstrom. Tolakan terjadi jika jarak antara atom-atom yang berinteraksi memendek kurang dari jumlah jari-jari kontak. Suku energi yang mendiskripsikan tarikan/tolakan sering dimodelkan dengan persamaan 6-12.

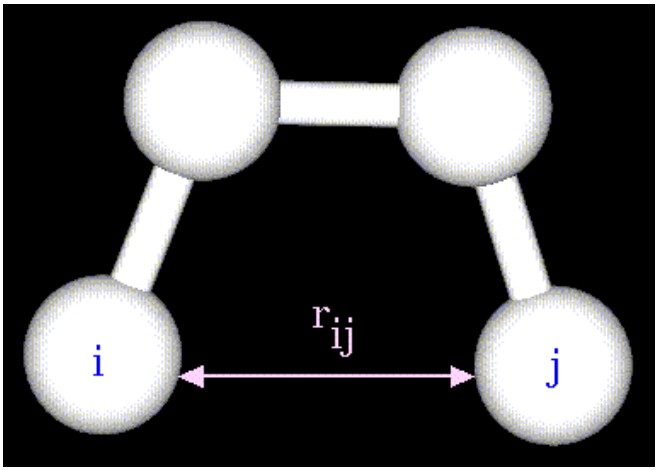
Parameter A dan B mengontrol kedalaman dan posisi (jarak antar atom) dari sumur energi potensialnya untuk suatu

pasangan atom-atom yang berinteraksi secara bukan ikatan (misalnya C:C, O:C dan O:H). A menyatakan derajat "*stickiness*" dari tarikan van der Waals dan B menentukan derajat "*hardness*" dari atom.

Parameter *A* dapat diperoleh dari pengukuran polarizabilitas atomik, atau dapat dihitung dengan mekanika kuantum. Parameter *B* biasanya diturunkan dari data kristalografi sebagai hasil observasi jarak kontak antara atom dalam keadaan berbeda dalm kristal pada berbagai variasi molekul.

Kontribusi elektrostatik dimodelkan dengan memakai suatu potensial Coulomb. Energi elektrostatik merupakan suatu fungsi muatan pada atom-atom tak-berikatan, jarak antar atomnya dan suatu ekspresi dielektrik molekul yang memperhitungkan besarnya interaksi elektrostatik dengan lingkungannya (misalnya pelarut atau molekul itu sendiri). Sering dielektrik molekul diatur pada suatu harga yang tetap antara 1,0 dan 5,0. Suatu variasi linear dielektrik yang tergantung jarak (1/r) kadang-kadang digunakan untuk menerangkan kenaikan dalam lingkungan *bulk* jika pemisahan antara atom-atom yang berinteraksi bertambah.

Muatan parsial atomik dapat dihitung untuk molekul yang kecil menggunakan teknik *ab initio* atau semempiris. Beberapa program menandai muatan menggunakan aturan atau *template* terutama untuk makromolekul. Dalam medan gaya, potensial torsi dikalibrasi dengan metode perhitungan muatan.



Gambar 2. 8 ***Non-Bonded energy***

Untuk perhitungan dari energi tak-terikat (*Non-bonded energy*) adalah sebagai berikut :

$$E_{non-bonded} = E_{vdw} + E_{elec}$$

Dengan :

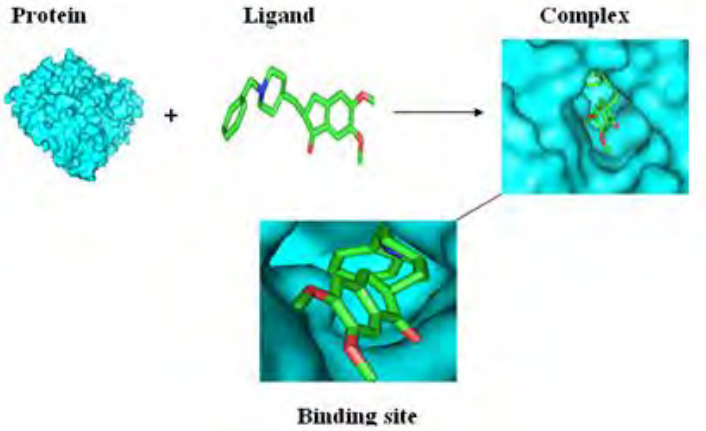
$$E_{vdw} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{A_{ij}}{r_{ij}^{6}} - \frac{B_{ij}}{r_{ij}^{12}} \, , i > j \qquad (4)$$

$$E_{elec} = \sum_{i=1}^{N}\sum_{j=1}^{M} \frac{q_i q_j}{r_{ij}} \, , i > j \qquad (5)$$

2.5 *Molecular Docking*

Di era modern ini, metode pengembangan obat – obatan sudah mulai masuk ke ranah ilmu komputasi, dalam memahami struktur biologi molekuler dan penemuan obat berdasarkan struktur. Hal ini disebut '*Computer aided drug design*' (desain obat berbantu komputer). Salah satu metode yang digunakan adalah Molecular Docking.

Secara bahasa, ‘molecular’ berarti molekuler, dan ‘docking’ berarti penambatan, sehingga dapat diartikan ‘penambatan molekuler’. Dalam hal ini, yang ditambatkan adalah molekul obat (ligan) pada reseptornya (target obat).



Gambar 2. 9 Penambatan ligan pada protein

Molecular docking digunakan untuk mengetahui bagaimana ligan berinteraksi dengan situs tambat reseptornya sehingga dapat diprediksi aktifitasnya. Ligan biasanya berupa molekul kecil, atau dapat pula berupa protein. Sedangan reseptor berupa rangkaian susunan asam amino protein, atau dapat pula berupa DNA atau RNA yang terdapat dalam sel tubuh.

Gambar 2. 10 Filosofi *Docking Molecule*

Docking dilakukan untuk mensimulasikan secara komputasi proses pengenalan molekul. Tujuan dari docking adalah untuk mencapai konformasi protein dan ligan yang optimal sehingga energi bebas dari sistem secara keseluruhan diminimalkan. docking membantu dalam mempelajari obat / ligan atau interaksi reseptor / protein dengan mengidentifikasi situs aktif yang cocok pada protein, mendapatkan geometri terbaik dari kompleks ligan – reseptor , dan menghitung energi interaksi dari ligan yang berbeda untuk merancang ligan yang lebih efektif.

2.5.1 Klasifikasi *Molecular Docking* berdasarkan jenis ligan

Berdasarkan jenis ligannya, metode molecular docking dibedakan menjadi beberapa golongan, yaitu:

1. Protein-protein docking (protein reseptor dengan protein ligan)
2. Protein-ligand docking.
3. Protein-small molecule docking (protein reseptor dengan molekul kecil).
4. Protein-DNA/RNA docking (protein reseptor dengan asam nukleat).

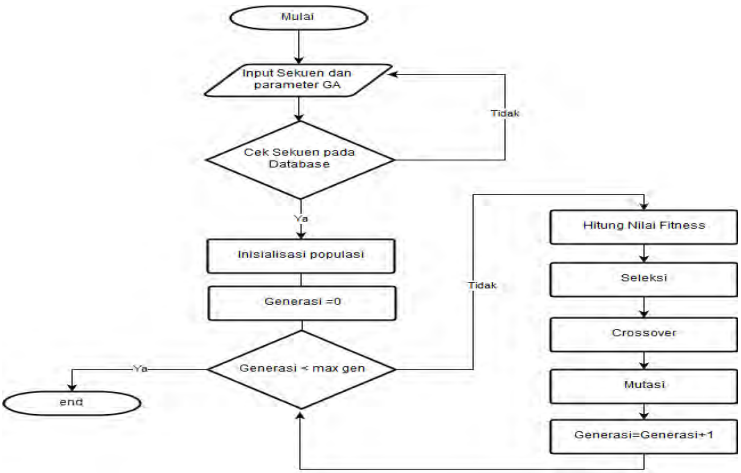
2.5.2 Klasifikasi *Molecular Docking* berdasarkan fleksibilitas Molekul

Berdasarkan fleksibilitas molekulnya, metode molecular docking dibedakan menjadi beberapa golongan, yaitu:

1. *Rigid docking*
   Rigid docking adalah salah satu cabang dari metode *molecular docking* yang memposisikan molekul yang akan dipasangkan (di-docking-kan) sebagai obyek yang bersifat *rigid*. Definisi *rigid* di sini diartikan sebagai suatu kondisi dimana kedua molekul yang digunakan dikondisikan untuk tetap (tidak mengalami perubahan konformasi) selama proses docking sedang berlangsung.
2. *Semi-flexible docking*
   Semi-flexible docking adalah metode molecular docking yang memperlakukan dua molekul yang akan dipasangkan secara berbeda (asimetris). Salah satu molekul (biasanya molekul yang berukuran lebih kecil; ligan) diperlakukan sebagai molekul yang fleksibel sedangkan molekul yang berukuran lebih besar (protein reseptor) diperlakukan sebagai molekul yang *rigid*.
3. *Flexible docking*
   Flexible docking adalah metode *molecular docking* yang memperlakukan dua molekul yang akan dipasangkan sebagai molekul yang fleksibel meskipun pada aplikasinya, fleksibilitas salah satu atau kedua molekul yang akan di-docking.

2.6 Algoritma Genetika

Algoritma genetika merupakan evolusi atau perkembangan dunia komputer dalam bidang kecerdasan buatan (*artificial intelligent*), yang sebenarnya terinspirasi oleh teori Darwin dimana individu yang kuat adalah pemenang dalam kompetisi dilingkungannya.



Gambar 2. 11 Diagram Alir Algoritma Genetika

Algoritma Genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup. pada dasarnya ada 4 kondisi yang sangat mempengaruhi proses evaluasi yaitu :

1. kemampuan organisme untuk melakukan reproduksi
2. keberadaan populasi organisme yang bisa melakukan reproduksi
3. keberagaman organisme dalam suatu populasi
4. perbedaan kemampuan untuk *survive*

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

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

Nilai fitness dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi berikut dikenal dengan istilah anak (***offspring***) terbentuk dari gabungan 2 kromosom generasi sekarang yang bertindak sebagai induk (***parent***) dengan menggunakan operator penyilangan (***crossover***). Selain operator penyilangan, suatu kromosom juga dapat dimodifikasi dengan menggunakan operator **mutasi**.

Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai *fitness* dari kromosom induk (*parent*) dan nilai fitness dari kromosom anak (*offspring*), serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah kromosom dalam suatu populasi) konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik.

2.6.1 Algoritma Genetika Komponen-komponen Utama

Ada 6 komponen utama dalam algoritma genetika, yaitu :

1. Teknik Penyandian
   Teknik penyandian disini meliputi penyadian gen dari kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Gen dapat dipresentasikan dalam bentuk: string bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.
2. Prosedur Inisialisasi
   Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus inisialisai terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada.
3. Fungsi Evaluasi
   Ada 2 hal yang harus dilakukan dalam melakukan evaluasi kromosom, yaitu: evaluasi fungsi objektive (fungsi tujuan) dan konversi fungsi objektive kedalam fungsi fitness. Secara umum, fungsi fitness diturunkan dari fungsi objektive dengan nilai tidak negatif. Apabila ternyata fungsi objektive memiliki nilai negatif, maka perlu ditambahkan suatu konstanta C agar nilai fitness yang terbentuk menjadi tidak negatif.
4. Seleksi
   Seleksi ini bertujuan untuk memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling fit.

5. Operator Genetika
   Ada 2 operator genetika, yaitu:
   a. Operator untuk melakukan rekombinasi, yang terdiri dari: rekombinasi bernilai biner (*crossover*).
   b. Mutasi. mutasi bernilai biner
6. Penentuan parameter
   Yang disebut parameter disini adalah parameter kontrol algoritma genetika, yaitu: ukuran populasi (popsize), peluang c*rossover* (P_c), dan peluang mutasi (P_m). Nilai parameter ini ditentukan juga berdasarkan permasalahan yang akan dipecahkan. Ada beberapa rekomendasi yang bisa digunakan, antara lain:
   a) Untuk permasalahan yang memiliki solusi cukup besar, De Jong merekomendasikan untuk nilai parameter kontrol:
      (popsize; P_c; P_m) = (50; 0,6; 0,001)
   b) Bila rata-rata fitness setiap generasi digunakan sebagai indikator, maka Grefenstette merekomendasikan:
      (popsize; P_c; P_m) = (30; 0,95; 0,01)
   c) Bila fitness dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah:
      (popsize; P_c; P_m) = (80; 0,45; 0,01)
   Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

2.6.2 Istilah-istilah dalam algoritma genetika

Algoritma genetika merupakan algoritma pencarian yang bekerja berdasarkan mekanisme seleksi alam dan genetika. Pada genetika, kromosom terdiri dari gen-gen. Tiap gen mempunyai sifat tertentu (*allele*), dan posisi tertentu (*locus*). Satu atau lebih kromosom bergabung membentuk paket genetika yang disebut *genotif*. Interaksi *genotif* dengan lingkungannya disebut *fenotif*. Pada algoritma genetika,

kromosom berpadanan dengan string dan gen dengan karakter. Setiap karakter mempunyai posisi (*locus*) dan arti tertentu (*allele*). Satu atau lebih string bergabung membentuk struktur (*genotif*), dan apabila struktur tersebut di-*decode*-kan akan diperoleh salah satu alternative solusi (*fenotif*).

Tabel 2. 1Tabel susunan gen ilmu genetika

Ilmu genetika	Algoritma genetika
Kromosom	String
Gen	Karakter
Allele	Nilai karakter
Locus	Posisi dalam individu
Genotif	Struktur
Fenotif	Parameter

2.6.3 Aplikasi Algoritma Genetika

Sejak dirintis oleh John holland, Algoritma Genetika telah dipejari, diteliti dan diaplikasikan secara luas pada berbagai bidang. Algoritma Genetika banyak digunakan pada masalah praktis yang berfokus pada pencarian parameter-parameter optimal. Hal ini membuat banyak orang mengira bahwa Algoritma Genetika hanya bisa digunkan untuk masalah optimasi. Pada kenyataannya, Algoritma Genetika juga memiliki performansi yang bagus untuk masalah-masalah selain optimasi.

Keuntungan penggunaan Algoritma Genetika terlihat dari kemudahan implementasi dan kemampuan untuk menemukan solusi yang "bagus" (bisa diterima) secara cepat untuk masalah-masalah berdimensi tinggi. Algoritma Genetika sangat berguna dan efisien untuk masalah dengan karakteristik sebagai berikut :

a. Ruang masalah sangat besar, kompleks, dan sulit dipahami.

b. Kurang atau bahkan tidak ada pengetahuan yang memadai untuk merepresentasikan masalah ke dalam ruang pencarian yang lebih sempit.
c. Ketika metode-metode konvensional sudah tidak mampu menyelesaikan masalah yang dihadapi.
d. Solusi yang diharapkan tidak harus paling optimal, tetapi cukup "bagus" atau bisa diterima.
e. Terdapat batasan waktu, misalnya dalam *real time system* atau sistem waktu nyata.

Algoritma Genetika telah banyak diaplikasikan untuk menyelsaikan masalah dan pemodelan dalam bidang teknologi, bisnis, dan entertainment, seperti :

a) Optimasi
   Algoritma Genetika digunakan untuk optimasi numerik dan optimasi kombinatorial seperti Travelling Salesman Problem (TSP), perancangan IC, Job Shop Sceduling, optimasi suara dan dan video.
b) Pemrograman Otomatis
   Algoritma genetika digunakan untuk melakukan proses evolusi terhadap program komputer untuk merancang struktur komputasional, seperti cellular automata dan sorting networks
c) Machine Learning
   Algoritma genetika berhasil diaplikasikan untuk memprediksi struktur protein. Algoritma genetika juga berhasil dalam perancangan neural networks (jaringan syaraf tiruan).
d) Model Ekonomi
   Algoritma genetika digunkan untuk memodelkan sistem-sistem inovasi, memprediksi produksi dan lab perusahaan.

2.6.4 Operator Genetika

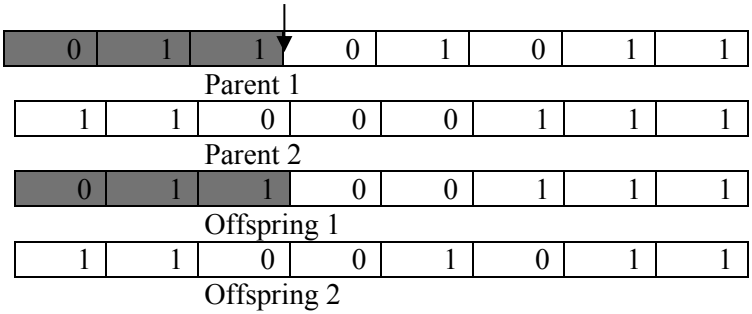
Operator genetika digunakan untuk mengkombinasikan (modifikasi) individu dalam aliran populasi untuk menghasilkan individu baru. Ada dua operator genetika yaitu crossover dan mutasi.

2.6.4.1 *Crossover*

Pada *crossover* akan dipilih secara acak dua individu dan tempat pertukaran, dimana kromosom yang ditandai diantara kedua tempat pertukaran akan bertukar tempat satu sama lain. Proses *crossover* akan membangkitkan offspring baru dengan mengganti sebagian informasi dari parents (orang tua atau induk).

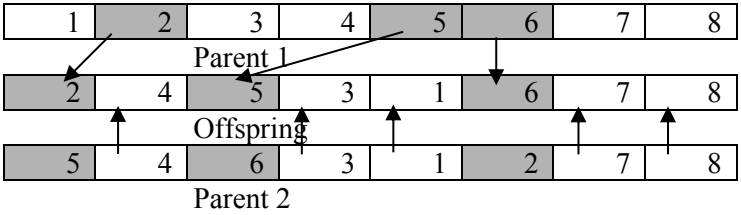
Tujuan dari proses *crossover* adalah untuk menambahkan keanekaragaman individu dalam populasi dengan mengawinkan individu-individu pada populasi sehingga menghasilkan keturunan berupa individu-individu baru untuk ditempatkan pada populasi selanjutnya. Ada beberapa tipe *crossover*, antara lain :

1. *One-cut-point-crossover*
   Pada tipe ini, akan dibuat satu titik *crossover* dimana individu yang dihasilkan akan diambil dari bilangan biner parent pertama dari awal sampai titik *crossover* dan sisanya dari parent kedua. Algoritmanya adalah :
   a. Memilih site secara random dari parent pertama
   b. Isi disebelah kanan site pada parent pertama ditukar dengan parent kedua untuk menghasilkan offspring.



Gambar 2. 12 Contoh single point crossover

2. *Order-based crossover*
   Pada tipe ini, offspring yang dihasilkan hanya satu hasil dari kombinasi kedua parent.



Gambar 2. 13 Contoh Order Based Crossover

2.6.4.2 Mutasi

Mutasi menciptakan individu baru dengan melakukan modifikasi satu atau lebih gen dalam individu yang sama. Mutasi berfungsi untuk menggantikan gen yang hilang dari populasi selama proses seleksi serta menyediakan gen yang tidak ada dalam populasi awal. Sehingga mutasi akan meningkatkan variasi populasi. *Shif mutation* dilakukan dengan cara :

- Menentukan dua site secara random
- Site pertama ditempatkan ke site kedua, untuk selanjutnya digeser ke kiri seperti terlihat pada gambar berikut.

Parent	1	2	3	4	5	6	7	8
Offspring	1	2	7	4	5	6	3	8

Gambar 2. 14 Contoh Shift Mutation

2.6.5 Parameter Genetika

Parameter-parameter genetika berguna dalam pengendalian operator-operator genetika. Pemilihan penggunaan nilai-nilai parameter genetika sangat berpengaruh terhadap kinerja algoritma genetika dalam menyelesaikan suatu masalah. Parameter-parameter genetika yang digunakan antara lain :

1. Ukuran populasi
   Ukuran popolasi mempengaruhi ukuran efektivitas dan kinerja algoritma genetika. Tidak ada aturan yang pasti tentang berapa nilai ukuran populasi. Apabila ukuran populasi kecil berarti hanya tersedia sedikit pilihan untuk *crossover* dan sebagian kecil dari domain solusi saja yang dieksplorasi untuk setiap generasinya. Sedangkan apabila terlalu besar, kinerja algoritma genetika akan menurun. Penelitian menunjukan ukuran populasi besar tidak mempercepat pencarian solusi. Disarankan ukuran populasi berkisar antara 20 – 30.
2. Jumlah generasi
   Jumlah generasi berpengaruh terhadap banyaknya iterasi yang akan dikerjakan dan domain solusi yang akan dieksplorasi untuk setiap generasinya. Semakin besar jumlah generasi berarti semakin banyak iterasi yang dilakukan, dan semakin besar solusi yang dieksplorasi. Sedangkan semakin kecil jumlah generasinya, maka akan semakin kecil iterasi yang akan dilakukan, dan semakin kecil pula solusi yang dieksplorasi untuk tiap generasinya.

3. Probabilitas *crossover* (P_c)

Probabilitas *crossover* akan mengendalikan operator *crossover* dalam setiap generasi dalam populasi yang mengalami *crossover*. Semakin besar nilai probabilitas *crossover*, akan semakin cepat struktur individu baru terbentuk ke dalam populasi. Sedangkan apabila nilai probabilitas *crossover* terlalu besar, individu yang merupakan kandidat solusi terbaik mungkin akan dapat hilang lebih cepat pada generasi selanjutnya. Disarankan nilai probabilitas *crossover* berkisar antara 80 % - 95 %.

4. Probabilitas mutasi (P_m)

   Probabilitas mutasi akan mengendalikan operator mutasi pada setiap generasi. Peluang mutasi yang digunakan biasanya lebih kecil daripada peluang *crossover*. Pada seleksi alam murni, mutasi jarang sekali muncul. Oleh karena itu, operator mutasi pada algoritma genetika juga tidak selalu terjadi. Untuk itulah nilai peluang mutasi dibuat lebih kecil untuk setiap generasi. Disarankan nilai probabilitas mutasi kecil berkisar antara 0.5 % - 1 %.

2.6.6 Tahapan Algoritma genetika:

Pada dasarnya, algoritma genetika dapat dilakukan melalui lima tahap yaitu sebagai berikut :

1. Membentuk Populasi Awal

   Langkah pertama dalam algoritma genetika adalah membentuk sebuah populasi untuk sejumlah gen. populasi itu sendiri merupakan sekumpulan solusi yang akan digunakan dalam proses regenerasi selanjutnya untuk mencari solusi terbaik. Solusi-solusi yang ada selanjutnya disebut sebagai individu.

2. Mencari *Fitness Cost*

Pada tahap ini setiap individu yang terbentuk dicari fitness cost-nya sebagai nilai pembanding antara individu satu dengan yang lainnya.

3. Pengurutan (Sorting)
   Pada tahap ini, individu yang terdapat pada populasi diurutkan berdasarkan *fitness costnya*. Tujuan utamanya adalah untuk mencari individu terbaik pada populasi yang ada, yang dapat dikatakan sebagai solusi sementara.
4. Proses Regenerasi
   Proses ini terdapat dua metode yaitu
   a. *Elitism Method*
      Metode dimana individu-individu yang akan mengalmi proses regenerasi, yaitu proses mutasi dan crossover adalah individu-individu dengan nilai fitness yang rendah, sedangkan individu dengan nilai fitness tertinggi atau gen terbaik akan dipertahankan untuk dibandingkan lagi dengan individu hasil proses regenerasi.
   b. *Non Elitms*
      Suatu metode regenerasi yang melibatkan semua individu baik individu / gen terbaik maupun gen yang kurang baik (individu dengan nilai fitness rendah).
5. Tahapan Pengulangan
   Setelah proses regenerasi selesai, maka dilakukan pengulangan proses ini sampai sejumlah generasi yang dikehendaki.

Algorithm 1 A Genetic Algorithm Pseudo-Code

1: Choose an initial random population of individuals
2: Evaluate the fitness of the individuals
3: **repeat**
4: Select the *best* individuals to be used by the genetic operators
5: Generate new individuals using crossover and mutation
6: Evaluate the fitness of the new individuals
7: Replace the *worst* individuals of the population by the best new individuals
8: **until** some stop criteria

Gambar 2. 15 *Pseudocode* Algoritma Genetika

2.7 Fungsi Evaluasi untuk *Fleksibel Docking*

Fungsi evaluasi dari *flexible docking* adalah persamaan sederhana dari mekanika molekul fungsi energinya. Fungsi energi didapatkan sesuai dengan data telah dideskripsikan dari atom dan ikatan yang digunakan. Dalam *molecular docking* energi yang digunakan adalah energi torsi untuk bentuk *helix* dari protein dan energi interaksi pada non-ikatan.

Pada implementasi *molecular docking*, setiap individu kromosom memiliki tiga gen yang mewakili translasi suatu ligan, 4 gen yang mewakili orientasi ligan dan gen yang lain mewakili konformasi ligan [5]. Gen trasnlasi adalah koordinat X,Y,Z referensi atom (biasanya atom yang paling dekat dengan pusat massa ligan), gen oriental adalah quaternion dibentuk oleh vektor satuan dan satu sudut oriental. Konformasi gene adalah sudut dihedral ligan (satu gen untuk setiap sudut dihedral). Fungsi energi ligan-protein yang menggunakan GROMOS96 [12,13] medan gaya klasik diimplementasikan dalam program THOR molekul mekanik / dinamika [14]. Parameter medan gaya disesuaikan untuk mereproduksi hasil eksperimen (misalnya struktur dan termodinamika) atau tingkat tinggi ab initio perhitungan kuantum. Medan kekuatan

GROMOS yang diberikan mengunakan persamaan (3), (4), dan (5) :

$$\sum_{\substack{i=1\\ \text{Protein}}}^{N}\sum_{\text{j= Ligan}}^{M}\left(\frac{A_{ij}}{r_{ij}^{12}}-\frac{B_{ij}}{r_{ij}^{6}}+\frac{q_i q_j}{r_{ij}}\right)+\sum_{\substack{i=1\\ \text{Ligan}}}^{M}\sum_{\substack{j=1\\ \text{Ligan}}}^{M}\left(\frac{A_{ij}}{r_{ij}^{12}}-\frac{B_{ij}}{r_{ij}^{6}}+\frac{q_i q_j}{r_{ij}}\right)$$
$$+\sum_{\substack{k=1\\ \text{Dihedral Angle}}}^{N}\frac{V_k}{2}\left(1+\cos(n\tau-\phi)\right) \qquad (6)$$

BAB III
METODE PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang digunakan dalam penyusunan Tugas Akhir. Disamping itu, dijelaskan pula prosedur dan proses pelaksanaan tiap-tiap langkah yang dilakukan dalam menyelesaikan Tugas Akhir.

3.1 Tahapan Penelitian

Langkah-langkah sistematis yang dilakukan dalam proses pengerjaan tugas akhir ini sebagai berikut :

1. **Studi Literatur**

Studi Literatur ini dilakukan untuk identifikasi permasalahan dengan mencari referensi yang menunjang penelitian yang berupa Tugas Akhir, jurnal internasional, buku, maupun artikel yang berhubungan dengan topik Tugas Akhir ini.

2. **Analisis metode pendekatan algoritma genetika**

Analisi metode pendekatan algoritma genetika merupakan tahap untuk analisis metode yang diperlukan dalam pengerjaan Tugas Akhir ini yaitu gaya intermolekul, analisis metode pendekatan untuk *flexible docking*, nilai parameter yang dibutuhkan untuk perhitungan, serta modeling *docking molecule*. Adapun parameter yang akan dipakai berdasarkan fungsi evaluasi *flexible docking* parameter Lennard-jones, jarak antara atom, muatan atom, serte parameter yang lain. Parameter energi ikatan dan non-ikatan didapatkan dari GROMOS96. Untuk jarak antar molekul didapatkan setelah proses modeling yang dilakukan menggunakan Web server SWISS-MODEL.

3. **Perancangan Simulasi**

Pada tahap ini penulis membuat implementasi perancangan simulasi agar mudah dipahami. Implementasi tersebut dibuat dengan bahasa pemrograman Java dan menggunakan aplikasi NETBEANS IDE 8.1 untuk proses

Generate Genetic Algorithm dan Matlab untuk mengecek homologi gen setelah dilakukan docking dengan protein awal.

Pada tahap ini dilakukan proses dimana persoalan diterjemahkan melalui suatu rangkaian aktivitas sesuai model proses, metode, dan alat bantu yang akan digunakan. Tahap-tahap yang dilakukan sebagai berikut :

a. Menentukan jenis protein dan ligan

   Pada tahap ini akan dilakukan penentuan jenis protein dan ligan yang digunakan untuk simulasi.

b. Menentukan jumlah Populasi dan Generasi

   Pada tahap ini akan dilakukan inisialisasi banyaknya populasi serta banyaknya generasi yang akan di *generate* dalam algoritma genetika.

c. Menghasilkan *Chromosome*

   Untuk mendapatkan chromosome setiap generasi dilakukan random dimana nilai setiap array adalah 0 atau 1.

d. Mendapatkan *Docking Sequence*

   Docking sequence didapatkan dari pengubahan chromosome biner ke dalam desimal. Setelah didapatkan desimal kita tempatkan ligan pada posisi

e. Mendapatkan jarak antar molekul

   Untuk mendapatkan nilai jarak antar moleku dilakukan prediksi model menggunakan web server SWISS-MODEL.

f. Menghasilkan nilai evaluasi

   Nilai parameter yang dibutuhkan untuk perhitungan didapatkan dari GROMOS96. Kemudian dilakukan perhitungan dengan menggunakan Netbeans IDE Java 8.1

g. Proses seleksi

   Proses seleksi dilakukan untuk mendapatkan individu atau chromosome terbaik dari generasi.

h. Proses Crossover dan Mutasi

Setelah proses seleksi dilakukan proses crossover dan mutasi chromosome atau individu setiap generasi. Kemudian dilakukan generate dimulai dengan generate chromosome hingga mutasi sampai generasi terakhir

i. Menampilkan representasi hasil implementasi

Setelah generasi terakhir dilakukan Pengecekan minimum energi untuk setiap generasi. Setelah didapat minimum generasi dilakukan pensejajaran protein atau cek homologi protein dengan program Matlab.

4. Implementasi

Setelah dibuat perancangan simulasi, akan dilakukan pengujian simulasi untuk memeriksa apakah hasil implementasi sudah sesuai atau terjadi eror.

5. Kesimpulan dan Saran

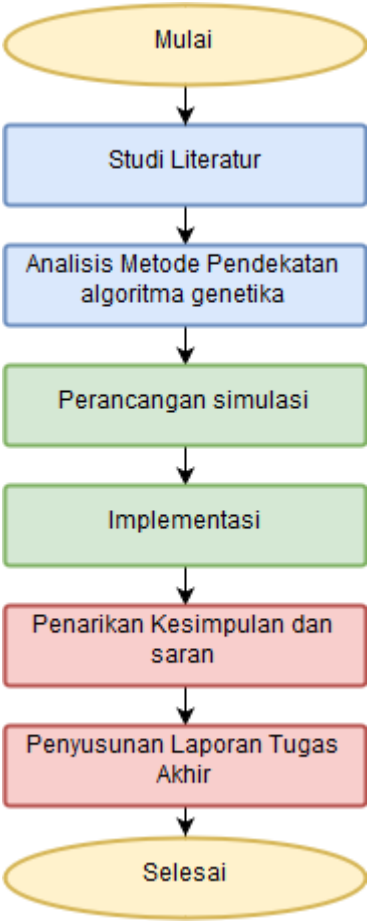
Setelah dilakukan analisis dan pembahasan maka dapat ditarik suatu kesimpulan dan saran sebagai masukan untuk pengembangan penelitian lebih lanjut.

6. Penyusunan Laporan Tugas Akhir

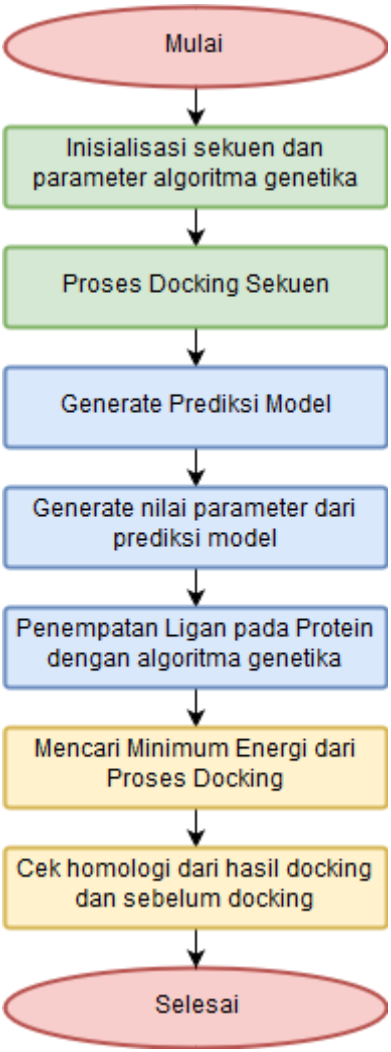
Setelah semua proses selesai dilakukan maka tahap terakhir adalah penyusuan laporan Tugas Akhir.

3.2 Diagram Alir Penelitian

Berdasarkan uraian tersebut diatas, penelitian Tugas Akhir ini dapat dinyatakan dalam diagram alir sebagai berikut.



Gambar 3. 1 Diagram Alir Metode Penelitian



Gambar 3. 2 Diagram Alir Implementasi

BAB IV
PERANCANGAN IMPLEMENTASI

Bab ini menjelaskan rancangan implementasi yang digunakan sebagai acuan untuk implementasi. Perancangan implementasi menggambarkan proses rancang bangun secara terperinci dari awal tahap analisis metode pendekatan algoritma genetika hingga implementasi atau simulasi penempatan ligan pada protein dengan algoritma genetika.

4.1 Analisis metode pendekatan

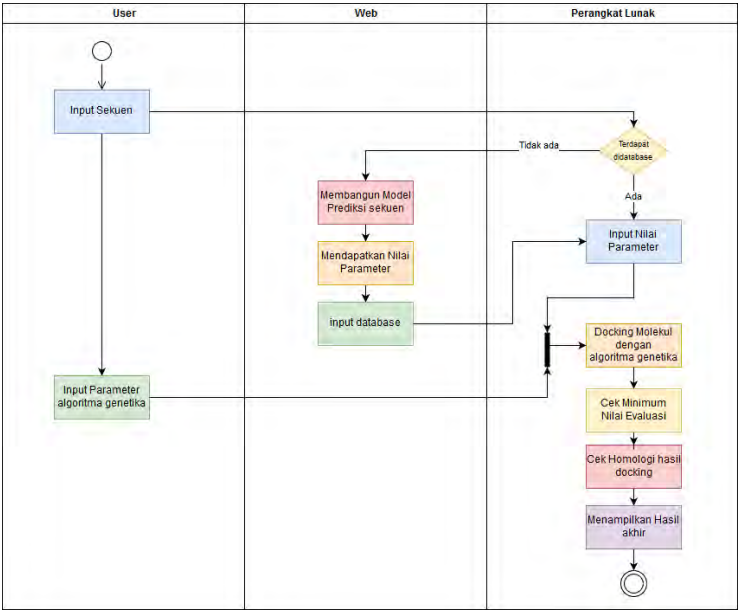
Tahap ini bertujuan untuk menjelaskan tentang proses-proses yang dilakukan terhadap data awal yang diperoleh, sehingga pendekatan untuk penemempatan ligan dapat dilakukan. Ada 3 hal yang akan dibahas dalam analisis metode yaitu : Perancangan model data, perancangan diagram alir dan perancangan class diagram .

4.1.1 Perancangan Model Data

Tahapan ini bertujuan untuk menjelaskan *docking molecule* untuk menepatkan ligan pada protein dengan menggunakan algoritma genetika. Data yang digunakana adalah protein virus malaria dan DBD serta ligan dari tumbuhan kina dan jambu merah. Data yang didapatkan dari *website https://www.ncbi.nlm.nih.gov/* .Untuk membantu user melakukan proses *docking molecule* berdasarkan dengan jenis protein yang digunakan yaitu *fleksible docking.* Sehingga ligan dapat ditempatkan pada semua titik protein. Selain itu aplikasi ini menggunakan DBMS MySQL untuk membaca basis data, mulai dari pencocokan sekuen input dengan data sekuen data pada database hingga perhitungan nilai evaluasi. DBMS berfungsi sebagai media penyimpana data sekuen dan nilai evaluasi data yang akan diolah dan dianalisis

4.1.2 Perancangan Diagram Alir

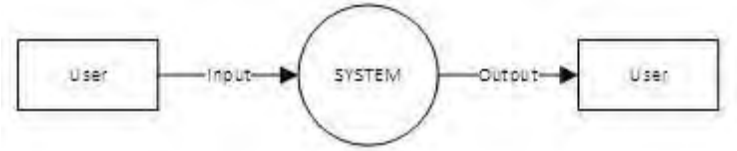
Pada tahap ini menjelaskan tentang alir dari program ini. Langkah pertama dalam diagram ini adalah memodelkan sekun docking. Pada tahap pemodelan sekun, pemodelan dilakukan dengan menggunakan bantuan *website* yaitu *https://swissmodel.expasy.org/*. Kemudian dilakukan perhitungan dengan menggunakan software java untuk docking molekul. Setelah didapatkan hasil docking minimum hasil, dilakukan proses pensejajaran sekuen dengan menggunakan software MATLAB. Proses pensejajaran sekuen menggunakan algoritma Needleman-wunsch dan Smith-waterman. Sesuai dengan Gambar 4.1.



Gambar 4. 1 *Activity Diagram*

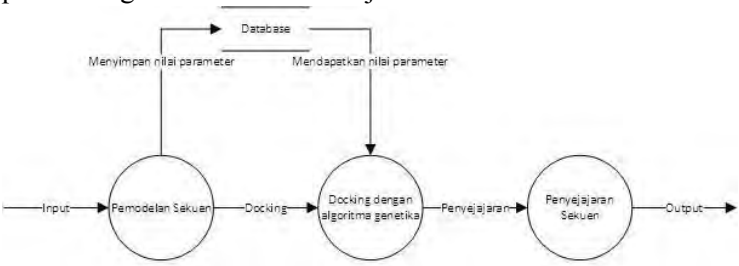
4.1.3 Data Flow Diagram

Data flow diagram adalah representasi grafik yang menggmbarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan (input) dan keluaran (output) [11]. Kemudian dibuat suatu data flow diagram untuk menggambarkan alur proses. DFD Level 0 biasa disebut dengan diagram sistem inti. Pada diagram sistem ini menjelaskan sebuah proses umum yang nantinya akan di dekomposisi menjadi proses-proses yang lebih detail. Seperti gambar 4.2



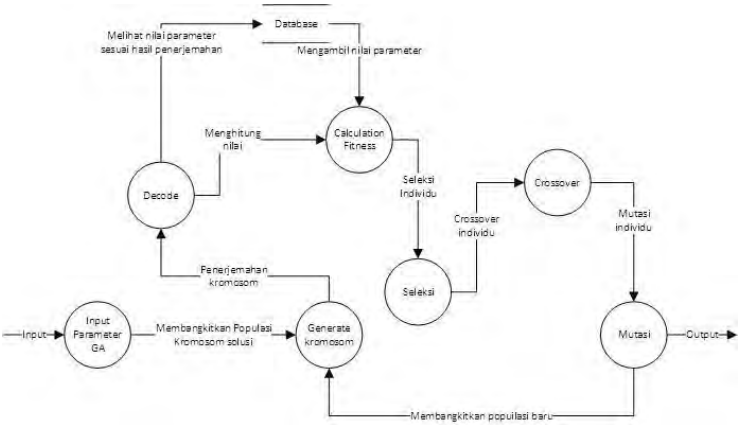
Gambar 4. 2 *Data Flow Diagram* Level 0

DFD level 1 menggambarkan dekomposisi dari proses-proses diagram sistem inti menjadi lebih detail.



Gambar 4. 3 *Data Flow Diagram* Level 1

DFD level 2 menggambarkan dekompoisis dari proses-proses diagram sistem algoritma genetika menjadi lebih detail



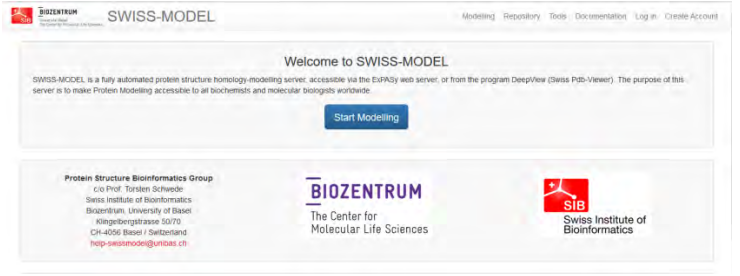
Gambar 4. 4 *Data Flow Diagram* level 2

4.2 Perancangan Proses

Dalam perancangan proses, ada beberapa tahapan yang harus dilakukan. Tahapan-tahapan tersebut adalah sebagai berikut :

4.2.1 Pemodelan Data sekuen

Pada tahapan ini dilakukan pemodelan dari kombinasi sekuen menggunakan *website* *https://swissmodel.expasy.org/* sesuai Gambar 4.5. Pemodelan dilakukan dengan menginputkan kombinasi ligan pada protein sesuai dengan panjang sekuen protein.

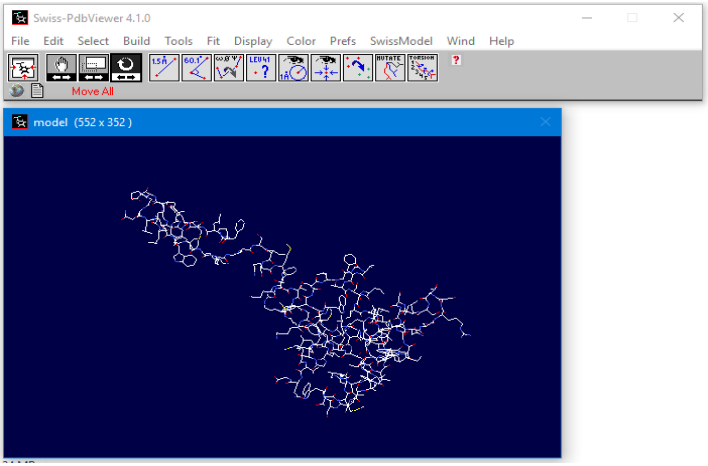


Gambar 4. 5 Web *predic modeling molecule* SWISS-MODEL

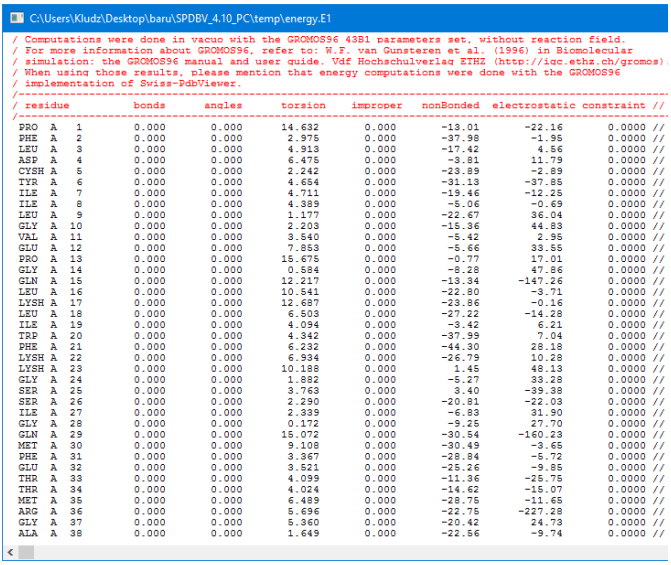
Pemodelan digunakan untuk mendapatkan parameter x, y, z lennard jonnes parameter r dimana $r = \sqrt{x^2 + y^2 + z^2}$. Selain itu didapatkan parameter ϕ sudut torsi dihedral molekul untuk persamaan (6).

4.2.2 Mendapatkan Nilai parameter dari model

Prediksi model yang sudah didapatkan kemudian di generate untuk mendapatkan nilai parameter untuk perhitungan penempatan ligan pada protein. Aplikasi yang digunakan untuk mendapatkan nilai parameter tersebut adalah DeepView (Swiss Pdb-Viewer) dengan menginputkan hasil modeling yaitu berupa file PDB.



Gambar 4. 6 DeepView (Swiss Pdb-Viewer)



C:\Users\Kludz\Desktop\baru\SPDBV_4.10_PC\temp\energy.E1

```
/ Computations were done in vacuo with the GROMOS96 43B1 parameters set, without reaction field.
/ For more information about GROMOS96, refer to: W.F. van Gunsteren et al. (1996) in Biomolecular
/ simulation: the GROMOS96 manual and user guide. Vdf Hochschulverlag ETHZ (http://igc.ethz.ch/gromos).
/ When using those results, please mention that energy computations were done with the GROMOS96
/ implementation of Swiss-PdbViewer.
/-------------------------------------------------------------------------------------------------
/ residue          bonds      angles      torsion     improper   nonBonded  electrostatic constraint //
/-------------------------------------------------------------------------------------------------
 PRO  A   1        0.000       0.000      14.632       0.000      -13.01        -22.16       0.0000 //
 PHE  A   2        0.000       0.000       2.975       0.000      -37.98         -1.95       0.0000 //
 LEU  A   3        0.000       0.000       4.913       0.000      -17.42          4.56       0.0000 //
 ASP  A   4        0.000       0.000       6.475       0.000       -3.81         11.79       0.0000 //
 CYSH A   5        0.000       0.000       2.242       0.000      -23.89         -2.89       0.0000 //
 TYR  A   6        0.000       0.000       4.654       0.000      -31.13        -37.85       0.0000 //
 ILE  A   7        0.000       0.000       4.711       0.000      -19.46        -12.25       0.0000 //
 ILE  A   8        0.000       0.000       4.389       0.000       -5.06         -0.69       0.0000 //
 LEU  A   9        0.000       0.000       1.177       0.000      -22.67         36.04       0.0000 //
 GLY  A  10        0.000       0.000       2.203       0.000      -15.36         44.83       0.0000 //
 VAL  A  11        0.000       0.000       3.540       0.000       -5.42          2.95       0.0000 //
 GLU  A  12        0.000       0.000       7.853       0.000       -5.66         33.55       0.0000 //
 PRO  A  13        0.000       0.000      15.675       0.000       -0.77         17.01       0.0000 //
 GLY  A  14        0.000       0.000       0.584       0.000       -8.28         47.86       0.0000 //
 GLN  A  15        0.000       0.000      12.217       0.000      -13.34       -147.26       0.0000 //
 LEU  A  16        0.000       0.000      10.541       0.000      -22.80         -3.71       0.0000 //
 LYSH A  17        0.000       0.000      12.687       0.000      -23.86         -0.16       0.0000 //
 LEU  A  18        0.000       0.000       6.503       0.000      -27.22        -14.28       0.0000 //
 ILE  A  19        0.000       0.000       4.094       0.000       -3.42          6.21       0.0000 //
 TRP  A  20        0.000       0.000       4.342       0.000      -37.99          7.04       0.0000 //
 PHE  A  21        0.000       0.000       6.232       0.000      -44.30         28.18       0.0000 //
 LYSH A  22        0.000       0.000       6.934       0.000      -26.79         10.28       0.0000 //
 LYSH A  23        0.000       0.000      10.188       0.000        1.45         48.13       0.0000 //
 GLY  A  24        0.000       0.000       1.882       0.000       -5.27         33.28       0.0000 //
 SER  A  25        0.000       0.000       3.763       0.000        3.40        -39.38       0.0000 //
 SER  A  26        0.000       0.000       2.290       0.000      -20.81        -22.03       0.0000 //
 ILE  A  27        0.000       0.000       2.339       0.000       -6.83         31.90       0.0000 //
 GLY  A  28        0.000       0.000       0.172       0.000       -9.25         27.70       0.0000 //
 GLN  A  29        0.000       0.000      15.072       0.000      -30.54       -160.23       0.0000 //
 MET  A  30        0.000       0.000       9.108       0.000      -30.49         -3.65       0.0000 //
 PHE  A  31        0.000       0.000       3.367       0.000      -28.84         -5.72       0.0000 //
 GLU  A  32        0.000       0.000       3.521       0.000      -25.26         -9.85       0.0000 //
 THR  A  33        0.000       0.000       4.099       0.000      -11.36        -25.75       0.0000 //
 THR  A  34        0.000       0.000       4.024       0.000      -14.62        -15.07       0.0000 //
 MET  A  35        0.000       0.000       6.489       0.000      -28.75        -11.65       0.0000 //
 ARG  A  36        0.000       0.000       5.696       0.000      -22.75       -227.28       0.0000 //
 GLY  A  37        0.000       0.000       5.360       0.000      -20.42         24.73       0.0000 //
 ALA  A  38        0.000       0.000       1.649       0.000      -22.56         -9.74       0.0000 //
```

Gambar 4. 7 Hasil Generate DeepView (Swiss Pdb Viewer)

Hasil dari generate seperti Gambar 4.7 diatas diperoleh nilai van derwaals, dihedral, serta nilai electrostatic dari PDB sekuen. Kemudian kita simpan nilai tersebut seperti pada Gambar 4.8 dibawah ini

	A	B	C	D	E	F	G
1	1	2ANL_2	ALS30314.	0	2735.55	-10049	-3962.57
2	2	2ANL_2	ALS30314.	1	2733.58	-10013.7	-4029.52
3	3	2ANL_2	ALS30314.	2	2721.73	-9975.18	-4050.55
4	4	2ANL_2	ALS30314.	3	2716.88	-9935.38	-3915.9
5	5	2ANL_2	ALS30314.	4	2709.75	-9904.68	-3844.03
6	6	2ANL_2	ALS30314.	5	2708.91	-9923.43	-3845.85
7	7	2ANL_2	ALS30314.	6	2872.74	-9833.01	-4019.21
8	8	2ANL_2	ALS30314.	7	2598.16	-10565.3	-4584.92
9	9	2ANL_2	ALS30314.	8	2838.42	-10094.9	-4062.93
10	10	2ANL_2	ALS30314.	9	2554.02	-10584.8	-4723.14
11	11	2ANL_2	ALS30314.	10	2602.41	-10021	-4425.27
12	12	2ANL_2	ALS30314.	11	2689.41	-10017.7	-4566.15
13	13	2ANL_2	ALS30314.	12	2827.02	-9920.39	-4137.69
14	14	2ANL_2	ALS30314.	13	2918.38	-9013.06	-4018.48
15	15	2ANL_2	ALS30314.	14	2645.53	-10073	-4477.62
16	16	2ANL_2	ALS30314.	15	2545.49	-10443.7	-4630.14
17	17	2ANL_2	ALS30314.	16	2899.73	-10066.5	-4173.99
18	18	2ANL_2	ALS30314.	17	2744.95	-9733.07	-4328.91
19	19	2ANL_2	ALS30314.	18	2641.14	-10256.1	-4555.03
20	20	2ANL_2	ALS30314.	19	2687.48	-10136.8	-4277

Gambar 4. 8 Hasil Dari Generate dari beberapa PDB

Data tersebut kemudian disimpan dalam format ".csv" agar bisa diimport ke DBMS MySQL. Database untuk penelitian ini diberi nama docking dan tabel untuk data tersebut disimpan dan diberi nama eval. Gambar 4.9 dan Gambar 4.10 berikut adalah tampilan *database* pada tabel protein dan tabel eval.

	jenis	pdb	nama	panjang	sekuen
Edit Copy Delete	protein	2ANL_2	Plasmodium	327	SENDVIELDDVANLMFYGEGEVGDNHQKFMLIFDTGSANLWVPSKKCNSI...
Edit Copy Delete	protein	3GPQ_D	chikunguny	168	MKHHHHHHIAPSYRVKRMDIAKNDEECVVNAANPRGLPGDGVCKAVYKKWP...
Edit Copy Delete	protein	AAD42780.1	Dengue Vir	140	PFLDCYIILGVEPGQLKLIWFKKGSSIGQMFETTMRGAKRMAILGDTAWD...
Edit Copy Delete	ligan	ACD86354.1	Zingiber o	16	NFPLDLAAVEVSSTNG
Edit Copy Delete	ligan	AHA94377.1	Psidium Gu	26	MEVMHERNAHNFPLDLAAVEAPSTNG
Edit Copy Delete	ligan	ALS30314.1	Chinchona	13	FPMIRKQGISGPL

Gambar 4. 9 Tampilan tabel protein pada *database* docking pada MySQL

←T→	NO	pdb1	pdb2	posisi	diherdal	vanderwaals	electrostatic
Edit Copy Delete	1	2ANL_2	ALS30314.1	0	2735.55	-10048.98	-3962.57
Edit Copy Delete	10	2ANL_2	ALS30314.1	9	2554.02	-10584.77	-4723.14
Edit Copy Delete	100	2ANL_2	ALS30314.1	99	2592.80	-10044.50	-4509.88
Edit Copy Delete	101	2ANL_2	ALS30314.1	100	2567.67	-10448.87	-4707.94
Edit Copy Delete	102	2ANL_2	ALS30314.1	101	2575.57	-10628.04	-4740.68
Edit Copy Delete	103	2ANL_2	ALS30314.1	102	2464.70	-10674.63	-4895.77
Edit Copy Delete	104	2ANL_2	ALS30314.1	103	2603.35	-10291.58	-4550.85
Edit Copy Delete	105	2ANL_2	ALS30314.1	104	2523.89	-10486.69	-4760.42
Edit Copy Delete	106	2ANL_2	ALS30314.1	105	3004.03	-9375.37	-3922.82
Edit Copy Delete	107	2ANL_2	ALS30314.1	106	2630.36	-10014.94	-4475.65
Edit Copy Delete	108	2ANL_2	ALS30314.1	107	2621.80	-10116.28	-4511.40
Edit Copy Delete	109	2ANL_2	ALS30314.1	108	2904.18	-9812.49	-4281.74
Edit Copy Delete	11	2ANL_2	ALS30314.1	10	2602.41	-10020.95	-4425.27
Edit Copy Delete	110	2ANL_2	ALS30314.1	109	2619.00	-9892.71	-4540.76
Edit Copy Delete	111	2ANL_2	ALS30314.1	110	2835.35	-10082.29	-4023.02
Edit Copy Delete	112	2ANL_2	ALS30314.1	111	2665.04	-10021.22	-4380.08
Edit Copy Delete	113	2ANL_2	ALS30314.1	112	2590.25	-9927.82	-4485.06
Edit Copy Delete	114	2ANL_2	ALS30314.1	113	2740.89	-10404.96	-4148.05
Edit Copy Delete	115	2ANL_2	ALS30314.1	114	2698.50	-10171.77	-4084.85

Gambar 4. 10 Tampilan tabel eval di *database* docking pada MySQL

4.3 Proses Docking Molekul

Tahap setelah pemodelan sekuen adalah penempatan ligan pada protein. Penempatan ligan menggunakan algoritma genetika untuk docking molekul. Algoritma genetika membangkitkan kromosom, dimana kromosom merupakan solusi dari permasalahan penemapatan ligan. Solusi tersebut merepresentasikan posisi *binding site* atau sisi aktif dari molekul. Sesuai dengan algoritma genetika tahapan dalam docking molekul adalah :

1) Pembangkitan kromosom awal
2) *Decode* atau penerjemahan kromosom
3) Hitung nilai fitness
4) Seleksi, Crossover, dan Mutasi

4.4 Proses Pensejajaran Sekuen

Tahap terakhir proses adalah pensejajaran sekuen (*Sequence alignment*) hasil docking dan sebelum docking. Beberapa metode *alignment* antar lain metode "Needleman-Wunsch" dan "Smith-Waterman". Metode Needleman-Wunsch digunakan untuk menyusun *alignment* global di antara dua atau lebih sekuens, yaitu *alignment* atas keseluruhan panjang sekuens tersebut. Metode Smith-Waterman menghasilkan alignment lokal, yaitu alignment atas bagian-bagian dalam sekuens. Pensejajaran molekul menggunakan algoritma Needleman-wunsch dan Smith-Waterman. Pensejajaran dilakukan untuk mengetahui berapa persentasi kemiripan sekuen hasil docking dengan sekuen awal protein.

4.5 Pengambilan Data

Data yang telah disimpan pada proses sebelumnya, selanjutnya sudah siap untuk diproses menggunakan algoritma genetika. Data tersebut disimpan ke dalam DBMS MySQL, karena pembuatan program pada penelitian ini menggunakan Java, maka perlu dibuat koneksi antara Java dengan MySQL. Kode yang digunakan adalah sebagai berikut :

```
try {
          koneksi.connectFirst();
          if(koneksi.isConnected())
             System.out.println("connected");
          else
             System.out.println("not connected");
       } catch (SQLException ex) {

Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE,
null, ex);
       }
public void connectFirst() throws SQLException{
       String server = "localhost", db="docking", user = "root", pass = "";
       getConnection(server, db, user, pass);
    }
    public boolean isConnected() {
```

```
        if(koneksi != null){
            return true;
        }
        return false;
    }
    public boolean getConnection(String server, String db, String user, String
pass)throws SQLException, IllegalStateException{
        this.destroyConnection();
        String dbDriver = "com.mysql.jdbc.Driver";
        String dbUrl = "jdbc:mysql://" + server + "/" + db;
        try{
            Class.forName(dbDriver);
            koneksi = DriverManager.getConnection(dbUrl, user, pass);
            return true;
        }
        catch(ClassNotFoundException ex){
        }
        catch(SQLException ex){
        }
        return false;
    }
```

Tahapan awal setelah inisialiasai adalah pengecekan data input sekuen. Penulis menggunakan query untuk mengecek sekuen tersebut ada pada *database.* Untuk pengecekan data tersebut query yang digunakan adalah

” "SELECT pdb FROM protein WHERE sekuen = '"+protein+"'"

Dalam melakukan perhitungan nilai fitness untuk setiap individu penulis menggunakan query untuk mendapatkan niliai *dihedral energy, van der waals energy* dan *electrostatic energy*. Untuk Mengambil nilai tersebut query yang digunakan adalah

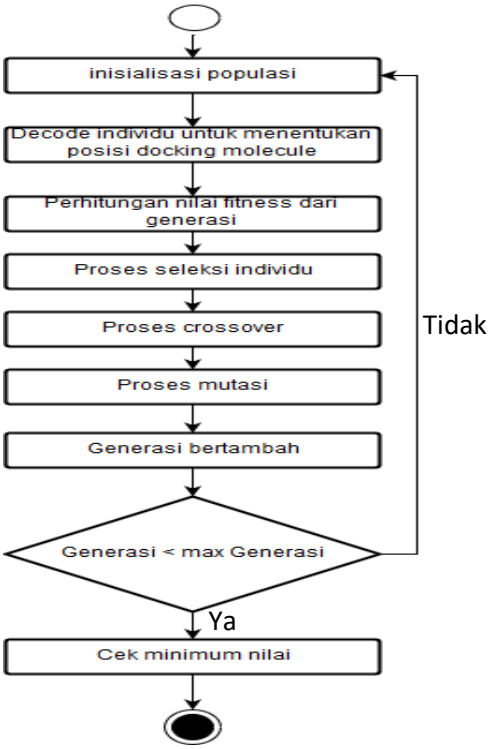
"SELECT * FROM eval WHERE pdb1 = '"+pdb1+"' AND pdb2 ='"+pdb2+"' AND posisi='"+index+"'"

BAB V
IMPLEMENTASI

Pada bab ini dijelaskan tentang implementasi dalam bahasa pemrograman Java dan hasil uji coba program menggunakan data hasil modeling sekuen yang telah dilakukan.

5.1 Algoritma Genetika

Pada tahap ini akan dijelaskan implementasi algoritma genetika pada *docking molecule* untuk penempatan ligan pada protein: Alur Algoritma Genetika :



Gambar 5. 1 Alur proses algoritma genetika

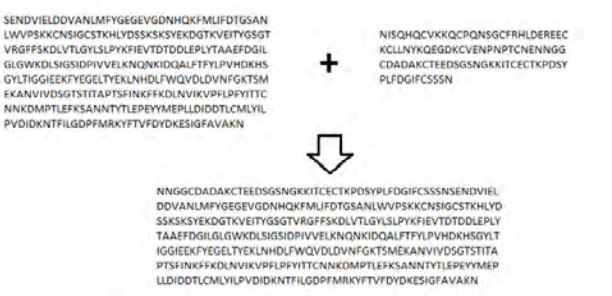
Algoritma genetika digunakan sebagai sebuah solusi docking. Setiap individu merupakan solusi permasalahan dari protein-ligan docking, dimana individu merupakan posisi ligan sebagai reseptor dari protein [5]. Oleh karena itu, konformasi ligan diwakili oleh kromosom yang dibentuk oleh nyata saat gen yang mewakili ligan translasi, orientational dan konformasi derajat kebebasan. Penelitian molekul docking memfokuskan pada produk simulasi Proses pengenalan molekuler. Hal ini bertujuan untuk mencapai konformasi dioptimalkan untuk kedua protein dan ligan dan relatif orientasi antara protein dan ligan sedemikian rupa sehingga energi bebas dari sistem keseluruhan diminimalkan.

5.1.1 Mekanisme Pendekatan Penempatan Ligan pada Protein

Permasalahan utama dalam penempatan ligan pada protein adalah menyelesaikan persamaan komplek dan berderajat tinggi yang sulit untuk diselesaikan.

1. Pada tahap awal dilakukan inisialisasi parameter input berupa protein, ligan, banyak populasi, maksimum generasi, *rate* mutasi dan crossover, serta *binding site*.
2. Pada tahap kedua acak indvidu tiap populasi, dimana merupakan solusi dari permasalahan docking. Individu merupakan solusi docking berupa posisi penempatan ligan pada protein.
3. Kemudian generate algoritma genetika untuk menentukan posisi penempatan ligan pada protein sehingga didapat sekuen baru.
4. Setelah didapatkan sekuen baru kita, kemudian kita hitung fungsi evaluasi.
5. Kita cari nilai terkecil dari evaluasi setiap penempatan ligan pada protein.

6. Kemudian kita generate hingga max generasi, untuk setiap generasi kita simpan nilai terkecil setiap generasi.
7. Setelah generasi terakhir kita cek homologi setiap sekuen cek homologi dari sekuen apakah masih sama dengan sekuen induk atau tidak.



Gambar 5. 2 Mekanisme penempatan dengan Algoritma Genetika

Berikut merupakan salah satu contoh pencarian data posisi *docing molecule* minimum dengan algoritma genetika:

a. Pada data sekuen kita cek terlebih dahulu berada pada *database* atau tidak.
b. Dilakukan pengisian parameter untuk banyak populasi, banyak generasi, rate crossover serta rate mutase seperti dibawah ini :
   Banyak popualsi >0, banyak generasi >0, nilai rate crossover >=0, dan nilai rate mutase >=0
c. Dilakukan proses inisialisasi atau generate individu dari populasi untuk generasi pertama.
d. Setelah didapatkan generasi, dilakukan decode dari setiap individu untuk menenetukan posisi docking ligan pada protein.
e. Kemudian, dilakukan generate probabilitas setiap kromoson dan dibandingankan apakah kurang dari rate

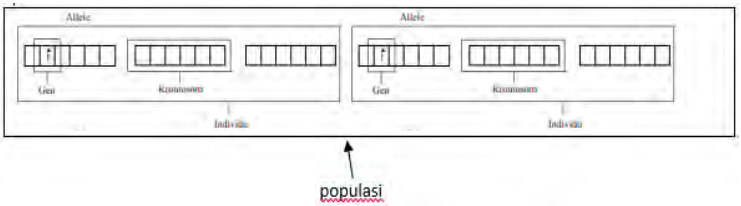
crossover, jika iya dilakukan proses crossover untuk individu tersebut. Dilakukan juga untuk proses mutasi setiap individu tersebut.

f. Proses tersebut mengalami pengulangan sampai banyak generasi yang telah input.
g. Setelah melebihi banyak generasi dilakukan proses pengecekan nilai minimun.

5.1.2 Algoritma Genetika dalam docking molekul

Algortima genetika digunakan sebagai pemecahan solusi pada docking ligan protein. Algoritma genetika digunakan untuk membangun atau mengenerate titik penempatan. Titik penempatana dilakukan pada *binding site* atau sisi aktif sebagai reseptor untuk proses docking. Dalam algoritma genetika terdapat istilah generasi, populasi, individu, kromosom serta alel.

1. Alel adalah nilai dari gen.
2. Kromosom adalah gabungan gen-gen yang membentuk nilai tertentu.
3. Individu adalah satu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
4. Populasi adalah sekumpulan individu yang akan diproses bersama dalam satu siklus proses evaluasi
5. Generasi adalah satu siklus proses evolusi atau iterasi dalam algoritma genetika.



Gambar 5. 3 Ilustrasi representasi Algoritma genetika

Binding site diperoleh dari nilai random yang unik dengan batas maksimumnya adalah panjang protein. Banyaknyak binding site tergantung pada nilai inputan yang diberikan. Algoritma genetika menghasilkan nilai 0 sampai banyaknya *binding site* secara acak disetiap individu dalam populasi.

a. Tahap awal dalam pembentukan individu adalaha menghitung panjang bit kromosom tiap individu. Individu sendiri terdiri dari kumpulam alel atau kromosom.

$$\mathrm{P} = \quad {}^{2}\log\left(\left((b-a)*10^{d}\right)+1\right)$$

Dengan

$\mathrm{P} =$ panjang kromsom

$\mathrm{b} =$ banyaknya $binding\ site$,

$\mathrm{a} =$ batas awal,

$\mathrm{d} =$ ketepatan angka dibelakang koma.

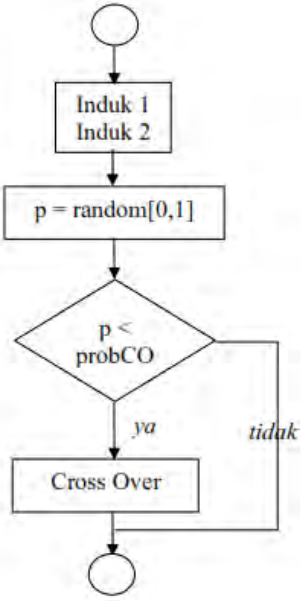
b. Setelah diperoleh panjang bit kromosm kemudian dilakukan inisialisasi acak dengan cara memberikan nilai biner 1 atau 0 secara acak untuk setiap alel dari gen.

c. Kemudian bilangan biner pada setiap individu yang telah diperoleh dari tahap inisialisasi acak diubah ke dalam bentuk decimal.

$$110 = (2^2 * 1) + (2^1 * 1) + (2^0 * 0) = 4 + 2 + 0 = 6$$

d. Nilai decimal yang diperoleh kemudian dicocokan dengan urutan random binding site yang diperoleh dari tahap pembentukan individu. Kemudian dihitung nilai evaluasi atau nilai fitness dengan index posisi sesuai dengan binding site. Jika nilai hasil perubahan biner ke decimal melebihi atau tidak cocok dengan urutan random dair inding site maka tidak dapat dilakukan docking sehingga nilai evaluasi bernilia 0 dengan kata lain sequence tidak memenuhi.
e. Tahap selanjutnya adalah operator algoritma genetika. Dalam algoritma genetika, operator yang digunakan ada 2 yaitu
    1. Operasi Evaluasi yang melibatkan proses seleksi (*selection*) didalamnya.
    2. Operasi Genetika yang melibatkan operator pindah silang (crossover) dan mutasi (*mutation*)
f. Tahap seleksi digunakan untuk memilih individu-individu mana saja yang akan dipilih untuk proses kawin silang dan mutasi. Langkah pertama yang dilakukan dalam seleksi adalah pencarian nilai fitness. Nilai fitnesss akan digunakan pada tahap-tahap seleksi berikutnya. Masing-masing individu dalam selskis akan menerima probabilitas reproduski yang tergantng pada nilai obyektif terhadap nilai obyektif dari semua individ dalam seleksi tersebut.
g. Tahap Selanjutnya adalah kawin silang atau (*crossover*). Kawin silang adalah operator dari algoritma genetika tidak melibatkan dua induk untuk membentuk kromosom baru. Pindah silang menghasilkan titik baru dalam ruang pencarian yang

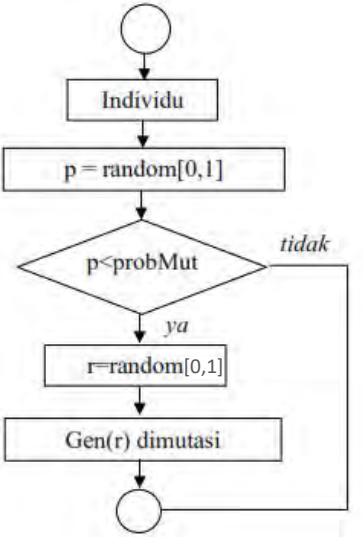
diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada. Individu dipilih secara acak untuk dilakukan crossing dengan probabilitas kawin silang antara 0 sampai dengan 1. Jika dipindah silang tidak dilakukan, maka nilai induk akan diturunkan kepada keturunan.



Gambar 5. 4 Diagram alir proses crossover

h. Tahap berikutnya pada algoritma genetika adalah mutasi gen. Operator ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi. Kromosom anak dimutasi dengan menambahkan nilai acak yang sangat kecil dengan

probabilitas yang rendah. Jika nilai probabilitas tiap gen lebih kecil dari probabilitas mutase maka gen mengalami mutasi dan terjadi perubahan biner dari 1 ke 0 dan sebaliknya. Sedangkan jika tidak memenuhi maka tidak terjadi perubahan nilai.

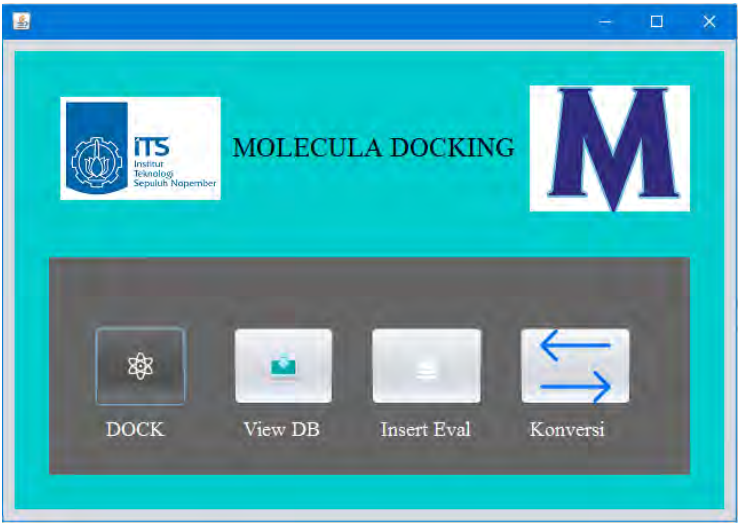


Gambar 5. 5 Diagram alir proses mutasi

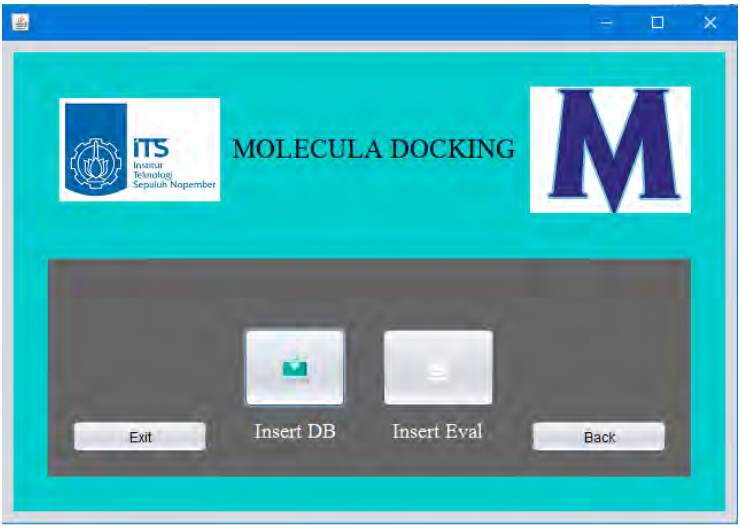
i. Kemudian kembali ke tahap generate kromosom sampai ke tahap mutasi. Algoritma genetika berhenti ketika generasi melebihi maksimum generasi.
j. Tahap selanjutnya mencari nilai minimum dari semua generasi. Hasil digunakan untuk cek homology kecocokan hasil docking dengan menggunakan software MATLAB

5.2 *Graphical User Interface (GUI)* Program

Perancangan program yang telah dibuat selanjutnya diimplementasikan dalam bahasa pemrograman Java dengan menggunakan *software* Netbeans IDE 8.1. Web server yang digunakan adalah XAMPP 3.2.1 dengan database MySQL. Gambar 5.6 merupakan tampilan menu utama dan Gambar 5.7 merupakan tampilan menu sekunder dari *insert database*.



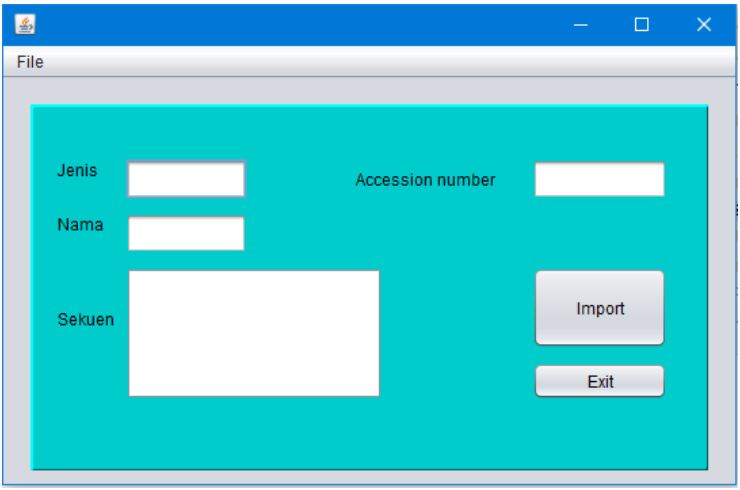
Gambar 5. 6 Tampilan Menu Utama



Gambar 5. 7 Tampilan Menu Sekunder

5.2.1 Form Input Data protein

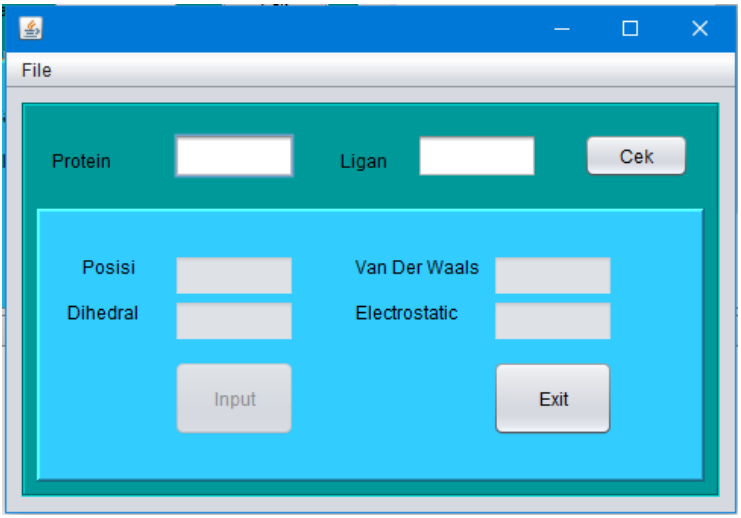
Gambar 5.8 merupakan form untuk menambahkan data protein dengan input data berupa sekuen, jenis, nama dan *accession number*. Jika ada salah satu bagian yang kosong maka data tidak dapat dimasukkan. Jika sebelumnya nama protein sudah dimasukkan kedalam basis data, maka tidak dapat dimasukkan lagi. Setelah memasukkan data, tabel akan otomatis menampilkan isi basis data.



Gambar 5. 8 Form Input Data Protein

5.2.2 Form Input Data

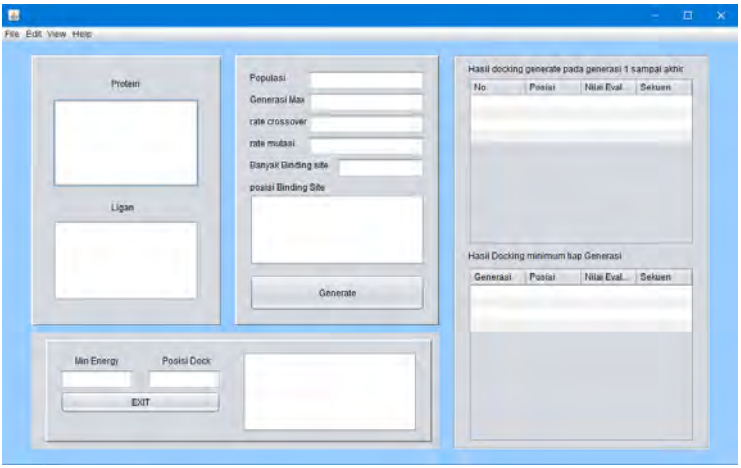
Gambar 5.9 merupakan form untuk menambahkan data hasil modeling sekuen dengan inputam accession number dari protein dan ligan, posisi, *diherdral*, *van der waals*, dan *electrostatic*. Jika sebelumnya posisi sudah dimasukkan kedalam basis data, maka tidak dapat dimasukkan lagi.



Gambar 5. 9 Form Input Parameter Data

5.2.3 Menu Generate Algoritma Genetika

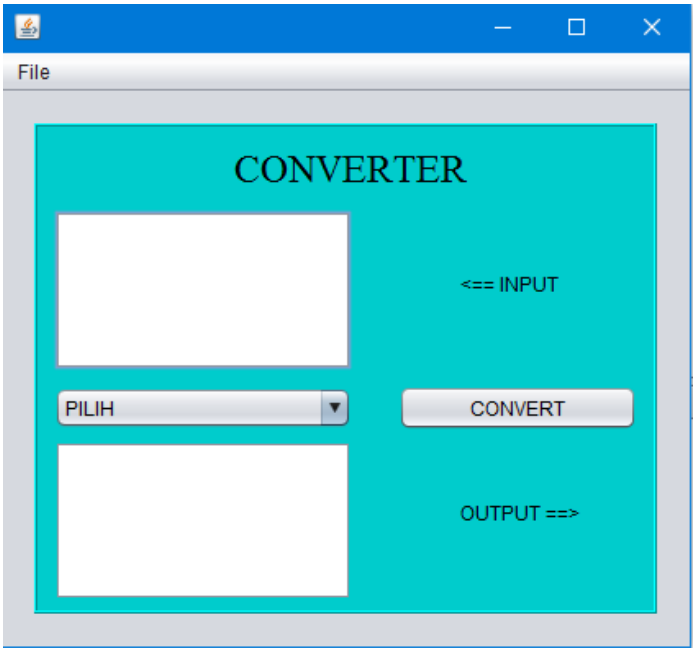
Gambar 5.10 merupakan menu untuk generate algoritma genetika untuk menetukan posisi docking ligan pada protein dengan input sekuen yang sudah berada pada *database* dan input parameter algortima genetika seperti populasim maksimum generasi, *rate crossover*, dan *rate mutasi*. Terdapat 2 tombol yaitu tombol generate dan tombol *exit*. Tombol generate untuk menjalankan proses algoritma genetika untuk melakukan docking ligan pada protein.



Gambar 5. 10 Menu Generate Algoritma Genetika

5.2.4 Menu *Converter*

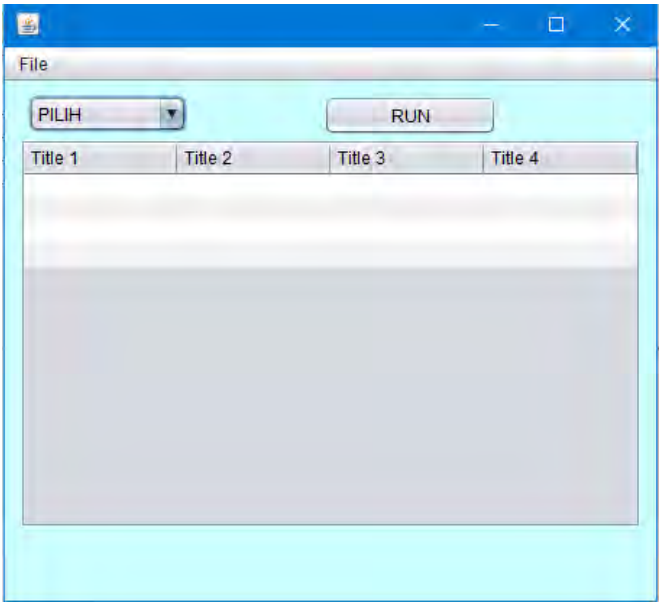
Menu untuk *converter* atau merubah dari protein ke DNA atau DNA ke protein. Terdapat *Combobox* yang berisi pilihan menu *converter* dan tombol yaitu tombol convert. Tombol convert untuk menjalankan proses merubah sekuen dari *input* sesuai dengan pilihan *Combobox.* Gambar 5.11 merupakan interface dari menu *converter.*



Gambar 5. 11 Menu Converter DNA to Protein

5.2.5 Menu menampilkan *Database*

Menu untuk menampilkan database protein dan nilai evaluasi. Terdapat *Combobox* yang berisi pilihan menu *database* yang akan dipilih. Tombol *run* untuk menjalankan proses untuk menampilkan database pada table sesuai dari *input* yang dipilihan pada *Combobox*. Gambar 5.12 merupakan tampilan interface.



Gambar 5. 12 Menu untuk menampilkan database

5.3 Implementasi dengan Java

Pada sub bab ini akan ditunjukkkan hasil dari implementasi program. Pada penelitian ini digunakan sekuen protein virus plasmodium malariae (2ANL_2) dan ligan dari tumbuhan yang dipercaya dapat menyebuhkan malaria, dengan populasi 50, banyak generasi 100, banyak binding site 100, probabilitas *crossover* 0.6 dan probabilitas *mutasi* 0,001. Tahapan implementasi dengan java :

a. Nilai awal didapatkan jarak antara titik, jumlah titik pada domain, dan panjang bit :

   $\text{Jarak antar titik} = 10^{-d}$, dengan $d = 0$

   $\text{Jarak antar titik} = 10^{-0} = 1$

   $\text{Jumlah titik pada domain} = (b - a) * 10^{d}$

   $a\ (\text{batas bawah}) = 0$

   $b\ (\text{batas atas}) = \text{Banyak binding site} = 100$

Jumlah titik pada domain $= (100 - 0) * 10^1 = 100$

Jumlah bit $= \left[\quad ^2\log\left(\left((b-a) * 10^d\right) + 1\right)\right] = 7$

b. Generate kromosom awal setiap individu, dilakukan dengan memberi nilai acak antara 1 atau 0 sebanyak panjang bit kromosom.

Tabel 5. 1 Kromosom acak setiap individu

NO	Individu	NO	Individu
1	0100111	26	0011110
2	0101010	27	1011000
3	0000110	28	1101101
4	1101101	29	1111101
5	1011011	30	1100101
6	0010011	31	1111001
7	1111100	32	1101000
8	1011100	33	1111001
9	1110110	34	1011110
10	0110101	35	0011100
11	1010000	36	0011000
12	0111101	37	1110000
13	0100010	38	1100111
14	0010001	39	1100111
15	0010000	40	1011000
16	1100000	41	1010110
17	0100110	42	1000001
18	1110001	43	0100001
19	1010110	44	0111000
20	1001011	45	1010010
21	1000010	46	1111001
22	0011000	47	1010000
23	1101100	48	0001011
24	0111010	49	0110101
25	0101001	50	1110010

c. Setelah didapatkan generate kromosom kemudian di lakukan proses *decode* atau penerjemahan biner ke desimal sehingga didapatkan nilai sebagai berikut :

Tabel 5. 2 Hasil Decode Kromosom setiap individu

No	Decode	Binding site	No	Decode	Binding site
1	39	85	26	30	135
2	42	193	27	88	195
3	6	67	28	109	TM
4	109	TM	29	125	TM
5	91	167	30	101	TM
6	19	305	31	121	TM
7	124	TM	32	104	TM
8	92	19	33	121	TM
9	118	TM	34	94	196
10	53	53	35	28	14
11	80	228	36	24	1
12	61	208	37	112	TM
13	34	288	38	103	TM
14	17	156	39	103	TM
15	16	261	40	88	195
16	96	17	41	86	309
17	38	157	42	65	222
18	113	TM	43	33	71
19	86	309	44	56	273
20	75	186	45	82	325
21	66	317	46	121	TM
22	24	1	47	80	228
23	108	TM	48	11	31
24	58	174	49	53	53
25	41	269	50	114	TM

d. Setelah didapat hasil decode dilakukan pengujian fungsi evaluasi fitness dengan nilai decode sebagai posisi docking protein dan ligan dan didapat nilai evaluasi sebagai berikut:

Tabel 5. 3 Hasil Evaluasi setiap individu

No	Hasil	No	Hasil
1	-12422.49	26	-11421.01
2	-12422.49	27	-11421.01
3	-13378.93	28	-11812.12
4	-13378.93	29	-11812.12
5	-12528.31	30	-11345.00
6	-12528.31	31	-11345.00
7	0.00	32	0.00
8	-10864.37	33	-11775.43
9	-10864.37	34	-11775.43
10	-11706.40	35	-11504.79
11	-11706.40	36	-11504.79
12	0.00	37	-11319.36
13	-11273.64	38	-11319.36
14	-11273.64	39	-12753.89
15	0.00	40	-12753.89
16	-11843.65	41	0.00
17	-11843.65	42	-11007.22
18	-12986.19	43	-11007.22
19	-12986.19	44	-12355.27
20	-11469.08	45	-12355.27
21	-11469.08	46	-11455.51

22	-12667.30	47	-11455.51
23	-12667.30	48	-12249.02
24	-11447.06	49	-12249.02
25	-11447.06	50	0.00

e. Tahap selanjutnya adalah proses seleksi dimana probabilitas dari setiap individu diacak antara 0 dan 1.

Tabel 5. 4 Hasil Proses Seleksi

No	Seleksi	No	Seleksi
1	seleksi 20	26	seleksi 25
2	seleksi 2	27	seleksi 0
3	seleksi 1	28	seleksi 10
4	seleksi 40	29	seleksi 19
5	seleksi 15	30	seleksi 41
6	seleksi 10	31	seleksi 44
7	seleksi 39	32	seleksi 41
8	seleksi 43	33	seleksi 21
9	seleksi 40	34	seleksi 34
10	seleksi 46	35	seleksi 24
11	seleksi 25	36	seleksi 9
12	seleksi 9	37	seleksi 44
13	seleksi 15	38	seleksi 35
14	seleksi 18	39	seleksi 34
15	seleksi 39	40	seleksi 14
16	seleksi 4	41	seleksi 26
17	seleksi 20	42	seleksi 33
18	seleksi 23	43	seleksi 4

19	seleksi 25	44	seleksi 21
20	seleksi 11	45	seleksi 33
21	seleksi 40	46	seleksi 40
22	seleksi 15	47	seleksi 43
23	seleksi 13	48	seleksi 4
24	seleksi 10	49	seleksi 1
25	seleksi 23	50	seleksi 5

f. Setelah proses seleksi dilakukan proses crossover dimana probabilitas untuk setiap individu diacak kemudian dibandingan dengan probabilitas crossover yang sudah diinputkan jika lebih dari tidak terjadi crossover jika kurang dari maka individu tersebut mengalami crossover.

Tabel 5. 5 Hasil Crossover

No	Crossover	No	Crossover
1	index 0	13	index 22
2	index 2	14	index 25
3	index 3	15	index 29
4	index 5	16	index 30
5	index 6	17	index 32
6	index 8	18	index 33
7	index 10	19	index 34
8	index 13	20	index 35
9	index 18	21	index 36
10	index 19	22	index 43
11	index 20	23	index 46
12	index 21	24	index 47

g. Tahap Selanjutnya adalah tahap mutasi, merupakn tahap dimana setiap alel atau kromosom dari setiap individu diberikan nilai acak antara 0 dan 1, kemudian setiap kromosom dicek apakah lebih dari probabilitas mutasi maka akan terjadi prose mutasi dari angka biner 0 menjadi 1 dan sebaliknya. Hasil mutasi tahap awal sebagai berikut:

Tabel 5. 6 Hasil Mutasi

i = 14
index indiv yang mutasi 1
index kromosom yang mutasi 6
i = 17
index indiv yang mutasi 2
index kromosom yang mutasi 2
i = 149
index indiv yang mutasi 21
index kromosom yang mutasi 1
i = 206
index indiv yang mutasi 29
index kromosom yang mutasi 2
i = 249
index indiv yang mutasi 35
index kromosom yang mutasi 3

h. Setelah tahap mutasi selesai , generasi akan bertambah dan dilakukan lagi proses b sampai g hingga tidak memenuhi banyak generas.

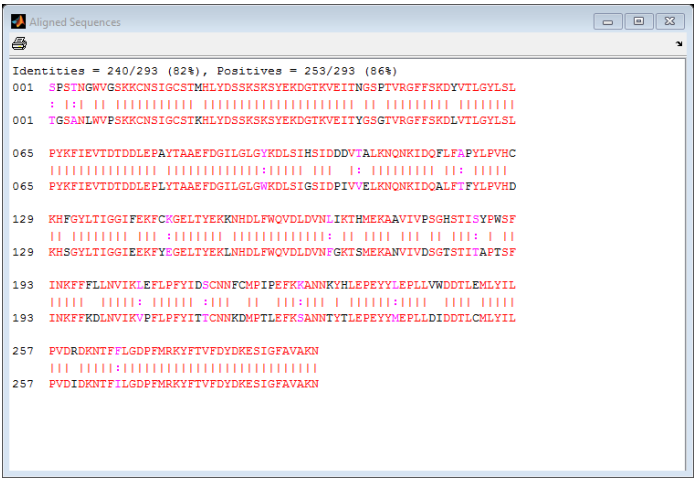
i. Tahap terakhir algoritma genetika untuk ligan docking adalah pengecekan nilai minimum dari setiap generasi dimana setiap populasi terbaik diambil individu dengan nilai minimun.

Tabel 5. 7 Hasil Akhir Algoritma Genetika

Energi terendah = -13378.93 Posisi docking = 193 Hasil sekuen = SEEDFIELHDVANLMFYGEGEVPDNHQKFMIIK LCGSANLWKPSKRCNSIPRMSKHLYDSVKSDFY EKDGTKSCATYGEGTARGLFSKDLVTLGYLSLP IKFIEVTDTDDLGPLYVAAEFIKILGLGWKDLSIG SIDPIILYLKTMNKIDQALFCFWLPVTDGHSGYL TCGGIEEKFNDGYSTYEKLNHDLSDFPMIQKQGI SRPINVDLDVNGGKTSMEKVNMIVDSGISTHTG PTSFIGKFFKQLNVIPDVFLPFYIMTCLNGLMRC WEFKHANWKYTLCPELYPEPLLDIDDTHCMLYI LPVDIDKNTFILGGQFSRKYFTGFDYDKESIGFA VAKN Generasi = 0

j. Setelah didapatkan nilai minimum kemudian akan dilakukan pengecek homologi dengan toolbox MATLAB dengan algoritma nedleeman-wunsch dan smith-waterman. Hasil pensejajaran dapat dilihat pada Gambar 5.13 dan Gambar 5.14

Gambar 5. 13 Hasil pensejajaran dengan algoritma nedleeman-wunsch

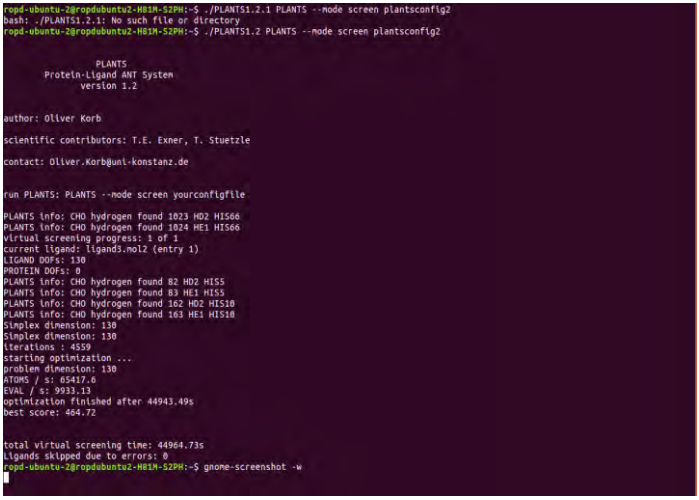


Gambar 5. 14 Hasil pensejajaran dengan algoritma Smith Waterman

5.4 Implementasi dengan PLANTS

Pada sub bab ini akan ditunjukkkan hasil dari implementasi program PLANTS. Program menggunakan algoritma *ant colony* (koloni semut) untuk melakukan docking, selain itu program juga menggunakan klustering untuk *docking molecule*. Pada program tersebut dibutuhkan file configurasi untuk menjalankan software tersebut dimana dibutuhkan file berekstensi mol2 untuk protein dan ligan file dengan radius serta struktur kluster dan rmsd.

Pada penelitian ini digunakan sekuen protein virus plasmodium malariae (2ANL_2) dan ligan dari tumbuhan yang dipercaya dapat menyebuhkan malaria, dengan binding site center (-4.12761 57.2797 -10.175) , radius 12.8396, dengan cluster struktur 10 dan cluster rmsd 2.



Gambar 5. 15 Gambar Running dengan PLANTS

Hasil running dengan menggunakan PLANTS berupa file mol2, *best ranking* berisi berisi nilai Skor, jumlah penilaian

fungsi evaluation dan waktu docking untuk setiap pose ligan terlaris peringkat, *constraints* berisi informasi tentang kendala untuk semua ligan pose, *corresponding , features* berisi informasi tentang persyaratan fungsi mencetak parsial untuk semua ligan pose, *optimizer* berisi informasi mengenai pengaturan algoritma pencarian, *ranking* berisi sama seperti bestranking tetapi untuk semua ligan pose, dan *skipped ligands*. Untuk setiap kluster di didapatkan nilai evaluasi score.

Tabel 5. 8 Hasil running *best rangking* dengan PLANTS

SCORE_RB_PEN	SCORE_NORM_HEVATOMS	SCORE_NORM_CRT_HEVATOMS	SCORE_NORM_WEIGHT	SCORE_NORM_CRT_WEIGHT
459.813	707.813	2.31062	78.7583	0.161441

Keterangan :
SCORE_RB_PEN : TOTAL SCORE ditambah hukuman nilai setiap ligan *rotatable Bond*
SCORE_NORM_HEVATOMS : TOTAL SCORE dibagi dengan jumlah ligan berat atom
SCORE_NORM_CRT_HEVATOMS : TOTAL SCORE dibagi dengan akar kubik jumlah ligan berat atom
SCORE_NORM_WEIGHT : TOTAL SCORE dibagi dengan berat molekul ligan
SCORE_NORM_CRT_WEIGHT : dibagi dengan akar kubik berat molekul ligan

Tabel 5. 9 Hasil running *best rangking* dengan PLANTS

EVAL	TIME	SCORE_RB_PEN_NORM_CRT_HEVATOMS	SCORE_NORM_CONTACT
9898.66	44568026 4	32.4384	121.237

Keterangan :
SCORE_RB_PEN_NORM_CRT_HEVATOMS : SCORE RB PEN dibagi dengan akar kubik jumlah ligan berat atom
SCORE_NORM_CONTACT : TOTAL SCORE dibagi dengan jumlah protein ligan kontak
EVAL : jumlah penilaian evaluasi fungsi
TIME : waktu docking

5.5 Hasil Akhir

Pada sub bab ini ditunjukkan hasil dari implemtasi dengan program java dan PLANTS didapatkan hasil untuk nilai evaluasi sebagai berikut:

Tabel 5. 10 Hasil implementasi

No	Protein	Ligan	Nilai evaluasi	Posisi
1	Malaria	Kina	-18234.47	134
2	Malaria	Jambu merah	-13851.972	41
3	DBD	Kina	-3291.149	28
4	DBD	Jambu merah	-3833.07	52

Tabel 5. 11 Hasil docking dengan PLANTS

No	Protein	Ligan	Nilai evaluasi
1	Malaria	Kina	9898.66
2	Malaria	Jambu merah	9933.13
3	DBD	Kina	27641.1
4	DBD	Jambu merah	27559.9

Pada tabel 5.10 dan tabel 5.11 menunjukan perbandingan hasil implementasi menggunakan java parameter yang

digunakan adalah parameter CHARMM dengan nilai hasil docking evaluasi sekitar -13254.93. Implementasi menggunakan software PLANTS parameter yang digunakan adalah parameter GOLD didapatkan nilai 9898.66 Terlihat bahaw nilai yang dihasilkan memiliki selisih yang besar di karena pamater yang digunakan berbeda. Dapat dilihat pada tabel 5.10 didapatkan nilai evaluasi bernilai negatif diartikan bahwa gaya atau benda itu tidak melakukan usaha dan benda tidak mengeluarkan energi, tetapi mendapatkan energi. Sebagai contoh adalah sebuah benda yang dilemparkan vertikal ke atas. Selama benda bergerak ke atas, arah gaya berat benda berlawanan dengan perpindahan benda. Hal itu dapat dikatakan bahwa gaya berat benda melakukan usaha yang negatif.

Tabel 5. 12 Hasil Pensejajaran

No	Protein	Ligan	Persentasi kemiripan	
			Needleman-wunsch	Smith-Waterman
1	Malaria	Kina	**75%**	**82%**
2	Malaria	Jambu merah	76%	82%
3	DBD	Kina	75%	75%
4	DBD	Jambu merah	**64%**	**64%**

Pada tabel 5.12 menunjukan hasil pensejajaran untuk protein virus Malaria dan DBD, untuk ligan adalah kina dan jambu merah. Dari tabel terlihat untuk protein virus malaria dan ligan kina memiliki persentasi kemiripan nilai lebih kecil sama dengan dibandingkan dengan protein virus malaria dan ligan jambu merah. Kemudian dari tabel juga terlihat untuk protein virus DBD dan ligan jambu merah memiliki persentasi kemiripan nilai lebih kecil dibandingkan dengan nilai dari protein virus DBD dan ligan kina. Persentasi kemiripan

mempengaruhi suatu docking, untuk nilai persentasi kemiripan (homologi) kecil maka protein tersebut sudah memiliki sifat yang berbeda dari sekuen awal protein.

BAB VI
PENUTUP

Pada bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan.dan saran yang dapat digunakan jika penelitian ini dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian program, maka dapat diambil kesimpulan bahwa Algoritma Genetika dapat diterapkan untuk *molecule docking* ligan dan protein dengan protein virus malaria dan DBD, ligan tumbuhan kina dan jambu merah. Penerapan algoritma genetika dalam docking tidak berlaku untuk semua protein dan ligan. Dalam penerapannya tingkat homologi mempengaruhi keberhasilan. Semakin tinggi homologi sekuen setelah docking dan sekuen awal virus , maka docking tidak berhasil.

6.2 Saran

Ada beberapa hal yang penulis sarankan untuk pengembangan penelitian selannjutnya :

1. Pada Tugas Akhir ini modeling dapat dilakukan sesuai konformasi.
2. Data yang digunakan dalam Tugas Akhir ini berjumlah 928 data posisi docking. Jumlah tersebut belum mencakup keseluruhan docking molecule untuk jenis protein yang berbeda. Sehingga, lebih baik apabila data yang digunakan dapat lebih banyak.

DAFTAR PUSTAKA

[1] Motiejunas, D., & Wade, R. (2006). *Structural, Energetics, and Dynamic Aspects of Ligand-Receptor Interactions. In J.B. Taylor & D. J. Triggle (Eds.).* Comprehensive Medicinal Chemistry II Volume 4: Computer-Assited Drug Design, Vol.4, pp. 193-214. Elsevier.

[2] Gane PJ, & Dean PM. (2000). *Recent Advances in Structure-Based Rational Drug Design.* Current Opinion in Structural Biology, 10:401-404.

[3] I. Belda et al. (2005). *ENPDA: An Evolutionary Structure-Based De Novo Peptide Design Algorithm.* Journal of Computer-Aided Molecular Design 19, 585-601.

[4] Osterberg, F., Morris, G., Sanner, M., Olson, A., & Goodsell, D. (2002). *Automated Docking to Multiple Target Structure: Incorporation of Protein Mobility and Structural Water Heterogeneity in Auto Dock.* Protein 46, 34-40.

[5] Magalhães, C. d. (2004). *A Genetic Algorithm for The Ligand-Protein Docking Problem.* Genetics and Molecular Biology 27, 605-610.

[6] Floquet, N., Marechal, J., Badet-Denisot, M., Robert, C., Dauchez, M., & Perahia, D. (2006). *Using Normal Modes Analysis as A Prerequisite for Drug Design : Application to Matix Metalloproteinase Inhibitors.* FEBS Letters 580, 5130-5136.

[7] Mouawad, L., Perahia, D., Cui, Q., & Bahar, I. (2006). *Normal Mode Analysis Theory and Applications to Biological and Chemical Systems.* Chapman & Hall.

[8] Spyrakis, F. (2007). *The Consequences of Scoring Docked Ligand Conformations Using Free Energy*

Correlations. European Journal of Medicinal Chemistry 42, 921-933.

[9] Thomsen, R., Christensen, H., & MolDock. (2006). *A New Technique for High-Accuracy Molecular Docking.* Journal of Medicinal Chemistry 49, 3315-3321.

[10] Huang, S., & Zou, X. (2007). *Efficient Molecular Docking of NMR Structures : Application to HIV-1 Protease.* Protein Science 16, 43-51.

[11] Guerler, A., S.Moll, M.Weber, Meyer, H., & Cordes, F. (2008). *Selection and Flexible Optimization of Binding Modes from Comformation Ensembles.* Biosystem 92, 42-48.

[12] Van Gunsteren. G., & Berendsen HJC. (1987). *Groningen Molecular Simulation (GROMOS) Library Manual.* Biomos, Groningen.

[13] Smith LJ, Mark AE, Dobson CM and van Gunsteren WF (1995*). Comparison of MD Simulation and NMR Experiments for Hen Lysozyme. Analysis of Local Fluctuations, Cooperative, and Global Changes.* Biochemistry, 34: 1091810931.

[14] Pascutti PG, Mundim KC, Ito AS and Bisch PM (1999). *Polarization Effects on Peptide Conformation at Water-Membrane Interface by Molecular Dynamics Simulation.* J Comp Chem, 20 : 971-982.

[15] Lima, A., Philot, E., Perahia, D., Braz, A., & Scott, L. (2012). *GANM: A Protein-Ligand Docking Approach Based on Genetic Algorithm and Norma Modes.* Appl.Math.Comput.219, 511-520.

[16] Mahdiyah, U., Irawan, M. I., & E.M.Imah. (2016). *Integrating Data Selection and Extreme Learning Machine to Predict Protein-Ligand Binding Site.*

[17] Shahab, M. L. & Irawan, M. I. (2016). *Algoritma Genetika Ganda untuk Capacitated Vehicle Routing Problem.* Journal of Sains dan Seni ITS

[18] Muhtaromi, M. & Irawan, M. I. (2016). *Perancangan Prototipe Perangkat Lunak Untuk Penempatan Pegawai Dengan Model Pilihan Dari Perspektif Dua Arah Berbasis Algoritma Genetika.* Journal of Sains dan Seni ITS

[19] Firmansyah, A., Utomo , D. B. & Irawan, M. I. (2016). *Algoritma Genetika Ddengan Modifikasi Kromosom Untuk Penyelesaian Masalah Penjadwalan Flowshop.* Journal of Sain dan Seni Vol 1 no. 1

[20] Pramsistya, Y., Utomo , D. B. & Irawan, M. I. (2010). *Optimasi Penempatan BTS dengan Menggunakan Algoritma Genetika.* Journal of Sain dan Seni Vol 1 no. 1

LAMPIRAN A
Source Code

1. Menu Home

```
package ta;
public class Menu_Utama extends javax.swing.JFrame {
    public Menu_Utama() {
        initComponents();
        setLocationRelativeTo(this);
    }
        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        new GUi_TA1().show();
        this.dispose();
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            new db().show();
        } catch (SQLException ex) {
            Logger.getLogger(Menu_Utama.class.getName()).log(Level.SEVERE, null,
ex);
        }
        this.dispose();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new Menu_Sekunder().show();
        this.dispose();
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        new Convert().show();
        this.dispose();
    }
```

2. Menu Sekunder

```
package ta;
public class Menu_Utama extends javax.swing.JFrame {
    public Menu_Utama() {
        initComponents();
        setLocationRelativeTo(this);
    }
            private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new Import_Eval_Ligan().show();
        this.dispose();
    }
```

```
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        new Import_Protein().show();
        this.dispose();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        new Menu_Utama().show();
        this.dispose();
    }
```

3. *Input* Protein dan ligan *database*

```
package ta;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Locale;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import ta.Tools.Database;
public class Import_Protein extends javax.swing.JFrame {
    ResultSet rst;
    public Import_Protein() {
        initComponents();
        setLocationRelativeTo(this);
        try {
            koneksi.connectFirst();
        } catch (SQLException ex) {
            Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    Database koneksi= new Database();
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        int panjang = sk.getText().length();

if(jns.getText().toUpperCase().equals("PROTEIN")||jns.getText().toUpperCase().equ
als("LIGAN")){
```

```
        String query="INSERT INTO protein VALUES
('"+jns.getText().toUpperCase()+"','"+acn.getText().toUpperCase()+"','"+nm.getText(
)+"','"+panjang+"','"+sk.getText().toUpperCase(Locale.ITALY)+"')";
        try {
           koneksi.executeUpdate(query);
        } catch (SQLException | IllegalStateException ex) {
           Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE,
null, ex);
        }
      }
      else{
        JOptionPane.showMessageDialog(null, "INPUT JENIS
SALAH\nMASUKKAN PROTEIN ATAU LIGAN");
      }
   }

   private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
      this.setVisible(false);
   }

   private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
      new Menu_Utama().show();
      this.dispose();
   }
```

4. *Input Evaluasi Data*

```
package ta;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import ta.Tools.Database;
public class Import_Eval_Ligan extends javax.swing.JFrame {
   Database koneksi= new Database();
   ResultSet rst;
   public Import_Eval_Ligan() {
      initComponents();
      setLocationRelativeTo(this);
      try {
        koneksi.connectFirst();
        if(koneksi.isConnected())
           System.out.println("connected");
        else
           System.out.println("not connected");
        ps.setEnabled(false);
        vdw.setEnabled(false);
        dh.setEnabled(false);
```

```
            elc.setEnabled(false);
            jButton1.setEnabled(false);
        } catch (SQLException ex) {
            Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        int p=Integer.parseInt(ps.getText());
        double d=Double.parseDouble(dh.getText());
        double vd=Double.parseDouble(vdw.getText());
        double elec=Double.parseDouble(elc.getText());
        int count =0;
        ResultSet hsl;
        String q1="SELECT count(NO) FROM eval";
        try {
            hsl=koneksi.executeSelect(q1);
            hsl.next();
            count=hsl.getInt(1)+1;
        } catch (SQLException | IllegalStateException ex) {
            Logger.getLogger(Import_Eval_Ligan.class.getName()).log(Level.SEVERE,
null, ex);
        }
        String query="INSERT INTO eval VALUES
('"+count+"','"+prt.getText().toUpperCase()+"','"+lgn.getText().toUpperCase()+"','"+
p+"','"+d+"','"+vd+"','"+elec+"')";
        try {
            koneksi.executeUpdate(query);
            ps.setText("");
            dh.setText("");
            vdw.setText("");
            elc.setText("");
        } catch (SQLException | IllegalStateException ex) {
            Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        ResultSet hsl;
        try {
            koneksi.connectFirst();
            //SELECT pdb FROM protein WHERE pdb = '2ANL_2'
```

```
        String query="SELECT pdb,jenis FROM protein WHERE pdb =
'"+prt.getText().toUpperCase()+"'";
        String query1="SELECT pdb,jenis FROM protein WHERE pdb =
'"+lgn.getText().toUpperCase()+"'";
        hsl=koneksi.executeSelect(query);
        String pdb_name=null,pdb_name1=null,jenis1=null,jenis=null;
        while(hsl.next()){
           pdb_name=hsl.getString("pdb");
           jenis=hsl.getString("jenis");
        }
        hsl=koneksi.executeSelect(query1);
        while(hsl.next()){
           pdb_name1=hsl.getString("pdb");
           jenis1=hsl.getString("jenis");
        }
        System.out.println(pdb_name);
        System.out.println(jenis);
        System.out.println(pdb_name1);
        System.out.println(jenis1);

if((jenis!=null&&jenis1!=null)&&(jenis.toUpperCase().equals("PROTEIN")&&jenis
1.toUpperCase().equals("LIGAN"))){
           ps.setEnabled(true);
           vdw.setEnabled(true);
           dh.setEnabled(true);
           elc.setEnabled(true);
           jButton1.setEnabled(true);
        }
        else if(jenis==null&&jenis1==null){
           System.out.println("salah");
        }
        else
if(!jenis.toUpperCase().equals("PROTEIN")||!jenis1.toUpperCase().equals("LIGAN"
)){
           System.out.println("input salah");
        }
        else{
           System.out.println("Input tidak sesuai");
        }
     } catch (SQLException ex) {
        Logger.getLogger(Import_Eval_Ligan.class.getName()).log(Level.SEVERE,
null, ex);
     }
  }

  private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
     this.setVisible(false);
  }
```

```
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Menu_Utama().show();
        this.dispose();       // TODO add your handling code here:
    }
```

5. Menu Generate Algoritma Genetika

```
package ta;
import java.awt.Color;
import ta.Tools.FileChecker;
import ta.Tools.protein_ligan_eval;
import ta.Tools.Database;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintStream;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.PrimitiveIterator;
import java.util.Random;
import java.util.concurrent.ThreadLocalRandom;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.BadLocationException;
import javax.swing.text.DefaultHighlighter;
public class GUi_TA1 extends javax.swing.JFrame {
    int p_protein=0,p_ligan=0;              //panjang protein dan ligan
    int populasi,mxgen,defaultSize=100,n=0;    //banyak populasi , max generasi dan
iterasi
    char[]prot;                             //sequence protein
    double pc,pm;
    String []dock;                    //Simpan docking
    int krom[][],krom_seleksi[][];      //kromoson
    int x[],s[];                      //posisi docking dan kromosom seleksi
    double fx[],p[],q[],r[];          //fungsi evaluasi, probabilitas , total dan random value
    double min_eval[];
    int min_pos[];
    int d=0;                          //ketepatan angka dibelakang koma
    int a=0;                          //batas bawah
    int b=0;                          //batas atass
    int jAntarTitik=0;                //jarak antara titik
```

```
    int bit=0;                      //panjang bit
    String lgn="",prt;                  //Ligan sequence
    static String mm,dl;
    int hp;
    Database koneksi= new Database();
    ResultSet rst;
    double cal_db[];
    int bs[];
    int bbs;
    String min_dock[];
    int ind=0;
    private String hs;
    private double[] s_eval;
    private int[] s_pos;
    private String[] s_dock;
    int in_s_eval=0;
    int in_s_pos=0;
    int in_s_dock=0;
    public GUi_TA1() {
        initComponents();
        bs_list.setEditable(false);
        in_p.setForeground(Color.blue);
        in_l.setForeground(Color.red);
        setLocationRelativeTo(this);
        try {
            koneksi.connectFirst();
            if(koneksi.isConnected())
                System.out.println("connected");
            else
                System.out.println("not connected");
        } catch (SQLException ex) {
            Logger.getLogger(Import_Protein.class.getName()).log(Level.SEVERE,  null,
ex);
        }
    }
    public void inisialisasi(String protein, String ligan, int pop,int mxgen,double
pc,double pm) throws SQLException, BadLocationException{
        p_protein = protein.length();                   //mengambil panjang protein dan
ligan
        p_ligan = ligan.length();
        this.pc=pc;
        this.pm=pm;
        this.mxgen=mxgen;
        b=bbs;                              //menentukan batas atas dan dimana batas bawah
=0
        populasi=pop;                               //mengambil banyak populasi
        jAntarTitik=(int) ((b - a) * Math.pow(10, (d)));        //jumlah titik
        System.out.println("jAntarTitik = "+jAntarTitik);
```

```
    bit=(int)Math.ceil(Math.log(jAntarTitik+1)/Math.log(2));//mengambil    panjang
bit protein
    System.out.println("bit = "+bit);
    lgn=ligan;                                  //sequence protein
    prt=protein;
    prot = new char[p_protein];                 //inisialisasi array untuk menyimpan
protein
    prot = protein.toCharArray();               //menyimpan protein pada array
    krom = new int[populasi][bit];              //inisialisasi individu
    krom_seleksi = new int[populasi][bit];
    dock=new String[populasi];                  //inisialisasi array penyimpanan
docking
    x=new int[populasi];                        //inisialisasi x (posisi docking)
    fx=new double[populasi];                    //inisialisasi fungsi evaluasi
    p=new double[populasi];                     //probabilitas
    q=new double[populasi];                     //jumlahan propabilitas
    r=new double[populasi];                     //random value;
    s=new int[populasi];
    s_eval=new double [populasi*mxgen];
    s_pos=new int[populasi*mxgen];
    s_dock=new String[populasi*mxgen];
    min_eval=new double[mxgen];
    min_pos=new int[mxgen];
    min_dock=new String[mxgen];
    generateKromosom();
  }
  public void generateKromosom() throws SQLException, BadLocationException,
BadLocationException {
    if(n<mxgen){
      try{
        String A="E:\\Running\\generate\\generate kromosom "+n+".txt";
        PrintStream myconsole=new PrintStream(new File(A));
        System.setOut(myconsole);
        myconsole.println("Tahap Generate kromosom ke "+n);
        for (int j = 0; j <populasi; j++) {
          String str="";
          for(int i=0;i<bit;i++){
            int gene1 = (int) Math.round(Math.random());
            krom[j][i] = gene1;
            str=str+gene1;
            //System.out.print(gene1+" ");
          }
          myconsole.println(str);
          //System.out.println("");
        }
        myconsole.println("Tahap generate kromosom "+n+" selesai");
      }catch(FileNotFoundException Generate){
        System.out.println(Generate);
      }
```

```
    decode();}
    else{
       DefaultTableModel d1tm=(DefaultTableModel) jTable2.getModel();
       d1tm.setRowCount(0);
       String data1[]=new String [4];
       int ii=0;
       while(ii<s_eval.length){
          data1[0]=String.valueOf(ii+1);
          data1[1]=String.valueOf(s_pos[ii]);
          data1[2]=String.valueOf(s_eval[ii]);
          data1[3]=s_dock[ii];
          d1tm.addRow(data1);
       ii++;
       }
       int i=0;
       DefaultTableModel dtm=(DefaultTableModel) jTable1.getModel();
       dtm.setRowCount(0);
       String data[]=new String [4];
       while(i<min_eval.length){
          data[0]=String.valueOf(i+1);
          data[1]=String.valueOf(min_pos[i]);
          data[2]=String.valueOf(min_eval[i]);
          data[3]=min_dock[i];
          dtm.addRow(data);
          i++;
       }
       min();
       System.out.println("ALGORITMA GENETIKA SELESAI");
    }
}
 public void  min() throws BadLocationException{
    double min=0;
    int pos=0;
    int gen=0;
    String sekuen="";
    try{
       String A="E:\\Running\\Hasil.txt";
       PrintStream myconsole=new PrintStream(new File(A));
       System.setOut(myconsole);
       myconsole.println("Tahap akhir Cek Minimum");
       for(int i=0;i<min_pos.length;i++){
          if(min_eval[i]<min){
             min=min_eval[i];
             pos=min_pos[i];
             sekuen=min_dock[i];
             gen=i;
          }
       }
       h_me.setText(String.valueOf(min));
```

```
            hp=pos;
            hs=sekuen;
            h_pd.setText(String.valueOf(pos));
            myconsole.println("Hasil akhir");
            myconsole.println();
            myconsole.println("Energi terendah = "+min);
            myconsole.println("Posisi docking = "+pos);
            myconsole.println("Hasil sekuen = "+hasil_docking());
            myconsole.println("Generasi = "+gen);
            JOptionPane.showMessageDialog(null, "Generate Selesai");
        }catch(FileNotFoundException aa){
            System.out.println(aa);
        }
    }
    public String hasil_docking() throws BadLocationException{
        String str=hs;
        int pos=str.indexOf(lgn);
        h_sd.setText(str);
        h_sd.getHighlighter().addHighlight(0, pos, new
DefaultHighlighter.DefaultHighlightPainter(Color.cyan));
        h_sd.getHighlighter().addHighlight(pos, pos+lgn.length(), new
DefaultHighlighter.DefaultHighlightPainter(Color.red));
        h_sd.getHighlighter().addHighlight(pos+lgn.length(),str.length() , new
DefaultHighlighter.DefaultHighlightPainter(Color.cyan));
        return str;
    }
    public void decode() throws SQLException, BadLocationException{
        String temp[]=new String[populasi];
        String str="";
        try{
            String A="E:\\Running\\decode\\decode "+n+".txt";
            PrintStream myconsole=new PrintStream(new File(A));
            System.setOut(myconsole);
            myconsole.println("Tahap Decode ke "+n);
        for (int j = 0; j <populasi; j++){
            for(int i=0;i<bit;i++){
                str=str+krom[j][i];
            }//System.out.println(str);
            temp[j]=str;
            str="";
            }
        for (int j = 0; j < x.length; j++) {
            x[j]= Integer.parseInt(temp[j],2);
            if(x[j]>=jAntarTitik){
                x[j]=jAntarTitik-1;
            }
        }
        for (int j = 0; j < x.length; j++) {
            x[j]=bs[x[j]];
```

```
        myconsole.println(x[j]);
        s_pos[in_s_pos]=x[j];
          in_s_pos++;
      }
      myconsole.println("Tahap decode "+n+" selesai");
      calcFitness();
      }catch(FileNotFoundException bb){
        System.out.println(bb);
      }
    }
    public void calcFitness() throws SQLException, BadLocationException{
      String str="";
      /*for(int j=0;j<prot.length;j++)
      System.out.print(prot[j]);
      System.out.println("");*/
      String coba="E:\\Running\\docking\\docking "+n+".txt";
      String coba1="E:\\Running\\calcfit\\calcFitnes "+n+".txt";
      try{
        PrintStream myconsole=new PrintStream(new File(coba));
        PrintStream myconsole1=new PrintStream(new File(coba1));
        System.setOut(myconsole);
        System.setOut(myconsole1);
        myconsole.println("Tahap docking ke "+n);
        for(int i=0;i<x.length;i++){
          if(x[i]==0){
            str=str+lgn;
          }
          for(int j=0;j<prot.length;j++){
            if(j==x[i]-1){
              str=str+prot[j]+lgn;
            }
            else{
              str=str+prot[j];
            }
          }
          dock[i]=str;

          str="";
        }
        Random aa=new Random();
        char
Asam_amino[]={'A','R','N','D','C','Q','E','G','H','I','L','K','M','F','P','S','T','W','Y','V'};
        for(int i=0;i<dock.length;i++){
          StringBuilder a=new StringBuilder(dock[i]);
          int banyak=aa.nextInt(5);
          for(int j=0;j<banyak;j++){
            a.setCharAt(aa.nextInt(p_protein),
Asam_amino[aa.nextInt(Asam_amino.length)]);
          }
```

```
            dock[i]=a.toString();
            myconsole.println(dock[i]);
            s_dock[in_s_dock]=dock[i];
            in_s_dock++;
        }
        double total = 0;
        double min=0;
        int pos=0;
        String sekuen ="";
        boolean cek = false;
        myconsole1.println("Tahap evaluasi ke "+n);
        for (int j = 0; j < x.length; j++) {
            total=total+cal_db[x[j]];
            System.out.println(cal_db[x[j]]);
            fx[j]=cal_db[x[j]];
            s_eval[in_s_eval]=fx[j];
            in_s_eval++;
            myconsole1.println(fx[j]);
            if(cal_db[x[j]]<min){
                min=cal_db[x[j]];
                pos=x[j];
                sekuen= dock[j];
                cek=true;
            }
        }
        if(cek){
        myconsole1.println("nilai minimum = "+min);
        myconsole1.println("posisi minimum = "+pos);
        min_eval[n]=min;
        min_pos[n]=pos;
        min_dock[n]=sekuen;
        double sum=0;
        for (int j = 0; j < x.length; j++) {
            p[j]=fx[j]/total;
            sum=sum+p[j];
            q[j]=sum;
            //System.out.println(q[j]);
        }
        myconsole.println("Tahap docking "+n+" selesai");
        myconsole1.println("Tahap evaluasi "+n+" selesai");
        seleksi();
        //System.out.println(sum);
        }else
            JOptionPane.showMessageDialog(null, "ERROR");
    }catch(FileNotFoundException ax){
        System.out.println(ax);
    }

}
```

```
public void seleksi() throws SQLException, BadLocationException{
   try{
      String coba="E:\\Running\\seleksi\\seleksi "+n+".txt";
      PrintStream myconsole=new PrintStream(coba);
      System.setOut(myconsole);
      myconsole.print("Tahap Seleksi ke "+n);
      for(int i=0;i<r.length;i++)
         r[i]=Math.random();
      myconsole.println(Arrays.toString(r));
      for(int i=0;i<q.length;i++){
         for(int j=0;j<=q.length-1;j++)
            if(r[i]>=q[j]&&r[i]<=q[j+1])
               s[i]=j+1;
            else if (r[i]<=q[0])
               s[i]=0;
         myconsole.println("seleksi "+s[i]);
      }
      myconsole.println();
      myconsole.println("sebelum seleksi");
      myconsole.println(Arrays.deepToString(krom));
      for(int i=0;i<q.length;i++){
         System.arraycopy(krom[s[i]], 0, krom_seleksi[i], 0, bit);
      }
      for(int i=0;i<q.length;i++){
         System.arraycopy(krom_seleksi[i], 0, krom[i], 0, bit);
      }
      myconsole.println("setelah seleksi");
      myconsole.println(Arrays.deepToString(krom));
      myconsole.println("Tahap Seleksi "+n+" Selesai");
   }catch(FileNotFoundException cc){
      System.out.println(cc);
   }
   crossover();
}
public void crossover() throws SQLException, BadLocationException{
   try{
      String coba="E:\\Running\\crossover\\crossover"+n+".txt";
      PrintStream myconsole=new PrintStream(new File(coba));
      System.setOut(myconsole);
      myconsole.print("Tahap crossover ke "+n);
      for(int i=0;i<r.length;i++){
         r[i]=Math.random();
         if(r[i]<pc)
            a++;
      }int cr[];
      if(a%2==0)
         cr=new int[a];
      else
         cr=new int[a-1];
```

```
        myconsole.println("banyak indiv cros "+a);
        a=0;
        int z=0;
        for(int i=0;i<r.length;i++){
          if(r[i]<pc&&a<cr.length){
            cr[z]=i;
            z++;
            myconsole.println("index "+i);
            a++;
          }
        }
        Random ack= new Random();
        int de;
        for (int j = 0; j <a; j=j+2) {
          int c=ack.nextInt(bit-1);
          for(int i=c;i<bit;i++){
          de=krom[cr[j]][i];
          krom[cr[j]][i] = krom[cr[j+1]][i];
          krom[cr[j+1]][i] = de;
        }
          myconsole.println("hasil crossover "+cr[j]+" dengan "+cr[j+1]);
          myconsole.println(Arrays.toString(krom[cr[j]]));
          myconsole.println(Arrays.toString(krom[cr[j+1]]));
        }
        myconsole.println("tahap crossover "+n+" selesai");
      }catch(FileNotFoundException dd){
        System.out.println(dd);
      }
      mutasi();
    }
    public void mutasi() throws SQLException, BadLocationException{
      try{
        String coba="E:\\Running\\mutasi\\mutasi"+n+".txt";
        PrintStream myconsole=new PrintStream(new File(coba));
        System.setOut(myconsole);
        myconsole.print("Tahap Mutasi ke "+n);
        int pbit=populasi*(bit);
        myconsole.println("panjang bit = "+pbit);
        double mut[] =new double[pbit];
        int indiv ;
        int inkr ;
        for(int i=1;i<mut.length+1;i++){
          mut[i-1]=Math.random();
          if(mut[i-1]<pm){
            if(i<bit){
              inkr=i-1;
              indiv=0;
              myconsole.println("i 1 = "+i);
              myconsole.println("index indiv yang mutasi "+indiv);
```

```
                myconsole.println("index kromosom yang mutasi "+inkr);
            }
            else if(i%(bit)!=0){
                inkr=i%(bit)-1;
                indiv=i/(bit);
                myconsole.println("i 3 = "+i);
                myconsole.println("index indiv yang mutasi "+indiv);
                myconsole.println("index kromosom yang mutasi "+inkr);
            }
            else{
                myconsole.println("i 4 = "+i);
                inkr=bit-1;
                indiv=i/(bit)-1;
                myconsole.println("index indiv yang mutasi "+indiv);
                myconsole.println("index kromosom yang mutasi "+inkr);
            }
            if(krom[indiv][inkr]==0)
                krom[indiv][inkr]=1;
            else
                krom[indiv][inkr]=0;
          }
        }
        myconsole.println("Tahap mutasi "+n+" selesai");
    }catch(FileNotFoundException ee){
        System.out.println(ee);
    }
    n++;
    generateKromosom();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String pr=in_p.getText();
    String l=in_l.getText();
    int pop=Integer.parseInt(in_pop.getText());
    int gen=Integer.parseInt(in_gen.getText());
    //System.out.println("Max generasi 100");
    double cross=Double.parseDouble(in_rc.getText());
    double mut=Double.parseDouble(in_rm.getText());
    clear();
    double calfi[]=new double[pr.length()];
    protein_ligan_eval ada = null;
    try {
        ada = new protein_ligan_eval();
    } catch (SQLException ex) {
        Logger.getLogger(GUi_TA1.class.getName()).log(Level.SEVERE, null, ex);
    }
    calfi=ada.Hitung1(pr, l);
    boolean c=false;
    for(int i=0;i<calfi.length;i++){
```

```
            if(calfi[i]!=0){
                c=true;
            }
        }
        bbs=Integer.parseInt(in_bbs.getText());
        bs= new int [bbs];
        Random rnd= new Random();
        String pp=in_p.getText();
        String ll=in_l.getText();
        PrimitiveIterator.OfInt                    iterator                    =
ThreadLocalRandom.current().ints(0,pp.length()).distinct().iterator();
        for(int i=0;i<bbs;i++){
            bs[i]=iterator.next();
            bs_list.append(bs[i]+" \n");
        }
        /*char a[]=pp.toCharArray();
        String str="";
        String h[]=new String [bss];
        for(int i=0;i<bs1.length;i++){
            for(int j=0;j<a.length;j++){
                if(j==0&&bs1[i]==0)
                    str=str+ll+a[j];
                else if(j==bs1[i])
                    str=str+a[j]+ll;
                else
                    str=str+a[j];
            }
            h[i]=str;
            //bs_list.append(str+" \n");
            str="";
        }
        System.out.println(Arrays.toString(h));*/
        if(c){
            cal_db=calfi;
            try {
                inisialisasi(pr, l, pop, gen, cross, mut);
            } catch (SQLException ex) {
                Logger.getLogger(GUi_TA1.class.getName()).log(Level.SEVERE,      null,
ex);
            } catch (BadLocationException ex) {
                Logger.getLogger(GUi_TA1.class.getName()).log(Level.SEVERE,      null,
ex);
            }
        }else{
            System.out.println("Data Tidak Ada");
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    System.exit(0);
}

private void exitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void open_pActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));
    int result = fileChooser.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        System.out.println("Selected file: " + selectedFile.getAbsolutePath());
        String FILENAME = selectedFile.getAbsolutePath();
        BufferedReader br = null;
        FileReader fr = null;
        try {
            fr = new FileReader(FILENAME);
            br = new BufferedReader(fr);
            String sCurrentLine;
            br = new BufferedReader(new FileReader(FILENAME));
            while ((sCurrentLine = br.readLine()) != null) {
                in_p.setText(sCurrentLine);
            }
        } catch (IOException e) {
        } finally {
            try {
                if (br != null)
                    br.close();
                if (fr != null)
                    fr.close();
            } catch (IOException ex) {
            }
        }
    }
}

private void open_lActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));
    int result = fileChooser.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        System.out.println("Selected file: " + selectedFile.getAbsolutePath());
        String FILENAME = selectedFile.getAbsolutePath();
        BufferedReader br = null;
        FileReader fr = null;
        try {
```

```
            fr = new FileReader(FILENAME);
            br = new BufferedReader(fr);
            String sCurrentLine;
            br = new BufferedReader(new FileReader(FILENAME));
            while ((sCurrentLine = br.readLine()) != null) {
               in_l.setText(sCurrentLine);
            }
         } catch (IOException e) {
         } finally {
            try {
               if (br != null)
                  br.close();
               if (fr != null)
                  fr.close();
            } catch (IOException ex) {
            }
         }
      }
   }

   private void clearActionPerformed(java.awt.event.ActionEvent evt) {
      clear();
      JOptionPane.showMessageDialog(null, "Log Sudah Terhapus");
   }

   private void rndActionPerformed(java.awt.event.ActionEvent evt) {
      Random gn=new Random();
      int p1=gn.nextInt(300);
      int q1=gn.nextInt(500);
      double c1=gn.nextDouble();
      int a=gn.nextInt(100);
      double start = 0;
      double end = c1;
      double random = new Random().nextDouble();
      double result = start + (random * (end - start));
      in_pop.setText(String.valueOf(p1));
      in_gen.setText(String.valueOf(q1));
      in_rc.setText(String.valueOf(c1));
      in_rm.setText(String.valueOf(result));
      in_bbs.setText(String.valueOf(a));

   }

   private void import_db_prot_ligActionPerformed(java.awt.event.ActionEvent evt)
{
      Import_Protein imp = new Import_Protein();
      imp.setVisible(true);
   }
```

```
private void import_nilai_evalActionPerformed(java.awt.event.ActionEvent evt) {
    Import_Eval_Ligan imp1=new Import_Eval_Ligan();
    imp1.setVisible(true);
}

private void saveActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser chooser = new JFileChooser();
    int returnVal = chooser.showSaveDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        FileOutputStream stream = null;
        PrintStream out = null;
        try {
            File file = chooser.getSelectedFile();
            stream = new FileOutputStream(file);
            String text = "========DOCKING MOLEKUL========";
            String text1="\n\nProtein sekuen = "+in_p.getText();
            String text2="\nLigan sekuen = "+in_l.getText();
            String text3="\nBanyak Populasi = "+in_pop.getText();
            String text4="\nBanyak Generasi = "+in_gen.getText();
            String text5="\nProbabiliti Crossover = "+in_rc.getText();
            String text6="\nProbabiliti Mutasi = "+in_rm.getText();
            String text7="==========HASIL MOLEKUL==========";
            String text8="\n\nMinimum nilai = "+h_me.getText();
            String text9="\n Posisi Docking = "+h_pd.getText();
            String text10="\nHasil Docking = "+h_sd.getText();
            String text11="\n\n================================";
            out = new PrintStream(stream);
            out.print(text);                //This will overwrite existing contents
            out.print(text1);
            out.print(text2);
            out.print(text3);
            out.print(text4);
            out.print(text5);
            out.print(text6);
            out.print(text7);
            out.print(text8);
            out.print(text9);
            out.print(text10);
            out.print(text11);
        } catch (Exception ex) {
            //do something
        } finally {
            try {
                if(stream!=null) stream.close();
                if(out!=null) out.close();
            } catch (Exception ex) {
                //do something
            }
        }
```

```
    }
  }

  private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    db n0;
    try {
      new db().show();
      this.dispose();
    } catch (SQLException ex) {
      Logger.getLogger(GUi_TA1.class.getName()).log(Level.SEVERE, null, ex);
    }

  }

  private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    help h=new help();
    h.setVisible(true);
  }

  private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
    new Menu_Utama().show();
    this.dispose();
  }
```

6. Menu Convert DNA to Protein dan sebaliknya

```
package ta;
import ta.Tools.ConvertToDNA;
import ta.Tools.ConvertToProtein;
public class Convert extends javax.swing.JFrame {
  public Convert() {
    initComponents();
    setLocationRelativeTo(this);
  }
  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String a=IN.getText();
    if(pilih.getSelectedItem().equals("Protein to DNA")){
      ConvertToDNA dna= new ConvertToDNA();
      OUT.setText(dna.ConvertToDNA(a));
    }else if(pilih.getSelectedItem().equals("DNA to Protein")){
      ConvertToProtein pr =new ConvertToProtein();
      OUT.setText(pr.translation(pr.ConvertToProtein(a)));
    }else{

    }
  }
```

```
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Menu_Utama().show();
        this.dispose();
    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
```

7. Method Converter

```
package ta.Tools;
import java.util.Random;
public class ConvertToDNA {
    public String ConvertToDNA(String protein){
        int a=0;
        int b=0;
        char simpan[]=new char[protein.length()];
        Random ini=new Random();
        simpan=protein.toCharArray();
/*1*/       String d[][]={{"GCT","GCC","GCA","GCG"},
/*2*/                     {"GCT","GCG","CGG","AGA","AGG"},
/*3*/                     {"ATT","ACC"},
/*4*/                     {"GAT","GAC"},
/*5*/                     {"TGT","TGC"},
/*6*/                     {"CAA","CAG"},
/*7*/                     {"GAA","GAG"},
/*8*/                     {"GGT","GGC","GGA","GGG"},
/*9*/                     {"CAT","CAC"},
/*10*/                    {"ATT","ATC","ATA"},
/*11*/                    {"TTA","TTG","CTT","CTC","CTA","CTG"},
/*12*/                    {"AAA","AAG"},
/*13*/                    {"ATG"},
/*14*/                    {"TTT","TTC"},
/*15*/                    {"CCT","CCC","CCA","CCG"},
/*16*/                    {"TCT","TCC","TCA","TCG","AGT","AGC"},
/*17*/                    {"ACT","ACC","ACA","ACG"},
/*18*/                    {"TGG"},
/*19*/                    {"TAT","TAC"},
/*20*/                    {"GTT","GTC","GTA","GTG"}
        };
        String coba="";
        for(int i=0;i<protein.length();i++){
        switch(simpan[i]){
            case 'A':a=0;b=4;break;
            case 'R':a=1;b=6;break;
            case 'N':a=2;b=2;break;
```

```
            case 'D':a=3;b=2;break;
            case 'C':a=4;b=2;break;
            case 'Q':a=5;b=2;break;
            case 'E':a=6;b=2;break;
            case 'G':a=7;b=4;break;
            case 'H':a=8;b=2;break;
            case 'I':a=9;b=3;break;
            case 'L':a=10;b=6;break;
            case 'K':a=11;b=2;break;
            case 'M':a=12;b=1;break;
            case 'F':a=13;b=2;break;
            case 'P':a=14;b=4;break;
            case 'S':a=15;b=6;break;
            case 'T':a=16;b=4;break;
            case 'W':a=17;b=1;break;
            case 'Y':a=18;b=2;break;
            case 'V':a=19;b=4;break;
            default :System.out.println("input salah");System.exit(0);break;
        }
        coba=coba+d[a][ini.nextInt(b)];
        }
        return coba ;
    }
}

public class ConvertToProtein {
    public String ConvertToProtein(String Protein){
        String coba=Protein.toUpperCase().replaceAll("T", "U");
        System.out.println(coba);
        boolean c=false;
        for(int i=0;i<coba.length();i++){

if(coba.charAt(i)=='A'||coba.charAt(i)=='C'||coba.charAt(i)=='G'||coba.charAt(i)=='U'
){
                c=true;
            }else{
                c=false;
                break;
            }
        }
            return coba;
    }
    public String translation(String temp) {
    int i = 0;
    String result = "";
    while (i <= temp.length() - 3) {
    String triplet = temp.substring(i,i+=3);
    System.out.println(triplet);
    if (triplet.equals("UUU") || triplet.equals("UUC"))
```

```
    result += 'F';
if (triplet.equals("UUA") || triplet.equals("UUG")
        || triplet.equals("CUU") || triplet.equals("CUC")
        || triplet.equals("CUA") || triplet.equals("CUA")
        || triplet.equals("CUG"))
    result += 'L';
if (triplet.equals("AUU") || triplet.equals("AUC")
        || triplet.equals("AUA"))
    result += 'I';
if (triplet.equals("AUG"))
    result += 'M';
if (triplet.equals("GUU") || triplet.equals("GUC")
        || triplet.equals("GUA") || triplet.equals("GUG"))
    result += 'V';
if (triplet.equals("UCU") || triplet.equals("UCC")
        || triplet.equals("UCA") || triplet.equals("UCG"))
    result += 'S';
if (triplet.equals("AGA") || triplet.equals("AGG"))
    result += 'R';
if (triplet.equals("AGU") || triplet.equals("AGC"))
    result += 'S';
if (triplet.equals("UGG"))
    result += 'W';
if (triplet.equals("UGU") || triplet.equals("UGC"))
    result += 'C';
if (triplet.equals("GAA") || triplet.equals("GAG"))
    result += 'E';
if (triplet.equals("GAU") || triplet.equals("GAC"))
    result += 'D';
if (triplet.equals("AAA") || triplet.equals("AAG"))
    result += 'K';
if (triplet.equals("AAU") || triplet.equals("AAC"))
    result += 'N';
if (triplet.equals("CAA") || triplet.equals("CAG"))
    result += 'Q';
if (triplet.equals("CAU") || triplet.equals("CAC"))
    result += 'H';
if (triplet.equals("UAU") || triplet.equals("UAC"))
    result += 'Y';
if (triplet.equals("CCG") || triplet.equals("CCA")
        || triplet.equals("CCC") || triplet.equals("CCU"))
    result += 'P';
if (triplet.equals("ACG") || triplet.equals("ACA")
        || triplet.equals("ACC") || triplet.equals("ACU"))
    result += 'T';
if (triplet.equals("GCG") || triplet.equals("GCA")
        || triplet.equals("GCC") || triplet.equals("GCU"))
    result += 'A';
if (triplet.equals("CGG") || triplet.equals("CGA")
```

```
        || triplet.equals("CGC") || triplet.equals("CGU"))
      result += 'R';
   if (triplet.equals("GGG") || triplet.equals("GGA")
        || triplet.equals("GGC") || triplet.equals("GGU"))
      result += 'G';
   if (triplet.equals("UAG") || triplet.equals("UAA")
        || triplet.equals("UGA"))
      result += '.';
   }
   return result;
   }
}
```

8. Menu Untuk menampilkan *Database*

```
package ta;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import ta.Tools.Database;
public class db extends javax.swing.JFrame {
   public db() throws SQLException {
      initComponents();
      setLocationRelativeTo(this);
   }
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      System.out.println(jComboBox1.getSelectedItem());
      if(jComboBox1.getSelectedItem().equals("EVAL")){
         ResultSet rst;
         Database koneksi=new Database();
         try {
            koneksi.connectFirst();
            String sql="SELECT * FROM eval;";
            rst=koneksi.executeSelect(sql);
            DefaultTableModel dtm=(DefaultTableModel) jTable1.getModel();
            dtm.setRowCount(0);
            String []data=new String[7];
            int i=1;
            while(rst.next()){
               data[0]=rst.getString("NO");
               data[1]=rst.getString("pdb1");
               data[2]=rst.getString("pdb2");
               data[3]=String.valueOf(rst.getInt("posisi"));
               data[4]=rst.getString("diherdal");
```

```
                data[5]=rst.getString("vanderwaals");
                data[6]=rst.getString("electrostatic");
                dtm.addRow(data);
            }
        } catch (SQLException ex) {
                Logger.getLogger(db.class.getName()).log(Level.SEVERE, null, ex);
        }String[][]a= {{"a"},{"B"}};
        System.out.println(Arrays.deepToString(a));
    }else if(jComboBox1.getSelectedItem().equals("PDB")){
        ResultSet rst;
        Database koneksi=new Database();
        try {
            koneksi.connectFirst();
            String sql="SELECT * FROM protein;";
            rst=koneksi.executeSelect(sql);
            DefaultTableModel dtm=(DefaultTableModel) jTable1.getModel();
            dtm.setRowCount(0);
            String []data=new String[5];
            int i=1;
            while(rst.next()){
                data[0]=rst.getString("jenis");
                data[1]=rst.getString("pdb");
                data[2]=rst.getString("nama");
                data[3]=String.valueOf(rst.getInt("panjang"));
                data[4]=rst.getString("sekuen");
                dtm.addRow(data);
            }
        } catch (SQLException ex) {
                Logger.getLogger(db.class.getName()).log(Level.SEVERE, null, ex);
        }
    }else{
        JOptionPane.showMessageDialog(null, "ANDA BELUM MEMILIH");
    }
  }

  private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    new Menu_Utama().show();
    this.dispose();
  }

  private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
System.exit(0);        // TODO add your handling code here:
  }
```

BIODATA PENULIS

Penulis bernama lengkap **Hartanto Setiawan**, lahir di Madiun, 11 Maret 1995. Anak ketiga dari pasangan Harto Wiryono dan Sri Pudji Astuti. Penulis memiliki kakak perempuan Rosalia Novianti, kakak laki-laki Fanny Ade Hanafi, dan saudara kembar Hartanto Setiabudi. Penulis mengikuti pendidikan dasar dari Sekolah Dasar hingga Sekolah Menegah Atas di Kota Ngawi. Penulis menempuh pendidikan di SD Negeri Margomulyo 3, SMP Negeri 2 Ngawi, dan SMA Negeri 2 Ngawi. Setelah Lulus dari SMAN 2 Ngawi pada tahun 2013 yang lalu, penulis melanjutkan pendidikan tingginya di Institut Teknologi Sepeluh Nopember (ITS) Surabaya dengan mengambil Departemen Matematika dengan bidang minat Ilmu Komputer. Selama mengikuti perkuliahan di ITS, penulis turut aktif dalam beberapa kegiatan kemahasiswaan sebagai staff Departemen ristek UKM Robotika, Asisten Dosen pada mata kuliah rumpun ilkom dan Asisten Dosen PAPSI ITS. Informasi lebih lanjut mengenai Tugas Akhir ini dapat ditujukan ke penulis melalui email: zetiawan.h@gmail.com.