Tesis – KM185401

BAT ALGORITHM UNTUK *MOLECULAR DOCKING* SENYAWA ALKALOID SA2014 TERHADAP PROTEIN CYCLIN D1 PADA KANKER

FEDRIC FERNANDO
06111750010013

Dosen Pembimbing
Prof. Dr. Mohammad Isa Irawan, M.T
Dr. Arif Fadlan, S.Si, M.Si

Departemen Matematika
Fakultas Matematika Komputasi Dan Sains Data
Institut Teknologi Sepuluh Nopember
2019

Thesis – KM185401

BAT ALGORITHM FOR SOLVING MOLECULAR DOCKING OF ALKALOID COMPOUND SA2014 TOWARDS CYCLIN D1 PROTEIN IN CANCER

FEDRIC FERNANDO
06111750010013

Supervisor
Prof. Dr. Mohammad Isa Irawan, M.T
Dr. Arif Fadlan, S.Si, M.Si

Mathematics Department
Faculty of Mathematics Computation And Data Science
Institut Teknologi Sepuluh Nopember
2019

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember
Oleh:

FEDRIC FERNANDO
NRP: 06111750010013

Tanggal Ujian: 10 Juli 2019
Periode Wisuda: September 2019

Disetujui oleh:
Pembimbing:

1. Prof. Dr. Mohammad Isa Irawan, MT .....
   NIP: 196312251989031001

2. Dr.Arif Fadlan, S.Si, M.Si .....
   NIP: 198108092008121001

Penguji:

1. Dr. Budi Setiyono, S.Si, M.T. .....
   NIP: 197202071997021001

2. Dr. Mahmud Yunus, M.Si. .....
   NIP: 196204071987031005

Kepala Departemen Matematika
Fakultas Matematika, Komputasi, dan Sains Data

Dr. Imam Mukhlash, S.Si, MT
NIP: 197008311994031003

BAT ALGORITHM UNTUK MOLECULAR DOCKING SENYAWA ALKALOID SA2014 TERHADAP PROTEIN CYCLIN D1 PADA KANKER

Nama : Fedric Fernando
NRP : 06111750010013
Pembimbing : Prof. Dr. Mohammad Isa Irawan, M.T
Dr. Arif Fadlan, S.Si, M.Si

ABSTRAK

Bioinformatika adalah bidang interdisipliner yang menggabungkan biologi, ilmu komputer, teknik informatika, matematika, dan statistik untuk menganalisis dan menginterpretasikan data biologi. Akhir-akhir ini, bioinformatika memiliki peran penting dalam penemuan obat baru. Salah satu langkah dalam penemuan obat baru adalah *molecular docking. Molecular docking* meniru interaksi antara ligan dan protein tujuan dalam uji *in-vitro.* Menyelesaikan permasalahan *molecular docking* bukanlah permasalahan yang mudah, dikarenakan *molecular docking* melibatkan banyak derajat kebebasan. Banyak metode yang telah digunakan dalam menyelesaikannya, salah satunya adalah menggunakan *artificial intelligence*. Penelitian ini menggunakan *bat algorithm* dalam menyelesaikan permasalahan *molecular docking*. *Bat algorithm* adalah suatu algoritma yang meniru karakteristik kelelawar dalam mencari mangsa. Penelitian ini menggunakan permasalahan *molecular docking* atas senyawa alkaloid SA2014 terhadap protein cyclin D1 dalam kanker. Senyawa alkaloid SA2014diisolasikan dari spons laut *Cinachyrella anomala*. Fungsi objektif dari permasalahan ini adalah meminimalkan nilai energi bebas, semakin kecil nilai energi bebas, maka ikatan antar protein dan ligan semakin kuat. Peneliti menggunakan *root mean squared deviation* (RMSD) dari struktur yang telah ter-docking dibandingkan dengan struktur nyata, untuk memastikan apakah *bat algorithm* dapat digunakan sebagai suatu metode pengerjaan. Struktur yang dipakai dalam validasi adalah struktur 3ptb dan 2cpp. Hasil validasi menunjukkan bahwa *bat algorithm* merupakan suatu metode yang valid untuk menyelesaikan permasalahan *molecular docking* dikarenakan nilai dari RMSD tidak melebihi 2Å dan nilai energi bebas yang didapatkan bernilai negatif. Untuk *molecular docking* SA2014 terhadap cyclin D1, *bat algorithm* memunculkan nilai energi bebas sebesar -2.3009868.

Kata kunci: *molecular docking,* algoritma kelelawar, SA2014*, Cinachyrella anomala,* cyclin D1

BAT ALGORITHM FOR SOLVING MOLECULAR DOCKING OF ALKALOID COMPOUND SA2014 TOWARDS CYCLIN D1 PROTEIN IN CANCER

Name : Fedric Fernando

NRP : 06111750010013

Supervisor : Prof. Dr. Mohammad Isa Irawan, M.T

Dr. Arif Fadlan, S.Si, M.Si

ABSTRACT

Bioinformatics is an interdisciplinary field that combines biology, computer science, information engineering, mathematics, and statistic to analyze and interpret biological data. Recently, bioinformatics had an important role in drug discovery. One of the steps for drug discovery is molecular docking. Molecular docking mimics the interaction between ligand and the target protein for in-vitro testing. Solving molecular docking problem isn't an easy task, because molecular docking involves many degrees of freedom. A lot of methods had been applied for this problem, one of them is artificial intelligence. This research studied the usage of bat algorithm in solving the molecular docking problem. Bat algorithm is an algorithm that uses the characteristic of the bats in searching prey. The problem that this research used is to solve the molecular docking of alkaloid compound SA2014 towards cyclin D1 protein in cancer. Alkaloid Compound SA2014 is isolated from marine sponge Cinachyrella anomala. The objective function for this problem is to minimize the binding energy, the lower energy means the bound of protein and ligand will be stronger. We use root mean squared deviation (RMSD) of protein structures to check the validation of bat algorithm. For validation, we used structure 3ptb and 2cpp. The validation shows that bat algorithm is a valid method to solve the molecular docking problem because of the RMSD is not over 2Å and the free binding energy is negative. For docking SA2014 towards cyclin D1, bat algorithm shows a negative value -2.3009868.

Keywords: *molecular docking; bat algorithm; SA2014, Cinachyrella anomala; cyclin D1*

KATA PENGANTAR

Puji syukur atas kebaikan Tuhan YME, dengan rahmat dan kasih karunia-Nya, penulis dapat menyelesaikan tesis yang berjudul

***Bat algorithm* untuk molecular docking senyawa alkaloid SA2014 terhadap protein cyclin D1 pada kanker**

tesis ini dibuat untuk memenuhi salah satu syarat dalam memperoleh gelar Magister Program Magister Matematika, Fakultas Matematika, Komputasi, dan Sains Data, Institut Teknologi Sepuluh Nopember.

Penyusunan tesis ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis menyampaikan terima kasih kepada pihak-pihak tersebut diantaranya:

1. Rektor Institut Teknologi Sepuluh Nopember.
2. Dekan Fakultas Matematika, Komputasi dan Sains Data, Institut Teknologi Sepuluh Nopember
3. Kepala Departmen Matematika, FMKSD, Institut Teknologi Sepuluh Nopmeber
4. Kepala Prodi Studi Magister, Departmen Matematika, FMKSD, Institut Teknologi Sepuluh Nopember
5. Prof. Dr. Mohammad Isa Irawan, MT dan Dr. Arif Fadlan, S.Si, M.Si, yang telah membimbing penulis dalam penyelesaian tesis ini
6. Dr. Budi Setiyono, S.Si, M.T. dan Dr. Mahmud Yunus, M.Si., yang telah memberi banyak masukan kepada penulis
7. Dosen-dosen dan staff-staff Departmen Matematika, FMKSD, Institut Teknologi Sepuluh Nopember, yang telah memberikan banyak ilmu dan telah membantu penulis dalam masa studinya
8. Mahasiswa Magister Departmen Matematika, FMKSD, Insitut Teknologi Sepuluh Nopember, terkhusus mahasiswa magister angkatan ganjil 2017.
9. Dan pihak lain yang telah membantu penulis dalam pengerjaan tesis

Penulis menyadari bahwa dalam tesis ini masih terdapat kelemahan dan kekurangan, oleh karena itu penulis sangat terbuka menerima saran dan ide demi

kesempurnaan penulisan selanjutnya. Penulis berharap semoga tesis ini dapat bermanfaat bagi pembaca.

Surabaya, Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Komputasi biologi atau lebih sering dikenal sebagai bioinformatika, adalah kombinasi biologi dan komputasi yang menggunakan aplikasi dari alat komputasi dan analisis untuk menangkap dan menginterpretasikan data-data biologi. Metode komputasi dalam biologi memiliki peran yang penting dalam penemuan obat di beberapa tahun terkahir ini (Jorgensen, 2004). Salah satu metode komputasi biologi untuk penemuan obat adalah *molecular docking. Molecular docking* adalah salah satu metode dari *structure-based drug discovery* yang paling sering digunakan dikarenakan kemampuannya untuk memprediksi konformasi antara ligan dan target dengan tingkat akurasi yang besar (Ferreira, Dos Santos, Oliva, & Andricopulo, 2015).

Interaksi antara protein dan ligan, memiliki banyak fungsi, interaksi yang paling umum adalah hemogoblin dengan oksigen dan karbon dioksida. Selain itu, obat, yang biasanya merupakan senyawa kecil, dapat berinteraksi, mengikat, dan mengontrol fungsi biologikal dari suatu protein (Tymoczko, John L. Berg, Jeremy Mark Stryer, 2002). Oleh karena itu, *molecular docking* diperlukan untuk menentukan apakah suatu ligan dapat bereaksi terhadap suatu protein.

Molecular docking meniru peristiwa interaksi suatu molekul ligan dengan protein yang menjadi targetnya pada uji *in-vitro* (Gane & Dean, 2000). *Molecular docking* merupakan suatu permasalahan yang sulit dengan melibatkan banyak derajat kebebasan, sehingga perkembangan dari metode dan algoritma *docking* yang efisien akan menjadi sangat berguna dalam desain obat baru (de Magalhães, Barbosa, & Dardenne, 2004). Metode-metode *artificial intelegence* telah banyak diterapkan untuk permasalahan *molecular docking*, dengan contoh, metode-metode *machine learing,* yaitu *extreme learning machine* (Mahdiyah, Imah, & Irawan, 2016) dan *random forest* (Ballester & Mitchell, 2010). Selain *machine learning,* telah juga digunakan metode *evolutionary algorithm* (EA), yaitu *particle swarm optimization* (Liu, Li, & Ma, 2012), *differential evolution,* dan juga *genetic*

algorithm (López-Camacho, García Godoy, García-Nieto, Nebro, & Aldana-Montes, 2015).

Dalam penelitian kali ini, dikaji penggunaan algoritma baru yaitu *bat algorithm* (BA) untuk menyelesaikan permasalahan *molecular docking*. BA merupakan EA dengan menggunakan karakteristik kelelawar dalam mencari mangsa. BA telah banyak digunakan dalam permasalahan optimasi, sebagai contoh: menyelesaikan kasus optimasi numerik (Tsai, Pan, Liao, Tsai, & Istanda, 2012), *tuning* dari *power system stabilizer* (Sambariya & Prasad, 2014), dsb. Kasus yang diambil dalam penelitian ini adalah *molecular docking* senyawa alkaloid SA2014 dari spons laut *Cinachyrella anomala* terhadap protein cyclin D1 pada kanker. Cyclin D1 adalah suatu protein yang berperan penting dalam pembelahan sel. Cyclin D1 berperan dalam memicu terjadinya pembelahan sel. Dalam sel kanker, cyclin D1memiliki jumlah yang banyak, sehingga pembelahan sel tidak terkendali. Tujuan pemberian senyawa alkaloid SA2014 kepada cyclin D1 adalah untuk menyetabilkan protein cyclin D1, dikarenakan senyawa SA2014 memiliki sifat antikanker. Dengan *molecular docking* , kita akan tahu bagaimana keadaan pengikatan SA2014 dengan cyclin D1.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merumuskan *bat algorithm* untuk malasah *molecular docking*?
2. Bagaimana hasil simulasi yang didapatkan dengan *bat algorithm*?
3. Bagaimana cara menampilkan hasil simulasi *molecular docking* dengan menggunakan *bat algorithm*?

1.3 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian ini adalah sebagai berikut:

1. *Molecular docking* yang digunakan adalah *flexible docking*.
2. Jenis *molecular docking* yang dikerjakan adalah *protein-ligand docking*.
3. Penelitian yang dilakukan hanya sampai simulasi *docking*.

4. Ukuran *Grid Box* adalah $40 \times 40 \times 40$ dengan panjang tiap titik sebesar 0.375Å
5. Parameter yang digunakan dalam pengerjaan permasalahan *molecuar docking* dengan *bat algorithm* ini ada pada Tabel 5.1.

1.4 Tujuan

Tujuan dari pengerjaan penelitian ini adalah sebagai berikut:

1. Menyelesaikan permasalahan *molecular docking* dengan menggunakan *bat algorithm*.
2. Mengetahui hasil yang diperoleh dalam permasalahan *molecular docking* dengan menggunakan *bat algorithm*.
3. Menampilkan hasil simulasi *molecular docking* dengan menggunakan *bat algorithm*.

1.5 Manfaat

Manfaat yang bisa diperoleh dari pengerjaan penelitian ini adalah:

Adanya kajian baru dalam penggunaan *Bat Algorithm* untuk permasalahan *molecular docking*. Metode ini masih belum digunakan sebelumnya. Dengan menggunakan BA, akan didapatkan performasi BA terhadap *molecular docking,* dibandingkan dengan metode yang telah ada. Penelitian ini menggunakan senyawa cyclin D1 dari penderita kanker dan senyawa alkaloid SA2014 sebagai ligan. Hasil *molecular docking* ini dapat membantu penemuan obat baru untuk kanker

BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

1. *A new metaheuristic Bat-inspired Algorithm* (X. S. Yang, 2010)

   Sebuah metode optimasi baru yang berdasarkan sikap kelelawar dalam menemukan mangsa digagas dalam penelitian. Dari percobaan, disimpulkan bahwa algortima kelelawar ini berhasil untuk menyelesaikan permasalahan optimasi.

2. *Solving Molecular Flexible Docking Problems with Metaheuristics: A Comparative Study* (López-Camacho et al., 2015).

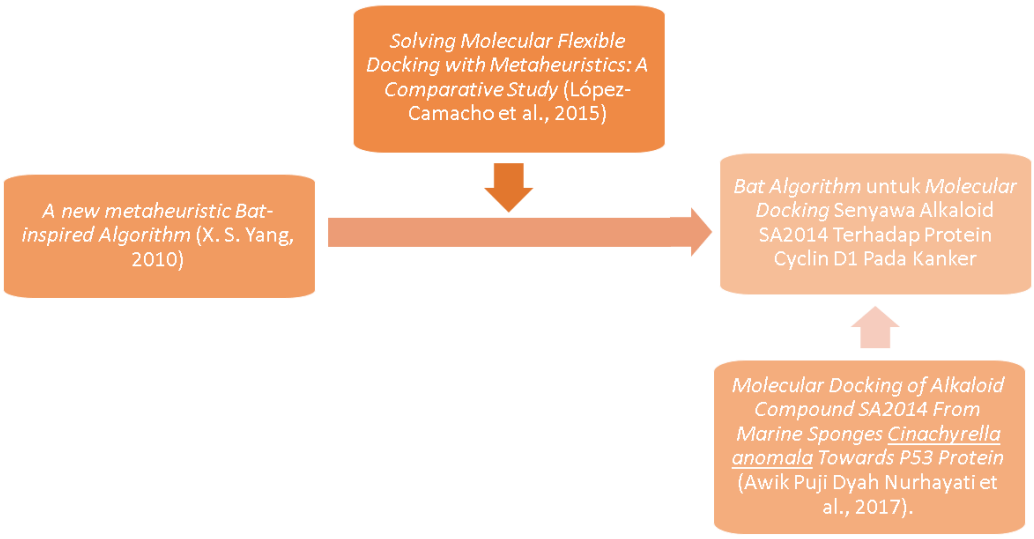
   Penelitian ini melakukan percobaan dengan 3 metode *Metaheuristic*, yaitu: *Genetic Algorithm, Particle Swarm Optimization*, dan *Differential Equation*. Fungsi objektif yang digunakan adalah penilaian energi yang direkomendasikan oleh AutoDock 4.2. Hasil yang didapatkan menunjukkan *Differential Evolution* adalah metode terbaik dari 3 metode *Metaheuristic* yang digunakan.

3. *Molecular Docking of Alkaloid Compound SA2014 From Marine Sponges Cinachyrella anomala Towards P53 Protein* (Awik Puji Dyah Nurhayati, Santoso, Setiawan, & Lianingsih, 2017).

   Penelitian ini menkaji tentang doking senyawa SA2014 dan *Doxurubicin* terhadap protein P53 pada kaker payudara. Dengan berbagai percobaan *docking,* didapatkan bahwa *docking* dengan SA2014 lebih stabil dibanding *Doxurubicin*.

Alur penelitan dapat dilihat pada Gambar 2.1. Dimulai dari *Solving Molecular Flexible Docking Problems with Metaheuristics: A Comparative Study* (López-Camacho et al., 2015), kemudian dengan mengggunakan algoritma kelelawar yang telah dijabarkan pada *A new metaheuristic Bat-inspired Algorithm* (X. S. Yang, 2010), didapatkan *molecular docking* dengan menggunakan BA. Selanjutnya dengan peneltian *Molecular Docking of Alkaloid Compound SA2014 From Marine Sponges Cinachyrella anomala Towards P53 Protein* (Awik Puji

Dyah Nurhayati et al., 2017). Senyawa SA2014 akan digunakan sebagai ligan dalam penelitian kali ini.



Gambar 2.1 : *Roadmap* Penelitian

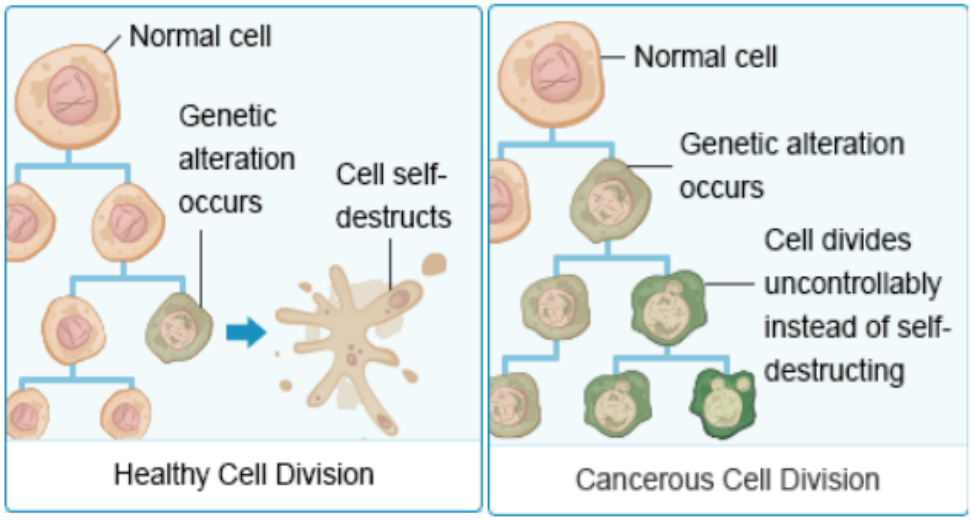
2.2. Kanker

Kanker adalah segolongan penyakit yang ditandai dengan pembelahan sel tidak terkendali dan kemampuan sel menyerang jaringan biologis lainnya, baik pertumbuhan langsung ke tetangganya (invasif) maupun migrasi ke tempat yang lebih jauh (metastatis) (Diandana, 2009). Kanker terjadi dikarenakan hilangnya kontrol dalam siklus sel. Sel kanker tidak memiliki sistem kontrol yang mencegah sel tumbuh berlebih dan penyusupan sel ke jaringan lain. Berikut adalah gen-gen yang mengalami perubahan dalam sel kanker (Subianto, 2018):

a. Proto-onkogen, gen yang menghasilkan protein-protein yang meningkatkan pembelahan sel atau menghambat kematian sel. Jika mengalami mutasi, disebut dengan onkogen.
b. Gen penekan tumor, gen yang menghasilkan protein-protein yang mencegah pembelahan sel atau menyebabkan kematian sel.
c. Gen perbaikan DNA, gen pencegah mutasi yang dapat mengakibatkan kanker.

Perbandingan siklus sel normal dan sel kanker dapat dilihat pada Gambar 2.2. Dalam siklus sel normal, jika terjadi perubahan genetic, maka gen akan

memerintah sel untuk menghancurkan diri sendiri, namun pada siklus sel kanker, gen yang mengatur mengalami mutasi, sehingga tidak dapat dihentikan.



Gambar 2.2 : Pembelahan sel (Subianto, 2018)

2.3. Siklus Sel

Siklus sel merupakan untaian kejadian dimulai dengan sel menduplikasi gen, selanjutnya mensintesis komponen sel, dan melakukan pembelahan menjadi 2 buah sel. Ketiga proses diatas, berada dibawah kontrol genetik. Berikut adalah kontrol genetic dari proses siklus sel (Sarmoko & Larasati, 2012):

a. Cyclin (CYC)

   Jenis cyclin utama dalam siklus sel adalah cyclin D, E, A, dan B. Cyclin diekspresikan secara periodik sehingga konsentrasi cyclin berubah-ubah pada setiap fase siklus sel. Berbeda dengan cyclin yang lain, cyclin D tidak diekspresikan secara periodik akan tetapi selalu disintesis selama ada stimulasi growth factor.

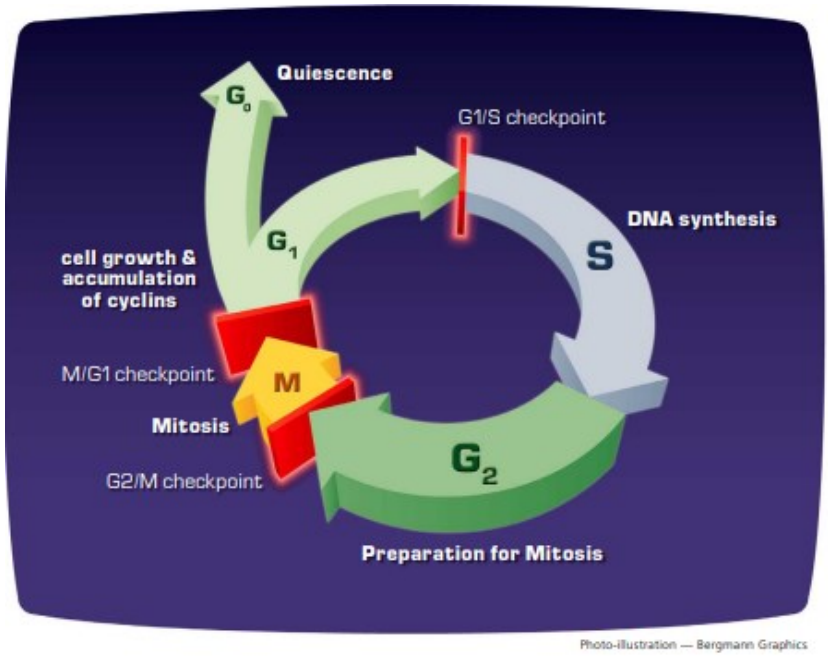
b. Cyclin-dependent kinases (CDK)

   CDK utama dalam siklus sel adalah CDK 4, 6, 2, dan 1. CDK merupakan treonin atau serin protein kinase yang harus berikatan dengan cyclin untuk aktivasinya. Ketika diaktifkan, CDK akan memacu proses *downstream* dengan cara memfosforilasi protein spesifik.

c. Cyclin–dependent kinase inhibitor (CKI)

CKI merupakan protein yang dapat menghambat aktivitas Cdk dengan cara mengikat Cdk atau kompleks cyclinCdk. Cyclin–dependent kinase inhibitor terdiri dari dua kelompok protein yaitu INK4 (p15, p16, p18, dan p19) dan CIP/KIP (p21, p27, p57). (Vermeulen, Berneman, & Bockstaele, 2003).

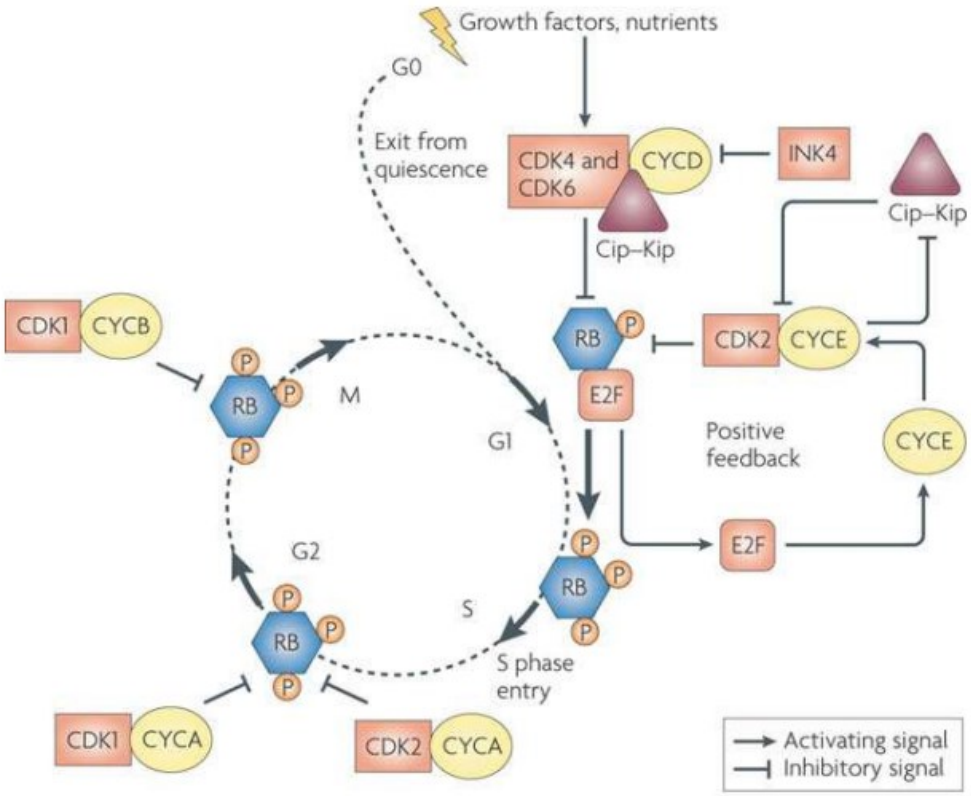
Fase siklus sel dapat dibagi menjadi 5 fase, yaitu fase G1 (Gap 1), fase S (*Synthesis*), fase G2 (Gap 2), fase M (Mitosis), dan fase G0 (Gap 0) (Subianto, 2018). Urutan dari siklus sel dapat dilihat pada Gambar 2.3. Siklus sel dimulai dari masuknya sel dari fase G0 (*quiescent*) ke fase G1 karena adanya stimulus oleh *growth factor*.



Gambar 2.3: Siklus Sel (*Cell Biology and Cancer*, 2001)

Gambar 2.4 menunjukkan peran pengontrol dalam siklus sel. Pada awal fase G1, Cdk 4 dan/atau 6 diaktifkan oleh cyclin D (CYCD). Kompleks CDK4/6 dengan CYCD akan menginisiasi fosforilasi dari keluarga protein retinoblastoma (P-RB) selama awal G1. Efek dari fosforilasi ini, fungsi histon deasetilasi (HDAC) yang seharusnya menjaga kekompakan struktur kromatin menjadi terganggu. Akibatnya struktur DNA menjadi longgar dan faktor transkripsi yang

semula diikat pRb menjadi lepas dan transkripsi dari E2F *responsive genes* yang dibutuhkan dalam progresi siklus sel ke fase S menjadi aktif. Gen tersebut antara lain CYCE, CYCA, CDC25, DNA polimerase, timidilat kinase, timidilat sintetase, DHFR, dll (Satyanarayana & Kaldis, 2009; Vermeulen et al., 2003)

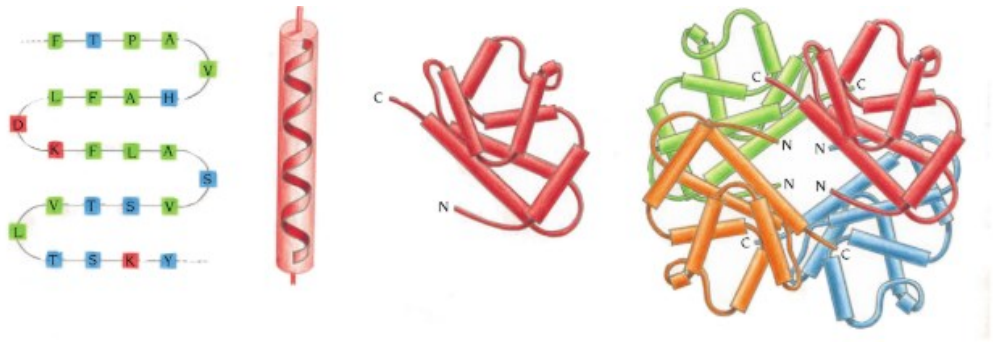


Gambar 2. 4: Ilustrasi Umum Siklus Sel (Lapenna & Giordano, 2009)

2.4. Protein

Protein terbentuk atas asam amino yang berbeda-beda, dimana ada 20 macam asam amino yang dapat ditemui di alam. Asam amino yang berbeda-beda, disusun sedemikian hingga, sehingga terbentuklah struktur utama protein (Shen & Tuszynki, 2008). Struktur protein terbagi menjadi 4, yaitu struktur utama atau primer, struktur sekunder, struktur tersier, dan struktur kuartener. Seperti yang dijelaskan sebelumnya, struktur utama adalah rantai asam amino dari protein. Struktur sekunder dari protein adalah bentuk berlekuk yang diakibatkan oleh ikatan hidrogen pada peptida. Struktur tersier dari protein adalah bentuk 3 dimensi protein, yang diakibatkan reaksi antar rantai peptida. Sedangkan struktur

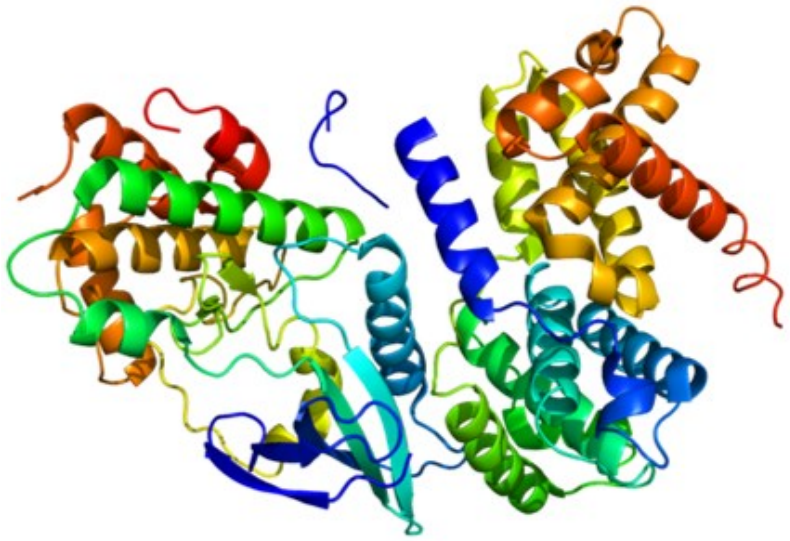
kuartener adalah bentuk 3 dimensi untuk protein yang memiliki rantai asam amino. Perbedaan bentuk struktur protein dapat dilihat pada Gambar 2.5 (Branden, 1999).



Gambar 2.5: Struktur Protein (Branden, 1999)

2.5. Protein Cyclin D1

Protein cyclin D (CYCD) merupakan salah satu regulator positif pada siklus sel. Cyclin D tidak diekspresikan secara periodik, akan tetapi selalu disintesis selama ada *growth factor.* Cyclin D terdiri dari 3 tipe yaitu cyclin D1, D2, dan D3. Siklus sel secara lengkap dapat dilihat pada Gambar 2.4. Pada siklus sel, cyclin D1 tidak hanya berperan pada fase G1 saja. Pada fase G2, tingkat cyclin D1 cenderung meningkat, sedangkan di fase S, tingkat cyclin D1 cenderung turun (K. Yang, Hitomi, & Stacey, 2006). Bentuk dari cyclin D1 dapat dilihat pada Gambar 2.6.



Gambar 2.6: Bentuk Protein Cyclin D1 (Wikipedia, 2019a)

Overekspresi cyclin D1 dapat menjadi onkogen pemicu melalui fungsi pengaturan selnya. Dalam beberapa tumor manusia, cyclin D1 diperkuat dan/atau diekspresikan secara berlebih. Penambahan jumlah cyclin D1 sering terjadi pada masa awal tumorigenesis. Memahami mekanisme dimana cyclin D1 melimpah dalam studi in-vivo, dapat memberikan wawasan penting tentang kanker manusia (Fu, Wang, Li, Sakamaki, & Pestell, 2004).

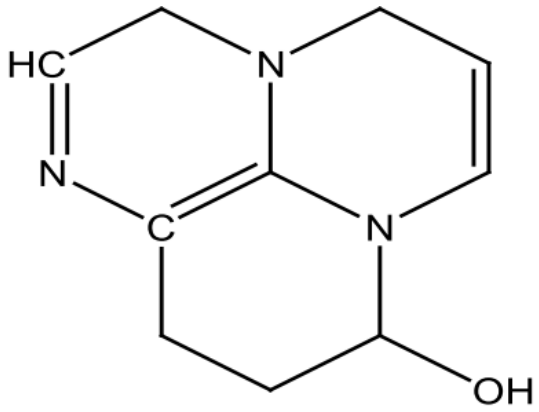
2.6. Ligan

Ligan adalah molekul sederhana yang dalam senyawa kompleks bertindak sebagai donor pasangan elektron (basa Lewis). Ligan akan memberikan pasangan elektronnya kepada atom pusat yang menyediakan orbital kosong. Interaksi antara ligan dan atom pusat menghasilkan ikatan koordinasi. Ligan adalah sebuah molekul sinyal kecil yang terlibat dalam proses anorganik dan biokimia. Dalam kimia koordinasi, ligan memungkinkan pembentukan kompleks koordinasi, atau hubungan antara molekul yang berbeda dalam larutan. Dalam biokimia umumnya mendefinisikan ligan sebagai molekul *messenger*, seperti hormon, substrat, atau aktivasi dan faktor inhibisi (Horton, 2006). Ligan juga dapat bertindak sebagai label untuk protein tertentu dalam proses modifikasi paska-translasi. Ligan dapat mengaktifkan atau menghambat protein yang berbeda berdasarkan kondisi yang mengikatnya, menargetkan protein untuk mengarahkan mereka ke berbagai daerah ke dalam sel, protein atau label untuk degradasi (Gesteland, 2004).

Bidang kimia sering mengklasifikasikan ligan berdasarkan pola ikatan ligan, ukuran, dan muatan listrik. Sifat kimia ligan menggambarkan jumlah formasi ikatan yang terjadi antara ligan dan logam lain atau molekul dalam kompleks koordinasi. Jumlah ikatan yang berbeda akan menghasilkan struktur kompleks yang berbeda secara keseluruhan dalam tiga dimensi. Secara umum, stabilitas kompleks tergantung pada ikatan yang dibentuk oleh ligan tunggal, yang meningkatkan struktur dan sifat ikatan. Ukuran dan muatan ligan sangat bervariasi, yang tidak hanya akan menentukan berapa banyak ikatan dapat terbentuk dengan atom lain, tetapi juga menentukan jenis atom yang akan dibawa ke dalam koordinasi kompleks. Ukuran massa dan besar juga akan mengubah sudut di mana ligan mengikat atom lain dalam kompleks (Berg, 2002).

2.7. Senyawa SA2014

Senyawa SA2014 merupakan derivate dari kelompok cinachyramine dan termasuk ke dalam golongan alkaloid. Alkaloid merupakan senyawa yang memiliki unsur nitrogen dan biasanya berbentuk siklik. Senyawa ini diisolasi dari spons laut *Cinachyrella anomala.* Senyawa ini memiliki formulasi $C_{10}H_{13}N_3O$ dengan nama struktur 1,4,9-triazatricyclo[7,3,1,0]trideca-3,5(13),10-trien-8-ol (A.P.D Nurhayati, Pratiwi, Wahyuono, Fadlan, & Syamsudin, 2014). Dengan visualisasi struktur SA2014 pada Gambar 2.7.

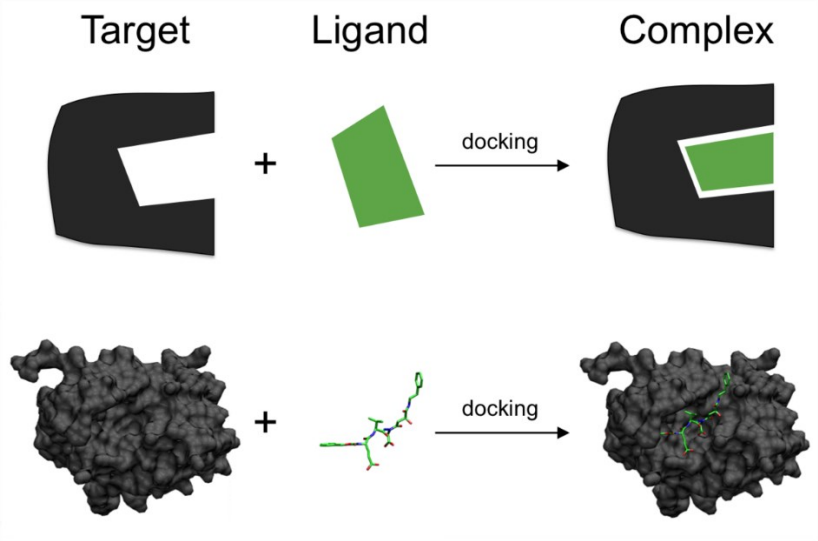


Gambar 2.7. Struktur Senyawa SA2014 (A.P.D Nurhayati et al., 2014)

Telah dilakukan pengujian senyawa SA2014 sebagai antikanker, dan hasil yang didapatkan menunjukkan bahwa SA2014 dapat berfungsi sebagai antikanker.

2.8. *Molecular Docking*

Dalam bidang pemodelan molekul, *docking* adalah metode untuk memprediksi orientasi yang lebih diutamakan dari suatu molekul ketika terikat satu sama lain untuk membentuk kompleks yang stabil. Informasi tentang orientasi ini dapat digunakan untuk memprediksi kekuatan hubungan atau afinitas ikatan antara dua molekul. *Docking* sering digunakan untuk memprediksi orientasi ikatan kandidat obat bermolekul kecil terhadap target proteinnya untuk memprediksi afinitas dan aktivitas molekul kecil. *Docking* memainkan peran penting dalam desain obat secara rasional (Mukesh & Rakesh, 2011).



Gambar 2.8: *Molecular Docking* (Wikipedia, 2019b)

Secara mudahnya, *molecular docking* dapat dijabarkan dengan Gambar 2.8. Target dalam Gambar 2.8 merupakan protein yang akan di*docking* dengan ligan. *Complex* yang dimaksud dalam Gambar 2.8 merupakan hasil *docking*, dimana hasil tersebut merupakan senyawa kompleks dari gabungan target dan ligan yang ada.

Molecular docking atau penambatan molekul mensimulasikan secara komputasi proses pengenalan molekul. Tujuan dari *molecular docking* adalah untuk mencapai konformasi yang optimal untuk protein dan ligan serta orientasi relatif antara protein dan ligan sehingga energi bebas dari sistem secara keseluruhan diminimalkan. Proses komputasi mencari ligan yang cocok baik secara geometris dan energi ke *binding site* dari protein ini disebut *molecular docking*. *Molecular docking* membantu dalam mempelajari obat/ligan atau interaksi reseptor/protein dengan mengidentifikasi *active site* yang cocok pada protein, mendapatkan geometri terbaik dari ligan-kompleks reseptor, dan menghitung energi interaksi dari ligan yang berbeda untuk merancang ligan yang lebih efektif. Keberhasilan program *docking* bergantung pada dua komponen yaitu algoritma pencarian dan *scoring function* (Mukesh & Rakesh, 2011).

Scoring function digunakan untuk menghitung afinitas kompleks ligan-protein yang terbentuk dan untuk mengurutkan peringkat senyawa. Identifikasi ini didasarkan pada beberapa teori seperti teori energi bebas Gibbs. Nilai energi bebas

Gibbs yang kecil menunjukkan bahwa konformasi yang terbentuk adalah stabil, sedangkan nilai energi bebas Gibbs yang besar menunjukkan tidak stabilnya kompleks yang terbentuk. Sedangkan penggunaan algoritma berperan dalam penentuan konformasi (*docking pose*) yang paling stabil dari pembentukan kompleks (Funkhouser, 2007). Konformasi adalah penataan ruang tertentu dari atom – atom dalam molekul. Gugus-gugus fungsional ligan akan berinteraksi dengan residu-residu asam amino protein reseptor sehingga membentuk ikatan intermolekular. Kekuatan ikatan inilah yang dihitung dan diperingkatkan (*ranking*) dengan *scoring function*.

2.8.1 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)

2.8.2 Klasifikasi Molecular Docking Berdasarkan Fleksibilitas Molekul

Berdasarkan fleksibilitas molekulnya, motede *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 sebagai obyek yang bersifat *rigid*. Definisi *rigid* adalah suatu kondisi di mana 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. 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*

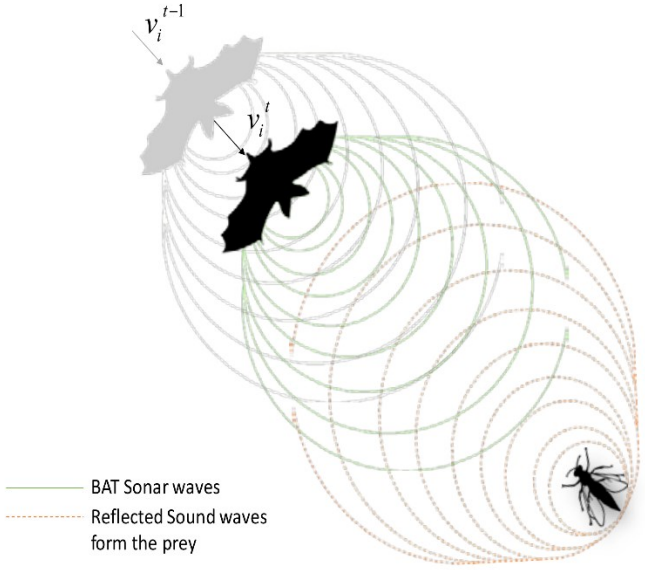
   Flexible docking adalah metode *molecular docking* yang memperlakukan dua molekul yang akan dipasangkan sebagai molekul yang fleksibel.

2.9. Bat Algorithm

Bat Algorithm atau Algoritma Kelelawar diperkenalkan pertama kali oleh Xin-She Yang pada tahun 2010. *Bat algorithm* meniru karakteristik kelelawar dalam pencarian mangsa.Aturan yang didefinisikan adalah (X. S. Yang, 2010)

1. Setiap kelelawar menggunakan ekolokasi. Ekolokasi adalah kemampuan kelelawar untuk menemukan mangsa dengan menggunakan gelombang berfrekuensi tinggi yang dihasilkan oleh kelelawar. Saat gelombang sampai kepada mangsa, gelombang akan kembali kepada si kelelawar sehingga kelelawar dapat mengetahui berapa jarak dia dengan mangsa. Gambar 2.9 menunjukkan kejadian ekolokasi.
2. Setiap kelelawar terbang secara random dengan kecepatan v_i di posisi x_i , frekuensi f_i, Panjang gelombang λ_i, kebisingan suara A_0. Mereka dapat mengatur Panjang gelombang (atau frekuensi) secara otomatis dari gelombang yang dipancarkan dengan laju emisi $r \in [0,1]$, bergantung atas kedekatannya dengan target mereka.
3. Meskipun kebisingan suara dapat berbeda-beda, diasumsikan kebisingan suara bervariasi dari suatu nilai positif besar A_0 ke suatu nilai minimum yang konstan A_{min}

Nilai v_i, x_i, f_i, λ_i, A_i, r dapat berupa scalar, array, dan juga vektor. Bergantung pada pembahasan yang akan digunakan. *Bat algorithm* akan membuat setiap kelelawar menuju nilai kelelawar dengan nilai *fitness* terbaiknya.



Gambar 2.9 Ekolokasi (Seyedmahmoudian et al., 2018)

Langkah untuk *update* dari setiap iterasi dapat ditulis sebagai berikut:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{2.1}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*)f_i \tag{2.2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.3}$$

dengan:

- $\beta \in [0,1]$, suatu nilai acak dari distribusi uniform
- x_* adalah lokasi solusi optimal pada saat itu, yang telah dibandingkan terhadap semua kelelawar pada saat itu
- $\lambda_i f_i$ menghasilkan penambahan kecepatan, f_i (atau λ_i) digunakan untuk mengatur perubahan kecepatan, dan memperbaiki faktor lain λ_i (atau f_i), bergantung atas permasalahan yang ada
- *Random walk* akan meng*update* kelelawar dengan menggunakan:

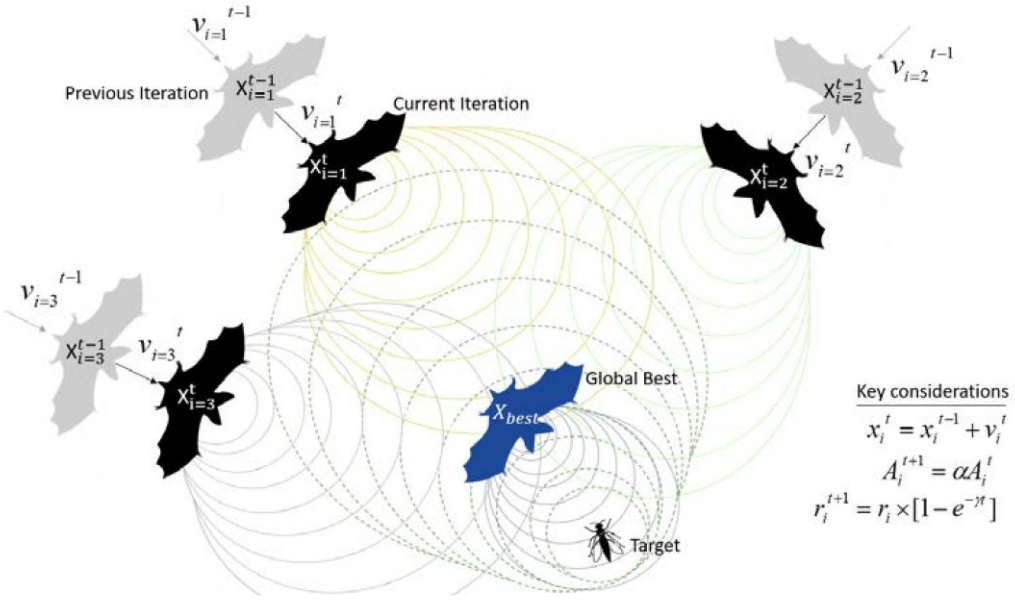
$$x_{new} = x_{old} + \epsilon A^t \tag{2.4}$$

dimana:

ϵ = angka vector acak dari $[-1,1]$

A^t = rata-rata dari kebisingan suara setiap kelelawar pada waktu t

Visualisasi atas algoritma kelelawar dapat dilihat pada Gambar 2.10. Dalam visualisasi di Gambar 2.10, kelelawar dengan warna abu-abu adalah kelelawar pada waktu ke-t. Langkah pertama adalah menentukan kelelawar terbaik dengan menghitung *fitness* dari semua kelelawar yang ada. Selanjutnya akan semua kelelawar akan di*update* dengan menggunakan persamaan 2.1 hingga 2.3.



Gambar 2.10 Visualisasi Algoritma Kelelawar (Seyedmahmoudian et al., 2018)

dimana:

x_i^t : posisi kelelawar ke-i pada waktu ke-t

v_i^t : kecepatan kelelawar ke-i pada waktu ke-t

x_{best} : kelelawar dengan fitness terbaik pada waktu ke-$t-1$

Pada umumnya, kebisingan suara kelelawar akan cenderung berkurang jika semakin dekat dengan mangsanya, dan emisi gelombangnya akan cenderung naik. Sehingga:

$$A_i^{t+1} = \alpha A_i^t, r_i^{t+1} = r_i^0(1 - \exp(-yt)),$$

dengan:

$0 < \alpha < 1$ dan $y > 0$

Berikut adalah *pseudocode*dari algoritma kelelawar:

Objective function $f(x), x = (x1, \ldots, xd)^T$

Initialize the bat population x_i $(i = 1, 2, \ldots, n)$ *and* v_i

Define pulse frequency f_i *at* x_i

Initialize pulse rates r_i *and the loudness* A_i

while *(*t *< Max number of iterations)*

Generate new solutions by adjusting frequency,

and updating velocities and locations/solutions [equations (2.1) to (2.3)]

    if *(rand >* r_i*)*

    Select a solution among the best solutions

    Generate a local solution around the selected best solution

    end if

    Generate a new solution by flying randomly

    if *(rand <* A_i *&* $f(x_i) < f(x^*)$*)*

    Accept the new solutions

    Increase r_i *and reduce* A_i

    end if

Rank the bats and find the current best x^*

end while

Postprocess results and visualization

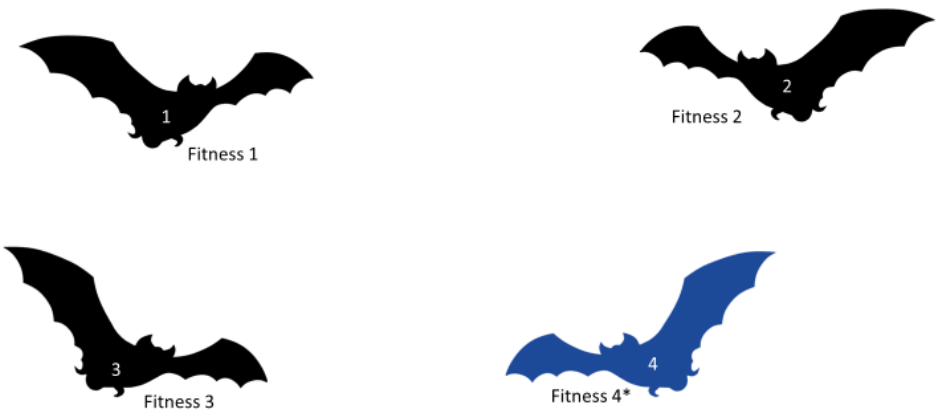
Agar lebih mudah dipahami, berikut adalah langkah-langkah jalannya algoritma kelelawar dengan disertai gambar. Dimulai dengan setelah dilakukan inisialisasi parameter, populasi awal dibentuk dengan cara *random*. Misalnya pada Gambar 2.11 adalah populasi awal yang didapatkan dengan cara *random*.

Gambar 2.11: Populasi Awal Algoritma Kelelawar

Setelah kita miliki populasi awal tersebut. Langkah awal adalah mengitung nilai *fitness* setiap kelelawar dalam populasi tersebut. Seperti pada Gambar 2.12

Gambar 2.12: Mendapatkan *Fitness* Setiap Kelelawar

Setelah *fitness* didapatkan untuk setiap kelelawar, kita pilih kelelawar dengan nilai *fitness* terbaik dan akan dijadikan sebagai kelelawar terbaik pada iterasi tersebut. Dalam Gambar 2.13, kelelawar terbaik diubah warna menjadi biru.



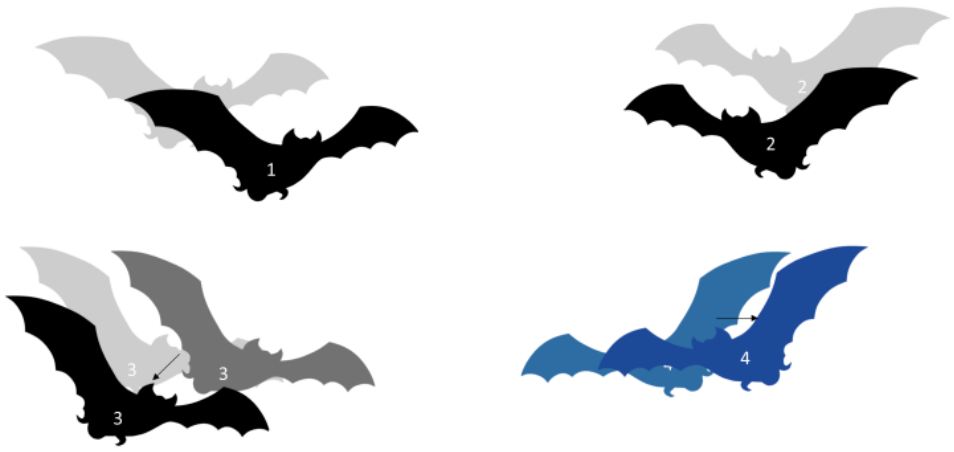
Gambar 2.13: Pemilihan Kelelawar Terbaik

Selanjutnya, kita *update* setiap kelelawar dengan persamaan 2.1 hingga 2.3dan juga dengan kelelawar terbaik yang telah kita pilih sebelumnya. Gambar 2.14 adalah perubahan posisi kelelawar dengan kelalawar 4 sebagai kelelawar terbaik. Kelelawar dengan warna abu-abu adalah kelelawar sebelum di*update.*

Gambar 2.14: Pergerakan Kelelawar

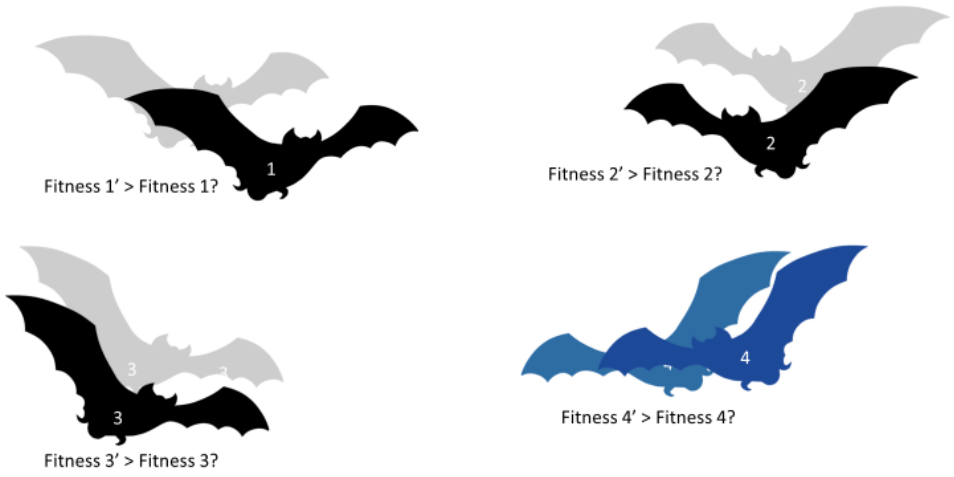
Selanjutnya, selain perhitungan dengan persamaan 2.1 hingga 2.3. Dilakukan juga *random walk* yang dapat dilihat pada persamaan 2.4 untuk beberapa kelelawar yang terpilih. Dalam Gambar 2.15, kelelawar 3 dan kelelawar

4 keduanya terpilih untuk melakukan *random walk*. Gambar kelelawar dengan warna lebih terang adalah lokasi kelelawar yang sebelumnya.



Gambar 2.15: *Random Walk* Kelelawar

Setelah update, dilakukan tahap validasi. Validasi akan memastikan apakah kelelawar akan di*update* atau akan di*revert* ke keadaan sebelumnya. Syarat suatu kekelawar akan boleh di*update* apabila kelelawar yang baru memiliki nilai *fitness* yang lebih baik dan juga kebisingan suaranya tidak lebih dari batasan kebisingan. Gambar 2.16 menunjukkan pengecekan untuk setiap kelelawar.



Gambar 2.16: Validasi Kelelawar

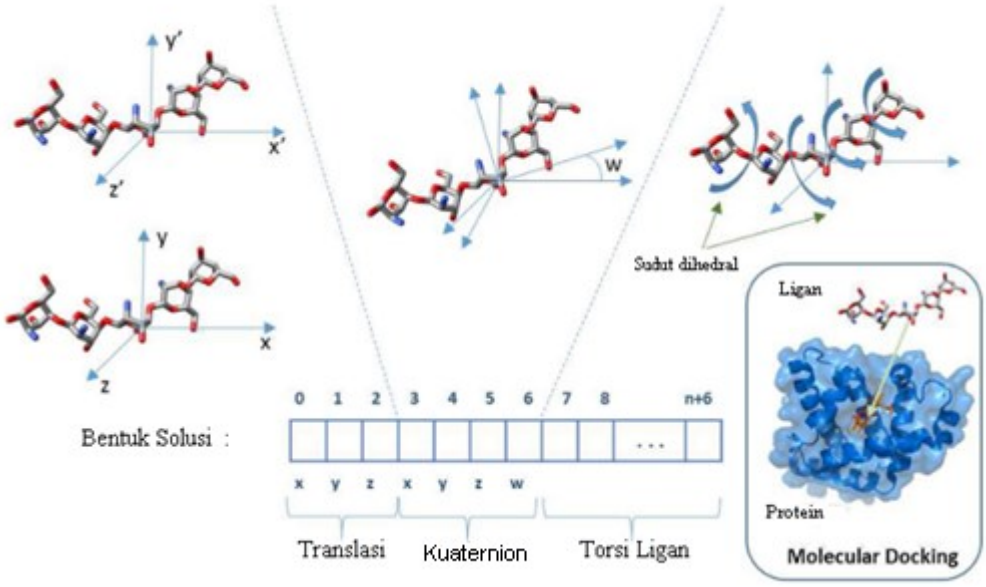
Setelah semua dilakukan validasi, maka akan disimpan sebagai generasi selanjutnya, Gambar 2.17 menunjukkan kelelawar yang akan digunakan dalam

iterasi selanjutnya. Dan apabila batas berhenti belum terpenuhi, akan dilakukan semua kegiatan dari Gambar 2.12 hingga 2.16 kembali.

Gambar 2.17: Kelelawar Untuk Iterasi Selanjutnya

2.10. Solusi BA dalam permasalahan *Molecular Docking*

Tujuan utama dalam pengerjaan *molecular docking* adalah untuk menemukan suatu konformasi antar ligan dan target. Representasi solusi atas permasalahan ini, harus dapat mewakili hasil dari *molecular docking*. Representasi solusi yang akan digunakan oleh algoritma BA dapat dilihat pada Gambar 2.18. Solusi akan direpresentasikan sebagai suatu barisan nilai, dengan 3 nilai pertama merupakan lokasi translasi ligan (variable x, y, z), 4 nilai selanjutnya adalah lokasi *quaternion* (variable w, x, y, z), dan sisanya adalah sudut rotasi dari semua titik ligan yang dapat berputar. Untuk variable x, y, z dari lokasi translasi, digunakan satuan Å (angstrom) atau 10^{-10}meter. Sedangkan untuk w dan sudut rotasi titik ligan, digunakan derajat, dengan besaran $[-180,180]$. Untuk variable x, y, z dari lokasi quaternion, digunakan besaran $[-1,1]$.



Gambar 2.18 Representasi Solusi BA untuk permasalahan *Molecular Docking* (López-Camacho et al., 2015)

2.11. Fungsi Objektif BA dalam permasalahan *Molecular Docking*

Fungsi objektif merupakan salah satu bagian terpenting dari *evolutionary algorithm.* Fungsi objektif memberikan acuan mengenai seberapa cocok akan suatu individu menjadi suatu solusi. Dalam penelitian ini, meminimalkan *free binding energy* adalah fungsi objektif dari permasalahan ini. Sebab semakin kecil nila *free binding energy* maka semakin erat ikatan yang terjadi.

Terjadi atau tidaknya suatu proses ditentukan oleh energi bebasnya. Hal ini dikarenakan tanpa adanya energi dari luar, suatu sistem cenderung akan berubah menuju kondisi bebas energi terendah mereka (Gapsys, Michielssens, Peters, de Groot, & Leonov, 2015). Nilai positif dari energi ikatan bebas menunjukkan tidak adanya ikatan, dan energi ikatan bebas negatif menunjukkan adanya ikatan. Semakin negatif nilai energi ikatan bebas, maka semakin besar energi yang dibutuhkan untuk melepaskan ikatan.

Free binding energy dapat dihitung dengan cara mengurangi energi saat *bound* dengan energi saat *unbound* dan ditambahkan energi torsional, saat ligand berubah dari *unbound* ke *bound* (Morris et al., 2009). Sehingga dapat dituliskan sebagai:

$$\Delta G = \Delta G_{bound} - \Delta G_{unbound} + N_{tor} \quad (2.5)$$

dimana:

ΔG : total energi bebas

ΔG_{bound} : total energi mekanik molekul saat terikat

$\Delta G_{unbound}$: total energi mekanik molekul saat tidak terikat

N_{tor} : jumlah titik torsional ligan

Energi mekanik molekul yang digunakan adalah sebanyak 4, yaitu: van der Waals, ikatan hidrogen, elektrostatis, dan desolvasi. Berikut adalah masing-masing cara perhitungannya (Morris et al., 2009):

$$vdW = \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} \right) \quad (2.6)$$

$$hBound = \sum_{i,j} E(t) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}} \right) \quad (2.7)$$

$$elec = \sum_{i,j} \frac{q_i q_j}{\varepsilon(r_{ij}) r_{ij}} \quad (2.8)$$

$$sol = \sum_{i,j} \left(S_i V_j + S_j V_i \right) e^{-\frac{r_{ij}^2}{2\sigma^2}} \quad (2.9)$$

dimana:

vdW : energi van der Waals

$hBound$: energi ikatan hidrogen

$elec$: energi elektrostatis

sol : energi desolvasi

$A_{ij}, B_{ij}, C_{ij}, D_{ij}$: parameter Lennard-Jones untuk atom i dan j

r_{ij} : jarak antar atom i dan j

$E(t)$: direksionalitas bergantung sudut

q_i : muatan di atom i

$\varepsilon(r_{ij})$: persamaan permitivitas

S_i : persamaan solvasi atom i

V_i : volume fragmental atom i

σ : konstanta jarak gaussian

Dengan menggunakan persamaan 2.5 hingga 2.9, dan dengan penambahaan bobot untuk tiap-tiap energi, persamaan 1 dapat ditulis ulang menjadi:

$$\Delta G = w_{vdW} \times (vdW_{bound} - vdW_{unbound}) + w_{hBound} \times (hBound_{bound} - hBound_{unbound}) + w_{elec} \times (elec_{bound} - elec_{unbound}) + w_{sol} \times (sol_{bound} - sol_{unbound}) + w_{tor} \times N_{tor}$$

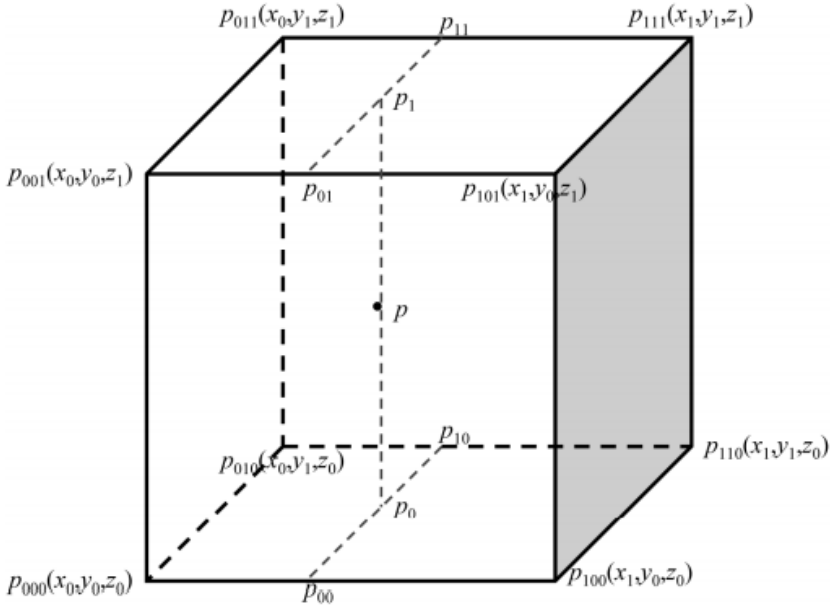
Parameter yang digunakan dalam Autodock 4 dapat dilihat pada tabel (Morris et al., 2009):

Tabel 2.1: Nilai Parameter di AutoDock4

Parameter	**Nilai**
σ	3.5 Å
w_{vdW}	0.1560
w_{hBound}	0.0974
w_{elec}	0.1465
w_{sol}	0.1159
w_{tor}	0.2744

2.12. Interpolasi Trilinier

Sama seperti pengerjaan dalam Autodock, dalam pengerjaan penelitian ini digunakan interpolasi trilinier untuk mempermudah perhitungan energi bebas. Interpolasi adalah suatu metode untuk menentukan nilai yang berada diantara dua nilai dari suatu persamaan. Interpolasi linier adalah interpolasi dari persamaan linier. Persamaan interpolasi trilinier didapatkan dari mengaplikasikan interpolasi linier sebanyak 7 kali. Gambar 2.19 menunjukkan ilustrasi dari interpolasi trilinier yang menggunakan 8 buah titik (Kang, 2006).



Gambar 2.19 Interpolasi Trilinier (Kang, 2006)

Dilakukan interpolasi sebanyak 3 kali untuk mendapatkan nilai $p_1$dan p_0. Kemudian 1 kali lagi untuk mendapatkan nilai p. Persamaan umum dari interpolasi trilinier dapat diexpresikan sebagai berikut:

$$p(x,y,z) = c_0 + c_1\Delta x + c_2\Delta y + c_3\Delta z + c_4\Delta x\Delta y + c_5\Delta y\Delta z + c_6\Delta x\Delta z + c_7\Delta x\Delta y\Delta z$$

dimana:

$\Delta x = \frac{x-x_0}{x_1-x_0}$;

$\Delta y = \frac{y-y_0}{y_1-y_0}$;

$\Delta z = \frac{z-z_0}{z_1-z_0}$;

$c_0 = p_{000}$;

$c_1 = (p_{100} - p_{000})$;

$c_2 = (p_{010} - p_{000})$;

$c_3 = (p_{001} - p_{000})$;

$c_4 = (p_{110} - p_{100} - p_{010} + p_{000})$;

$c_5 = (p_{011} - p_{010} - p_{001} + p_{000})$;

$c_6 = (p_{101} - p_{100} - p_{001} + p_{000})$;

$c_7 = (p_{111} - p_{110} - p_{101} - p_{011} + p_{100} + p_{010} + p_{001} - p_{000})$;

2.13. Validasi Hasil *Molecular Docking* Menggunakan BA

Validasi dilakukan agar kita dapat mengetahui apakah hasil *molecular docking* yang didapatkan menggunakan BA adalah valid. Dalam kasus ini, akan digunakan *Root Mean Squared Deviation*. *Root Mean Squared Deviation* (RMSD) dari jarak antar struktur protein menunjukkan kemiripan antar struktur protein. RMSD dengan nilai lebih kecil dari 2 Å, dapat dikatakan sebagai prediksi *molecular bonding* yang benar (Trott & Olson, 2010). Semakin kecil nilai RMSD maka dapat dikatakan bahwa prediksi *molecular docking* mendekati kejadian nyata.

Untuk dua buah struktur a dan b, RMSD didefinisikan sebagai (Trott & Olson, 2010):

$$RMSD_{ab} = \max(RMSD'_{ab}, RMSD'_{ba}) \tag{2.10}$$

$$RMSD'_{ab} = \sqrt{\frac{1}{N}\sum_{i} \min_{j} r_{ij}^2} \tag{2.11}$$

dimana:

$RMSD_{ab}$: Nilai RMSD dari struktur a dan b

N : jumlah atom berat di struktur a

$\min_j r_{ij}$: jarak minimum antara semua atom di b dengan elemen yang sama dengan atom i di struktur a terhadap posisi atom i di struktur a

Perhitungan dengan persamaan 2.10 dan 2.11 adalah untuk perhitungan RMSD untuk ligan yang berbeda dengan *native ligand* struktur kristal tersebut. Karena dalam pengerjaan penelitian ini digunakan ligan yang sama dengan *native ligand* dari struktur kristalnya, maka persamaan 2.10 dan 2.11 dapat disederhanakan menjadi:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i} r_i^2} \quad (2.12)$$

dimana:

$RMSD$: Nilai RMSD dari *molecular docking*

N : Banyaknya atom dari ligan

r : jarak atom ligan hasil *molecular docking* dan *native ligand* ke$-i$

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan tahapan-tahapan penelitian untuk menyelesaikan permasalahan sesuai dengan rumusan masalah yang telah disebutkan.

3.1 Tempat Penelitian

Penelitian dilaksanakan di Laboratorium Ilmu Komputer, Departemen Matematika, Fakultas Matematika Komputasi dan Sains Data.

3.2 *Software* Pendukung

Berikut adalah *software-software* pendukung yang digunakan dalam penelitian ini:

1. MarvinSketch (ChemAxon, 2019)

   MarvinSketch memiliki fitur yang mempermudah *user* untuk menggambar senyawa kimia, reaksi kimia, struktur Markush, dsb dengan mudah, cepat, dan akurat. Dalam pengerjaan penelitian ini, MarvinSketch digunakan untuk menggambar senyawa SA2014.

2. Chimera (Pettersen et al., 2004)

   UCSF Chimera adalah program yang sangat luas untuk visualisasi interaktif dan analisis struktur molekul dan data terkait, termasuk peta kerapatan, rakitan supramolekul, penyejajaran urutan, hasil *docking*, lintasan, dan ansambel konformasi. Selain visualisasi, Chimera dapat menghapus suatu senyawa atau atom dalam suatu sturktur. Dalam pengerjaan penelitian ini, Chimera akan digunakan untuk memisahkan protein dan ligan, dan juga memisahkan data cyclin D1 dari struktur yang dimiliki.

3. AutoDock 4 (Morris et al., 2009)

   AutoDock adalah seperangkat alat *docking* otomatis. AutoDock dirancang untuk memprediksi bagaimana molekul kecil, seperti substrat atau

kandidat obat, berikatan dengan reseptor struktur 3D yang diketahui. Dalam pengerjaan penelitian ini, AutoDock digunakan untuk mendapatkan nilai-nilai awal, yang nantinya akan dijadikan sebagai data perhitungan nilai *fitness*.

3.3 Tahapan Penelitian

Beberapa tahapan penelitian yang akan dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Dilakukan studi literatur untuk mendukung pengerjaan penelitian ini dan pemahaman yang lebih mendalam mengenai *molecular docking* dan *bat algorithm.* Literatur yang dipelajari dapat bersumber dari jurnal, buku, internet, maupun bimbingan dengan dosen pembimbing.

2. Pengumpulan Data

   Data yang digunakan dalam peneltitian ini berupa kompleks protein-ligan yang diunduh dari PDB database.

3. Perancangan *Bat Algorithm* untuk Menyelesaikan *Molecular Docking*

   Pada tahap perancangan algoritma ini akan dirumuskan representasi solusi, cara membentuk solusi awal, dan cara membentuk solusi baru yang berdasarkan pada *bat algorithm* untuk menyelesaikan *molecular docking*, serta penentuan beberapa parameter yang dibutuhkan untuk perhitungan.

4. Pembuatan Simulasi

   Bat algorithm yang telah dirumuskan untuk *molecular docking* akan disimulasikan dengan *source code* dari simulasi yang telah dibuat akan ditampilkan di Lampiran. Penjelasan lebih lengkap mengenai metodologi simulasi dapat dilihat pada sub-bab 3.3.

5. Analisis Hasil

   Pada tahap ini peneliti melakukan analisis hasil *molecular docking* yang diperoleh dengan *bat algorithm,* mengecek apakah hasilnya sesuai dengan aturan standar *molecular docking*, bahwa *free energy binding* bernilai negatif dan RMSDnya kurang dari 2Å.

6. Penyusunan Makalah dan Buku

Pada tahap ini, dilakukan penyusunan makalah dan buku yang sistematis. Hal ini dilakukan agar pembaca dapat dengan mudah memahami penelitian yang dilakukan.

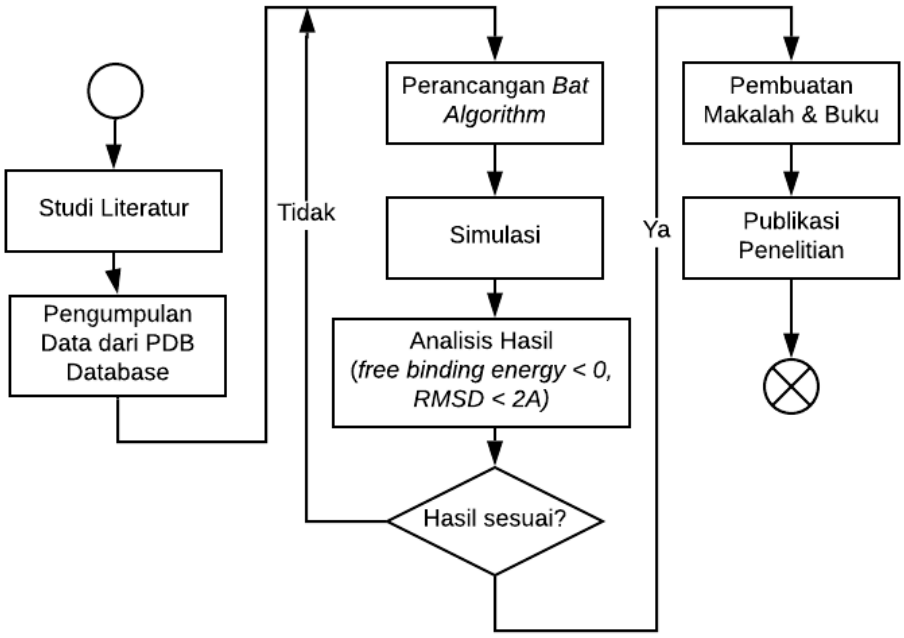
7. Publikasi Penelitian

   Pada tahapan ini, dilakukan publikasi penelitian dengan berpartisipasi pada seminar internasional. Kegiatan ini bertujuan untuk menunjukkan hasil penelitian yang telah dilakukan yang dapat dijadikan sebagai bahan kajian untuk melakukan penelitian selanjutnya.

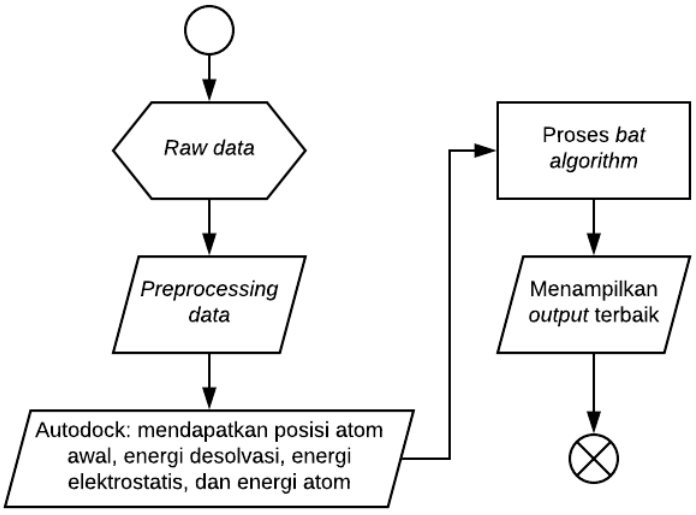
Diagram Alir tahapan penelitian dapat dilihat pada Gambar 3.1.

3.4 Methodologi Simulasi

Simulasi yang dilakukan dalam penelitian ini melibatkan bantuan AutoDock untuk dapat merubah dari rantai asam amino menjadi bentuk 3 dimensi. Nilai x, y, z dari setiap atom akan menjadi batasan peletakan ligan, apakah ligan dapat diletakkan pada posisi tertentu tersebut. Selanjutnya, program akan mendapatkan calon posisi *docking* dan selanjutnya masuk ke dalam proses BA, mulai dari hitung fungsi objektif dll.. Proses tersebut dilakukan secara terus menerus hingga batas iterasi atau syarat berhenti terpenuhi. Diagram alir simulasi dapat dilihat pada Gambar 3.2.



Gambar 3.1: Diagram Alir Penelitian



Gambar 3.2: Diagram Alir Simulasi

BAB IV
PERSIAPAN DATA DAN
PERANCANGAN ALGORITMA

4.1 Persiapan Data

4.1.1 *Raw Data*

Raw data didapatkan dari *Protein Data Bank* (PDB), dalam pengerjaan kali ini, akan digunakan 2 buah struktur kristal yang akan digunakan sebagai validasi, yaitu 3ptb (Marquart, Walter, Deisenhofer, Bode, & Huber, 1983) dan 2cpp (Poulos, Finzel, & Howard, 1987). 3ptb adalah kode untuk struktur kristal dengan protein β-Trypsin dan ligan benzamidine, sedangkan 2cpp adalah kode untuk struktur kristal Cytochrome P-450$_{cam}$ dan ligan camphor. Gambar 4.1 dan 4.2 masing-masing menampilkan 3ptb dan 2cpp di *website* PDB.

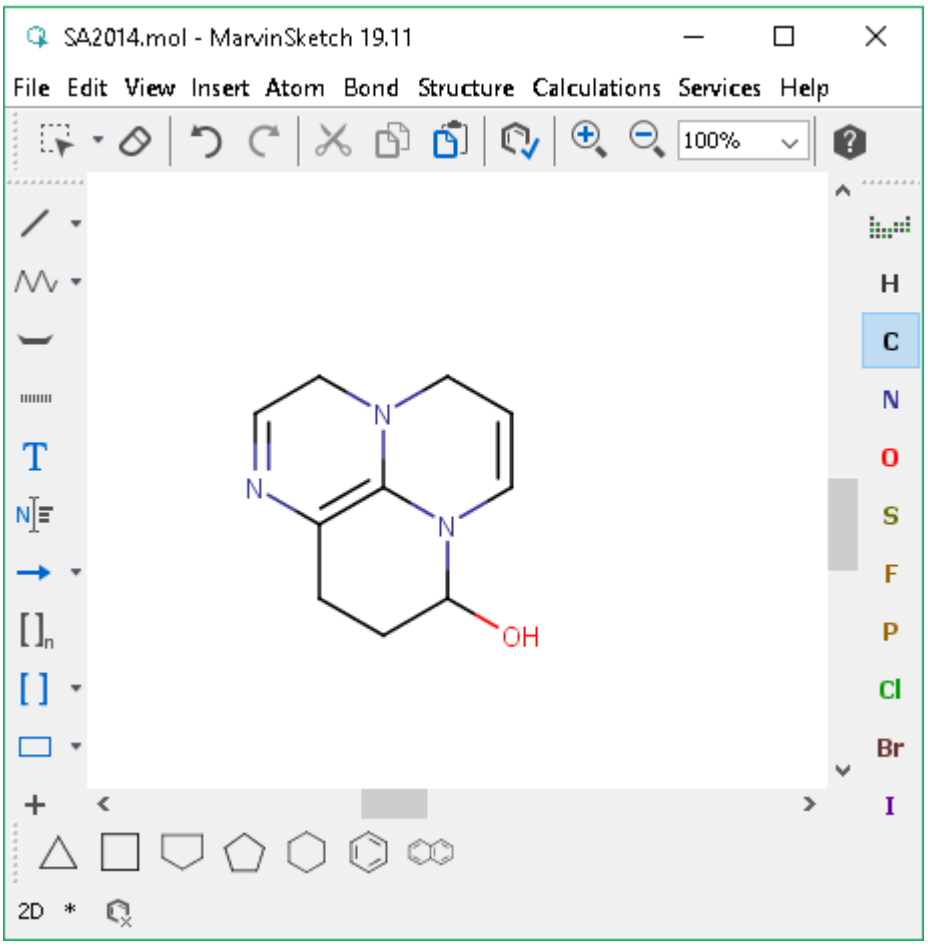
Gambar 4.1: Struktur 3ptb di PDB

Gambar 4.2: Struktur 2cpp di PDB

Untuk struktur cyclin D1, didapatkan dari dari PDB dengan kode 2w96 (Day et al., 2009), 2w96 adalah struktur dari kompleks CDK/cyclin D1, sehingga harus dilakukan pemotongan agar didapatkan struktur cyclin D1 saja. Gambar 4.3 menunjukan struktur 2w96 pada *website* PDB.

Gambar 4.3: Struktur 2w96 di PDB

Untuk senyawa SA2014, akan dibuat dengan menggunakan software MarvinSketch (ChemAxon, 2019). Gambar 4.4 adalah file yang telah dibuat untuk membentuk senyawa SA2014 menggunakan MarvinSketch.



Gambar 4.4: Membuat Struktur SA2014 di MarvinSketch

4.1.2 Pemrosesan Data dengan Chimera

Raw data yang didapatkan akan dilakukan pemrosesan awal, dimana akan didapatkan struktur yang siap dimasukkan ke dalam Autodock (Morris et al., 2009). Untuk struktur kristal 3ptb dan 2cpp, akan dilakukan pemotongan ligan dan proteinnya. Software yang digunakan adalah Chimera (Pettersen et al., 2004). Berikut adalah langkah-langkah untuk memotong ligan dan protein:

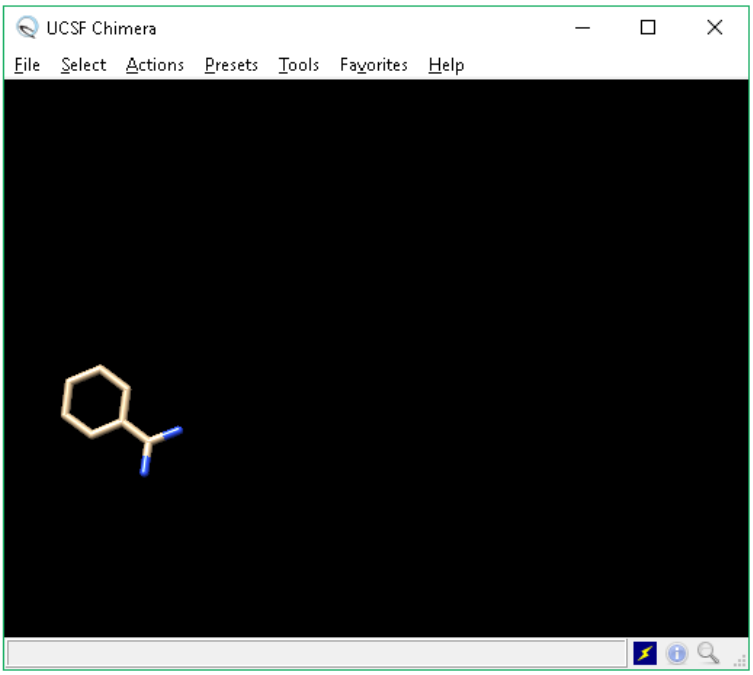
1. Buka file 3ptb.pdb dan 2cpp.pdb yang didapatkan dari *website* PDB. Gambar 4.5 dan Gambar 4.6 masing-masing menunjukkan struktur 3ptb dan 2cpp dalam software Chimera.

Gambar 4.5: Struktur 3ptb di Chimera

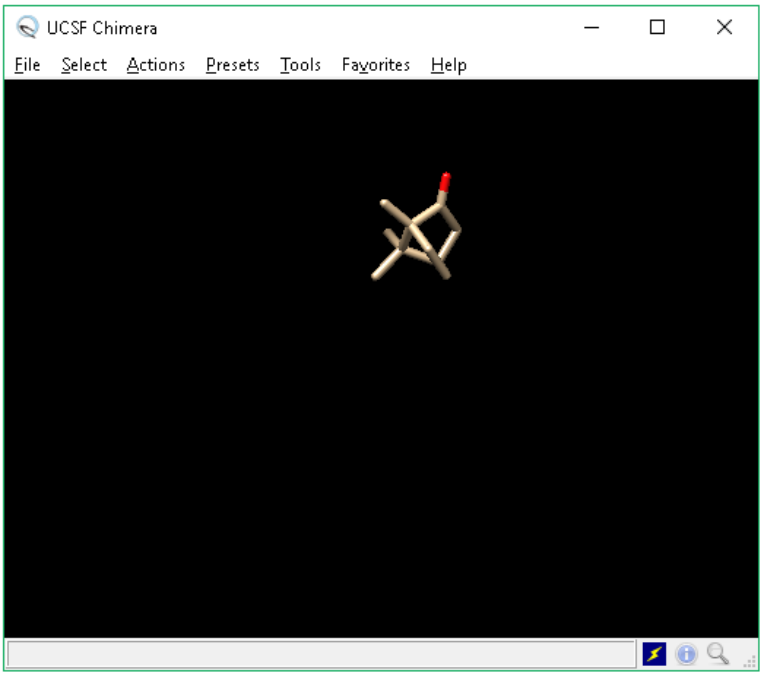
Gambar 4.6: Struktur 2cpp di Chimera

2. Selanjutnya akan didapatkan file ligan dari struktur yang telah didapatkan. Pilih *Select -> Structure -> ligand*, setelah ligan terpilih, *invert* yang terpilih, sehingga akan terpilih proteinnya, dengan cara:

Pilih *Select -> Invert (Selected models)*. Setelah itu pilih *Actions -> Atoms/Bonds -> delete*, dimana struktur yang terpilih (protein) akan dihapus, sehingga menyisakan ligan yang tertinggal. Dalam kasus 2cpp, aka nada lebih dari struktur yang tersisa, sehingga pilih yang tidak dibutuhkan dengan cara *Select -> Residue -> HEM,* dan hapus dengan *Actions -> Atoms/Bonds -> delete*. Gambar 4.7 dan 4.8 berturut-turut menunjukkan ligan dari 3ptb dan 2cpp, selanjutnya simpan ligan dengan cara *File -> Save PDB…*, beri nama dan klik *save*.



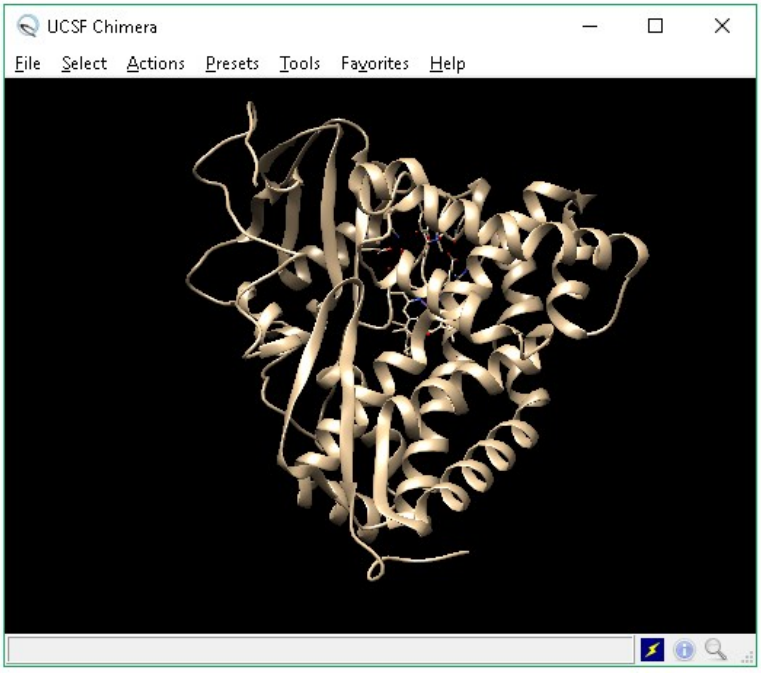
Gambar 4.7: Ligan dari struktur 3ptb



Gambar 4.8: Ligan dari Struktur 2cpp

3. Langkah selanjutnya adalah untuk mendapatkan file proteinnya, pertama pilih ligan dengan memilih *Select -> Structure -> ligand,* dan selanjutnya hapus ligan tersebut dengan memilih *Actions -> Atoms/Bonds -> delete* . Sehingga akan didapatkan proteinnya saja. Gambar 4.9 dan Gambar 4.10 menunjukkan protein dari struktur 3ptb dan 2cpp.

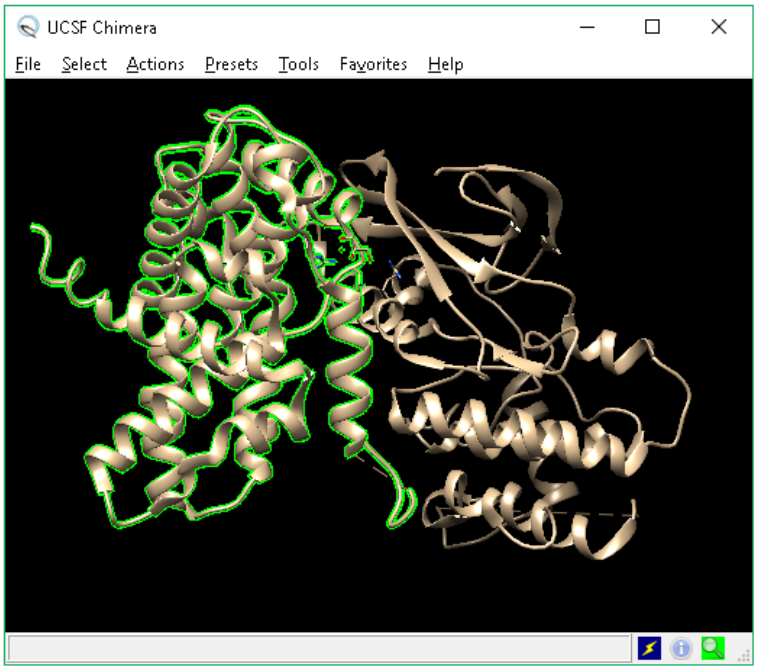
Gambar 4.9: Protein dari Struktur 3ptb



Gambar 4.10: Protein dari Struktur 2cpp

Untuk struktur 2w96, akan dilakukan pemotongan terhadap kompleks CDK/Cyclin D1. Langkah yang digunakan adalah:

1. Buka file 2w96 di Chimera, Gambar 4.11 adalah struktur 2w96 dalam program Chimera. Rantai protein yang dipilih (berselimut warna hijau) adalah cyclin D1.



Gambar 4.11: Struktur 2w96 di Chimera

2. Selanjutnya adalah menghapus rantai CDK dari kompleks 2w96. Chimera mengenali 2 buah rantai dari kompleks 2w96, dinamai rantai A dan B. Dimana rantai A adalah Cyclin D1 dan rantai B adalah CDK. Sehingga kita akan menghapus rantai B, dengan cara pilih *Select -> Chain -> B*, selanjutnya pilih *Actions -> Atoms/Bonds -> delete*. Sehingga akan tersisa rantai A seperti pada Gambar 4.12.

Gambar 4.12: Protein Cyclin D1 dari Struktur 2w96

3. Simpan dengan langkah yang sama seperti sebelumnya.

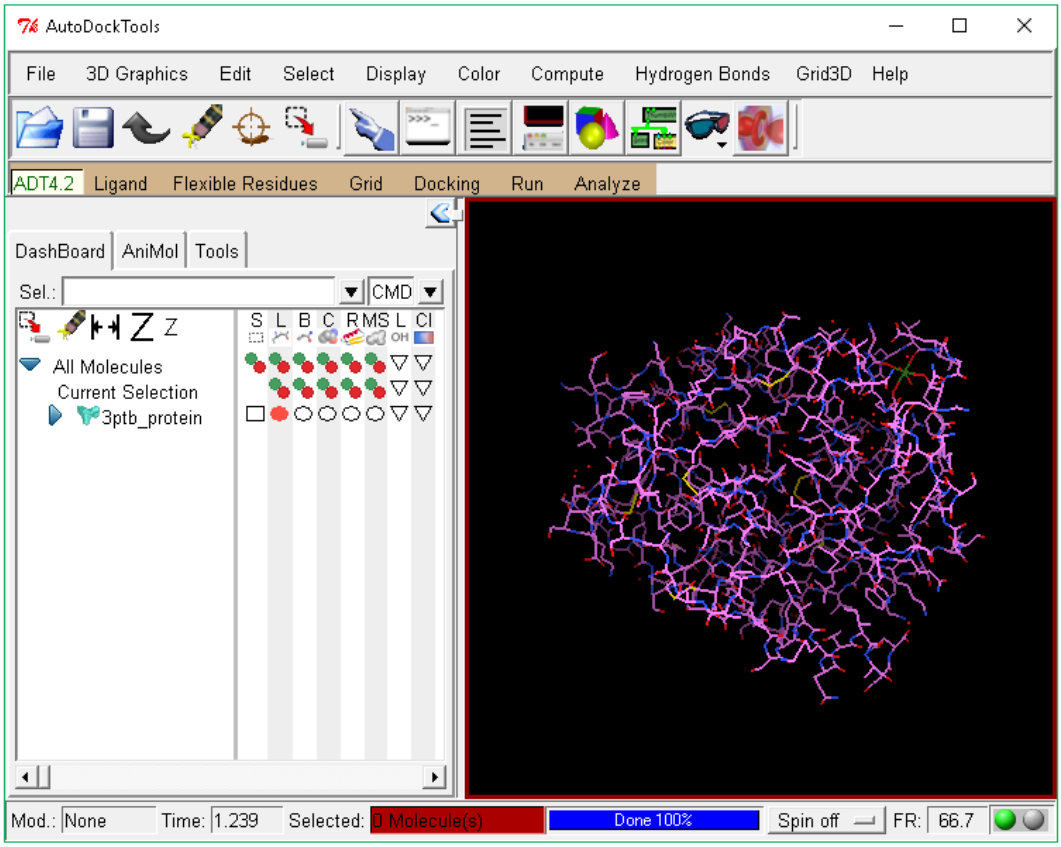
4.1.3 Pemrosesan dengan Menggunakan Autodock

Langkah ini adalah untuk mendapatkan nilai-nilai awal yang akan digunakan kedalam penyelesaian masalah *molecular docking* dengan menggunakan BA. Untuk proses di Autodock, 3ptb, 2cpp, dan cyclin D1-SA2014, mendapatkan perlakuan yang sama. Berikut adalah langkah-langkah yang dilakukan didalam Autodock:

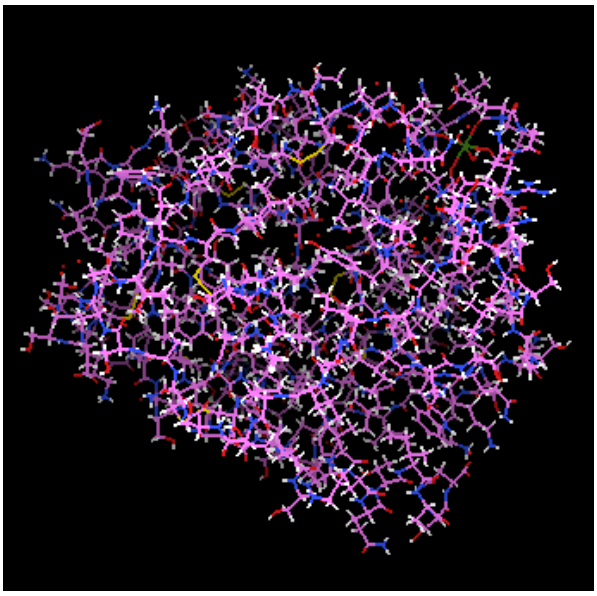
1. Preparasi Makromolekul (Protein)

   Langkah yang dilakukan adalah langkah umum yang biasa digunakan, dimulai dengan membuka *file*, dengan langkah: *File -> Read molecule* -> Pilih *file* protein yang akan digunakan. Gambar 4.13 dan 4.15 menunjukkan *file* protein yang telah dibuka, 4.13 adalah protein dari struktur 3ptb dan 4.15 adalah protein dari 2cpp. Selanjutnya tambahkan *hydrogen* dengan cara *Edit -> Hydrogens -> Add -> All Hydrogens, noBondOrder, yes -> OK*. Penambahan *hydrogen* dilakukan agar system yang akan dibuat, menyerupai sistem asli. Gambar 4.14

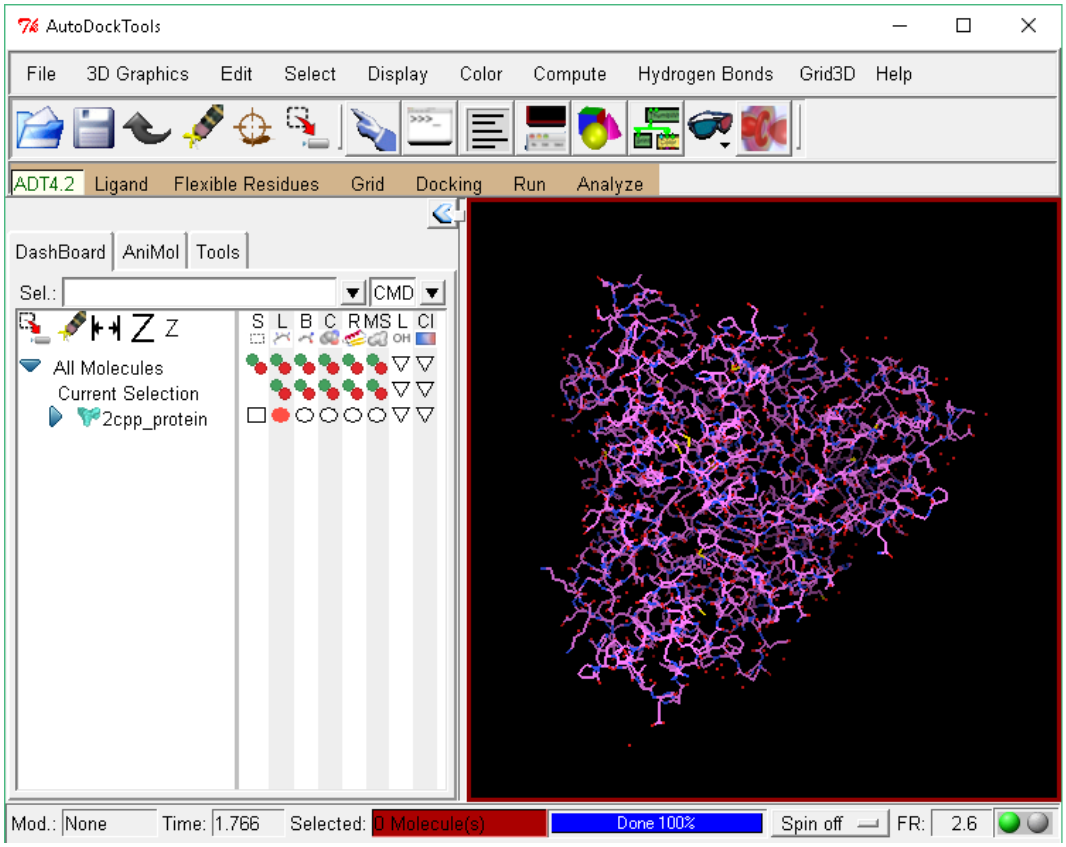
menampilkan protein 3ptb ditambahkan hidrogen dan 4.16 menampilkan protein 2cpp yang telah ditambahkan hidrogen.



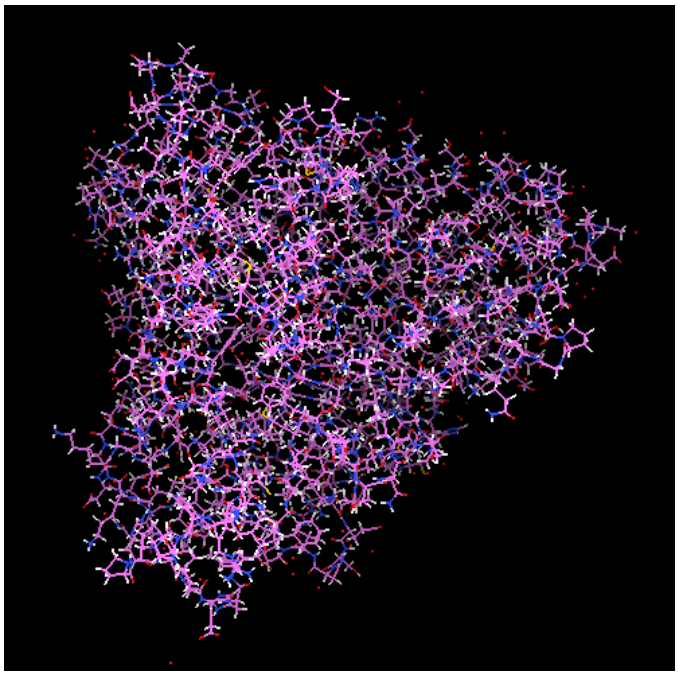
Gambar 4.13: Protein dari Struktur 3ptb di AutoDock



Gambar 4.14: Protein dari Struktur 3ptb dengan Penambahan Atom Hidrogen



Gambar 4.15: Protein dari Struktur 2cpp di AutoDock



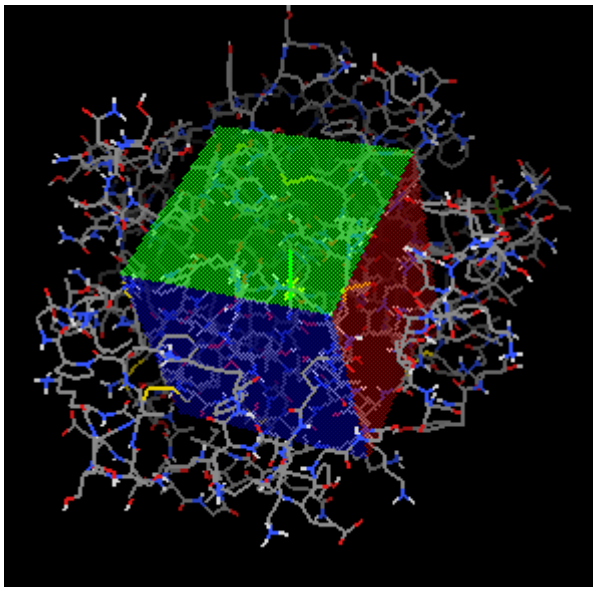
Gambar 4.16: Protein dari Struktur 2cpp dengan Penambahan Atom Hidrogen

2. Preparasi Ligan

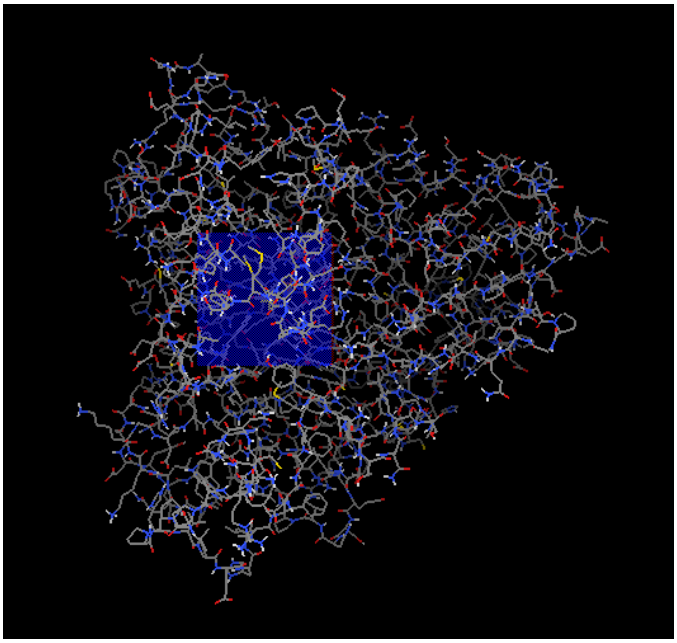
   Pertama akan dimasukkan ligan ke dalam *file* protein yang telah terbuka, dengan cara *Ligand -> Input -> Open ->* Pilih ligan yang akan digunakan. Selanjutnya simpan ligan dengan format PDBQT.

3. Menyiapkan Autogrid

   Pilih molekul yang akan dijadikan sebagai makromolekul dengan langkah *Grid -> Macromolecule -> Choose -> pilih Makromolekul -> Select Molecule -> Save as PDBQT*. Selanjutnya pilih molekul yang akan digunakan sebagai ligan, *Grid -> Set map types -> choose ligand -> Pilih Ligand -> Select Ligand.* Dalam penelitian kali ini akan digunakan *Grid Box* dengan ukuran $40 \times 40 \times 40$ dengan jarak 0.375 dengan pusat adalah ligan. Untuk mengatur *Grid Box*, dapat dilakukan dengan cara *Grid -> Grid Box*. Lalu untuk mengatur agar *Grid Box* berpusat pada ligand dapat dipilih dengan *Grid -> Grid Box -> Center -> Center on ligand*. Selanjutnya keluar dari pengaturan dan simpan pengaturan tersebut, *File -> Close saving current*, lalu *Grid -> Output -> Save GPF*. Gambar 4.17 menunjukkan *Grid Box* untuk 3ptb dan Gambar 4.18 untuk 2cpp.



Gambar 4.17: *Grid Box* 3ptb



Gambar 4.18: *Grid Box* 2cpp

4. Menjalankan Autogrid

   Langkah pertama yang dilakukan adalah menjalankan *autogrid* dengan cara pilih *Run -> Run Autogrid*. Selanjutnya akan muncul *box* yang harus diisi terlebih dahulu sebelum menjalankan *autogrid*. Untuk *Program Pathname,* untuk mempermudah, dapat diisi dengan menggunakan *browse* dan memilih aplikasi autogrid4.exe. Begitu juga dengan *Parameter Filename*, pilih file GPF yang telah dibuat pada seksi sebelumnya. Lalu untuk *Log Filename*, dapat diisi dengan nama terserah dengan *extension* .glg.

4.1.4 Menyimpan Data ke Database

Database dibentuk untuk mempermudah penyimpanan data dan pengoperasian data. Setiap *record* akan memiliki nilai koordinat x, y, z, energi desolvasi, energi elektrostatis, dan energi atom yang dimiliki oleh struktur tersebut. Gambar 4.19 dan 4.20 masing masing menampilkan beberapa contoh nilai yang tersimpan dari struktur 3ptb dan 2cpp.

koorx	koory	koorz	energiA	energiC	energiN	desolvasi	electrostatic
-9.356	6.866	9.248	23.732	23.655	9.939	0.937	-0.023
-8.981	6.866	9.248	24.816	24.735	10.355	0.974	-0.023
-8.606	6.866	9.248	24.004	23.92	10.02	1.012	0.015
-8.231	6.866	9.248	20.239	20.152	8.38	1.042	0.063
-7.856	6.866	9.248	13.568	13.479	5.372	1.076	0.095
-7.481	6.866	9.248	7.737	7.645	2.715	1.099	0.095
-7.106	6.866	9.248	5.707	5.615	1.776	1.114	0.071
-6.731	6.866	9.248	8.706	8.612	3.141	1.132	0.042
-6.356	6.866	9.248	17.428	17.333	7.158	1.155	0.015
-5.981	6.866	9.248	28.913	28.816	12.495	1.165	0.01
-5.606	6.866	9.248	35.144	35.047	15.371	1.171	0.033
-5.231	6.866	9.248	34.375	34.276	14.822	1.187	0.075
-4.856	6.866	9.248	44.887	44.787	18.87	1.201	0.141
-4.481	6.866	9.248	168.94	168.84	72.121	1.205	0.238
-4.106	6.866	9.248	914.764	914.664	393.328	1.207	0.372
-3.731	6.866	9.248	3505.56	3505.46	1520.84	1.206	0.513
-3.356	6.866	9.248	5864.5	5864.4	2603.28	1.213	0.647
-2.981	6.866	9.248	6344.41	6344.31	2926.6	1.216	1.053
-2.606	6.866	9.248	11410	11409.9	5397.83	1.22	1.729
-2.231	6.866	9.248	22176.9	22176.8	10537.7	1.215	2.189
-1.856	6.866	9.248	21861.8	21861.7	10391.8	1.207	1.969
-1.481	6.866	9.248	8602.37	8602.26	4082.78	1.203	1.18
-1.106	6.866	9.248	1978.36	1978.26	926.239	1.202	0.3
-0.731	6.866	9.248	1123.34	1123.24	511.094	1.233	-0.243
-0.356	6.866	9.248	1098.53	1098.42	500.847	1.236	-0.326

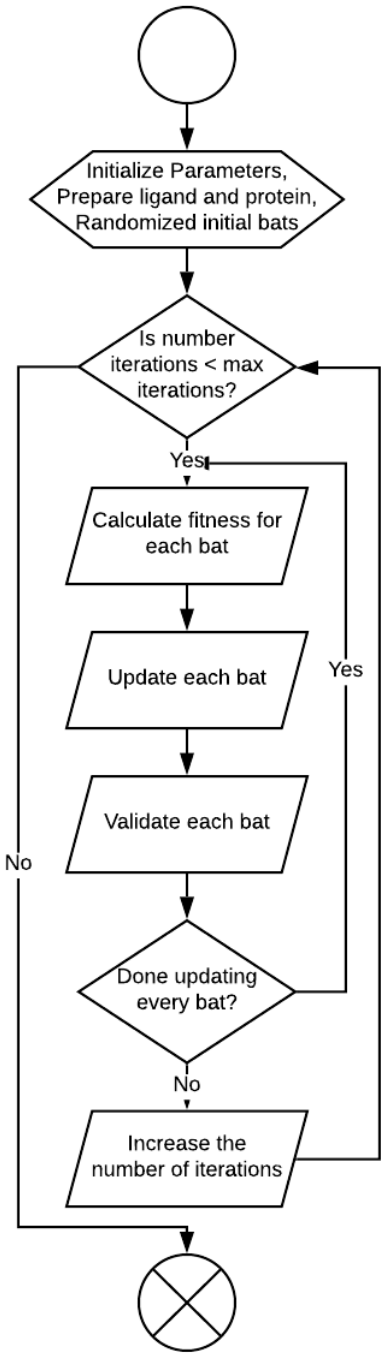
Gambar 4.19: Beberapa Data dari 3ptb

koorx	koory	koorz	energiC	energiOA	desolvasi	electrostatic
38.797	36.525	6.732	-0.041	-1.352	1.205	-0.023
39.172	36.525	6.732	0.243	-1.217	1.204	0.051
39.547	36.525	6.732	1.018	-0.914	1.205	0.098
39.922	36.525	6.732	4.516	0.232	1.212	0.123
40.297	36.525	6.732	22.124	5.859	1.22	0.125
40.672	36.525	6.732	109.581	33.772	1.227	0.108
41.047	36.525	6.732	515.441	164.834	1.22	0.068
41.422	36.525	6.732	1767.24	570.97	1.232	0.01
41.797	36.525	6.732	3368.26	1091.12	1.235	-0.067
42.172	36.525	6.732	3100.98	1003.68	1.231	-0.151
42.547	36.525	6.732	1496.47	481.954	1.234	-0.218
42.922	36.525	6.732	586.549	186.938	1.238	-0.25
43.297	36.525	6.732	213.509	66.536	1.236	-0.249
43.672	36.525	6.732	84.048	25.082	1.242	-0.234
44.047	36.525	6.732	44.336	12.582	1.232	-0.211
44.422	36.525	6.732	25.818	6.863	1.238	-0.19
44.797	36.525	6.732	13.944	3.249	1.24	-0.173
45.172	36.525	6.732	5.808	0.806	1.227	-0.155
45.547	36.525	6.732	1.966	-0.319	1.21	-0.139
45.922	36.525	6.732	0.524	-0.725	1.219	-0.121
46.297	36.525	6.732	0.883	-0.602	1.212	-0.1
46.672	36.525	6.732	3.134	0.087	1.214	-0.091
47.047	36.525	6.732	7.833	1.543	1.223	-0.102
47.422	36.525	6.732	14.66	3.653	1.2	-0.142
47.797	36.525	6.732	19.096	4.953	1.217	-0.236

Gambar 4.20: Beberapa Data dari 2cpp

4.2 Perumusan *Bat Algorithm* untuk *Molecular Docking*

Bat algorithm akan disesuaikan agar dapat menyelesaikan permasalahan *molecular docking*. Dalam subbab 2.10, telah dijelaskan bagaimana bentuk solusi yang akan ditawarkan sebagai penyelesaian permasalahan *molecular docking*. Solusi tersebut dapat dikembalikan menjadi posisi konformasi ligan. Langkah-langkah yang dilakukan dalam *bat algorithm* dilakukan pada Gambar 4.21.



Gambar 4.21: *Flowchart* BA

Berikut ini adalah implementasi dari langkah-langkah BA untuk *molecular docking*:

4.2.1 Inisialisasi Parameter

Inisialisasi parameter dilakukan untuk mendapatkan nilai-nilai parameter yang akan digunakan dalam BA. Parameter yang akan diisi adalah: ukuran populasi, jumlah iterasi, *loudness*, *pulse emission*, frekuensi minimum, frekuensi maksimum, α, dan γ.

4.2.2 Membuat Populasi Awal

Populasi awal akan di*random* berdasarkan kriteria yang telah dijelaskan di bab sebelumnya. *Source code function* untuk membuat populasi awal dapat dilihat pada Lampiran A. *value* adalah yang akan menjadi nilai dari populasi awal BA, sedangkan *boundary* adalah nilai yang akan menjadi Batasan dalam pengerjaan *molecular docking*.

4.2.3 Menghitung *Fitness*

Fitness dalam BA ini adalah menggunakan energi bebas dari gabungan protein-ligan tersebut. Dalam kasus ini, digunakan pendekatan menggunakan interpolasi. Langkah pertama adalah dengan mengambil data dari database, data tersebut berupa data desolvasi, elektrostatis, dan energi atom. Setelah data didapatkan, akan dihitung energi van der wall, desolvasi, dan elektrostatisnya, selanjutnya akan dihitung energi bebas dari struktur tersebut. *Source code* perhitungan nilai *fitness* dapat dilihat pada Lampiran C.

4.2.4 *Update*

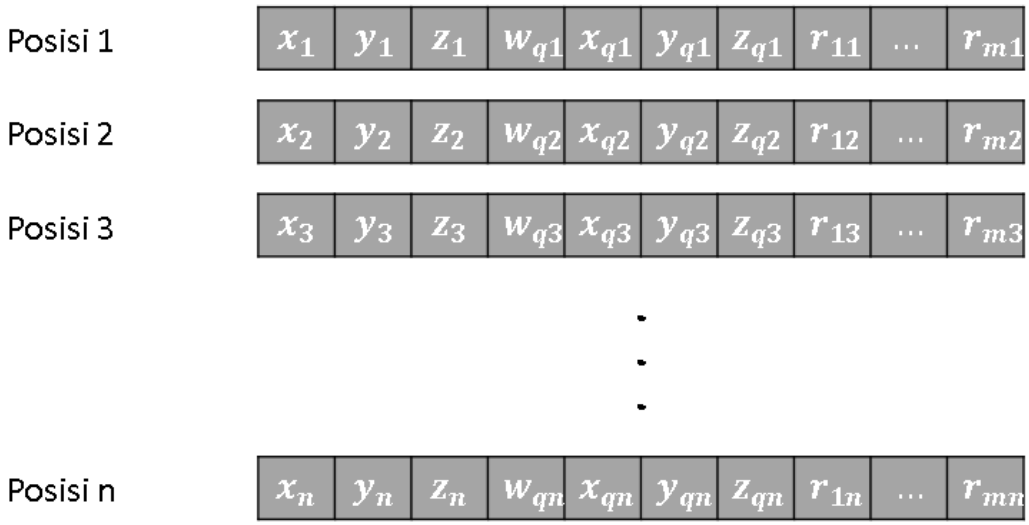
Proses *update* dimulai dengan mencari kelelawar terbaik di iterasi tersebut, setelah didapatkan, akan dilakukan perhitungan untuk mendapatkan frekuensi, posisi, dan kecepatan yang baru, sesuai dengan persamaan 2.1 hingga 2.3. Selain itu juga dilakukan *random walk* untuk beberapa kelelawar yang dipilih secara acak. *Source code* untuk *update* kelelawar dapat dilihat pada Lampiran D.

4.2.5 Validasi BA

Validasi BA dilakukan agar apabila kelelawar pada generasi sekarang lebih buruk dibandingkan generasi sebelumnya atau apabila terlalu bising, maka

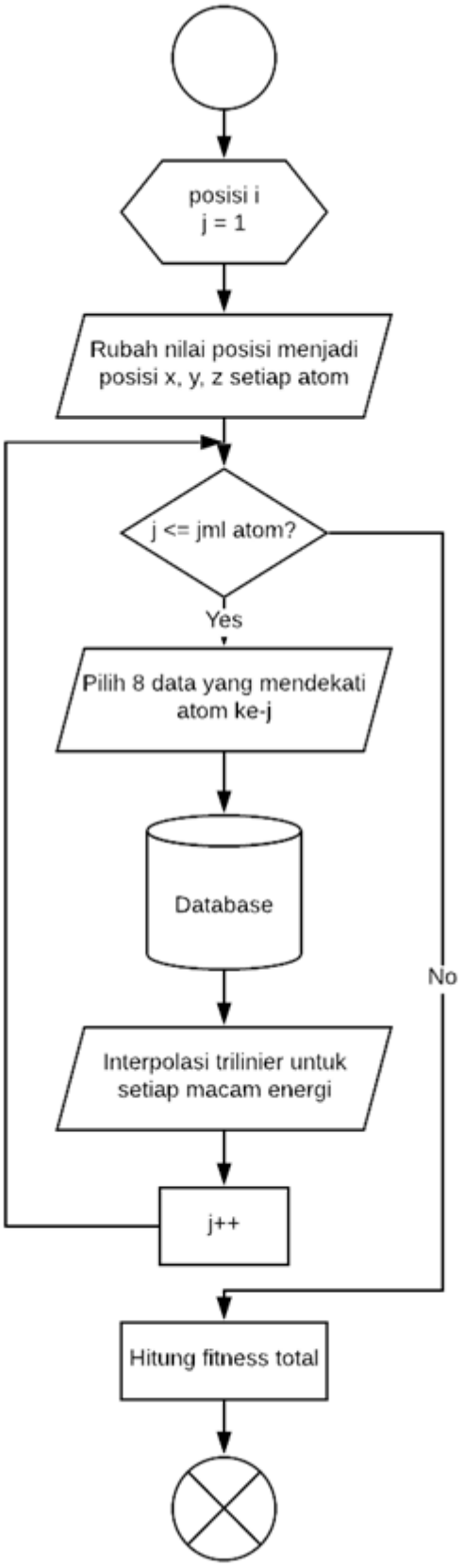
nilainya tidak akan di*update* (atau di*revert* ke nilai generasi sebelumnya), untuk *source code* lebih lengkapnya dapat dilihat pada Lampiran E.

Berikut adalah bentuk kelelawar yang akan dibuat beserta langkah-langkah yang dilakukan dalam BA. Kelelawar dalam algoritma adalah menunjukkan solusi dari permasalahan. Dalam kasus ini adalah posisi konformasi ligan. Kelelawar, yang selanjutnya akan disebut posisi, dalam kasus ini dimulai dengan mendapatkan nilainya dengan cara *random* dengan batasan yang didapat dari *database.* Gambar 4.22 adalah populasi pertama dari posisi.



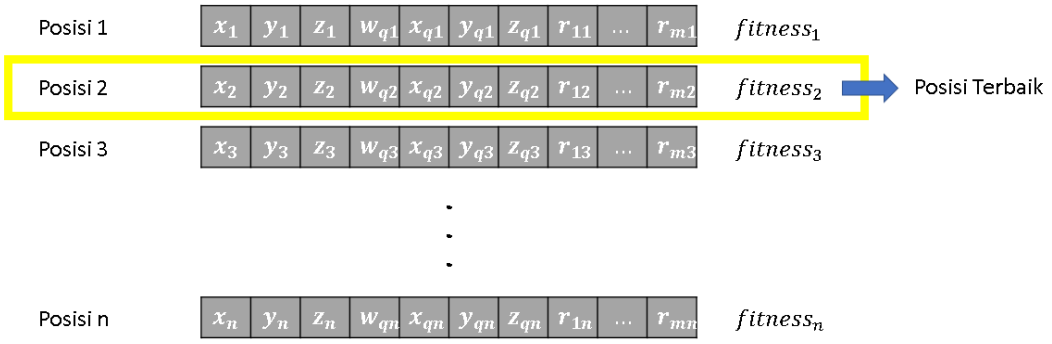
Gambar 4.22: Bentuk Posisi Pada Populasi Pertama

Setelah didapatkan populasi awal, langkah selanjutnya adalah menghitung nilai *fitness* untuk setiap posisi. Perhitungan fungsi *fitness* dapat dilihat pada Gambar 4.23. Langkah pertama adalah dengan mengubah kembali bentuk posisi menjadi bentuk posisi x, y, dan z untuk setiap atom dalam ligan. Selanjutnya adalah dengan mencari dari database 8 buah data yang memiliki posisi bersesuaian dengan posisi setiap atom. Lakukan interpolasi trilinier agar mendapatkan nilai energi tersebut. Dan ulangi untuk setiap atom yang ada pada ligan dan ketika selesai semua, jumlahkan semua energi setiap atom dan nilai energi bebas atau *fitness*nya didapatkan.



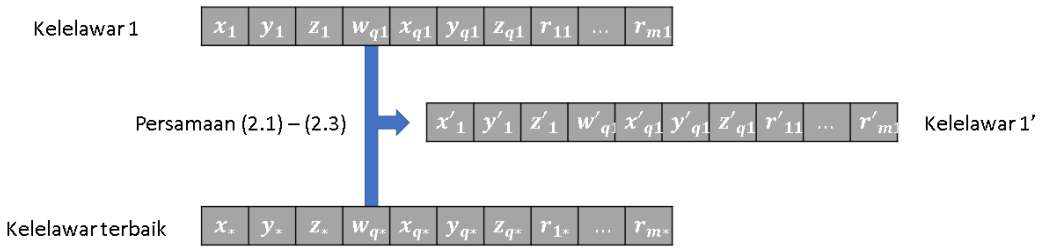
Gambar 4.23: *Flowchart* Perhitungan *Fitness* Penelitian

Setelah semua posisi memiliki nilai *fitness*nya, akan dicari posisi dengan *fitness* terbaik dan dijadikan kelelawar terbaik pada iterasi tersebut. Gambar 4.24 menunjukkan pemilihan posisi terbaik berdasarkan nilai *fitness*nya.



Gambar 4.24: Pemilihan Posisi Terbaik

Setelah dipilih posisi terbaik, setiap posisi akan di*update* berdasarkan persamaan 2.1 hingga 2.3, Gambar 4.25 menunjukkan langkah yang dilakukan dalam meng*update* posisi 1. Dengan cara yang sama, di*update* semua posisi sehingga didapatkan posisi baru.



Gambar 4.25: *Update* Posisi 1

Sebagai contoh *update,* misalkan nilai posisi 1 dan posisi terbaik secara berutan adalah sebagai berikut:

8.5	40.4	-4.4	0.42	-0.1	0.33	80.9	100

10.4	30.1	5	0.13	0.3	-0.2	-5.11	20.5

Setiap posisi memiliki kecepatan, dalam kasus ini dianggap bahwa kecepatannya masih bernilai 0 (iterasi pertama). Langkah *update* akan dimulai

dengan persamaan 2.1, yaitu menghitung frekuensi setiap nilai. Dengan anggapan nilai $f_{min} = 0$ dan $f_{max} = 2$.

$$f_1 = 0 + (2 - 0) * \beta$$

$$f_1 = 2\beta$$

dengan β adalah *random uniform* dari [0,1].

Misalkan nilai frekuensi posisi 1 didapatkan sebagai berikut:

0.3	0.52	1.1	0.88	0.43	1.4	0.22	0.71

Setelah didapatkan nilai frekuensi, langkah selanjutnya adalah melakukan *update* kecepatan dengan persamaan 2.2.

$$v_1^{t+1} = v_1^t + (x_i^t - x_*)f_1$$

Sehingga berikut adalah perhitungan kecepatan baru untuk posisi 1

0 $+ (8.5$ $- 10.4)$ $* 0.3$	0 $+ (40.4$ $- 30.1)$ $* 0.52$	0 $+ (-4.4$ $- 5)$ $* 1.1$	0 $+ (0.42$ $- 0.13)$ $* 0.88$	0 $+ (-0.1$ $- 0.3)$ $* 0.43$	0 $+ (0.33$ $+ 0.2)$ $* 1.4$	0 $+ (80.9$ $+ 5.11)$ $* 0.22$	0 $+ (100$ $- 20.5)$ $* 0.71$

Dan didapatkan hasil akhir kecepatan adalah:

0.57	5.356	-10.34	0.2552	-0.172	0.742	18.922	56.445

Seletah didapatkan kecepatan terbaru dari posisi 1, maka akan dihitung perubahan nilai posisi 1. Sesuai dengan persamaan 2.3, maka nilai dari posisi 1 berubah seperti berikut:

8.5	40.4	-4.4	0.42	-0.1	0.33	80.9	100

$+$

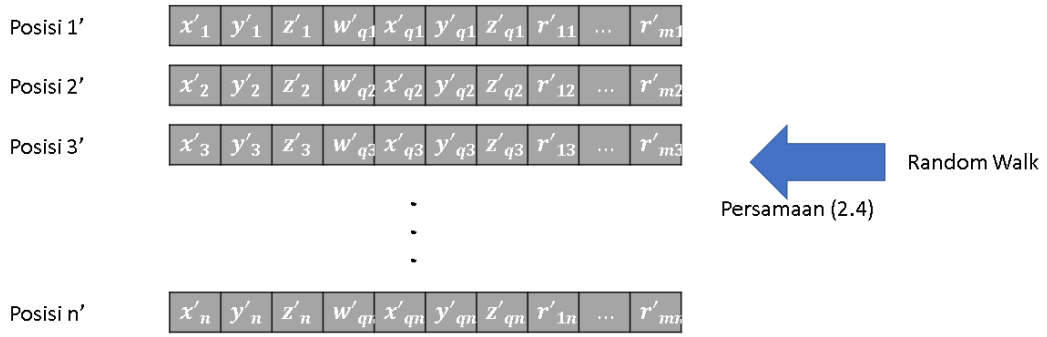
0.57	5.356	-10.34	0.2552	-0.172	0.742	18.922	56.445

$=$

9.07	45.756	-14.74	0.6752	-0.272	1.072	99.822	156.45

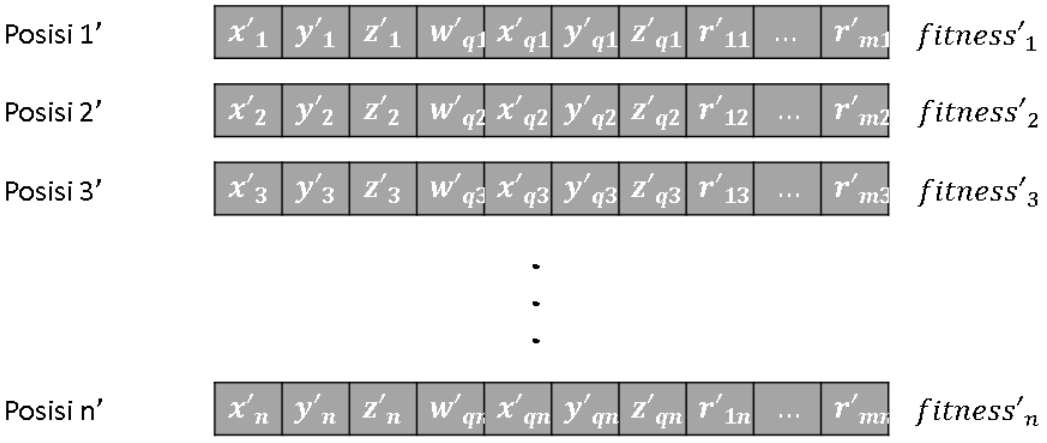
Setelah itu akan dilakukan *random walk*, pemilihan posisi yang dilakukan untuk *random walk* adalah dengan *random* juga. Sehingga tidak menentu posisi

mana yang akan diberi *random walk*. Gambar 4.26 menunjukkan ilustrasi *random walk*.



Gambar 4.26: Ilustrasi *Random Walk*

Dengan posisi-posisi yang telah ter*update*, kita hitung kembali nilai *fitness* posisi baru. Gambar 4.27 menunjukkan posisi baru dengan *fitness*nya masing-masing.



Gambar 4.27: Posisi Ter*update* dengan *Fitness*nya

Langkah terakhir untuk mengakhir iterasi adalah dengan melakukan validasi apakah nilai posisi akan diperbarui atau tidak. Fungsi perbaruan kurang lebih ada pada fungsi dibawah ini.

$$fitness_i' > fitness_i \wedge A_i < A \,?\, Posisi\ i' : Posisi\ i$$

Fungsi tersebut jika diartikan ke dalam bahasa sehari-hari, akan menjadi: Jika $fitness_i' > fitness_i$ dan $A_i < A$ adalah benar, maka yang akan dipilih adalah $Kelelawar\ i'$, namun apabila $fitness_i' > fitness_i$ dan $A_i < A$ bernilai

salah, yang terpilih adalah $Kelelawar\ i$ untuk dipilih pada iterasi selanjutnya. Dengan semua posisi tervalidasi, akan dilakukan semuanya dari Gambar 4.23 hingga 4.27 dan juga validasi jika iterasi masih belum terpenuhi.

4.3 Validasi Hasil *Molecular Docking*

Validasi akan dilakukan agar menentukan, apakah hasil yang didapatkan dengan menggunakan BA ini, dapat dikatakan hasil yang valid. Suatu hasil *molecular docking* dikatakan valid apabila RMSD dari struktur tersebut menghasilkan nilai yang lebih kecil dari 2 Å. Dengan rumus RMSD yang ada pada persamaan 2.12, dibuatlah *source code* yang dapat menghitungnya. *Source code* perhitungan RMSD dapat dilihat pada Lampiran F. Langkah perhituerhitungan RMSD dari posisi i adalah dengan merubah posisi i menjadi bentuk posisi x, y, z setiap atom dari ligan. Selanjutnya sesuai dengan jumlah atom yang dimiliki ligan, akan dihitung jarak antara atom prediksi dan atom asli. Karena 3 dimensi, perhitungan jarak menggunakan persamaan 4.1. dibawah ini.

$$r_j = \sqrt{\left(x_j - x_r\right)^2 + \left(y_j - y_r\right)^2 + \left(z_j - z_r\right)^2} \tag{4.1}$$

dimana:

r_j : jarak atom j prediksi dengan atom j pada *native ligand*

x_j : posisi x atom j prediksi

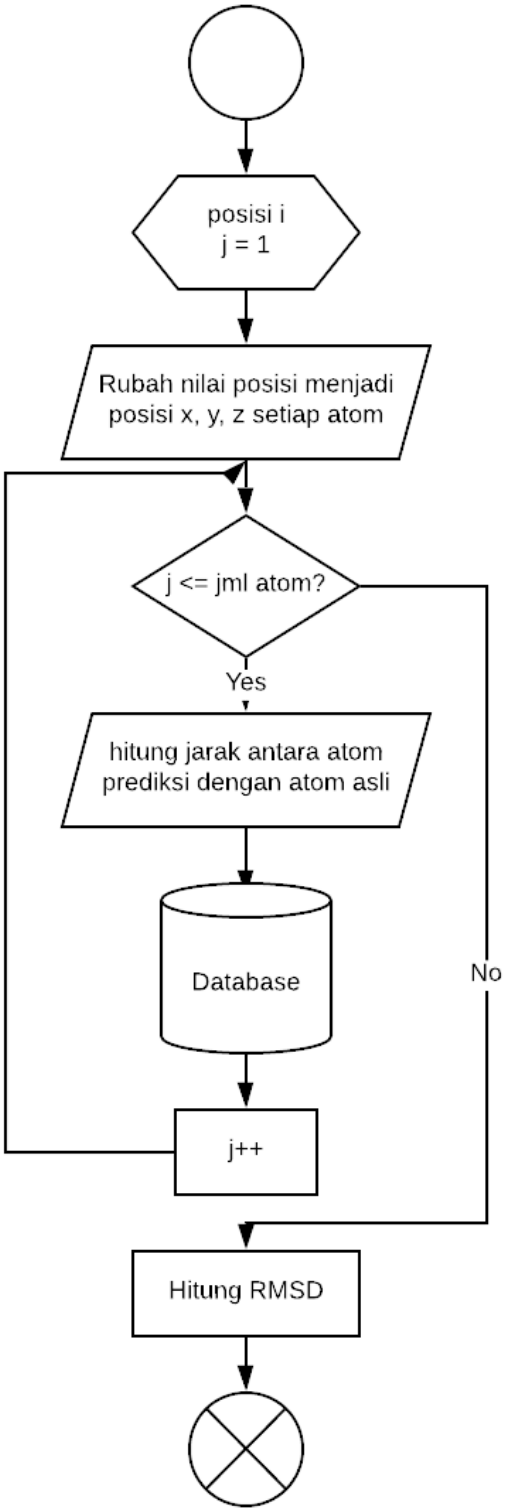
y_j : posisi y atom j prediksi

z_j : posisi z atom j prediksi

x_r : posisi x atom j pada *native ligand*

y_r : posisi y atom j pada *native ligand*

z_r : posisi z atom j pada *native ligand*



Gambar 4.28: *Flowchart* Perhitungan RMSD

4.4 Penyusunan Algoritma Lengkap

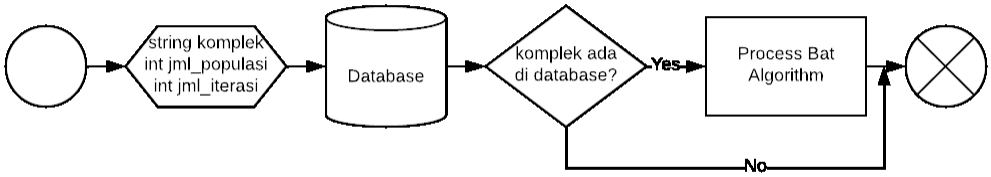
Gambar 4.29 menunjukkan alur program yang dibentuk. Program yang dibuat dimulai dengan meminta struktur apa yang akan dipakai dan juga parameter awal. Selanjutnya, akan dicoba untuk mengambil data struktur tersebut dari database yang telah dibentuk sebelumnya. Data yang akan diambil adalah, data translasi, data torsi, data ligan, dsb. Selanjutnya akan masuk ke *bat algorithm,* dimulai dengan pembentukan populasi awal.

Setelah populasi awal terbentuk,*fitness* dari populasi tersebut akan dihitung. Kemudian, program akan mulai masuk ke dalam *loop* hingga iterasi terakhir terpenuhi. Langkah-langkah yang dilakukan di dalam *loop* dimulai dengan melakukan proses *update*yang telah dijelaskan sebelumnya, kemudian akan dihitung nilai *fitness* untuk populasi yang baru. Selanjutnya dilakukan validasi BA apakah akan *update* akan dilakukan atau tidak, bergantung pada nilai *fitness* yang baru.

Apabila struktur yang dimasukkan memiliki data kristalisasi yang asli, maka akan dilakukan perhitungan RMSD, jika tidak ada data aslinya, akan di*skip*. Setelah *looping* selesai dilakukan, akan ditampikan semua kelelawar yang ada di populasi terakhir. *Source code* algoritma dapat dilihat pada Lampiran G.

Berikut adalah *pseudocode* dari penelitian ini:

Inisialsisasi semua parameter yang digunakan
Input nama struktur
Ambil data dari database untuk struktur yang di*input*kan
Inisialisasi kelelawar
Hitung *fitness* setiap kelelawar
Tampilkan kelelawar pada generasi pertama berserta *fitness* dan RMSD (jka ada)
while*(t<maxIteration)*
    update kelelawar
    validasi kelelawar
    tampilkan *fitness* terbaik dan RMSDnya (jika ada)
    t++
Tampilkan kelelawar pada generasi terakhir berserta *fitness* dan RMSD (jika ada)



Gambar 4.29: *Flowchart* Program Lengkap

BAB V
HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan parameter yang digunakan, *output* program, dan analisis hasil simulasi.

5.1 Parameter

Tabel 5.1 menunjukkan parameter yang digunakan di dalam pengerjaan *molecular docking* dengan BA

Tabel 5.1: Nilai Parameter dalam BA

Parameter	*Value*
Population size	5, 10, 20
Number of iterations	1000
Loudness (A_0)	0.5
Pulse emission (r_0)	0.5
Frequency minimum (f_{min})	0
Frequency maximum (f_{max})	2
α	1
γ	0

Alasan digunakan nilai *loudness* dan α seperti diatas adalah agar kemungkinan terjadi peng*update*an posisi adalah tetap 50%. Karena jika nilai α yang digunakan lebih kecil daripada 1, nilai *loudness* akan berkurang setiap kali terjadi *update*, sehingga untuk jumlah iterasi yang besar, kemungkinan nilai posisi berubah akan menjadi kecil. Begitu juga dengan pasangan *pulse emission* dan γ. Keduanya dipilih agar nilai pemilihan *random walk* akan tetap 50% dari awal hingga akhir. Dengan menggunakan α dan γ sesuai dengan nilai diatas, akan menyebabkan *loudness* dan *pulse emission* menjadi konstan. Dalam BA, nilai tersebut diperbolehkan berupa konstan (X. S. Yang, 2010).

5.2 *Output* Program

Berikut adalah contoh *output* yang dihasilkan dari program yang telah dibentuk. Dengan meminta nama kompleks yang akan digunakan (2cpp, 3ptb, atau SA2014), jumlah posisi yang akan dipakai, dan jumlah iterasi yang dilakukan. Setelah semua parameter yang diminta dimasukkan, program pertama akan menampilkan nilai-nilai yang ada dalam populasi awal posisi, Sebagai contoh, berikut adalah output struktur 3ptb dan SA2014.

```
Masukkan nama kompleks : 3ptb
Masukkan jumlah individu : 20
Masukkan jumlah iterasi : 1000
Koneksi berhasil
Inisialisasi
Populasi pertama
Value : [5.154741650537886, 16.323375165635607, 20.562753539416075, -
0.9795324423346741, 0.8845274962748662, 0.03224436678625753,
61.951741874395935, -0.9005499176672913]
Fitness : 2389.5452
RMSD : 8.371039
Value : [4.493631093640635, 15.079779037635145, 19.7102687240821, -
0.04342388704070399, 0.8197410478999723, -0.40532803919759863,
84.04913335098138, 106.27412599468136]
Fitness : 85314.05
RMSD : 7.511499
Value : [1.6521005539975242, 7.5428588492835695, 17.86571372003185, -
0.9401003351955715, 0.16765662562627082, 0.43273427174155343,
39.61623191976278, 48.273804164089825]
Fitness : 156675.94
RMSD : 7.3788443
```

```
Masukkan nama kompleks : SA2014
Masukkan jumlah individu : 20
Masukkan jumlah iterasi : 1000
Koneksi berhasil
Inisialisasi
Populasi pertama
Value : [15.250218525870537, 5.280574966969986, 53.70697067145804,
0.3253590429939297, 0.610104248216524, 0.030385146575681032, -
60.45826369829315]
Fitness : 537120.94
Value : [0.4673471545353509, 26.721968522414215, 24.289643161245724,
0.4883324148259618, 0.6312164739630943, -0.18247405585340215, -
54.28944048569626]
Fitness : 12.394951
```

Pada struktur 3ptb, yang dimunculkan dalam populasi pertama adalah *value* yang merupakan bentuk solusi keleawar, kemudian nilai *fitness* awal, dan juga RMSDnya. Sedangkan untuk SA2014, karena tidak memiliki nilai kristal awal

yang menjadi acuan RMSD, maka RMSD tidak akan ditampilkan untuk SA2014. Setelah memunculkan populasi awal, selanjutnya program akan menampilkan nilai *fitness* terbaik setiap iterasi beserta RMSD dari solusi tersebut, jika memiliki RMSD. Contoh dibawah secara berurut adalah milik percobaan 3ptb dan SA2014.

```
Fitness Terbaik : 25.660156            RMSD : 10.3398485
Fitness Terbaik : -1.0314249           RMSD : 6.2140284
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
Fitness Terbaik : -1.5201883           RMSD : 5.3999786
```

```
Fitness Terbaik : -0.2576634
Fitness Terbaik : -0.46008334
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
Fitness Terbaik : -0.76562804
```

Fitness akan dimunculkan sebanyak iterasi yang dimasukkan. Setelah program selesai dijalankan, akan ditampilkan semua posisi pada iterasi terakhir. Berikut adalah contoh penampilan generasi terakhir dari posisi. Dengan contoh pertama adalah dari permasalahan 3ptb dan kedua adalah SA2014.

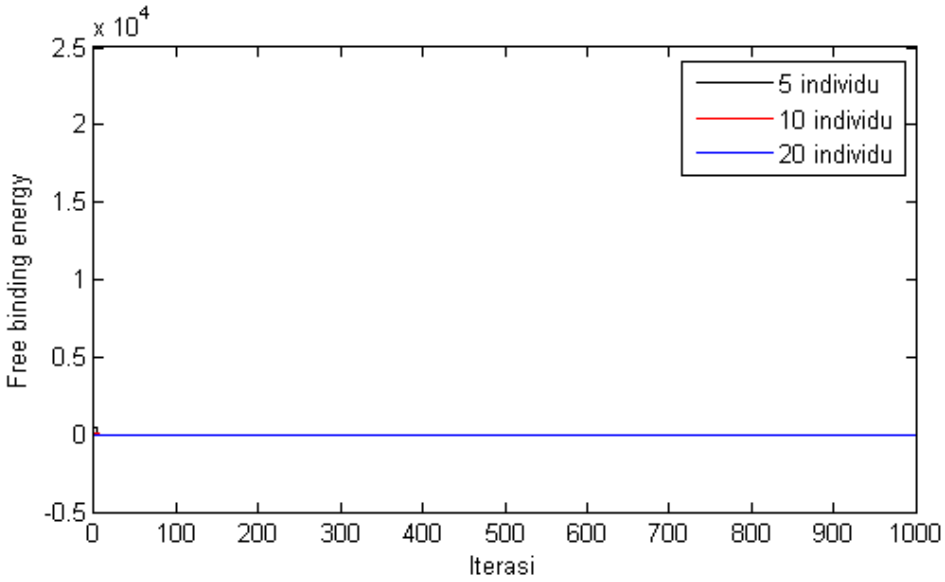
```
Generasi terakhir
Value  :  [-0.4107325723037576,  18.60713983827329,  17.56652092849124,
0.5909598390019231,      -0.07163840742854544,      -0.5500671471940519,
126.76953306855847, 178.27924560570273]
Fitness : -2.3968523
RMSD : 5.3743854
```

```
Value : [-1.9429781276019709, 17.878167843526022, 18.44458333785004,
0.35246450822724995, -0.26824664535514375, 0.8598540689809933, -
98.37632197626677, 72.27479267330102]
Fitness : -2.513617
RMSD : 3.6869576
Value : [-2.6495641000883356, 15.723505078715416, 17.63391361727603, -
0.9944702671489387, 0.20069720928594403, 0.0493763213500269,
32.603144220472984, 175.3287964838028]
Fitness : -1.8980064
RMSD : 2.2847521
```

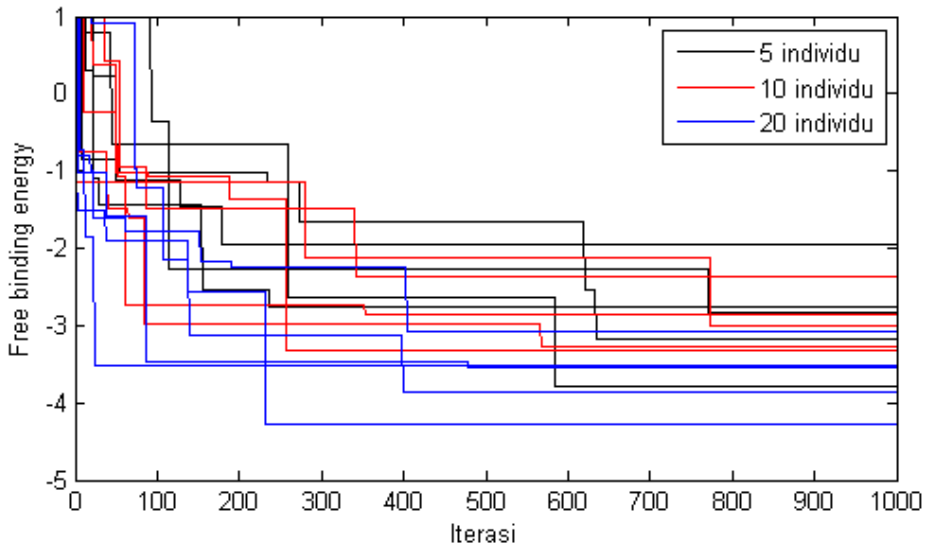
```
Generasi terakhir
Value : [7.932643291146475, -4.79055453074172, 55.839515963136066, -
0.8800508846902478, -0.10835281208799374, 0.9323951649270212,
170.6327915876256]
Fitness : -0.77879095
Value : [7.230228356330059, 26.088491691089416, 52.48350426160457,
0.6798570891699509, 0.43627866669637316, 0.2941364738706669, -
66.18503492216023]
Fitness : -1.9829301
Value : [10.471489089009047, 0.44309368032657126, 35.28896827100216, -
0.8810883725496735, 0.9083786561125999, 0.7022039682606522,
90.88583702949273]
Fitness : -1.3987818
Value : [26.08356469055355, 16.374924043963958, 33.982012441556776,
0.501387329667708, -0.03545121085959102, 0.3730445922006951, -
78.88646933627318]
Fitness : -0.8940432
```

5.3 Validasi

Untuk menilai apakah BA dapat menyelesaikan permasalahan *molecular docking*, kita harus melakukan validasi terlebih dahulu. Validasi dilakukan dengan mencoba menyelesaikan permasalahan *molecular docking* struktur 3ptb dan 2cpp. Percobaan dilakukan sebanyak 5 kali untuk masing-masing ukuran populasi. Gambar 5.1 dan 5.2 adalah perbandingan *fitness* terbaik untuk setiap iterasi dari semua percobaan pada 3ptb. Gambar 5.1 grafik asli perbandingan *fitness,* sedangkan Gambar 5.2 adalah grafik yang sama namun difokuskan pada nilai *fitness* diantara nilai 1 hingga -5. Garis berwarna hitam adalah percobaan dengan menggunakan ukuran populasi 5, garis berwarna merah adalah percobaan dengan menggunakan ukuran populasi 10, dan garis dengan warna biru adalah percobaan dengan menggunakan ukuran populasi 20.



Gambar 5.1: Nilai *Fitness* dari Percobaan Menggunakan 3ptb



Gambar 5.2: Nilai *Fitness* dari percobaan menggunakan 3ptb dengan Fokusan Nilai *Fitness* Diantara 1 Hingga -5

Dari Gambar 5.2 kita dapat melihat bahwa nilai *fitness* terkecil dimiliki oleh percobaan dengan menggunakan ukuran populasi 20. Sedangkan terburuk adalah dengan percobaan dengan menggunakan ukuran populasi 5. Tabel 5.2 menunjukkan *fitness* terbaik di awal dan akhir iterasi dari semua percobaan.

Tabel 5.2 *Fitness* Terbaik 3ptb dan RMSDnya di Awal dan Akhir Iterasi

Populasi	Percobaan ke-	*Fitness* awal	*Fitness* akhir	RMSD awal	RMSD akhir
	1	4.029	-2.838	8.584	2.607
	2	22747.871	-3.185	7.341	3.366
	3	147.291	-2.768	7.419	3.558
	4	487.492	-1.939	10.049	3.073
5	5	27.949	-3.778	8.756	1.185
	1	70.919	-2.852	9.328	2.296
	2	8.779	-3.275	10.932	2.093
	3	10.567	-2.364	11.360	3.824
	4	2.353	-2.992	8.801	3.055
10	5	2.665	-3.332	7.477	2.317
	1	2.029	-3.865	8.209	1.196
	2	5.935	-3.551	10.456	1.500
	3	3.696	-3.088	8.427	2.718
	4	7.599	-3.510	8.656	1.378
20	5	25.660	-4.283	10.340	1.240

Populasi awal posisi dan populasi akhir posisi dalam percobaan yang menghasilkan *fitness* terbaik dapat dilihat pada Tabel 5.3 dan Tabel 5.4. Dalam Tabel 5.3 dan Tabel 5.4 berisikan nilai solusi yang dinotasikan oleh posisi, dimulai dari x, y, dan z yang merupakan posisi translasi pusat ligan, diikuti dengan x_q, y_q, z_q, dan w_q yang merupakan posisi *quaternion* pusat ligan, dan terakhir adalah nilai r_1 yang merupakan derajat atom yang dapat diputar. Nilai r hanya 1 nilai dikarenakan 3ptb hanya memiliki 1 *rotateable atom*. Nilai dengan *fitness* terbaik yang dimiliki populasi awal, diwarnai dengan warna kuning pada Tabel 5.3, dimiliki oleh posisi ke-10 dengan nilai *fitness* 25.66 dan RMSD 10.34. Setelah dilakukan iterasi sebanyak parameter *number of iterations* pada Tabel 5.1, didapatkan populasi terakhir yang dapat dilihat pada Tabel 5.4. Semua posisi dalam Tabel 5.4 memiliki nilai *fitness* yang negatif, dengan nilai terbaik dimiliki oleh posisi ke-14, dengan nilai *fitness* -4.283 dan RMSD 1.24.

Tabel 5.3 Populasi Awal untuk Percobaan Terbaik Struktur 3ptb

Posisi ke-	x	y	z	x_q	y_q	z_q	w_q	r_1	$fitness$	$RMSD$
1	5.155	16.323	20.563	-0.980	0.885	0.032	61.952	-0.901	2389.545	8.371
2	4.494	15.080	19.710	-0.043	0.820	-0.405	84.049	106.274	85314.05	7.511
3	1.652	7.543	17.866	-0.940	0.168	0.433	39.616	48.274	156675.9	7.379
4	-2.219	14.718	23.374	0.960	-0.139	0.081	-101.042	-151.600	103.055	6.830
5	-4.722	8.970	17.045	-0.258	-0.293	-0.832	-172.003	-74.175	250077.1	6.864
6	-1.920	7.718	10.188	-0.382	-0.220	0.252	-38.2371	67.351	1844.8	9.671
7	5.089	20.469	11.018	0.247	0.960	0.435	113.993	-106.296	31511.37	11.360
8	2.193	16.362	22.052	-0.366	0.568	-0.710	71.151	129.703	229495.3	6.927
9	-7.950	8.126	14.905	0.013	-0.390	0.142	129.132	117.953	13315.9	10.167
10	-0.377	7.133	12.597	0.627	0.565	-0.747	-179.069	65.549	25.660	10.340
11	-4.862	12.966	23.282	-0.172	-0.770	-0.075	73.567	43.333	126279.4	8.568
12	-5.081	9.650	23.021	-0.599	0.577	-0.912	82.540	87.235	520623.8	8.001
13	4.309	19.493	23.982	0.530	-0.899	0.395	82.094	81.227	165.77	10.703
14	-4.381	14.180	13.224	-0.678	0.650	0.364	39.583	-8.582	276449.6	4.461
15	-2.197	11.554	15.929	0.301	-0.808	0.616	-127.794	30.831	3636.22	4.285
16	3.313	15.236	15.732	0.786	0.514	-0.265	34.401	64.386	270142.7	5.416
17	1.325	13.027	23.529	-0.434	-0.682	0.802	-25.823	173.619	226588.6	8.114
18	0.301	13.729	19.910	-0.207	0.692	-0.871	-33.332	70.675	102220.7	4.472
19	-3.434	15.833	10.089	0.494	-0.554	-0.755	43.180	-178.902	14087.36	6.487
20	-0.638	15.214	13.979	-0.372	-0.736	0.663	-95.973	104.957	147821.8	4.199

Tabel 5.4 Populasi Akhir Percobaan Terbaik untuk Struktur 3ptb

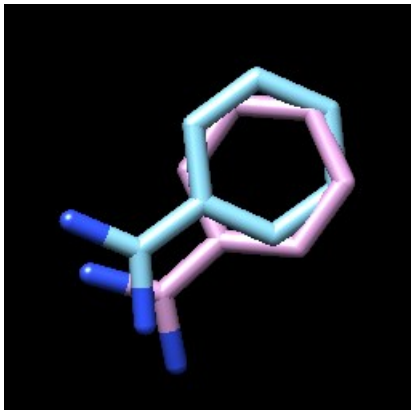
Posisi ke-	x	y	z	x_q	y_q	z_q	w_q	r_1	$fitness$	$RMSD$
1	-0.411	18.607	17.567	0.591	-0.072	-0.550	126.770	178.279	-2.397	5.374
2	-1.943	17.878	18.445	0.352	-0.268	0.8560	-98.376	72.275	-2.514	3.687
3	-2.650	15.724	17.634	-0.994	0.201	0.049	32.603	175.329	-1.898	2.285
4	-2.026	14.766	17.262	-0.440	-0.321	0.214	-155.312	21.496	-3.403	3.412
5	2.817	21.398	14.926	0.347	0.993	-0.704	-138.636	-155.793	-1.643	9.158
6	-0.655	16.372	17.832	0.308	0.946	0.586	-4.422	-173.164	-2.288	2.923
7	-1.221	15.847	17.429	-0.170	0.237	-0.846	-34.892	-120.967	-3.240	2.157
8	-1.732	14.206	16.696	0.773	0.370	0.185	-22.890	-12.203	-4.269	1.003
9	0.560	20.137	16.210	0.999	-0.963	0.287	-88.854	55.942	-1.301	7.536
10	-0.860	15.936	18.270	0.643	-0.052	-0.200	-38.220	66.405	-2.798	3.553
11	-1.407	17.594	19.328	-0.767	0.021	0.325	-158.546	51.305	-2.749	4.759
12	-1.084	17.741	21.380	-0.681	-0.796	-0.303	134.362	105.821	-1.196	6.632
13	-1.266	15.464	17.547	-0.983	0.819	0.496	56.508	5.300	-2.524	2.330
14	-1.172	13.824	16.790	0.671	0.943	0.046	-26.865	-13.762	-4.283	1.240
15	-1.241	17.722	19.912	0.826	0.452	0.891	-86.998	-25.851	-2.037	5.041
16	-1.827	18.664	20.269	0.719	0.441	0.433	164.212	62.703	-1.031	6.214
17	1.468	20.819	18.474	-0.341	-0.445	0.452	-164.127	145.281	-0.666	7.135
18	-1.994	17.357	17.732	0.482	-0.380	-0.355	14.349	-5.718	-2.078	3.490
19	-1.083	17.064	17.629	-0.655	-0.228	0.644	-136.351	-35.103	-2.574	4.391
20	-2.102	13.865	16.780	0.124	0.612	0.003	-4.311	-16.259	-3.955	1.085

Percobaan ini mendapatkan nilai negatif untuk energi bebasnya dan nilai RMSD yang kecil (< 2Å), dengan solusi terbaiknya ada pada posisi ke-14 pada Tabel 5.4. Sehingga, hasil percobaan untuk *molecular docking* struktur 3ptb dapat dikatakan valid sebagai solusi.

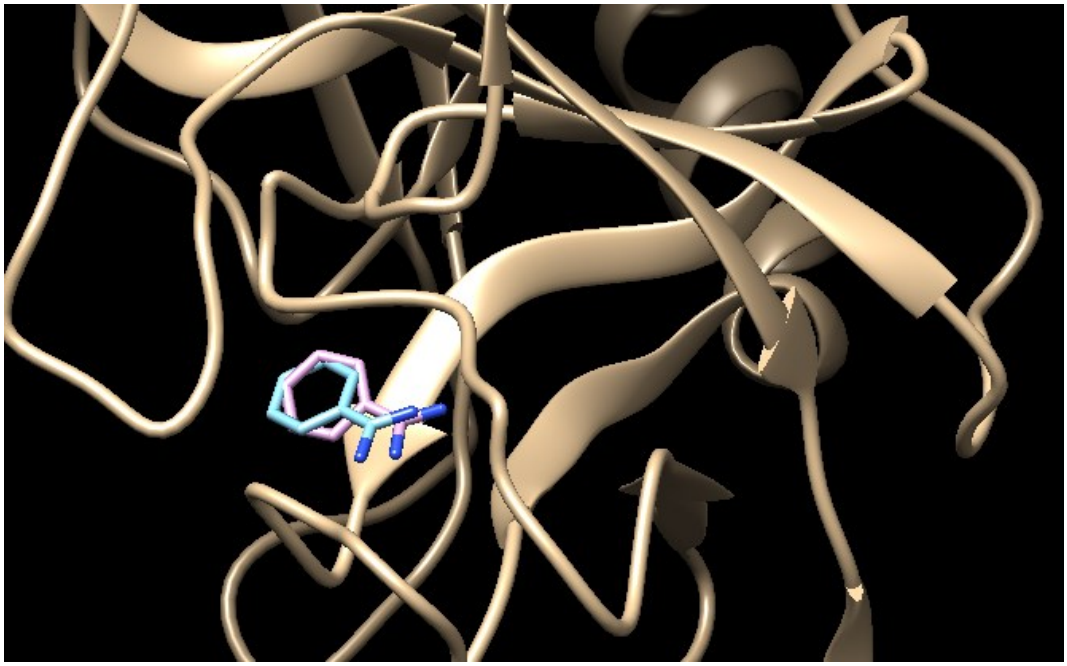
Selanjutnya, solusi yang telah didapatkan diubah menjadi koordinat x, y, dan z. Selanjutnya dimasukkan kembali ke *file* pdb yang asli. Tabel 5.5 menampilkan nilai x, y, dan z untuk setiap atom, pada ligan *native* dan juga prediksi penempatan ligan. Dengan posisi atom yang telah didapatkan, dikembalikan lagi kedalam *file* 3ptb dan ditampilkan menggunakan Chimera. Perbandingan prediksi *docking* 3ptb dengan keadaaan asli dapat dilihat pada Gambar 5.3 dan Gambar 5.4. Gambar 5.3 menunjukkan perbandingan antara kedua ligan saja, sedangkan Gambar 5.4 menunjukkan penempatan kedua ligan pada proteinnya. Ligan dengan warna biru adalah *native ligand* dan ligan berwarna ungu adalah ligan hasil prediksi.

Tabel 5.5: Nilai Posisi Prediksi Penempatan Ligan 3ptb dengan BA

Atom	Posisi x		Posisi y		Posisi z	
	Native	Prediksi	*Native*	Prediksi	*Native*	Prediksi
C	-1.853	-1.140	14.311	13.747	16.658	16.725
C	-2.107	-1.320	15.653	15.058	16.758	16.373
C	-1.774	-1.393	16.341	16.044	17.932	17.367
C	-1.175	-1.276	15.662	15.694	19.005	18.721
C	-0.914	-1.087	14.295	14.353	18.885	19.064
C	-1.257	-1.017	13.634	13.393	17.708	18.057
C	-2.193	-1.186	13.627	12.999	15.496	15.553
N	-2.797	-1.741	14.235	13.475	14.491	14.453
N	-1.762	-0.521	12.391	11.858	15.309	15.473

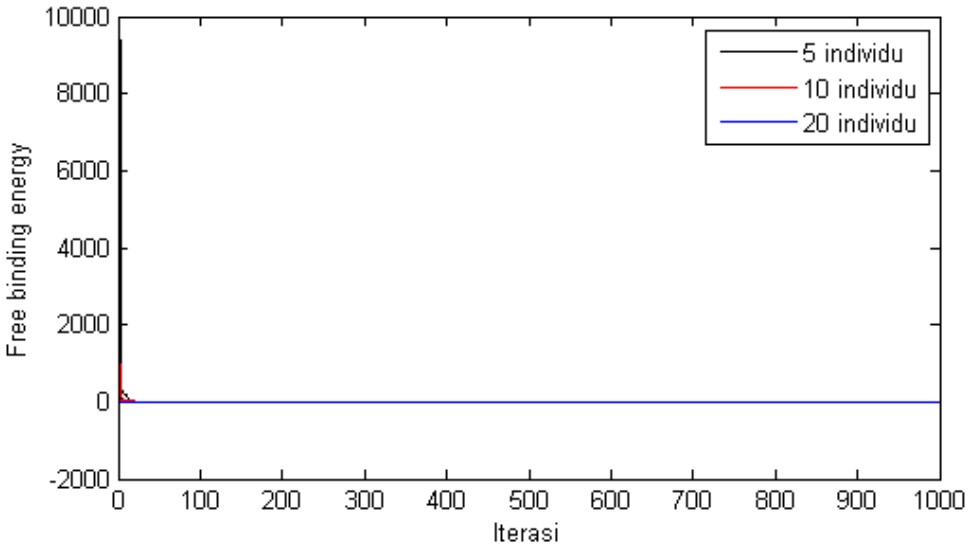


Gambar 5.3: Perbandingan Ligan Prediksi dan *Native Ligand* 3ptb

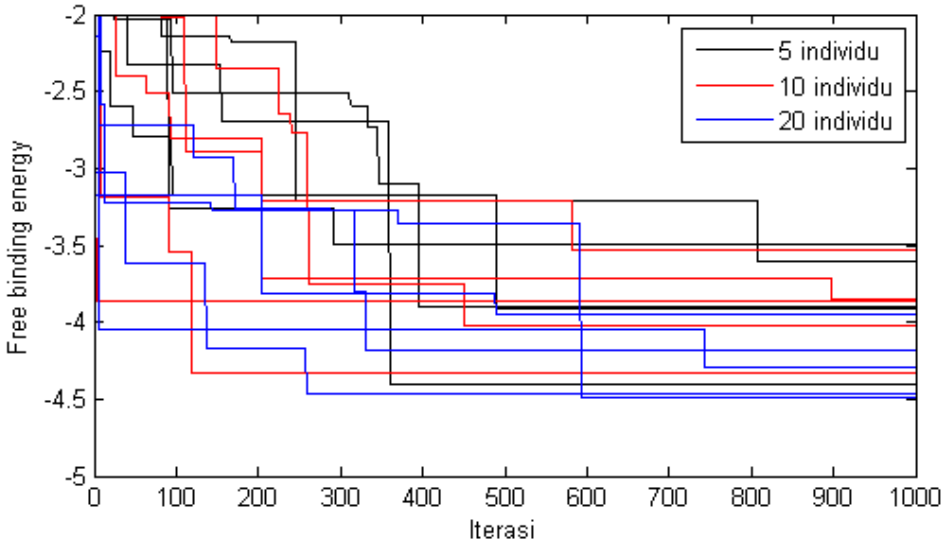


Gambar 5.4: Perbandingan Peletakaan Ligan Prediksi dengan *Native Ligand* untuk 3ptb

Selanjutnya adalah menggunakan struktur 2cpp sebagai validasi. Sama seperti percobaan sebelumnya, percobaan menggunakan ukuran populasi 5, 10, dan 20, yang dilakukan masing-masing sebanyak 5 kali. Gambar 5.5 grafik asli perbandingan *fitness,* sedangkan Gambar 5.6 adalah grafik yang sama namun difokuskan pada nilai *fitness* diantara nilai -2 hingga -5. Garis berwarna hitam adalah percobaan dengan menggunakan ukuran populasi 5, garis berwarna merah adalah percobaan dengan menggunakan ukuran populasi 10, dan garis dengan warna biru adalah percobaan dengan menggunakan ukuran populasi 20.



Gambar 5.4: Nilai *Fitness* dari Percobaan Menggunakan 2cpp



Gambar 5.6: Nilai *Fitness* dari Percobaan Menggunakan 2cpp Dengan Fokusan Nilai *Fitness* Diantara -2 Hingga -5

Dari Gambar 5.6 kita dapat melihat bahwa nilai *fitness* terkecil dimiliki oleh percobaan dengan menggunakan ukuran populasi 20. Sedangkan terburuk adalah dengan percobaan dengan menggunakan ukuran populasi 5. Tabel 5.6 menunjukkan *fitness* terbaik di awal dan akhir iterasi dari semua percobaan.

Tabel 5.6 *Fitness* terbaik 2cpp dan RMSDnya di Awal dan Akhir Iterasi

Populasi	Percobaan ke-	*Fitness* awal	*Fitness* akhir	RMSD awal	RMSD akhir
	1	958.381	-3.896	8.103	0.708
	2	8321.605	-3.500	9.687	6.870
	3	2.967	-4.401	1.622	0.732
	4	3.523	-3.913	5.829	6.020
5	5	9780.736	-3.606	3.417	1.089
	1	1438.432	-4.022	8.456	0.587
	2	101.840	-3.867	9.293	6.930
	3	0.727	-4.330	1.974	0.393
	4	31.405	-3.534	5.869	1.381
10	5	30.741	-3.851	9.440	1.196
	1	2.909	-4.292	2.295	1.061
	2	70.619	-4.178	4.309	0.653
	3	-2.975	-4.470	7.671	1.841
	4	3.036	-3.949	5.742	6.758
20	5	8.460	-4.491	7.158	0.488

Populasi awal posisi dan populasi akhir posisi dalam percobaan yang menghasilkan *fitness* terbaik dapat dilihat pada Tabel 5.7 dan Tabel 5.8. Tabel 5.7 berisikan populasi awal dan Tabel 5.8 berisikan populasi akhir posisi. Dalam Tabel tersebut, tidak ditemukan nilai r dikarenakan 2cpp tidak memiliki *rotateable atom*. Nilai dengan *fitness* terbaik yang dimiliki populasi awal, diwarnai dengan warna kuning pada Tabel 5.7, dimiliki oleh posisi ke-4 dengan nilai *fitness* 2.909 dan RMSD 2.295. Setelah dilakukan iterasi sebanyak parameter *number of iterations* pada Tabel 5.1, didapatkan populasi terakhir yang dapat dilihat pada Tabel 5.8. Semua posisi dalam Tabel 5.8 memiliki nilai *fitness* yang negatif, dengan nilai terbaik dimiliki oleh posisi ke-20, dengan nilai *fitness* -4.491 dan RMSD 0.488.

Tabel 5.7 Populasi Awal Percobaan Terbaik untuk Struktur 2cpp

Posisi ke-	x	y	z	x_q	y_q	z_q	w_q	$fitness$	$RMSD$
1	50.405	39.680	7.928	0.189	-0.759	0.151	-6.841	91553.625	8.679
2	53.208	37.447	7.483	0.450	0.236	-0.631	171.203	5592.369	12.446
3	41.588	51.105	16.632	0.559	-0.211	0.741	28.030	135004.02	8.966
4	46.024	42.611	12.751	0.367	-0.590	0.426	-31.241	2.909	2.295
5	50.434	49.976	15.605	-0.097	-0.150	0.324	-150.631	253723.28	8.038
6	40.087	39.672	17.830	0.678	0.605	-0.263	-122.189	212979.44	8.830
7	49.552	47.416	16.140	0.114	-0.483	-0.100	167.246	58649.965	6.251
8	45.314	40.283	9.058	-0.484	-0.122	-0.797	-171.489	1081.580	7.477
9	39.813	48.768	8.110	0.341	-0.385	-0.403	-8.064	1143.758	10.127
10	53.049	39.192	8.503	-0.966	-0.938	0.089	75.968	7868.642	10.698
11	49.913	37.579	19.932	0.885	0.374	-0.653	-177.875	247226.08	10.512
12	44.268	49.726	15.390	-0.743	-0.803	-0.891	-125.857	113272.34	6.678
13	40.039	42.880	14.097	-0.226	-0.461	-0.900	130.372	171498.89	6.800
14	39.154	44.230	12.048	-0.661	-0.388	0.887	24.732	231894.38	7.544
15	49.260	47.156	7.479	-0.967	-0.198	0.214	-82.809	467.628	7.875
16	39.189	41.056	21.721	0.629	-0.866	-0.441	136.114	253.299	11.491
17	53.015	51.342	8.196	-0.463	0.247	0.965	52.093	20.267	11.412
18	42.671	39.187	7.425	-0.728	0.159	0.581	-92.114	99699.2	8.852
19	45.647	42.856	14.828	-0.873	0.854	-0.603	-114.703	21.126	3.144
20	46.207	44.090	20.633	-0.536	-0.503	0.709	86.320	5443.992	6.642

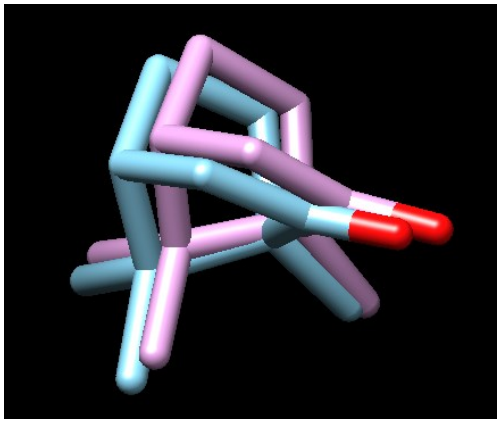
Tabel 5.8 Populasi Akhir Percobaan Terbaik untuk Struktur 2cpp

Posisi ke-	x	y	z	x_q	y_q	z_q	w_q	$fitness$	$RMSD$
1	46.406	43.449	14.209	0.006	0.210	0.205	0.229	-4.022	0.587
2	44.892	42.518	11.984	-0.392	-0.273	-0.572	-10.838	-2.983	3.061
3	48.305	44.237	9.340	0.798	-0.864	-0.492	133.047	-3.125	5.473
4	46.598	43.129	10.224	0.454	0.116	0.332	12.569	-2.984	4.065
5	46.857	44.398	13.104	-0.187	-0.198	0.632	-30.596	-2.819	1.565
6	46.176	44.100	14.991	0.108	0.393	0.679	-33.789	-4.047	1.199
7	45.984	44.981	12.208	0.308	-0.133	0.593	22.774	-2.730	2.327
8	45.660	43.0443	10.923	-0.711	0.305	0.322	8.589	-2.805	3.563
9	46.522	43.567	15.292	0.686	-0.295	-0.992	20.183	-3.418	1.338
10	46.319	49.100	9.631	-0.524	0.748	0.845	155.940	-2.372	6.664
11	45.927	43.116	10.442	0.633	0.825	-0.526	-121.878	-2.883	4.777
12	46.969	43.895	13.675	0.300	-0.561	0.284	22.530	-2.146	1.088
13	46.071	43.780	9.893	-0.601	0.739	0.855	108.078	-2.971	4.775
14	46.253	44.426	12.624	-0.115	-0.653	-0.904	150.974	-2.746	3.338
15	44.587	41.585	9.694	-0.920	-0.218	-0.006	-29.299	-2.679	5.288
16	44.645	48.929	9.790	0.322	0.373	-0.680	123.317	-3.268	7.115
17	46.398	42.458	9.519	0.333	0.417	-0.067	-120.012	-3.155	5.672
18	47.358	44.728	10.394	-0.688	0.038	0.575	99.423	-2.542	4.506
19	46.230	44.500	14.166	0.762	-0.518	-0.858	-71.755	-3.816	1.728
20	46.166	44.315	14.484	0.722	0.885	-0.399	6.940	-4.491	0.488

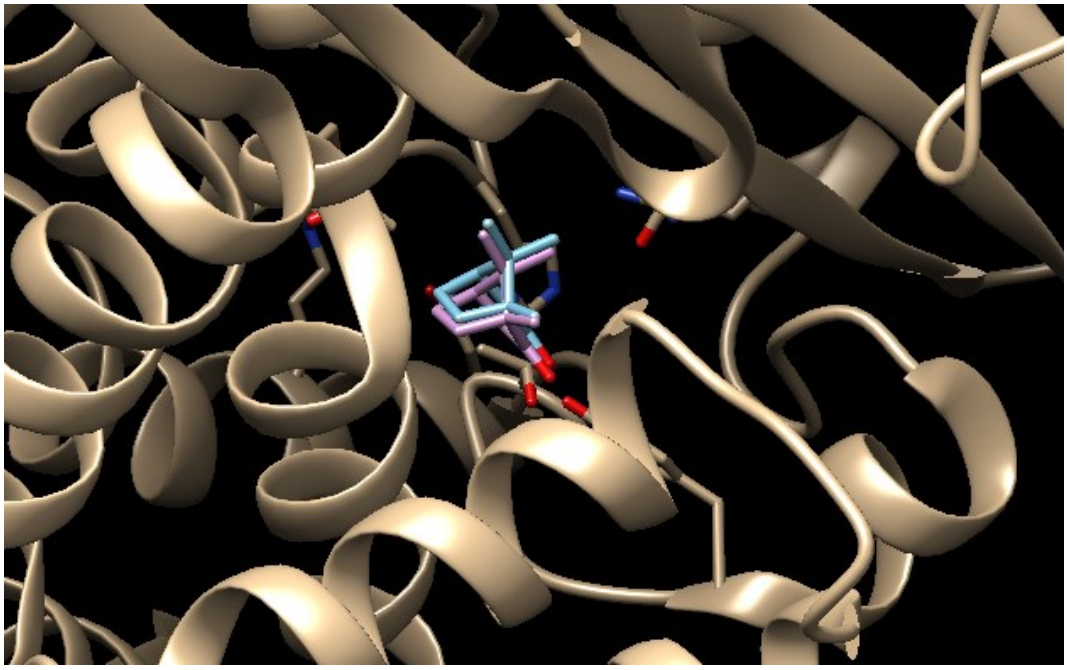
Sama dengan sebelumnya, dari nilai solusi terbaik yang dimiliki, diubah menjadi bentuk posisi x, y, dan z, yang nilainya dapat dilihat pada Tabel 5.9. Selanjutnya akan ditampilkan gambarnya dan dibandingkan dengan *real*nya, yang dapat dilihat pada Gambar 5.7 dan Gambar 5.8. Gambar 5.7 menampilkan perbandingan antara prediksi ligan dengan *native ligand* pada 2cpp, sedangkan Gambar 5.8 menunjukkan peletakan kedua ligan pada proteinnya. Ligan dengan warna biru adalah *native ligand* dan ligan berwarna ungu adalah prediksi penempatan ligan menggunakan BA untuk struktur 2cpp.

Tabel 5.9: Nilai Posisi Prediksi Penempatan Ligan 2cpp dengan BA

Atom	Posisi x		Posisi y		Posisi z	
	Native	Prediksi	*Native*	Prediksi	*Native*	Prediksi
C	46.508	46.434	44.528	44.778	14.647	14.913
C	44.913	44.838	44.451	44.764	14.586	14.990
O	44.197	44.213	44.497	44.763	15.571	16.037
C	44.562	44.353	44.319	44.756	13.094	13.530
C	45.987	45.719	44.434	44.867	12.432	12.753
C	46.538	46.342	45.892	46.290	12.569	12.942
C	46.856	46.795	45.977	46.250	14.100	14.440
C	46.875	46.652	43.585	43.916	13.416	13.591
C	46.482	46.208	42.074	42.413	13.565	13.668
C	48.396	48.130	43.529	43.832	13.036	13.073
C	47.083	47.113	44.219	44.347	16.031	16.214



Gambar 5.7: Perbandingan Ligan Prediksi dengan *Native Ligand* 2cpp



Gambar 5.8: Perbandingan Peletakaan Ligan Prediksi dengan *Native Ligand* untuk 2cpp

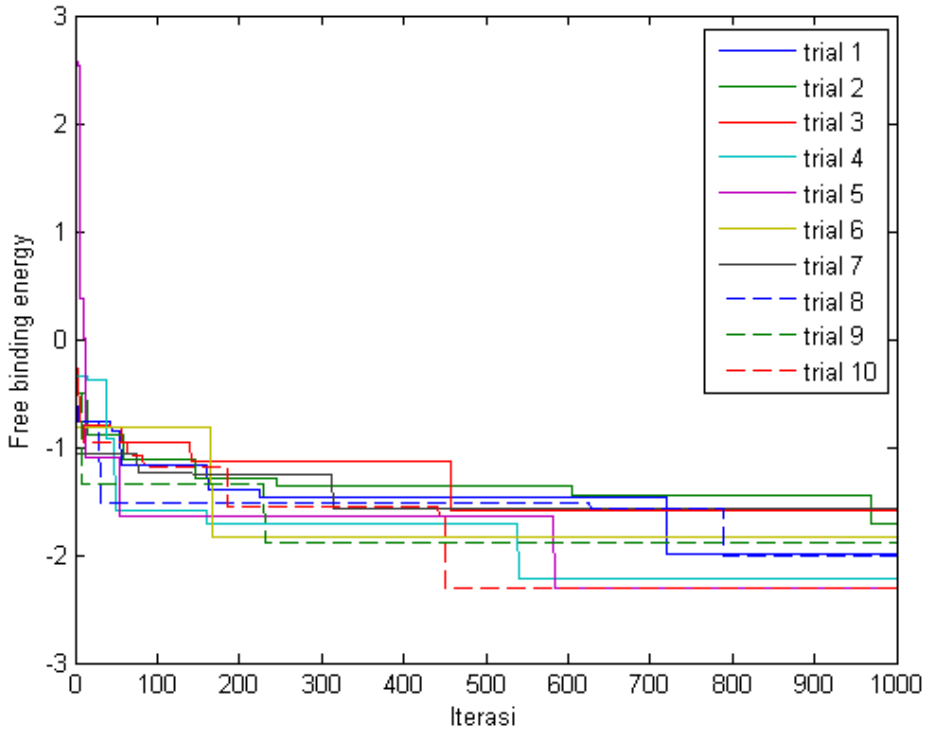
5.4 Prediksi

Dengan menggunakan parameter dan data yang telah disimpan sebelumnya, prediksi penempatan ligan SA2014 dilakukan. Percobaan dilakukan sebanyak 10 kali, dimana populasi awal posisi dapat dilihat pada Lampiran I dan posisi akhir dapat dilihat pada Lampiran J. Tabel 5.7 adalah posisi terbaik disetiap percobaan.

Tabel 5.10: Nilai Fitness Terbaik Prediksi Penempatan Ligan SA2014

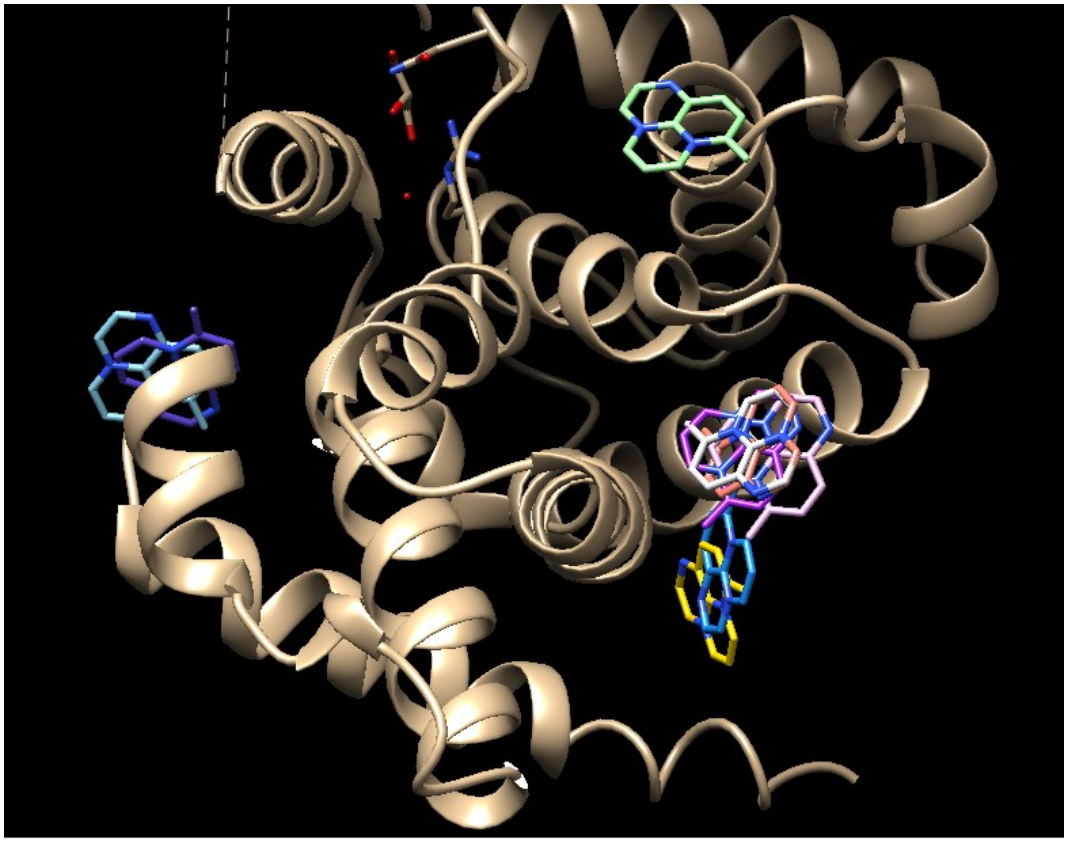
Percobaan Ke-	Posisi Terbaik	*Fitness*
1	Posisi ke-2	-1.9829301
2	Posisi ke-2	-1.7056981
3	Posisi ke-12	-1.5934348
4	Posisi ke-9	-2.2165177
5	Posisi ke-6	-2.3009868
6	Posisi ke-19	-1.8400162
7	Posisi ke-13	-1.5771536
8	Posisi ke-18	-2.0153031
9	Posisi ke-4	-1.8877426
10	Posisi ke-1	-2.2996628

Gambar 5.9 menunjukkan nilai *fitness* untuk setiap percobaan di setiap iterasinya.

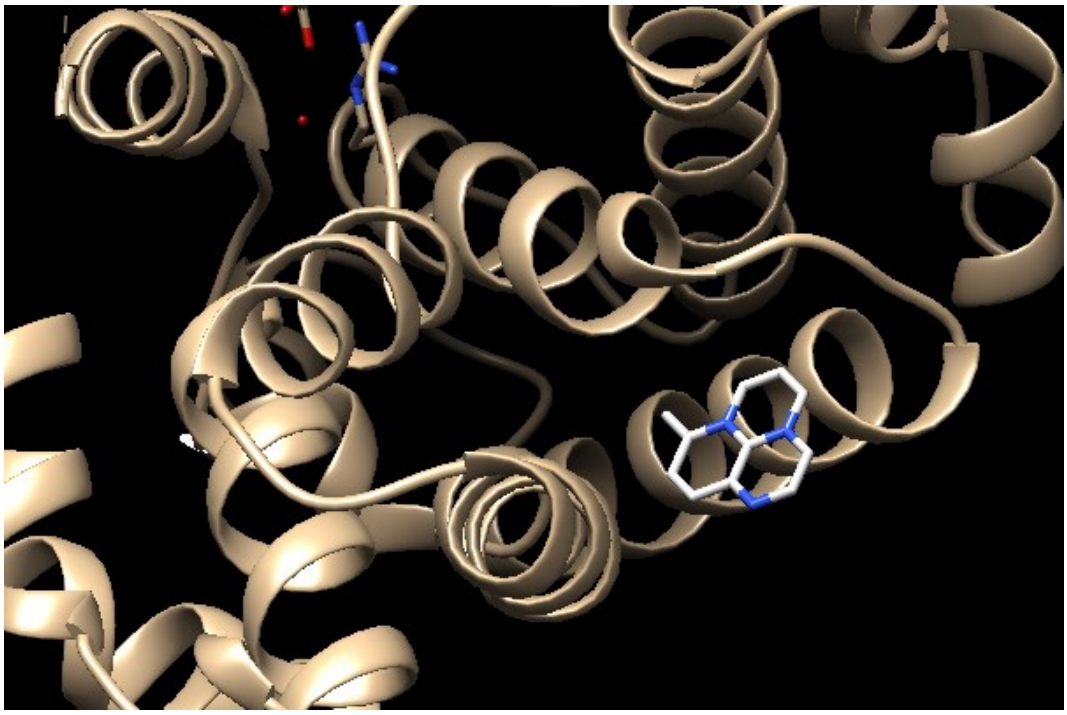


Gambar 5.9: Nilai *Fitness* dari Percobaan Menggunakan SA2014

Dengan posisi terbaik disetiap percobaannya, dikembalikan kedalam bentuk posisi x, y, dan z untuk setiap atomnya. Lampiran K menunjukkan nilai posisi x, y, dan z. Dengan posisi yang didapatkan, akan ditampilkan lokasi dari konformasi ligan SA2014 terhadap cyclin D1. Gambar 5.10 menunjukkan prediksi peletakan ligan SA2014 pada cylcin D1 dengan menggunakan BA. Secara berurutan, warna ligan dari percobaan 1 hingga 10 berwarna: biru langit, violet muda, hijau, blewah, abu-abu, magenta, kuning, biru muda, ungu, dan violet tua. Dengan ligan terbaik adalah yang memiliki warna abu-abu, yang dapat dilihat pada Gambar 5.11.



Gambar 5.10: *Molecuar Docking* SA2014 Terhadap Cyclin D1



Gambar 5.11: *Molecuar Docking* SA2014 Terhadap Cyclin D1Dengan *Fitness* Terbaik

5.5 Diskusi

Molecular docking atas struktur 3ptb dan 2cpp menunjukkan nilai energi yang negatif dan RMSD yang kecil. Dengan begitu, dapat disimpulkan bahwa prediksi yang dilakukan sudah cukup akurat.Terdapat naik turun nilai RMSD yang didapatkan dari kedua validasi tersebut. Semakin kecil nilai RMSD yang didapatkan, maka prediksi yang dilakukan akan semakin dekat dengan kejadian asli. Berdasarkan Tabel 5.2 dan 5.6, nilai RMSD tidak berbanding lurus dengan nilai energi bebasnya.

Gambar 5.3 dan 5.7 menunjukkan bahwa prediksi menggunakan BA sudah mendekati konformasi *native ligand*nya, hanya saja masih berbeda sedikit. Untuk percobaan dengan SA2014, ligan ke-2, ke-4, ke-5, ke-6, ke-7, ke-8, dan ke-10 memiliki lokasi ligan yang menyerupai, begitu juga dengan konformasi ke-1 dan ke-9. Sedangkan ligan ke-3 memiliki konformasi yang berbeda dengan yang lainnya. Dengan ligan ke-5 yang memiliki nilai *fitness* terbesar, maka prediksi konformasi ligan SA2014 dengan cyclin D1 diprediksi akan seperti konformasi ligan ke-5.

Dari Gambar 5.2, 5.6, dan 5.9, kita dapat melihat bahwa nilai energi bebas jarang berubah. Hal ini dikarenakan BA dimungkinan memiliki kekurangan untuk menyelesaikan permasalahan *molecular docking*. BA tidak memiliki langkah seperti mutasi pada algoritma genetika, yang menyebabkan BA mendapatkan solusi yang baru (tidak berhubungan dengan solusi lama) hanya melalui *random walk*. Meskipun telah ditambahkan aturan dimana suatu nilai *random* akan dimasukkan apabila nilai saat itu keluar dari batas, percobaan ini masih jarang mendapatkan nilai baru tersebut. Alasan lain kenapa jarang ada perubahan adalah dikarenakan BA tidak akan menggunakan solusi baru jika nilai *fitness* yang dimiliki lebih besar dibanding dengan nilai *fitness* dari solusi lama. Oleh karena itu, nilai yang didapatkan BA jarang berubah.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasar pembahasan analisis hasil yang telah dipaparkan, dapat diambil beberapa kesimpulan yaitu:

1. *Bat algorithm* dapat disesuaikan untuk menyelesaikan permasalahan *molecular docking*. Dari hasil validasi yang dilakukan, BA dapat digunakan sebagai salah satu algoritma untuk menyelesaikan permasalahan *molecular docking*dikarenakan dari hasil *molecular docking* struktur 3ptb dan 2cpp mendapatkan nilai energi bebas yang negatif dan nilai RMSD yang lebih kecil dari 2Å.
2. Dengan percobaan yang dilakukan pada 3ptb dan 2cpp menggunakan ukuran populasi 5, 10, dan 20. Ukuran populasi yang terbaik adalah 20, oleh karena itu dalam prediksi lokasi SA2014, digunakan ukuran populasi 20.
3. Hasil prediksi lokasi ligan SA2014 di dalam *docking* antara SA2014 terhadap cyclin D1 dengan menggunakan BA adalah [8.105923452343326, -1.348759472041527, 32.91612312919135, -0.7948815142662109, -0.1293564151513376, 0.2312338111539906, -103.51123999121]. Dikarenakan SA2014 tidak mimiliki titik torsi, sehingga solusinya hanyalah nilai translasi dan kuaternion saja.
4. Hasil yang didapatkan menggunakan BA cukup monoton dikarenakan BA tidak memiliki suatu langkah yang dapat memunculkan nilai *random*dengan mudah.

6.2 Saran

Saran yang diberikan oleh penulis untuk penelitian selanjutnya yaitu:

1. Mengubah ukuran *Grid Box*, ukuran *Grid Box* yang diberikan belum mencukupi ke semua lokasi yang mungkin menjadi tempat *docking*. Namun semakin besar *Grid Box*, akan menyebabkan waktu komputasi menjadi lebih lama.

2. Mencoba dengan berbagai parameter lainnya, dalam pengerjaan ini, hanya digunakan satu nilai untuk setiap parameter yang ada, selain ukuran populasi. Dan mencoba untuk ukuran populasi lebih dari 20, namun semakin besar jumlah ind
3. Menggabungkan BA dengan metode optimasi lain agar dapat menghasilkan nilai *random* lebih mudah.

DAFTAR PUSTAKA

Ballester, P. J., & Mitchell, J. B. O. (2010). A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. *Bioinformatics*, *26*(9), 1169–1175.

Branden, C. I. (1999). *Introduction to Protein Structure*. Taylor & Francis Group.

Cell Biology and Cancer. (2001). *Counseling about cancer: strategies for genetic counseling*.

ChemAxon. (2019). Marvin. Retrieved July 17, 2019, from https://chemaxon.com/products/marvin

Day, P. J., Cleasby, A., Tickle, I. J., O'Reilly, M., Coyle, J. E., Holding, F. P., Jhoti, H. (2009). Crystal structure of human CDK4 in complex with a D-type cyclin. *Proceedings of the National Academy of Sciences, 106*(11), 4166–4170.

de Magalhães, C. S., Barbosa, H. J. C., & Dardenne, L. E. (2004). A genetic algorithm for the ligand-protein docking problem. *Genetics and Molecular Biology*, *27*(4), 605–610.

Diandana, R. (2009). *Panduan Lengkap Mengenal Kanker* (1st ed.). Yogyakarta: Mirza Media Pustaka.

Ferreira, L. G., Dos Santos, R. N., Oliva, G., & Andricopulo, A. D. (2015). *Molecular docking and structure-based drug design strategies*. *Molecules*, *20*, 13384–13421

Fu, M., Wang, C., Li, Z., Sakamaki, T., & Pestell, R. G. (2004). Minireview: Cyclin D1: Normal and abnormal functions. *Endocrinology*, *145*(12), 5439–5447.

Gane, P. J., & Dean, P. M. (2000). Recent advances in structure-based rational drug design. *Current Opinion in Structural Biology*, *10*(4), 401–404.

Gapsys, V., Michielssens, S., Peters, J. H., de Groot, B. L., & Leonov, H. (2015). Calculation of binding free energies. *Methods in Molecular Biology (Clifton, N.J.),* 173–209.

Jorgensen, W. L. (2004). The Many Roles of Computation in Drug Discovery. *Science*, *303*(5665), 1813–1818.

Kang, H. R. (2006). Three-Dimensional Lookup Table with Interpolation. In *Computational Color Technology*, SPIE, 151–159

Lapenna, S., & Giordano, A. (2009). Cell cycle kinases as therapeutic targets for cancer. *Nature Reviews Clinical Oncology*, *8*(10), 547–556.

Liu, Y., Li, W., & Ma, R. (2012). Particle Swarm Optimization on Flexible Docking. *International Journal of Biomathematics*, *05*(05), 1250044.

López-Camacho, E., García Godoy, M. J., García-Nieto, J., Nebro, A. J., & Aldana-Montes, J. F. (2015). Solving molecular flexible docking problems with metaheuristics: A comparative study. *Applied Soft Computing Journal*, *28*, 379–393.

Mahdiyah, U., Imah, E. M., & Irawan, M. I. (2016). Integrating data selection and extreme learning machine to predict protein-ligand binding site. *Contemporary Engineering Sciences*, *9*, 791–797.

Marquart, M., Walter, J., Deisenhofer, J., Bode, W., & Huber, R. (1983). The geometry of the reactive site and of the peptide groups in trypsin, trypsinogen and its complexes with inhibitors. *Acta Crystallographica Section B Structural Science, 39*(4), 480–490.

Morris, G. M., Ruth, H., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., & Olson, A. J. (2009). Software news and updates AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of Computational Chemistry*, *30*(16), 2785–2791.

Nurhayati, A.P.D, Pratiwi, R., Wahyuono, S., Fadlan, A., & Syamsudin. (2014). Isolation and Identification of Alkaloid Compound of Marine Sponge Cinachyrella sp . (Family Tetillidae). *Journal of Advanced Botany and Zoology*, *2*(1), 1–4.

Nurhayati, Awik Puji Dyah, Santoso, M., Setiawan, E., & Lianingsih, F. (2017). Molecular Docking of Alkaloid Compound SA2014 From Marine Sponges Cinachyrella anomala Towards P53 Protein. *International Journal of Drug Discovery*, *8*(1), 247–249.

Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., & Ferrin, T. E. (2004). UCSF Chimera — A Visualization System for Exploratory Research and Analysis*, Journal of Computational Chemistry, 25*, 1605–1612.

Poulos, T. L., Finzel, B. C., & Howard, A. J. (1987). High-resolution crystal structure of cytochrome P450cam. *Journal of Molecular Biology, 195*(3), 687–700.

Sambariya, D. K., & Prasad, R. (2014). Robust tuning of power system stabilizer for small signal stability enhancement using metaheuristic bat algorithm. *International Journal of Electrical Power and Energy Systems*, *61*, 229–238.

Sarmoko, & Larasati. (2012). Regulasi siklus sel, Cancer Chemoprevention Research Center - Farmasi UGM.

Satyanarayana, A., & Kaldis, P. (2009). Mammalian cell-cycle regulation: Several cdks, numerous cyclins and diverse compensatory mechanisms. *Oncogene*, *28*(33), 2925–2939.

Seyedmahmoudian, M., Soon, T. K., Jamei, E., Thirunavukkarasu, G. S., Horan, B., Mekhilef, S., & Stojcevski, A. (2018). Maximum power point tracking for photovoltaic systems under partial shading conditions using bat algorithm. *Sustainability (Switzerland)*, *10*(5), 1–16. https://doi.org/10.3390/su10051347

Shen, S. N., & Tuszynki, J. A. (2007). *Theory and Mathematical Methods in Bioinformatics,* Springer.

Subianto, H. (2018). Perubahan Sel Normal Menjadi Sel Kanker. Retrieved July 17, from https://iccc.id/uploads/201801/Sel-sel_normal_menjadi_kanker.pdf

Trott, O., & Olson, A. J. (2010). Software news and update AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry, 31,* 455–461.

Tsai, P. W., Pan, J. S., Liao, B. Y., Tsai, M. J., & Istanda, V. (2012). Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems. *Applied Mechanics and Materials*, *148–149*, 134–137.

Tymoczko, John L. Berg, Jeremy Mark Stryer, L. (2002). *Biochemistry. 5th edition*. *Section 26.4Important Derivatives of Cholesterol Include Bile Salts and Steroid Hormones,* Freeman WH and Co., New York.

Vermeulen, K., Berneman, Z. N., & Bockstaele, D. R. Van. (2003). Cell cycle and apoptosis, *Cell Proliferation, 36*, 165–175.

Wikipedia. (2019a). Cyclin D1. Retrieved July 17, 2019, from https://en.wikipedia.org/wiki/Cyclin_D1

Wikipedia. (2019b). Docking (Molecular). Retrieved July 17, from https://en.wikipedia.org/wiki/Docking_(molecular)

Yang, K., Hitomi, M., & Stacey, D. W. (2006). Variations in cyclin D1 levels through the cell cycle determine the proliferative fate of a cell. *Cell Division*, *1*, 1–8.

Yang, X. S. (2010). A new metaheuristic Bat-inspired Algorithm. In *Studies in Computational Intelligence*, *284*, 65–74.

LAMPIRAN A

Source code function untuk membuat populasi awal

```
public double[][] inisialization(int jml_individu, int Ntor, float Xmin,
   float Xmax, float Ymin, float Ymax, float Zmin, float Zmax) {
       double[][] value = new double[jml_individu][Ntor+7];
       for(int i=0; i<jml_individu; i++) {
              value[i][0] = (double)((Xmax-Xmin)*Math.random()+(Xmin));
              value[i][1] = (double)((Ymax-Ymin)*Math.random()+(Ymin));
              value[i][2] = (double)((Zmax-Zmin)*Math.random()+(Zmin));
              value[i][3] = (double)((2)*Math.random()+(-1));
              value[i][4] = (double)((2)*Math.random()+(-1));
              value[i][5] = (double)((2)*Math.random()+(-1));
              for(int j = 6; j<7+Ntor; j++){
                     value[i][j] = (double)((360)*Math.random()+(-180));
              }
       }
       boundary = new double[2*(Ntor+7)];
       boundary[0] = Xmin;
       boundary[1] = Xmax;
       boundary[2] = Ymin;
       boundary[3] = Ymax;
       boundary[4] = Zmin;
       boundary[5] = Zmax;
       boundary[6] = -1; boundary[8] = -1; boundary[10] = -1;
       boundary[7] = 1; boundary[9] = 1; boundary[11] = 1;
       for(int i=6; i<7+Ntor;i++) {
              boundary[2*i]=-180;
              boundary[2*i+1]=180;
       }
       return value;
}
```

LAMPIRAN B

Source code function untuk membuat update populasi

```
public static void update() {
	bestIndex = getBestIndex();
	//updating bat
	for(int i=0; i<number; i++) {
		frequency[i] = fmin + (fmax - fmin) * Math.random();
		for(int j=0; j<length; j++) {
			velocity[i][j] = velocity[i][j] + (value[i][j] -
			  value[bestIndex][j])*frequency[i];
			value[i][j] = value[i][j] + velocity[i][j];
		}
	double averageLoudness = 0;
		for(int i=0; i<number; i++) {
			averageLoudness = averageLoudness +
			  loudness[i]/number;
		}
	//random walk
	for(int i=0; i<number; i++) {
		if(Math.random() > pulse[i]) {
			for(int j=0; j<length; j++) {
			value[i][j] = value[bestIndex][j] + (2 *
			  Math.random() - 1) * (boundary[2*j+1] -
			  boundary[2*j]) * 0.001;
			}
		}
	}
	randomBound();
}

public static int getBestIndex() {
	int index = 0;
	double min = 999999999;
	for(int i=0; i<number; i++) {
		if(min > fitness[i]) {
			min = fitness[i];
			index = i;
		}
	}
	return index;
}
```

LAMPIRAN C

Source code function untuk menghitung *fitness*

```
public static float[] hitung_fitness(float[][] xakhir, float[][] yakhir,
float[][] zakhir) throws SQLException{
      float[] fitness = new float[jml_individu];
       try{
            String url = "jdbc:mysql://localhost/tesis";
            String user = "root";
            String pass = "";
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection(url, user, pass);
            stm = con.createStatement();
            for(int i = 0;i<jml_individu; i++){
                for(int j = 0;j<Natom_ligan; j++){
                     if(xakhir[i][j]<Xmin || xakhir[i][j]>Xmax ||
                        yakhir[i][j]<Ymin || yakhir[i][j]>Ymax ||
                        zakhir[i][j]<Zmin || zakhir[i][j]>Zmax){
                        sx = (float)((xakhir[i][j]-Xmin)/0.375);
                    if(sx<0){
                       x0[i][j] = (float)(Xmin+(Math.ceil(sx)*0.375));
                    }
                    else{
                       x0[i][j] = (float)(Xmin+(Math.floor(sx)*0.375));
                    }
                    x1[i][j] = (float) (x0[i][j]+0.375);
                    sy = (float)((yakhir[i][j]-Ymin)/0.375);
                    if(sy<0){
                       y0[i][j] = (float)(Ymin+(Math.ceil(sy)*0.375));
                    }
                    else{
                       y0[i][j] = (float)(Ymin+(Math.floor(sy)*0.375));
                    }
                    y1[i][j] = (float) (y0[i][j]+0.375);
                    sz = (float)((zakhir[i][j]-Zmin)/0.375);
                    if(sz<0){
                       z0[i][j] = (float)(Zmin+(Math.ceil(sz)*0.375));
                    }
                    else{
                       z0[i][j] = (float)(Zmin+(Math.floor(sz)*0.375));
                    }
                    z1[i][j] = (float) (z0[i][j]+0.375);

                    d000[i][j] = 1;
                    e000[i][j] = 1;
                    q000[i][j] = 1;
                    d100[i][j] = 1;
                    e100[i][j] = 1;
                    q100[i][j] = 1;
                    d010[i][j] = 1;
                    e010[i][j] = 1;
                    q010[i][j] = 1;
                    d001[i][j] = 1;
                    e001[i][j] = 1;
                    q001[i][j] = 1;
```

```
d110[i][j] = 1;
e110[i][j] = 1;
q110[i][j] = 1;
d101[i][j] = 1;
e101[i][j] = 1;
q101[i][j] = 1;
d011[i][j] = 1;
e011[i][j] = 1;
q011[i][j] = 1;
d111[i][j] = 1;
e111[i][j] = 1;
q111[i][j] = 1;
}

else{
ResultSet query30 = stm.executeQuery("SELECT
   `koorx`, `koory`, `koorz` FROM
   `grid_data_"+kompleks+"` USE INDEX (`koorx`,
   `koory`, `koorz`) WHERE
   (`koorx`>="+xakhir[i][j]+"-0.375 AND
   `koorx`<"+xakhir[i][j]+") AND
   (`koory`>="+yakhir[i][j]+"-0.375 AND
   `koory`<"+yakhir[i][j]+") AND
   (`koorz`>="+zakhir[i][j]+"-0.375 AND
   `koorz`<"+zakhir[i][j]+")") ;
while(query30.next()){
   x0[i][j] = query30.getFloat("koorx");
   y0[i][j] = query30.getFloat("koory");
   z0[i][j] = query30.getFloat("koorz");
}
 ResultSet query31 = stm.executeQuery("SELECT
   `koorx`, `koory`, `koorz` FROM
   `grid_data_"+kompleks+"` USE INDEX (`koorx`,
   `koory`, `koorz`) WHERE
   (`koorx`>="+xakhir[i][j]+" AND
   `koorx`<"+xakhir[i][j]+"+0.375) AND
   (`koory`>="+yakhir[i][j]+" AND
   `koory`<"+yakhir[i][j]+"+0.375) AND
   (`koorz`>="+zakhir[i][j]+" AND
   `koorz`<"+zakhir[i][j]+"+0.375)") ;
while(query31.next()){
   x1[i][j] = query31.getFloat("koorx");
   y1[i][j] = query31.getFloat("koory");
   z1[i][j] = query31.getFloat("koorz");
}
ResultSet query22 = stm.executeQuery("SELECT
   `desolvasi`, `electrostatic`,
   `energi"+tipe_atom[j]+"` FROM
   `grid_data_"+kompleks+"` USE INDEX (`koorx`,
   `koory`, `koorz`) WHERE (`koorx`>="+x0[i][j]+"-
   0.05 AND `koorx`<"+x0[i][j]+"+0.05) AND
   (`koory`>="+y0[i][j]+"-0.05 AND
   `koory`<"+y0[i][j]+"+0.05) AND
   (`koorz`>="+z0[i][j]+"-0.05 AND
   `koorz`<"+z0[i][j]+"+0.05)") ;
while(query22.next()){
```

```
                    d000[i][j] = query22.getFloat("desolvasi");
                    e000[i][j] = query22.getFloat("electrostatic");
                    q000[i][j] =
                        query22.getFloat("energi"+tipe_atom[j]);
                }
                ResultSet query23 = stm.executeQuery("SELECT
                    `desolvasi`, `electrostatic`,
                    `energi"+tipe_atom[j]+"` FROM
                    `grid_data_"+kompleks+"` USE INDEX (`koorx`,
                    `koory`, `koorz`) WHERE (`koorx`>="+x1[i][j]+"-
                    0.05 AND `koorx`<"+x1[i][j]+"+0.05) AND
                    (`koory`>="+y0[i][j]+"-0.05 AND
                    `koory`<"+y0[i][j]+"+0.05) AND
                    (`koorz`>="+z0[i][j]+"-0.05 AND
                    `koorz`<"+z0[i][j]+"+0.05)") ;
                while(query23.next()){
                    d100[i][j] = query23.getFloat("desolvasi");
                    e100[i][j] = query23.getFloat("electrostatic");
                    q100[i][j] =
                        query23.getFloat("energi"+tipe_atom[j]);
                }
                 ResultSet query24 = stm.executeQuery("SELECT
                    `desolvasi`, `electrostatic`,
                    `energi"+tipe_atom[j]+"` FROM
                    `grid_data_"+kompleks+"` USE INDEX (`koorx`,
                    `koory`, `koorz`) WHERE (`koorx`>="+x0[i][j]+"-
                    0.05 AND `koorx`<"+x0[i][j]+"+0.05) AND
                    (`koory`>="+y1[i][j]+"-0.05 AND
                    `koory`<"+y1[i][j]+"+0.05) AND
                    (`koorz`>="+z0[i][j]+"-0.05 AND
                    `koorz`<"+z0[i][j]+"+0.05)") ;
                while(query24.next()){
                    d010[i][j] = query24.getFloat("desolvasi");
                    e010[i][j] = query24.getFloat("electrostatic");
                    q010[i][j] =
                        query24.getFloat("energi"+tipe_atom[j]);
                }
                ResultSet query25 = stm.executeQuery("SELECT
                    `desolvasi`, `electrostatic`,
                    `energi"+tipe_atom[j]+"` FROM
                    `grid_data_"+kompleks+"` USE INDEX (`koorx`,
                    `koory`, `koorz`) WHERE (`koorx`>="+x0[i][j]+"-
                    0.05 AND `koorx`<"+x0[i][j]+"+0.05) AND
                    (`koory`>="+y0[i][j]+"-0.05 AND
                    `koory`<"+y0[i][j]+"+0.05) AND
                    (`koorz`>="+z1[i][j]+"-0.05 AND
                    `koorz`<"+z1[i][j]+"+0.05)") ;
                while(query25.next()){
                    d001[i][j] = query25.getFloat("desolvasi");
                    e001[i][j] = query25.getFloat("electrostatic");
                    q001[i][j] =
                        query25.getFloat("energi"+tipe_atom[j]);
                }

                ResultSet query26 = stm.executeQuery("SELECT
`desolvasi`, `electrostatic`, `energi"+tipe_atom[j]+"` FROM
```

```
`grid_data_"+kompleks+"` USE INDEX (`koorx`, `koory`, `koorz`) WHERE
(`koorx`>="+x1[i][j]+"-0.05 AND `koorx`<"+x1[i][j]+"+0.05) AND
(`koory`>="+y1[i][j]+"-0.05 AND `koory`<"+y1[i][j]+"+0.05) AND
(`koorz`>="+z0[i][j]+"-0.05 AND `koorz`<"+z0[i][j]+"+0.05)") ;
                    while(query26.next()){
                        d110[i][j] = query26.getFloat("desolvasi");
                        e110[i][j] = query26.getFloat("electrostatic");
                        q110[i][j] =
query26.getFloat("energi"+tipe_atom[j]);
                    }

                    ResultSet query27 = stm.executeQuery("SELECT
`desolvasi`, `electrostatic`, `energi"+tipe_atom[j]+"` FROM
`grid_data_"+kompleks+"` USE INDEX (`koorx`, `koory`, `koorz`) WHERE
(`koorx`>="+x1[i][j]+"-0.05 AND `koorx`<"+x1[i][j]+"+0.05) AND
(`koory`>="+y0[i][j]+"-0.05 AND `koory`<"+y0[i][j]+"+0.05) AND
(`koorz`>="+z1[i][j]+"-0.05 AND `koorz`<"+z1[i][j]+"+0.05)") ;
                    while(query27.next()){
                        d101[i][j] = query27.getFloat("desolvasi");
                        e101[i][j] = query27.getFloat("electrostatic");
                        q101[i][j] =
query27.getFloat("energi"+tipe_atom[j]);
                    }

                    ResultSet query28 = stm.executeQuery("SELECT
`desolvasi`, `electrostatic`, `energi"+tipe_atom[j]+"` FROM
`grid_data_"+kompleks+"` USE INDEX (`koorx`, `koory`, `koorz`) WHERE
(`koorx`>="+x0[i][j]+"-0.05 AND `koorx`<"+x0[i][j]+"+0.05) AND
(`koory`>="+y1[i][j]+"-0.05 AND `koory`<"+y1[i][j]+"+0.05) AND
(`koorz`>="+z1[i][j]+"-0.05 AND `koorz`<"+z1[i][j]+"+0.05)") ;
                    while(query28.next()){
                        d011[i][j] = query28.getFloat("desolvasi");
                        e011[i][j] = query28.getFloat("electrostatic");
                        q011[i][j] =
query28.getFloat("energi"+tipe_atom[j]);
                    }

                    ResultSet query29 = stm.executeQuery("SELECT
`desolvasi`, `electrostatic`, `energi"+tipe_atom[j]+"` FROM
`grid_data_"+kompleks+"` USE INDEX (`koorx`, `koory`, `koorz`) WHERE
(`koorx`>="+x1[i][j]+"-0.05 AND `koorx`<"+x1[i][j]+"+0.05) AND
(`koory`>="+y1[i][j]+"-0.05 AND `koory`<"+y1[i][j]+"+0.05) AND
(`koorz`>="+z1[i][j]+"-0.05 AND `koorz`<"+z1[i][j]+"+0.05)") ;
                    while(query29.next()){
                        d111[i][j] = query29.getFloat("desolvasi");
                        e111[i][j] = query29.getFloat("electrostatic");
                        q111[i][j] =
query29.getFloat("energi"+tipe_atom[j]);
                    }


                    }
                }
            }
        }
```

```
        catch(Exception e){
            System.err.println("Koneksi gagal " + e.getMessage());
        }
         finally{
                stm.close();
                con.close();

        }

        //System.out.println("db selesai");
        for (int i = 0; i < jml_individu; i++){
            fitness[i] = 0;
            vdw[i] = 0;
            desol[i] = 0;
            elec[i] = 0;
            for(int k = 0; k < Natom_ligan;k++){

                vdw[i] = vdw[i] + triLerp(xakhir[i][k], yakhir[i][k],
zakhir[i][k], q000[i][k], q001[i][k], q010[i][k], q011[i][k],
q100[i][k], q101[i][k], q110[i][k], q111[i][k], x0[i][k], x1[i][k],
y0[i][k], y1[i][k], z0[i][k], z1[i][k]);
                desol[i] = desol[i] +
Math.abs(muatan[k])*triLerp(xakhir[i][k], yakhir[i][k], zakhir[i][k],
d000[i][k], d001[i][k], d010[i][k], d011[i][k], d100[i][k], d101[i][k],
d110[i][k], d111[i][k], x0[i][k], x1[i][k], y0[i][k], y1[i][k],
z0[i][k], z1[i][k]);
                elec[i] = elec[i] + muatan[k]*triLerp(xakhir[i][k],
yakhir[i][k], zakhir[i][k], e000[i][k], e001[i][k], e010[i][k],
e011[i][k], e100[i][k], e101[i][k], e110[i][k], e111[i][k], x0[i][k],
x1[i][k], y0[i][k], y1[i][k], z0[i][k], z1[i][k]);
            }
            fitness[i] = (float) (vdw[i]+desol[i]+elec[i]+ 0.2983*Ntor);
        }
        return fitness;
    }

    public static float lerp(float x, float x1, float x2, float q00,
float q01) {
        if(x2 != x1){
            return ((x2 - x) / (x2 - x1)) * q00 + ((x - x1) / (x2 - x1))
* q01;}
        else{
            return 0;
        }
    }

    public static float biLerp(float x, float y, float q11, float q12,
float q21, float q22, float x1, float x2, float y1, float y2) {
        float r1 = lerp(x, x1, x2, q11, q21);
        float r2 = lerp(x, x1, x2, q12, q22);
        return lerp(y, y1, y2, r1, r2);
    }

    public static float triLerp(float x, float y, float z, float q000,
float q001, float q010, float q011, float q100, float q101, float q110,
```

```
float q111, float x1, float x2, float y1, float y2, float z1, float z2)
{
        float x00 = lerp(x, x1, x2, q000, q100);
        float x10 = lerp(x, x1, x2, q010, q110);
        float x01 = lerp(x, x1, x2, q001, q101);
        float x11 = lerp(x, x1, x2, q011, q111);
        float r0 = lerp(y, y1, y2, x00, x01);
        float r1 = lerp(y, y1, y2, x10, x11);
        return lerp(z, z1, z2, r0, r1);
    }
```

LAMPIRAN D

Source code function untuk *update* BA

```
public static int getBestIndex() {
	int index = 0;
	double min = 999999999;
	for(int i=0; i<number; i++) {
		if(min > fitness[i]) {
			min = fitness[i];
			index = i;
		}
	}
	return index;
}

public static void update() {
	bestIndex = getBestIndex();
	//updating bat
	for(int i=0; i<number; i++) {
		frequency[i] = fmin + (fmax - fmin) * Math.random();
		for(int j=0; j<length; j++) {
			velocity[i][j] = velocity[i][j] + (value[i][j] -
value[bestIndex][j])*frequency[i];
			value[i][j] = value[i][j] + velocity[i][j];
		}
	}
	double averageLoudness = 0;
	for(int i=0; i<number; i++) {
		averageLoudness = averageLoudness + loudness[i]/number;
	}
	//random walk
	for(int i=0; i<number; i++) {
		if(Math.random() > pulse[i]) {
			for(int j=0; j<length; j++) {
				value[i][j] = value[bestIndex][j] + (2 *
Math.random() - 1) * (boundary[2*j+1] - boundary[2*j]) * 0.001;
			}
		}
	}
	randomBound();
}
```

LAMPIRAN E

Source code function untuk membandingkan kelelawar

```
public static void compareBat(int iteration, float[] currFitness) {
	for(int i=0; i<number; i++) {
		if(Math.random() < loudness[i] && currFitness[i] <
		    fitness[i]) {
			loudness[i] = loudness[i] * alpha;
			pulse[i] = pulse0[i] * (1 - Math.exp(-
			    1*gamma*iteration));
			fitness[i] = currFitness[i];
		} else{
			for (int j = 0; j<length; j++) {
				value[i][j] = currentValue[i][j];
			}
		}
	}
}
```

LAMPIRAN F

Source code function untuk menghitung RMSD

```
public static float hitung_rmsd(){
       rmsd = 0;
        x_akhir = new float[Natom_ligan];
        y_akhir = new float[Natom_ligan];
        z_akhir= new float[Natom_ligan];
        xtmp = new float[Natom_ligan];
        ytmp = new float[Natom_ligan];
        ztmp = new float[Natom_ligan];
        double deg = Math.toRadians(gbest[6]);
        double norm = Math.sqrt(Math.pow(gbest[3],2)+Math.pow(gbest[4],
2)+Math.pow(gbest[5],2));
        double w = Math.cos(deg/2);
        double xi = (gbest[3]*Math.sin(deg/2))/norm;
        double yi = (gbest[4]*Math.sin(deg/2))/norm;
        double zi = (gbest[5]*Math.sin(deg/2))/norm;

        for(int n = 0; n<Natom_ligan; n++){
                  xawal[n] = (float) (xrandom[n]-centerx);
                  yawal[n] = (float) (yrandom[n]-centery);
                  zawal[n] = (float) (zrandom[n]-centerz);
              }

              for(int m = 0; m<Natom_ligan; m++){
                  x_akhir[m] = (float)(((1-2*yi*yi-
2*zi*zi)*xawal[m]+(2*xi*yi-
2*w*zi)*yawal[m]+(2*xi*zi+2*w*yi)*zawal[m])+gbest[0]);
                  y_akhir[m] = (float)(((2*xi*yi+2*w*zi)*xawal[m]+(1-
2*xi*xi-2*zi*zi)*yawal[m]+(2*yi*zi-2*w*xi)*zawal[m])+gbest[1]);
                  z_akhir[m] = (float)(((2*xi*zi-
2*w*yi)*xawal[m]+(2*yi*zi+2*w*xi)*yawal[m]+(1-2*xi*xi-
2*yi*yi)*zawal[m])+gbest[2]);

              }
              if(Ntor!=0){
              for(int t = 0; t<Ntor; t++){
                  orix[t] = (float) (oriX[t]-centerx);
                  oriy[t] = (float) (oriY[t]-centery);
                  oriz[t] = (float) (oriZ[t]-centerz);
              }
              for(int j = 0; j<Ntor; j++){
                  ori_x[j] = (float)(((1-2*yi*yi-
2*zi*zi)*orix[j]+(2*xi*yi-
2*w*zi)*oriy[j]+(2*xi*zi+2*w*yi)*oriz[j])+gbest[0]);
                  ori_y[j] = (float)(((2*xi*yi+2*w*zi)*orix[j]+(1-
2*xi*xi-2*zi*zi)*oriy[j]+(2*yi*zi-2*w*xi*oriz[j]))+gbest[1]);
                  ori_z[j] = (float)(((2*xi*zi-
2*w*yi)*orix[j]+(2*yi*zi+2*w*xi)*oriy[j]+(1-2*xi*xi-
2*yi*yi)*oriz[j])+gbest[2]);

                  for(int k = 0; k<jml_atomtorsi[j]; k++){
                      xtmp[atomtorsi[j][k]-1]=x_akhir[atomtorsi[j][k]-1]-
ori_x[j];
```

```
                    ytmp[atomtorsi[j][k]-1]=y_akhir[atomtorsi[j][k]-1]-
ori_y[j];
                    ztmp[atomtorsi[j][k]-1]=z_akhir[atomtorsi[j][k]-1]-
ori_z[j];
                    x_akhir[atomtorsi[j][k]-1]=(float)
(ori_x[j]+xtmp[atomtorsi[j][k]-
1]*(Math.pow(Math.cos(Math.toRadians(gbest[7+j])),2))+ytmp[atomtorsi[j][
k]-
1]*(Math.cos(Math.toRadians(gbest[7+j]))*Math.sin(Math.toRadians(gbest[7
+j]))+Math.cos(Math.toRadians(gbest[7+j]))*Math.pow(Math.sin(Math.toRadi
ans(gbest[7+j])),2))+ ztmp[atomtorsi[j][k]-
1]*(Math.pow(Math.sin(Math.toRadians(gbest[7+j])),2)-
Math.pow(Math.cos(Math.toRadians(gbest[7+j])),2)*Math.sin(Math.toRadians
(gbest[7+j]))));
                    y_akhir[atomtorsi[j][k]-1] = (float)
(ori_y[j]+xtmp[atomtorsi[j][k]-1]*(-
Math.cos(Math.toRadians(gbest[7+j]))*Math.sin(Math.toRadians(gbest[7+j])
))+ ytmp[atomtorsi[j][k]-
1]*(Math.pow(Math.cos(Math.toRadians(gbest[7+j])),2)-
Math.pow(Math.sin(Math.toRadians(gbest[7+j])),3))+ztmp[atomtorsi[j][k]-
1]*(Math.cos(Math.toRadians(gbest[7+j]))*Math.sin(Math.toRadians(gbest[7
+j]))+Math.cos(Math.toRadians(gbest[7+j]))*Math.pow(Math.sin(Math.toRadi
ans(gbest[7+j])),2)));
                    z_akhir[atomtorsi[j][k]-1] = (float)
(ori_z[j]+xtmp[atomtorsi[j][k]-1]*Math.sin(Math.toRadians(gbest[7+j]))-
ytmp[atomtorsi[j][k]-
1]*Math.sin(Math.toRadians(gbest[7+j]))*Math.cos(Math.toRadians(gbest[7+
j]))+ ztmp[atomtorsi[j][k]-
1]*Math.pow(Math.toRadians(Math.cos(gbest[7+j])),2));
                }
            }}
        dis = 0;
        for(int i = 0; i<Natom_ligan; i++){
            dis = (float) (dis+Math.pow(x_akhir[i]-xAwal[i],
2)+Math.pow(y_akhir[i]-yAwal[i], 2)+Math.pow(z_akhir[i]-zAwal[i], 2));
            rmsd = (float) (Math.sqrt(dis/Natom_ligan));
        }
        return rmsd;

    }
```

LAMPIRAN G

Source code function main

```
public static void main(String[] args) throws SQLException {
  input = new Scanner(System.in);
  System.out.print("Masukkan nama kompleks : ");
  kompleks = input.nextLine();
  System.out.print("Masukkan jumlah individu : ");
  jml_individu = input.nextInt();
  System.out.print("Masukkan jumlah iterasi : ");
  iteration = input.nextInt();

  try{
       String url = "jdbc:mysql://localhost/tesis";
       String user = "root";
       String pass = "";
       Class.forName("com.mysql.jdbc.Driver");
       con = DriverManager.getConnection(url, user, pass);
       stm = con.createStatement();
       System.out.println("Koneksi berhasil");
       ResultSet query = stm.executeQuery("SELECT * FROM `kompleks`
           WHERE `nama_kompleks` = '"+kompleks+"'");
       while(query.next()){
          Ntor = query.getInt("ntor");
          Natom_ligan = query.getInt("natom_ligan");
          Natom_reseptor = query.getInt("natom_reseptor");
          Xmin = query.getFloat("xmin");
          Ymin = query.getFloat("ymin");
          Zmin = query.getFloat("zmin");
          Xmax = query.getFloat("xmax");
          Ymax = query.getFloat("ymax");
          Zmax = query.getFloat("zmax");
          centerx = query.getFloat("centerx");
          centery = query.getFloat("centery");
          centerz = query.getFloat("centerz");
       }
       tipe_atom = new String[Natom_ligan];
       xawal = new float[Natom_ligan];
       yawal = new float[Natom_ligan];
       zawal = new float[Natom_ligan];
       xAwal = new float[Natom_ligan];
       yAwal = new float[Natom_ligan];
       zAwal = new float[Natom_ligan];
       muatan = new float[Natom_ligan];
       xakhir = new float[jml_individu][Natom_ligan];
       yakhir = new float[jml_individu][Natom_ligan];
       zakhir = new float[jml_individu][Natom_ligan];
       x_temp = new float[jml_individu][Natom_ligan];
       y_temp = new float[jml_individu][Natom_ligan];
       z_temp = new float[jml_individu][Natom_ligan];
       for(int i = 0; i<Natom_ligan; i++){
           ResultSet query1 = stm.executeQuery("SELECT `tipe_atom`,
`x_atom`, `y_atom`, `z_atom`, `charge` FROM `atom_ligan` WHERE
`n_kompleks` = '"+kompleks+"' AND `no_atom` = "+(i+1));
```

```
            while(query1.next()){
               tipe_atom[i] = query1.getString("tipe_atom");
               xAwal[i] = query1.getFloat("x_atom");
               yAwal[i] = query1.getFloat("y_atom");
               zAwal[i] = query1.getFloat("z_atom");
               muatan[i] = query1.getFloat("charge");
            }
         }

         ResultSet quer = stm.executeQuery("SELECT MAX(`x_atom`),
MAX(`y_atom`), MAX(`z_atom`) FROM `atom_ligan` WHERE
`n_kompleks`='"+kompleks+"'");
         while(quer.next()){
            xligan = quer.getFloat("MAX(`x_atom`)");
            yligan = quer.getFloat("MAX(`y_atom`)");
            zligan = quer.getFloat("MAX(`z_atom`)");
         }

         torsi1 = new int [Ntor];
         torsi2 = new int [Ntor];
         for(int i = 0; i<Ntor; i++){
            ResultSet query1 = stm.executeQuery("SELECT `atom1`, `atom2`
FROM `torsi` WHERE `n_kompleks` = '"+kompleks+"' AND `no_torsi` =
"+(i+1));
             while(query1.next()){
                torsi1[i] = query1.getInt("atom1");
                torsi2[i] = query1.getInt("atom2");
            }
         }

         jml_atomtorsi = new int[Ntor];
         atomtorsi = new int[Ntor][100];
         orix = new float[Ntor];
         oriy = new float[Ntor];
         oriz = new float[Ntor];
         oriX = new float[Ntor];
         oriY = new float[Ntor];
         oriZ = new float[Ntor];
         ori_x = new float[Ntor];
         ori_y = new float[Ntor];
         ori_z = new float[Ntor];

         for(int i = 0; i<Ntor; i++){
            ResultSet query1 = stm.executeQuery("SELECT COUNT(*) FROM
`rotasi_torsi` WHERE `nama_kompleks` = '"+kompleks+"' AND `torsi` =
"+(i+1));
            while(query1.next()){
               jml_atomtorsi[i] = query1.getInt("COUNT(*)");
            }
            int j = 0;
            ResultSet query2 = stm.executeQuery("SELECT `no_atom` FROM
`rotasi_torsi` WHERE `nama_kompleks` = '"+kompleks+"' AND `torsi` =
"+(i+1));
            while(query2.next()){
               atomtorsi[i][j] = query2.getInt("no_atom");
               j=j+1;
```

```
            }

            ResultSet query3 = stm.executeQuery("SELECT `x_atom`, `y_atom`,
`z_atom` FROM `atom_ligan` WHERE `n_kompleks` = '"+kompleks+"' AND
`no_atom` = "+torsi1[i]);
            while(query3.next()){
                oriX[i] = query3.getFloat("x_atom");
                oriY[i] = query3.getFloat("y_atom");
                oriZ[i] = query3.getFloat("z_atom");
            }
        }
    }
    catch(Exception e){
         System.err.println("Koneksi gagal " + e.getMessage());
    }
    finally{
        stm.close();
        con.close();
    }

    value = new double[jml_individu][Ntor+7];
    currentValue = new double[jml_individu][Ntor+7];
    fitness = new float[jml_individu];
    currentFitness = new float[jml_individu];
    velocity = new double[jml_individu][Ntor+7];
    frequency = new double[jml_individu];
    loudness = new double[jml_individu];
    Arrays.fill(loudness, 0.5);
    pulse = new double[jml_individu];
    Arrays.fill(pulse, 0.5);
    number = jml_individu;
    length = Ntor+7;
    fmin = 0;
    fmax = 2;
    alpha = 1;
    gamma = 0;
    pulse0 = new double[jml_individu];
    pulse0 = pulse;

    x0 = new float[jml_individu][Natom_ligan];
    y0 = new float[jml_individu][Natom_ligan];
    z0 = new float[jml_individu][Natom_ligan];
    x1 = new float[jml_individu][Natom_ligan];
    y1 = new float[jml_individu][Natom_ligan];
    z1 = new float[jml_individu][Natom_ligan];
    q000 = new float[jml_individu][Natom_ligan];
    q100 = new float[jml_individu][Natom_ligan];
    q010 = new float[jml_individu][Natom_ligan];
    q001 = new float[jml_individu][Natom_ligan];
    q110 = new float[jml_individu][Natom_ligan];
    q101 = new float[jml_individu][Natom_ligan];
    q011 = new float[jml_individu][Natom_ligan];
    q111 = new float[jml_individu][Natom_ligan];
    d000 = new float[jml_individu][Natom_ligan];
    d100 = new float[jml_individu][Natom_ligan];
    d010 = new float[jml_individu][Natom_ligan];
```

```
  d001 = new float[jml_individu][Natom_ligan];
  d110 = new float[jml_individu][Natom_ligan];
  d101 = new float[jml_individu][Natom_ligan];
  d011 = new float[jml_individu][Natom_ligan];
  d111 = new float[jml_individu][Natom_ligan];
  e000 = new float[jml_individu][Natom_ligan];
  e100 = new float[jml_individu][Natom_ligan];
  e010 = new float[jml_individu][Natom_ligan];
  e001 = new float[jml_individu][Natom_ligan];
  e110 = new float[jml_individu][Natom_ligan];
  e101 = new float[jml_individu][Natom_ligan];
  e011 = new float[jml_individu][Natom_ligan];
  e111 = new float[jml_individu][Natom_ligan];

  sisax = (float) (centerx-Math.floor(centerx));
  sisay = (float) (centery-Math.floor(centery));
  sisaz = (float) (centerz-Math.floor(centerz));

  value = inisialization(jml_individu, Ntor, Xmin, Xmax, Ymin, Ymax,
Zmin, Zmax);
  vdw = new float [jml_individu];
  desol = new float [jml_individu];
  elec = new float [jml_individu];
  Posisi pos = new Posisi(Ntor, torsi1, Xmin, Xmax, Ymin, Ymax, Zmin,
Zmax, Natom_ligan, jml_atomtorsi, atomtorsi, value, centerx, centery,
centerz, xAwal, yAwal, zAwal, oriX, oriY, oriZ);
  xrandom = pos.get_xrandom();
  yrandom = pos.get_yrandom();
  zrandom = pos.get_zrandom();
  oriX = pos.get_oriX();
  oriY = pos.get_oriY();
  oriZ = pos.get_oriZ();
  centerx = pos.get_centerx();
  centery = pos.get_centery();
  centerz = pos.get_centerz();
  Posisi posisi = new Posisi(jml_individu, Ntor, Natom_ligan,
jml_atomtorsi, atomtorsi, value, centerx, centery, centerz, xrandom,
yrandom, zrandom, oriX, oriY, oriZ);
  xakhir = posisi.get_xakhir();
  yakhir = posisi.get_yakhir();
  zakhir = posisi.get_zakhir();
  System.out.println("Inisialisasi");
  fitness = hitung_fitness(xakhir, yakhir, zakhir);
  System.out.print("Fitness Terbaik : " + fitness[getBestIndex()]);
  gbest = getBestBat();
  rmsd = hitung_rmsd();
  System.out.println("\t\tRMSD : " + rmsd);
  it = 0;
  while(it < iteration) {
       for(int i= 0; i<number; i++) {
          for(int j=0; j<length; j++) {
             currentValue[i][j] = value[i][j];
          }
       }
       update();
```

```
        Posisi post = new Posisi(jml_individu, Ntor, Natom_ligan,
jml_atomtorsi, atomtorsi, value, centerx, centery, centerz, xrandom,
yrandom, zrandom, oriX, oriY, oriZ);
        compareBat(it, hitung_fitness(post.get_xakhir(),
post.get_yakhir(), post.get_zakhir()));
        System.out.print("Fitness Terbaik : " + fitness[getBestIndex()]);
        post = new Posisi(jml_individu, Ntor, Natom_ligan, jml_atomtorsi,
atomtorsi, value, centerx, centery, centerz, xrandom, yrandom, zrandom,
oriX, oriY, oriZ);
        xakhir = post.get_xakhir();
        yakhir = post.get_yakhir();
        zakhir = post.get_zakhir();
        gbest = getBestBat();
        rmsd = hitung_rmsd();
        System.out.println("\t\tRMSD : " + rmsd);
        it++;
        if(it==iteration) {
        System.out.println("Generasi terkahir");
        for(int i=0; i<jml_individu; i++) {
           System.out.println("Value : " + Arrays.toString(value[i]));
           gbest = value[i];
           rmsd = hitung_rmsd();
           System.out.println("Fitness : " + fitness[i]);
           System.out.println("RMSD : " + rmsd);
        }
    }
}
```

LAMPIRAN H

Source Code Posisi

```
public Posisi(int jml_individu, int Ntor, int Natom_ligan, int[]
jml_atomtorsi, int[][] atomtorsi, double[][] solution, float centerx,
float centery, float centerz, float[] xAwal, float[] yAwal, float[]
zAwal, float[] oriX, float[] oriY, float[] oriZ){
  xakhir = new float[jml_individu][Natom_ligan];
  yakhir = new float[jml_individu][Natom_ligan];
  zakhir = new float[jml_individu][Natom_ligan];
  x_temp = new float[jml_individu][Natom_ligan];
  y_temp = new float[jml_individu][Natom_ligan];
  z_temp = new float[jml_individu][Natom_ligan];
  ori_x = new float[Ntor];
  ori_y = new float[Ntor];
  ori_z = new float[Ntor];
  xawal = new float[Natom_ligan];
  yawal = new float[Natom_ligan];
  zawal = new float[Natom_ligan];
  orix = new float[Ntor];
  oriy = new float[Ntor];
  oriz = new float[Ntor];
  for(int i = 0; i<jml_individu; i++){
      double deg = Math.toRadians(solution[i][6]);
      double norm =
Math.sqrt(Math.pow(solution[i][3],2)+Math.pow(solution[i][4],
2)+Math.pow(solution[i][5],2));
      double w = Math.cos(deg/2);
      double xi = (solution[i][3]*Math.sin(deg/2))/norm;
      double yi = (solution[i][4]*Math.sin(deg/2))/norm;
      double zi = (solution[i][5]*Math.sin(deg/2))/norm;
      for(int n = 0; n<Natom_ligan; n++){
          xawal[n] = (float) (xAwal[n]-centerx);
          yawal[n] = (float) (yAwal[n]-centery);
          zawal[n] = (float) (zAwal[n]-centerz);
      }
      for(int m = 0; m<Natom_ligan; m++){
          xakhir[i][m] = (float)(((1-2*yi*yi-2*zi*zi)*xawal[m]+(2*xi*yi-
2*w*zi)*yawal[m]+(2*xi*zi+2*w*yi)*zawal[m])+solution[i][0]);
          yakhir[i][m] = (float)(((2*xi*yi+2*w*zi)*xawal[m]+(1-2*xi*xi-
2*zi*zi)*yawal[m]+(2*yi*zi-2*w*xi)*zawal[m])+solution[i][1]);
          zakhir[i][m] = (float)(((2*xi*zi-
2*w*yi)*xawal[m]+(2*yi*zi+2*w*xi)*yawal[m]+(1-2*xi*xi-
2*yi*yi)*zawal[m])+solution[i][2]);
      }
      if(Ntor!=0){
        for(int t = 0; t<Ntor; t++){
          orix[t] = (float) (oriX[t]-centerx);
          oriy[t] = (float) (oriY[t]-centery);
          oriz[t] = (float) (oriZ[t]-centerz);
            }
        for(int j = 0; j<Ntor; j++){
          ori_x[j] = (float)(((1-2*yi*yi-2*zi*zi)*orix[j]+(2*xi*yi-
2*w*zi)*oriy[j]+(2*xi*zi+2*w*yi)*oriz[j])+solution[i][0]);
```

```
          ori_y[j] = (float)(((2*xi*yi+2*w*zi)*orix[j]+(1-2*xi*xi-
2*zi*zi)*oriy[j]+(2*yi*zi-2*w*xi*oriz[j]))+solution[i][1]);
          ori_z[j] = (float)(((2*xi*zi-
2*w*yi)*orix[j]+(2*yi*zi+2*w*xi)*oriy[j]+(1-2*xi*xi-
2*yi*yi)*oriz[j])+solution[i][2]);
            for(int k = 0; k<jml_atomtorsi[j]; k++){
              x_temp[i][atomtorsi[j][k]-1]=xakhir[i][atomtorsi[j][k]-1]-
ori_x[j];
              y_temp[i][atomtorsi[j][k]-1]=yakhir[i][atomtorsi[j][k]-1]-
ori_y[j];
              z_temp[i][atomtorsi[j][k]-1]=zakhir[i][atomtorsi[j][k]-1]-
ori_z[j];
              xakhir[i][atomtorsi[j][k]-1]=(float)
(ori_x[j]+x_temp[i][atomtorsi[j][k]-1]*
(Math.pow(Math.cos(Math.toRadians(solution[i][7+j])),2))+y_temp[i][atomt
orsi[j][k]-1]*
(Math.cos(Math.toRadians(solution[i][7+j]))*Math.sin(Math.toRadians(solu
tion[i][7+j]))+Math.cos(Math.toRadians(solution[i][7+j]))*Math.pow(Math.
sin(Math.toRadians(solution[i][7+j])),2))+z_temp[i][atomtorsi[j][k]-1]*
(Math.pow(Math.sin(Math.toRadians(solution[i][7+j])),2)-
Math.pow(Math.cos(Math.toRadians(solution[i][7+j])),2)*Math.sin(Math.toR
adians(solution[i][7+j]))));
              yakhir[i][atomtorsi[j][k]-1] = (float)
(ori_y[j]+x_temp[i][atomtorsi[j][k]-1]*(-
Math.cos(Math.toRadians(solution[i][7+j]))*Math.sin(Math.toRadians(solut
ion[i][7+j])))+ y_temp[i][atomtorsi[j][k]-
1]*(Math.pow(Math.cos(Math.toRadians(solution[i][7+j])),2)-
Math.pow(Math.sin(Math.toRadians(solution[i][7+j])),3))+
z_temp[i][atomtorsi[j][k]-
1]*(Math.cos(Math.toRadians(solution[i][7+j]))*Math.sin(Math.toRadians(s
olution[i][7+j]))+Math.cos(Math.toRadians(solution[i][7+j]))*Math.pow(Ma
th.sin(Math.toRadians(solution[i][7+j])),2)));
              zakhir[i][atomtorsi[j][k]-1] = (float)
(ori_z[j]+x_temp[i][atomtorsi[j][k]-
1]*Math.sin(Math.toRadians(solution[i][7+j]))-y_temp[i][atomtorsi[j][k]-
1]*Math.sin(Math.toRadians(solution[i][7+j]))*Math.cos(Math.toRadians(so
lution[i][7+j]))+z_temp[i][atomtorsi[j][k]-
1]*Math.pow(Math.cos(Math.toRadians(solution[i][7+j])),2));
                }
            }
        }}
    }
    public float[][] get_xakhir(){
        return xakhir;
    }

    public float[][] get_yakhir(){
        return yakhir;
    }

    public float[][] get_zakhir(){
        return zakhir;
    }
```

LAMPIRAN I

Populasi awal Prediksi

Trial 1

```
Value : [15.250218525870537, 5.280574966969986, 53.70697067145804,
0.3253590429939297, 0.610104248216524, 0.030385146575681032, -
60.45826369829315]
Fitness : 537120.94
Value : [0.4673471545353509, 26.721968522414215, 24.289643161245724,
0.4883324148259618, 0.6312164739630943, -0.18247405585340215, -
54.28944048569626]
Fitness : 12.394951
Value : [12.558294247474652, 20.96984493708561, 40.81744675929915, -
0.515544816830745, 0.9824548442175405, -0.5629113730406452,
146.65614512953852]
Fitness : 458804.9
Value : [13.849226041917785, 3.455940923983004, 30.86318571282056, -
0.16627643091507815, -0.992036999560951, 0.7133732930524557,
81.34885421393375]
Fitness : 73459.305
Value : [19.596330224255663, 23.43664512075901, 57.71702735849079,
0.9221735149039099, 0.028754246293501984, -0.6360402441413433,
142.18185835161404]
Fitness : 0.02576268
Value : [7.55516409715209, 22.944332269781512, 33.44576880495982,
0.381384551527681, 0.14867871484333373, 0.43989609457295864, -
93.96673961617878]
Fitness : 324603.84
Value : [31.970815016814115, 10.148994863805296, 45.05290184859295,
0.4062657161941412, -0.9494519525461416, -0.8462620444302049,
33.662033834569115]
Fitness : 136036.8
Value : [-4.64781763437564, 1.1278663256320236, 39.63842857469048,
0.7097216225337986, -0.7288744966387759, 0.78778807297629,
80.94068019321014]
Fitness : 142074.45
Value : [22.49275846629832, 21.82423176607299, 36.22680039436874,
0.5282779128615522, 0.642375910542508, -0.1823451977835282,
120.21977439716778]
Fitness : -0.24854146
Value : [6.676117929841416, 23.798855153647004, 26.915386171203465,
0.14892578274687174, 0.24015701618305596, -0.8375851434627863, -
73.78971050837617]
Fitness : 3.7923486
Value : [18.15525218008706, 8.334401889997807, 29.07687341677574, -
0.08869729968086548, -0.8955337986097212, 0.027278552684615764,
7.387377747631007]
Fitness : 252098.56
Value : [12.18190901179091, -4.308515429336038, 53.92826325862149,
0.009664313317181206, -0.12242857555400355, 0.34495425299446447, -
86.31575713991955]
Fitness : 397647.8
```

```
Value : [9.669679877334026, 22.14900537514086, 53.3824193739117,
0.7502937056777812, -0.42711637889465925, 0.055237958941812604,
98.01500890679739]
Fitness : -0.16665506
Value : [11.576339358399832, 21.066169565798276, 42.27510430540793, -
0.3161282118233737, -0.6439684436827946, -0.26845771516121575, -
12.10664661554793]
Fitness : 504730.6
Value : [26.753189858810984, -2.3391427357071777, 25.60720565622806, -
0.16782696694037647, 0.23420375313311226, 0.9183400277689069, -
9.70755063254174]
Fitness : 19406.896
Value : [2.010772227387003, 25.432643470591536, 51.73021427205386,
0.8367134360398605, 0.5035736261583237, 0.7274057857672704,
42.8314819991044]
Fitness : 117106.94
Value : [15.15379947459494, 0.035757916720303484, 33.345246437440736,
0.34577936137704146, -0.09581894194110818, 0.5003772261602835,
171.5781745878304]
Fitness : 158662.42
Value : [10.389094571966798, 0.03960827175878734, 27.11835018905107, -
0.3527665325607583, 0.7917346357244626, 0.42295166984363197, -
78.06143087376898]
Fitness : -0.07845523
Value : [-0.799468012974403, 17.04066479842721, 26.230537524015187,
0.9519615694800376, -0.6693969746296091, -0.4316285545084526, -
167.37939395324707]
Fitness : 299447.28
Value : [9.456836921942134, -8.031954501826423, 37.692153259799404, -
0.4579434653559804, 0.24771381147113147, 0.25804369289519613, -
139.41203090022861]
Fitness : 71587.7
```

Trial 2

```
Value : [16.758451223097126, -8.319133498435011, 34.84172063246503, -
0.47364809157680154, -0.3589288884303372, 0.17398011602952357,
144.155951132432]
Fitness : 363439.94
Value : [31.37103169165953, 0.05471789897275414, 43.88594881216649, -
0.7547820279753672, -0.4405747315901354, 0.7289584198534662,
135.87363405244054]
Fitness : 42520.598
Value : [-4.346205401395531, 12.774848474645076, 39.32667639161683, -
0.7657089999611972, 0.23646836353746048, 0.8651537824089832, -
73.53493559702002]
Fitness : 298841.97
Value : [27.993820415885992, 14.575972205340523, 54.75368967712491,
0.5537098390135351, -0.33365139402587785, 0.8974399214575699,
8.015170811548444]
Fitness : 192447.66
Value : [2.341057001736024, 3.8405185273303886, 55.73579616506123,
0.2706069461140821, 0.938461946201931, 0.4112698365851186, -
36.637697806912456]
Fitness : 102.75587
```

```
Value : [14.99241764466814, 12.003242873544682, 47.04216580913092,
0.24052401558379222, 0.0343246353287765, 0.2967866900996068,
155.78404453115843]
Fitness : 121754.91
Value : [-1.0623545033720454, 8.724076040038582, 35.73512641081015,
0.6149775482044759, -0.9641068787638349, -0.9342645075598792, -
58.10374634390686]
Fitness : 334539.88
Value : [31.957514765523243, 28.33922407245243, 38.19758465755362,
0.5009376264127605, -0.7755847969126934, -0.6565251309198084, -
46.75922992758319]
Fitness : 4.702194
Value : [22.778149650106283, 28.589061642658308, 49.878305082452634,
0.8313163096169618, 0.07010567556183545, -0.3614104352956491,
130.60583098198703]
Fitness : 8.925463
Value : [-2.7920025919752183, -9.0178677769686, 31.70944959306749,
0.7553882149116671, 0.08541815258221352, 0.8319175377444623, -
143.5669368544257]
Fitness : 7.2634234
Value : [-3.2868048309893743, 16.258649087479, 58.339663200080935,
0.9612311017065365, -0.8594372519271058, -0.014773076900819415, -
150.19738577754666]
Fitness : 57410.36
Value : [28.039469052108963, -1.6956107954459014, 27.175606190186258,
0.49246454772060155, 0.2809806747362267, 0.17925788509317409,
79.3046375019037]
Fitness : -0.49666902
Value : [17.006339428093856, -6.919945608749412, 36.82457017527849, -
0.027658256581481355, -0.402926910115448, 0.9958938151290957,
125.37375934148298]
Fitness : 273202.0
Value : [11.638758287112598, -2.169346121221764, 61.406893550221156, -
0.17313855797999134, 0.38439045549566453, 0.22018024223521304, -
13.305649845052528]
Fitness : 56185.336
Value : [13.412506214403933, 13.343391584431064, 48.84071998344611, -
0.011714438395863391, 0.6180496527950512, -0.9504080740593657, -
76.82808623699295]
Fitness : 213507.31
Value : [0.24121195357543268, 25.19576366578258, 31.731055663312354,
0.7967855304346958, 0.7964084303692287, -0.9059001691675781,
75.35073042877846]
Fitness : 3.3512547
Value : [3.089485697254128, 17.57043541604133, 30.00636569119728,
0.6315569575053361, 0.1782603200722268, -0.3481652368644903,
61.182097353016616]
Fitness : 271576.38
Value : [25.10505326941228, 4.312794173144972, 28.010959285908704,
0.5293087679161224, -0.0677519408544216, -0.9041633962619049, -
28.10544287077863]
Fitness : 19906.518
Value : [32.352577232963114, 11.527211074128132, 40.12786710170886, -
0.18704168456282932, -0.0758137783165358, 0.03411878022390158, -
140.09849164863817]
Fitness : 82319.98
```

```
Value : [18.38771036960941, 20.065392086199587, 45.596641682897925,
0.5265651496870747, 0.6425522947700864, 0.8174281010953237,
43.98445228332409]
Fitness : 108063.39
```

Trial 3

```
Value : [-3.364378895921301, 15.159490949086845, 38.309723750660766, -
0.5001970719515028,      0.08513776175799004,      0.30556058912693573,
137.7475457015994]
Fitness : 313485.8
Value  :  [20.92490821099262,  5.576162085043894,  41.589373071598544,
0.18600895657722516,    -0.7193986170651541,    0.3321228264392875,    -
26.567460121803975]
Fitness : 117896.125
Value  :  [26.24616038373437,  -2.9961211047633167,  55.8487543242053,  -
0.5666202513612744,    -0.23840192724342146,    0.0966167963663438,    -
43.88889614211314]
Fitness : 240113.58
Value  :  [8.833020507546944,  8.926472723437005,  43.27602913279938,  -
0.5685613579968518,     0.6448066046910967,     0.5310545594652918,     -
52.422890660347264]
Fitness : 152458.5
Value  :  [9.505030986657339,  18.832833023225188,  49.60786411532369,  -
0.8626070380401676,     -0.6611801922899085,     -0.53401882936518,     -
46.07498082227485]
Fitness : 140012.45
Value  :  [29.527139990724663,  2.887206807975053,  40.175144256530366,  -
0.8448795829775007,      -0.9855640265785988,      -0.19061699479607608,
41.046215476565976]
Fitness : 31155.266
Value  :  [17.07479764395815,  -8.828153119506513,  43.047176849093574,  -
0.18161173527647367,       -0.3837131963505318,       0.264324632704412,
140.93375999852236]
Fitness : 131081.84
Value  :  [4.368694771090567,  14.61152916703103,  28.79524506886307,
0.5786816627421041,     0.6764247752165395,     0.7174813382252823,     -
51.46283464581984]
Fitness : 97.104805
Value  :  [-4.924339572462409,  8.488179935742522,  23.735170670095584,
0.38138952063983367,      0.06476289719483241,      -0.9554123053391088,
27.488986804797605]
Fitness : 33.605
Value  :  [17.139734479132944,  28.473792689027007,  35.401268871916145,
0.10270610907470323,      0.14354935216727038,      0.8209811016567827,
120.98056631855172]
Fitness : 31537.15
Value  :  [5.688864825823774,  17.063104434360284,  58.291223124708644,  -
0.6746881369186004,       -0.815093157325673,       0.3837054805346036,
26.844708386008904]
Fitness : 120459.54
Value  :  [-2.265818432297147,  -2.6044829538441796,  24.50521303722153,
0.7054199108032606,       0.7039270495204804,       -0.5488981188307738,
54.10299064392092]
Fitness : 4.4313035
```

```
Value : [24.417570533335564, 28.70146684697243, 27.50284241417929,
0.6020321954457479, -0.8008889115918818, 0.9340263445412924,
125.6043233137894]
Fitness : 11.6148205
Value : [29.018269381391107, 23.79759052412622, 35.01450135016071,
0.9199686029806167, -0.39787063081219665, 0.8893423482622622, -
136.16922015783882]
Fitness : 0.0011551692
Value : [11.310896439551176, 0.09185562777059353, 24.059891229249406, -
0.1227776991782552, -0.3707853324714514, 0.016608368316082256, -
133.54177121259383]
Fitness : -0.19100918
Value : [11.266505453875325, 3.0942025752220346, 23.811703153984684,
0.22599257351913593, 0.014103020346408979, -0.7594321087404192, -
70.64850898345519]
Fitness : 12.606792
Value : [17.877227827560784, -2.2202514429779434, 32.64196244920825,
0.03706487557420646, 0.3386880099632723, -0.28114806770791434, -
5.163705754867436]
Fitness : 21482.717
Value : [24.009361542419853, 13.167806777009787, 42.557439702411514,
0.6188910156620584, 0.24788051523348376, -0.7797211415829504,
172.333712481833]
Fitness : 287090.94
Value : [0.20571833905061965, 1.7337606858458017, 44.34743899615109, -
0.5894109374881313, -0.11019444084941687, -0.8762696549972648,
126.88311974939944]
Fitness : -0.2699281
Value : [11.12266954975588, 26.754594401753167, 44.73894215688202, -
0.39954468262655074, -0.41636648471321114, 0.9792507930476193, -
72.80092924204594]
Fitness : 140086.06
```

Trial 4

```
Value : [27.567555162361685, 8.12387583664928, 28.474729409174287, -
0.6435371755326913, 0.717205355416445, -0.41520799184706747,
29.15874931991226]
Fitness : 343256.34
Value : [6.77333240765517, 14.691312385334214, 48.921062487439706, -
0.235627650531405, -0.3515675843995254, 0.588461434558718, -
1.7609305632404926]
Fitness : 405954.4
Value : [9.616785544571634, -7.609908364666332, 39.47294425197052,
0.7132577862380414, 0.23239916291516405, 0.762485862147902, -
124.48566412539842]
Fitness : 12383.829
Value : [1.8206247956026687, -6.600300947016165, 58.09560477269324, -
0.2549227589376868, -0.019196075249563016, -0.3361817993428051,
149.4428410450862]
Fitness : 0.8489332
Value : [30.590463661227595, -8.183632560618072, 28.02504808146994,
0.09454566842320378, 0.4527834302468552, 0.36884273446511506, -
24.512358801001568]
Fitness : 1.2742332
```

```
Value : [21.21474932167703, 5.896810835808743, 38.72886147271758,
0.7262254670288197, 0.15125182964308714, -0.029522585799938827,
49.10507569661385]
Fitness : 484449.12
Value : [18.954639944985402, 14.089862710182615, 49.44356008918809, -
0.39166790893951986, -0.36217859030290844, -0.6320992234905838, -
19.522353648406607]
Fitness : 275561.4
Value : [4.0061768905965, 0.5780822734011899, 51.581372917598756, -
0.3335507020396251, 0.4587643821850309, -0.7077146715569278,
109.31361709306304]
Fitness : 86.62436
Value : [0.4531859892882446, -7.482145183693241, 28.250063536613947,
0.7281074120094577, 0.4837250509753692, 0.10023238430393278,
110.76671463574093]
Fitness : -0.016152315
Value : [13.181317205721449, 3.354656741851226, 29.20141607203669, -
0.19542932865078444, -0.4751488570007245, 0.08657085331723735,
90.92469273286684]
Fitness : -0.3377024
Value : [7.5373379196993024, 3.969656884280985, 54.72165314753392,
0.07953828320138068, -0.21508936074732277, -0.402786315971434,
25.09748110502369]
Fitness : 403889.06
Value : [6.769598994375496, 23.100762747113606, 26.372158532440473, -
0.9407408858354813, 0.6046107208752707, 0.8583692303439392, -
160.07882001286384]
Fitness : 161830.0
Value : [32.06431772240131, -2.3477585157335294, 41.99867938766562,
0.9575556022243541, 0.04004838189744486, -0.8240482800103617,
100.48825975720672]
Fitness : 11682.554
Value : [6.112902507484536, 3.51334884382268, 38.01013606925614,
0.7549362660900898, 0.2819826661655165, -0.17677205879381241, -
154.46632089615994]
Fitness : 194121.88
Value : [28.723094499303592, 10.467957794739675, 53.228918144033514, -
0.8655857019559925, -0.06489652194028528, 0.15384877807498976,
60.63047196281494]
Fitness : 126187.25
Value : [28.554674794403553, 22.188629550038286, 28.83949641748863, -
0.09368068930813789, 0.875963547382727, -0.2731247826194323, -
5.991338697787938]
Fitness : 0.044285327
Value : [-1.7376701044823344, 13.938121297783198, 60.51653066358929, -
0.40601633835241, -0.33099270353409027, -0.8902274823985554, -
30.07798656435537]
Fitness : 65905.586
Value : [9.964794554978248, 10.714061411188968, 27.981919006198563,
0.9540687681048381, 0.9773175252534392, 0.8702447407293439, -
19.470649926798984]
Fitness : 54228.87
Value : [16.922965911633078, 15.258808041620036, 54.94006893697204, -
0.8464449589772407, -0.05067048753555947, 0.6358680058776331, -
32.79097277022669]
Fitness : 118026.59
```

```
Value : [19.686196499737992, 7.4974557906114185, 26.733494540794407,
0.6001476021180032, 0.6528424871150604, 0.6079928460633715, -
135.6309406025228]
Fitness : 1915.2257
```

Trial 5

```
Value : [7.263575188830586, 9.504194709723471, 43.04065266621319, -
0.02300210027592864, 0.612293557267819, 0.5253401360533578,
124.70236525382597]
Fitness : 397560.75
Value : [26.842215240832367, -6.378019231205745, 49.2678323707899,
0.1448062620224717, 0.6958534593773202, -0.20793394165562384, -
147.1919967132564]
Fitness : 474207.84
Value : [28.521747803943377, 25.732089793630614, 30.832053058998277,
0.029598069589393372, -0.9153489027075814, -0.20251912883562317,
87.22279039692012]
Fitness : 3.7644591
Value : [-4.994605591001136, 8.44551239170002, 37.09963899034325, -
0.7664694295374042, -0.09297384764215244, -0.6668710509667304, -
6.148573429184495]
Fitness : 218114.84
Value : [27.45398810618383, -2.12127202857787, 57.839393172427805, -
0.3644805260610473, -0.8191389971722496, -0.603165179359832, -
104.23903236594177]
Fitness : 389054.2
Value : [9.083451376723264, 20.823302850302845, 44.3942629535877, -
0.6128141674327536, -0.19796530578280214, 0.2020042587865427,
2.194169963917574]
Fitness : 406863.5
Value : [26.281873032624524, 27.6351313571383, 23.297459438682974,
0.47863970492947794, -0.9506455939275482, 0.7433441265747645,
114.86630623549286]
Fitness : 9.852924
Value : [8.01429729222954, 1.7748478445893774, 45.01891162575475, -
0.4119591971895442, 0.3898458676745262, 0.9069137467013102, -
40.38535361974786]
Fitness : 368612.38
Value : [-5.678944546744991, -8.786762865189875, 26.565797134213348, -
0.33743881530922426, 0.8238898435698225, -0.006876865475053284, -
161.54994704909606]
Fitness : 2.644972
Value : [29.81353908253015, 16.96297595735106, 37.07025103779688,
0.9099870364003144, 0.9469777849372807, -0.42928598760067005, -
72.34481296160536]
Fitness : 4.276568
Value : [6.388289762077665, -2.393424500952623, 34.680977559501784, -
0.36483985178405587, -0.6230961782041728, -0.9810874399857046,
135.34838268175298]
Fitness : 22663.656
Value : [5.888542412901916, 11.384723078819608, 57.20553162178228,
0.1619400065073069, -0.8939996773441747, 0.2451994167578997, -
25.134256868288958]
Fitness : 467910.03
```

```
Value : [2.046096144848889, 27.91474514769334, 39.70450043015967,
0.795746733353947, -0.8494818669346058, -0.1289820289215191, -
7.071159258571413]
Fitness : 3425.0579
Value : [21.52513327189832, 26.479718797948742, 52.346723306608524, -
0.08735533592418498, 0.14973593204687008, 0.1216820864410375,
105.86769499480965]
Fitness : 2.679193
Value : [6.374112970224722, 15.275523624487846, 25.63709877705333, -
0.04355623868964864, -0.3077839078354958, -0.04407180367329455, -
147.63008174361684]
Fitness : 121677.39
Value : [-0.5813919973789599, 25.387666890968404, 36.5173923991156,
0.7182397919073413, -0.18166519734663722, -0.6925340078455604,
112.74611448041458]
Fitness : 234952.73
Value : [28.917849829123767, 5.814338532849476, 55.15547672874766,
0.37251296113669863, 0.6093230296366441, -0.49270007115982306, -
104.3698431091812]
Fitness : 185851.9
Value : [5.740953034412623, 14.960601905281369, 29.31905995223582, -
0.292264977250424, -0.8983642766680613, -0.9957680231163253, -
175.21249226669167]
Fitness : 103634.336
Value : [2.560694328769369, 27.509705119014328, 49.06115941760126, -
0.06480652682531285, -0.35128471782801696, -0.7132443929969441, -
54.795457154939]
Fitness : 6508.361
Value : [-0.8188680961602905, 22.324666007418728, 54.43464305197456,
0.5728595173068978, 0.2500374257396034, 0.5498085029271518,
28.365875590921462]
Fitness : 54415.64
```

Trial 6

```
Value : [31.16173262190744, 13.48849317546442, 33.512499099981696,
0.24806581797086946, -0.6783947236740926, -0.3584181654088139, -
167.36530065987102]
Fitness : 156959.38
Value : [-2.1930047741990704, 4.650085834453311, 50.952998064391736, -
0.5965437648194727, 0.7268098034727322, 0.8476932673546629,
93.06908795371822]
Fitness : -0.3073254
Value : [1.3479201976072037, -8.446102123099976, 42.66394471194404, -
0.4025721035620269, 0.0854831389427988, 0.8668149732876989,
161.00606050114834]
Fitness : 5.9329677
Value : [15.433455753946511, 9.204095019218038, 49.67932756820484, -
0.2973620563349588, -0.7728374315432254, -0.385674496048493, -
47.02921001704658]
Fitness : 306972.28
Value : [19.253748875554336, -8.058328567595739, 44.416713806618674,
0.7357136882543647, 0.9174223724488679, 0.20617932448328147,
20.50908852052291]
Fitness : 2747.2385
```

```
Value : [7.238545677847769, 4.345990151438734, 29.924936405342194,
0.8001408302474051, 0.27102292267637096, 0.926058666914723, -
179.82358738348677]
Fitness : 144488.33
Value : [12.417760733196747, 12.661883783444388, 47.93380582399563,
0.3633410791207161, 0.9843379868149502, -0.29523158042400865, -
77.66988438101977]
Fitness : 232386.52
Value : [24.35824231178394, 22.788194210538215, 47.72297629814268, -
0.21851923717830513, -0.5763342510730365, -0.5766058168690273,
16.949417394036686]
Fitness : 78.69698
Value : [9.106743484260228, 19.70128588690908, 33.9599031194045,
0.37421134040195136, -0.9346032000905289, -0.24562042706167242,
4.243516492442865]
Fitness : 155652.4
Value : [-0.599566140550241, 12.439012582248587, 31.772041062559254, -
0.614412905150904, 0.7033727534750969, 0.03645429884551321, -
132.1045948575112]
Fitness : 99186.69
Value : [-4.858540368063358, 26.730859371140404, 45.20713810715536,
0.029154288062176015, 0.08471454799161249, 0.314096664388158, -
155.52324403154418]
Fitness : 9246.338
Value : [27.713462393698343, -4.419496445016671, 56.596683150189094, -
0.6181609406604005, -0.09693531459835847, -0.3151262529869787,
47.747385281121325]
Fitness : 463326.78
Value : [26.46789143033879, 0.36462524190229395, 24.592718050750793, -
0.4242204223490478, 0.002687998229814248, -0.9206804623369447,
129.3604213595064]
Fitness : 16.694866
Value : [18.725371790056624, 20.240565690425512, 37.064890444892306,
0.44951404965782626, -0.9729277777354752, 0.5113614115700165, -
35.61031120193721]
Fitness : 8868.188
Value : [-5.527488318434098, 11.999819341140288, 26.575038810033696,
0.430684383912437, 0.07553942153752757, 0.12521846862849917,
6.2234165410455375]
Fitness : 58609.812
Value : [4.19651396452956, -0.6002617870054809, 26.13674713199444,
0.3004653743985404, 0.12180909251938554, 0.5505692942861988,
52.6552112101227]
Fitness : 89.168785
Value : [-0.3953019910940796, -8.05722487050917, 53.644186405461795, -
0.6295181440261914, 0.3097985849876965, 0.24548331225728348, -
46.89022833142113]
Fitness : -0.017233282
Value : [2.390168871762935, 25.450668496450163, 44.7462557423896, -
0.7677343013604121, -0.5148901224997282, -0.7716771670699922,
126.00565734231662]
Fitness : 330095.3
Value : [-5.9082682243641464, 19.00912857794888, 45.63808474576702, -
0.07157225902871489, 0.5220721064167297, -0.4300216422526981,
143.06176582229233]
Fitness : 190226.17
```

```
Value : [-2.9792824301145986, 20.60719726264211, 61.14982760150872, -
0.1970335821135425,       -0.830514278531095,       -0.16165206658215925,
107.3353205432735]
```

Trial 7

```
Value : [21.052852725588413, -3.0777544197317326, 24.1064084079497,
0.9338563454313344,    -0.8877028130732589,    0.3682227668615028,    -
114.18962196018315]
Fitness : 6.1808786
Value : [16.570068797425392, 20.31001955274355, 28.6781481691008, -
0.31744783337032634,   0.6569445958384279,    -0.029647392999076372,   -
7.402461571128612]
Fitness : 20908.703
Value : [3.950058473036396, 16.17243645346112, 24.69448777815127,
0.6835316955983566,       0.4329470171265799,       -0.17405851027027675,
42.68966432758987]
Fitness : 116151.54
Value : [1.6463730279373419, -8.2454527738748, 38.77965300083902,
0.6383915283192922,    -0.3181711205465747,    -0.2721080280830679,    -
151.7881834037936]
Fitness : 4.9345326
Value : [3.5414081802573456, -2.6782846036816093, 61.265077963506684,
0.362372388863873,       0.04874334924463741,       0.13517015919250808,
122.39974072834508]
Fitness : 7.257199
Value : [28.3196363244627, 8.228651583030892, 54.93333628045504, -
0.9408119784067956,       0.4075599920070463,       0.10861488643189388,
168.11874182344866]
Fitness : 453682.88
Value : [12.524492116923994, 15.646333265915466, 35.458498278453916, -
0.6604362676481692,    0.5752546943944741,    -0.20828525313481805,    -
117.37581675804569]
Fitness : 194982.72
Value : [24.00673168891383, 3.0181247548744103, 30.615881573161822,
0.618441770330143,    -0.6934278007370731,    0.16067169643377155,    -
12.484462817657743]
Fitness : 9010.961
Value : [16.98592122534367, -6.451291138772802, 53.153252459324015,
0.36252049769411143,     0.22419664470188372,     -0.027647026775763583,
126.2118033834173]
Fitness : 161127.42
Value : [25.0699872340303, 26.19272666635363, 36.15669715979068,
0.7495365111471497,       0.4404820040238744,       0.9525717372171636,
108.28016467398533]
Fitness : -0.09896149
Value : [-3.1525720634156755, 15.12115385542129, 55.62394125241431, -
0.8031101163261991,    -0.584067621179611,    -0.7413512393860695,    -
172.89345733140894]
Fitness : 291241.6
Value : [12.16324361767634, 24.41888504987927, 30.27930066914205,
0.726917589604396,    -0.1022892013485659,    0.09091456150987498,    -
66.48896103690562]
Fitness : 23131.05
```

```
Value : [24.22816434733854, 26.12190949198248, 43.2547078898034, -
0.7721191022617329, -0.23664953396057675, 0.019616968651080313,
145.19322296615996]
Fitness : -1.0458852
Value : [8.88899148072719, -6.426270850741161, 25.951690523949786,
0.857004911750266, 0.6115406468386961, 0.5308188782990018,
57.84523958726206]
Fitness : 1.0335333
Value : [20.520193165578736, 23.35559857715213, 50.77948676673673,
0.94611991100009, -0.24453137909201672, 0.49981241664492937,
52.120168489989794]
Fitness : 6927.3784
Value : [16.215902795166762, 21.526661937746198, 51.492769090994564, -
0.024700958224347813, -0.8799208878140543, 0.010661963000660135, -
51.658134737767085]
Fitness : 2.0060325
Value : [24.019844106054943, -5.799755783400496, 51.85166751626403, -
0.4841326411296578, 0.37222402193777326, -0.540921506821489,
166.93135216435712]
Fitness : 212341.1
Value : [21.482520136785354, -8.185165207793428, 32.649488035590515, -
0.270010170212978, -0.7058155402306332, 0.5397051193612314,
103.37679646246346]
Fitness : 14.713172
Value : [-4.770614880408799, 6.260800129771226, 39.545383389955276, -
0.8310584622741934, 0.6443002135245441, -0.7290414929810487, -
109.18833778075924]
Fitness : 87347.516
Value : [-0.9087347570379638, 10.743812619780499, 61.58354160522184, -
0.9994192218296876, -0.1817078117959614, 0.8043513585043471,
88.37270010650508]
Fitness : 16695.035
```

Trial 8

```
Value : [15.370863151090546, 21.840711926215747, 52.11570851108772,
0.5611469929369557, 0.28804003194761796, 0.6185515863071223, -
6.771374428532624]
Fitness : 13803.039
Value : [13.300482409824589, -0.9995458850448209, 24.19068842479733,
0.44547775016358737, -0.6599070320637184, -0.4669589131971945, -
120.80036870891809]
Fitness : 0.032715917
Value : [32.04604861345438, 7.567963135929634, 63.18879243101038,
0.5501494914082861, -0.7015638887645295, 0.1805033131778533, -
153.98525975191512]
Fitness : 560.40625
Value : [5.907357216039683, -7.9295924044975745, 27.46000101555816, -
0.2788992205809544, 0.14125814046811036, 0.13785055425072934,
174.67429072500573]
Fitness : 146.48102
Value : [4.39705585953681, 27.73567523684349, 57.416990926913485,
0.3713095149141261, 0.1444369343803391, 0.5421733275648772,
152.2364508523621]
Fitness : -0.056619212
```

```
Value : [27.247863621576414, 17.55632980053715, 58.816498093036884,
0.9675202335210362, 0.2720129119179193, 0.20676070582555028,
3.286047494016742]
Fitness : 2009.0627
Value : [2.1835995290606043, 26.434045451990997, 27.54346602945195, -
0.1757229897984347, -0.5058515156741852, -0.21598423660942334, -
140.39728484063207]
Fitness : 859.02606
Value : [26.24561031196832, 9.064804267961136, 53.8020120638361,
0.8939807691281567, 0.010950622281655997, 0.9494179315933249,
138.0431408653281]
Fitness : 317484.22
Value : [5.063487559148625, -6.3076083817076105, 32.65452942907179,
0.8997019544705951, -0.10617641372414188, -0.7580919425884978, -
25.427512536947233]
Fitness : -0.08600597
Value : [-0.2593450632472454, 28.387199825529528, 27.624084335003626,
0.95395708681804, -0.2895817549442097, 0.10098544849111812,
88.51518018069635]
Fitness : 10.619244
Value : [6.350550899715824, 26.715117541590544, 36.48268165027573, -
0.9613715145045849, 0.8380165220116553, -0.08943900704769248, -
139.3506094043822]
Fitness : 227999.47
Value : [6.502619319507678, 28.664538507173454, 60.876817267240504, -
0.8781671271406057, 0.27495715688754263, 0.0024208122189603998,
14.285394749852259]
Fitness : 7.440406
Value : [19.4410448908177, -8.699098695398678, 52.74865986576355, -
0.22365039008020093, -0.6339970592054662, -0.21920008107447342, -
36.45422057571665]
Fitness : 95318.34
Value : [17.411672964167916, 3.657302539380268, 43.63758925626743,
0.1900878403372559, -0.7684030108134621, -0.09399787507282653,
69.19466058396719]
Fitness : 242699.95
Value : [11.87991358491385, -7.72490590954454, 29.451614884625748, -
0.33743662252553186, -0.06325704777637275, 0.9337321923620079, -
136.6149318732243]
Fitness : 0.116932586
Value : [-1.4999346142268015, 9.12856182907673, 50.34077616591969, -
0.9595482523103536, -0.14160397889897358, -0.5540258821744437,
12.187740099274095]
Fitness : 190346.62
Value : [22.06474813026223, 2.115190676606284, 56.45750390520855, -
0.8939461489868752, -0.8971667362515459, -0.6238914274987437, -
33.62238838417181]
Fitness : 281705.94
Value : [11.316204292400336, 27.58032271889165, 39.71272292707373, -
0.7361535756186777, -0.7277689671231931, 0.44572877780662634, -
47.53364592760218]
Fitness : 36990.98
Value : [15.552567378802433, 19.845519421990268, 59.297330749502116,
0.11918883225322174, -0.869178428654986, -0.6297327190400444, -
43.38318035829977]
Fitness : -0.7224164
```

```
Value : [17.61557129300802, 2.7241041433883435, 25.200422005021323, -
0.9952293191041859, -0.9069368284351724, -0.028318752010514148,
173.41832793119147]
Fitness : 265.6708
```

Trial 9

```
Value : [29.551091364448205, 13.42396932430874, 43.05496663012791, -
0.3694643213336841, -0.3002624702838501, 0.5700525683307385, -
99.49021886980663]
Fitness : 262597.0
Value : [-4.130724651823344, 8.25411372592114, 32.03230754217219,
0.8122946529259079, -0.7154085903839764, -0.2921889185351214,
22.986339312129417]
Fitness : 440058.44
Value : [13.20397420806697, 11.688479192459031, 49.869694397482775,
0.14154314587048433, 0.22870523702621925, 0.5691229842283398,
10.846591425440124]
Fitness : 41272.508
Value : [-4.890622533073435, 21.224699300524037, 58.76663188714315,
0.2989342826645356, 0.056258758208845405, 0.9322657440738171, -
40.65440836307087]
Fitness : 216.9095
Value : [21.06567936309318, 17.291172854592155, 57.97187767484097,
0.21486405809713793, 0.15010107102652603, 0.2288028293465627,
16.902766281386562]
Fitness : 1491.6339
Value : [-6.21325265179985, 13.006287681014527, 37.07244143815723,
0.49982030786495346, 0.2033323242849825, 0.5183096521801791,
60.040391116587244]
Fitness : 117017.586
Value : [29.961306413070133, 13.097226858125087, 62.71023532204755,
0.9130446131123016, 0.6635371547938469, -0.20697568701123648,
20.60945617760578]
Fitness : 4.2653832
Value : [15.98925793231654, 17.055195611403338, 42.965927340208545, -
0.4205779833321479, -0.05439649205703678, -0.11065599585685315,
49.816410459833975]
Fitness : 73795.65
Value : [-5.124603677339461, 10.938582740234118, 59.41506060993552, -
0.14534021364966687, 0.3653301034797971, -0.019444856509782005,
98.03440885486697]
Fitness : 36548.355
Value : [-3.8139194638568057, 26.93494054349523, 49.39830354178956,
0.7062544805064002, 0.9055121744998167, 0.09088101665923598, -
68.86511204811697]
Fitness : 51478.59
Value : [28.55005987437322, 17.594909258103506, 42.7330078151855, -
0.014971534298202194, -0.729329605891432, 0.03922979301910168,
57.371431837382914]
Fitness : 82362.57
Value : [27.921986729310404, -9.554427319990475, 62.884592758566, -
0.9084948711878635, -0.29128692398397105, -0.38301732730281834,
106.86699586321879]
Fitness : 8.736661
```

```
Value : [20.20771014219928, 6.549389481202169, 44.63378668255169, -
0.6840181512985497, 0.8000138655889775, -0.610413411439644, -
46.52710462151376]
Fitness : 347090.9
Value : [32.06278342065005, 16.296065758475546, 26.115547167429256,
0.7199784808967575, -0.4513606041721323, -0.274334625434983, -
111.20589970562563]
Fitness : -0.3799953
Value : [2.245130256261085, -7.840387730185195, 57.41401864243798, -
0.5251610252239876, 0.7405610980777613, -0.1850309843984992, -
159.6885026148466]
Fitness : -8.83294E-4
Value : [26.176166311938424, 15.405655378229792, 45.489758683648304,
0.13194850578482797, -0.6409803625039427, -0.612227004701545,
169.60161543920543]
Fitness : 249967.98
Value : [2.3817004051571686, 17.86873198425795, 42.765483059826764,
0.27076302604253044, -0.035397463326551915, 0.6360860711657019, -
113.13407955119139]
Fitness : 243970.11
Value : [6.517057435557835, 1.9017665554318057, 48.786683301637254,
0.9204567773447054, -0.4146724401784472, -0.16985707473538736, -
74.12590803110783]
Fitness : 55391.336
Value : [-4.52617684094651, -6.390561942354312, 28.42740440922005, -
0.22795100186312922, 0.21755078008056117, 0.540300718997992, -
174.83810235363345]
Fitness : 1.0896871
Value : [12.552830574019172, -8.324122502951578, 53.496291150948636, -
0.3390014751865611, 0.20896378353837353, -0.9537193332627094,
163.87972681681163]
Fitness : 109665.83
```

Trial 10

```
Value : [18.106400179180056, 13.06129439455605, 61.093594507811936,
0.24172777218636732, -0.36520455636779303, 0.9330804495922971,
97.14472589683373]
Fitness : 766.4557
Value : [1.608158669657854, 14.621007730776519, 58.01619433020923,
0.7804941626941282, 0.588715110029709, 0.7896216504680227,
98.3221479161345]
Fitness : 165086.16
Value : [3.7988265523301727, 26.26184023967832, 32.25078082005894,
0.7475664216960844, -0.36266126016972233, 0.11454588129244114,
178.91510627553384]
Fitness : 32707.105
Value : [-5.487485611962318, 3.283868558317275, 37.311459014637066, -
0.9134068974947043, -0.16932100694613372, -0.6684586741300542, -
78.02836626919172]
Fitness : 227142.52
Value : [23.602588791100477, 4.16799492685006, 24.128816846315203,
0.4142811683009684, 0.3631620065653429, 0.8887634040812757, -
43.23222815954094]
Fitness : 20938.498
```

```
Value : [-4.5650598810311065, 12.364285138521822, 28.287259944783123,
0.39176376167049165, 0.48472874437902025, 0.13175897447764684, -
90.25296307694785]
Fitness : 258844.28
Value : [-6.002212475990032, 28.645194450293197, 58.155798564503776, -
0.7425704899947132, 0.589804015654015, -0.7135200382800746, -
111.2488388194618]
Fitness : 13.5089035
Value : [11.34901432846964, 4.987227805336651, 42.50065289493745,
0.5208566632683513, 0.19550463294722253, 0.08377826926685317,
1.174775228114754]
Fitness : 182975.98
Value : [22.691575210389903, 27.54952998788854, 47.18873211724629,
0.7666495314368946, -0.0426887408228398, -0.7916306545127285, -
133.8671323453358]
Fitness : -0.7407115
Value : [27.904576959861117, 6.355428420670936, 29.320596620998653, -
0.8397853839088119, -0.45904659091656463, -0.3600691067203372, -
80.58085189555078]
Fitness : 64280.523
Value : [12.272577198936563, -3.6722592316417506, 30.184294041174994,
0.9763723636409329, -0.4862928563549547, -0.6456824904581493, -
29.674168987328954]
Fitness : -0.16205889
Value : [30.015705173538308, -4.849159648654127, 28.408055138100696, -
0.38346108316025274, 0.8522676984315141, -0.7033917755238919,
7.612056989882831]
Fitness : 2.3250546
Value : [24.924959806879606, 12.346057572732278, 28.02138688671789, -
0.5501041740937447, 0.30688733701320126, 0.31604887598376563,
13.335508426464372]
Fitness : 166099.06
Value : [-1.764608307001481, 5.262920223383993, 51.902380488898686, -
0.12482401837276313, -0.5736628269352069, -0.9372730921124106,
41.20534278407655]
Fitness : 7261.847
Value : [3.725181097131383, 10.805809315907748, 26.8009648547147, -
0.9525274331670814, -0.6876780261795326, -0.9677392413752843,
162.02980903213216]
Fitness : 324266.38
Value : [5.852875928549597, 12.128335913132712, 24.827038291478797,
0.32301769250604484, 0.3723066563201085, 0.7590056169079935,
119.30711332364257]
Fitness : 11467.194
Value : [2.928748944336924, 27.78415623566879, 48.34903352534418,
0.9551827823767574, 0.28841241595414036, -0.11909001589130641,
113.13847120757276]
Fitness : 9.55377
Value : [-0.38336085370910755, 20.21236607351695, 56.84763511780185,
0.10336029097323185, -0.9365136396558214, -0.48071016555816404,
97.06673536616631]
Fitness : 523.9857
Value : [5.315486028554341, 22.938577432030712, 43.38069622969291, -
0.2644215048775962, 0.5918795140321091, 0.6927339716225771,
73.08647570284967]
Fitness : 264222.0
```

```
Value : [20.624531399079075, -5.346512603579654, 37.673688104815966, -
0.0627710597716793,    -0.840448387861735,    0.05031832578823825,    -
138.6416058409252]
Fitness : 148792.95
```

LAMPIRAN J

Populasi Akhir Prediksi

Trial 1

```
Value : [7.932643291146475, -4.79055453074172, 55.839515963136066, -
0.8800508846902478,      -0.10835281208799374,      0.9323951649270212,
170.6327915876256]
Fitness : -0.77879095
Value : [7.230228356330059, 26.088491691089416, 52.48350426160457,
0.6798570891699509,    0.43627866669637316,    0.2941364738706669,    -
66.18503492216023]
Fitness : -1.9829301
Value : [10.471489089009047, 0.44309368032657126, 35.28896827100216, -
0.8810883725496735,       0.9083786561125999,       0.7022039682606522,
90.88583702949273]
Fitness : -1.3987818
Value : [26.08356469055355, 16.374924043963958, 33.982012441556776,
0.501387329667708,    -0.03545121085959102,    0.3730445922006951,    -
78.88646933627318]
Fitness : -0.8940432
Value : [-5.465367678436473, 22.031932784530092, 49.108679463328016, -
0.5002955226753598,    -0.4030413958310235,    -0.20261112472322962,    -
21.600690653457377]
Fitness : -1.4572577
Value : [27.27267214116631, 22.527679785592284, 43.70467957721622,
0.8977841284674126,    0.2936053070323237,    -0.5037079078602067,    -
146.0462795910232]
Fitness : -1.0310336
Value : [10.438091832341112, 25.227982335130186, 28.75244194555195, -
0.4039391453068093,    -0.15795365517705884,    0.5255249617284312,    -
130.22498693833931
Fitness : -1.0549632
Value : [18.3959334884288, 21.958475276087924, 52.94422540199878,
0.674128624703652,       0.11220379171723671,       0.7431979552790942,
5.754510103674477]
Fitness : -0.7631226
Value : [-2.006912778439223, 0.42135521810002174, 53.893162589112166, -
0.9565992305865874,    0.42048626655052646,    0.8020206628426862,    -
73.36506679663208]
Fitness : -1.1231492
Value : [15.035652793711051, 28.639012053136746, 42.03177642079153, -
0.44335958270429243,     -0.27678846651320343,     -0.13889212126902994,
100.63458746286318]
Fitness : -0.751513
Value : [30.4521219600633, 22.239660947253917, 41.61666657883464,
0.2859019455491947,       0.7966147106906916,       0.4322375950703834,
151.82958949581024]
Fitness : -1.4114921
Value : [20.720044307847232, 23.16208256326633, 36.29157531269216,
0.41650938087421063,    0.6156450980921571,    0.18863002977384236,    -
174.05512662036486]
Fitness : -1.1660134
```

```
Value : [2.1533247053627242, 8.643910517067106, 61.080692163373584, -
0.7967094253379148,    -0.9656615591002371,    0.9314307709763332,    -
157.947967447995]
Fitness : -1.1341916
Value : [3.7383532019900017, -7.876775799008254, 51.84960370703236, -
0.5330592441672244,    -0.4314182699797904,    0.3661219387037309,    -
25.931887140544717]
Fitness : -0.9992304
Value : [19.296324521279697, 18.29937325885184, 57.30211581195983, -
0.543229720374705,    -0.19503385213916413,    -0.9573045798177835,    -
94.1875521520887]
Fitness : -1.1895329
Value : [25.266037915789376, 16.200728788536466, 36.51876379944147,
0.10398299072955575,      -0.8105925707094537,      -0.11752620008583547,
112.79272056821765]
Fitness : -1.7012948
Value : [14.751287303491786, 6.124918819458713, 32.04190025585667,
0.3331151918468509,   -0.004532504919650071,   -0.37080796638752767,   -
110.76210945297208]
Fitness : -1.2722436
Value : [-0.09879538750676975, 1.465275423647114, 51.014456759394335, -
0.9016637795828109,     0.6608748762823873,     0.3304961786374019,     -
12.140458123765626]
Fitness : -0.85211146
Value : [26.162379298279888, -9.171199243382897, 35.048870463663214,
0.12616592779640956,    -0.09808099321584307,    0.11879492735837394,    -
92.0022753892359]
Fitness : -1.0556862
Value : [24.670755262165805, 4.886248355901003, 31.02440650667864,
0.8330048453950025,       0.6011770671010555,       -0.7298813307636582,
135.6034896295189]
Fitness : -1.3648926
```

Trial 2

```
Value : [15.214635150212413, 17.100307520420504, 58.43310906348016,
0.1408953585636188,       -0.7983349201694443,       0.5237774480491504,
68.1231219186154]
Fitness : -1.1384112
Value : [10.706353814102638, -1.525892666141873, 36.252112215544244,
0.34967783759846394,      0.29301338216014994,      0.7647629383799519,
3.4328349719193056]
Fitness : -1.7056981
Value : [2.6521603201294823, -0.8486837032494101, 45.27660461623921, -
0.811781679213742,     0.4288673570060748,     0.643810877747595,     -
7.395938919736551]
Fitness : -1.3625687
Value : [12.282836058590533, 14.067123133325847, 24.754950027402906, -
0.7591572176797279,      -0.08209177391076561,      -0.6974380742274933,
176.5401471964144]
Fitness : -1.06265
Value : [23.108457882281364, 23.067832089429608, 40.46997842999707, -
0.5412146858303808,    0.42462865512714965,    0.4751821516756438,    -
158.53002595960186]
Fitness : -1.4395834
```

```
Value : [9.012393578838871, 4.67469177681253, 34.22649813555725,
0.06611823191696797, 0.07626783209266774, 0.1195928098208674, -
152.72100491492668]
Fitness : -1.3265364
Value : [22.303192401953616, 24.30334064072464, 43.82349864532673, -
0.6807631863594075, -0.06427067947499165, 0.605573048273297,
178.3150470833233]
Fitness : -0.98910743
Value : [4.086249336340401, -3.289717464938218, 52.894582379391686, -
0.09417200706542617, 0.029308481326737024, -0.3792687879946097, -
2.725419279342418]
Fitness : -1.1175725
Value : [19.529994572580517, 21.64421068784963, 38.41714956497256,
0.4274757911446523, -0.3986327379165393, -0.29874604823090056,
18.065511855637595]
Fitness : -1.1288289
Value : [14.867416114981033, -9.246968074185268, 58.70982522460132, -
0.49538636191913543, -0.0019589467362741697, -0.29117628486372404, -
82.94458337240911]
Fitness : -0.89487725
Value : [2.216801492008795, 0.9471196970004865, 50.93625134317252, -
0.697387171284171, 0.9617090681429321, -0.9625335160565134, -
138.45631275323453]
Fitness : -0.95440006
Value : [27.357880759835673, 4.324469761592644, 33.7270659100323, -
0.07395152236480373, -0.35469629440216566, -0.8733433671247499, -
172.39553447363608]
Fitness : -1.3261476
Value : [14.317361217689836, 18.493775230589378, 59.99843481449922, -
0.47313379120642973, -0.6773639786707859, -0.3531002343549017,
78.68533768180288]
Fitness : -1.2941121
Value : [25.814821301154375, 25.994880245749627, 48.64544366153293,
0.8097473712585699, -0.36766486769928863, -0.642749131983559, -
72.86474842683303]
Fitness : -0.9860452
Value : [13.76817220529793, 24.216007436654174, 53.84137115749359,
0.8772542579802849, 0.20569246913124095, 0.6004970331638755, -
100.855146067567]
Fitness : -0.8521777
Value : [2.9314939065297168, -2.61031013801491, 42.298599366914836, -
0.8550546254321527, -0.09316293505511575, -0.24180927625755655, -
3.625316218969175]
Fitness : -1.1987698
Value : [29.661395611493816, -0.28731203765127233, 35.76320815552326,
0.8822679621510188, -0.7024267251132792, -0.6295107011992975, -
60.97644747459785]
Fitness : -0.9007639
Value : [9.955930517566834, 22.945274609256643, 56.51274577261421,
0.22008411641295633, -0.1347117493464629, -0.3247229224028747, -
23.75748072472723]
Fitness : -0.834051
Value : [31.46517104191119, 11.334254072885109, 61.26681669257549,
0.6918823131088594, -0.5745850035514133, 0.4973334317723308,
50.40994466663946]
Fitness : -1.1590161
```

```
Value : [8.69033451140438, -0.1609673536220626, 35.74316589054993, -
0.11785881591200775,       0.6656846690901754,       -0.3892453637351152,
14.865221805981975]
Fitness : -0.98305136
```

Trial 3

```
Value : [6.738208291583703, -2.196720963717034, 58.05725052880482,
0.6865249561128055,      0.15018098147435555,      -0.47099125814307174,
171.1480160382888]
Fitness : -0.9278405
Value : [22.357536007684715, 23.72419417657089, 41.98089011904078, -
0.4634785729075117,      0.41388945262667787,      -0.13030758297817013,
48.99545712710963]
Fitness : -0.9821492
Value : [28.695851230276205, 14.780277568916276, 58.20754403340902,
0.4584793005224621,    0.35635449875318614,    0.07292560713936624,    -
163.9239150336056]
Fitness : -1.4683214
Value : [14.548713817409087, 4.578859045990328, 26.145884032214216, -
0.9170097706280294,    0.6169486505288908,    0.11776092405641969,    -
149.34343722512935]
Fitness : -1.1019083
Value : [19.971665929018, 23.101555083640065, 36.6322814909528,
0.9389943393066671,    0.030439116207955808,    -0.4375429698689304,    -
72.2149532318801]
Fitness : -1.5416254
Value : [8.904636760863799, 21.260413210980808, 51.97472768276821,
0.8557725913369401,      -0.11599783995485446,      -0.7437567891652543,
92.21327801948661]
Fitness : -0.8745466
Value : [21.27433326855815, 15.619857360884872, 58.31454405168671, -
0.10422519093258398,      0.8658759648874128,      -0.09629410353065682,
99.08117808805503]
Fitness : -0.8712174
Value : [2.418538445059097, -5.877930955446871, 52.50737141426633, -
0.5863491234127685,    0.6823609125551298,    -0.12938743878709058,    -
54.76850076305358]
Fitness : -0.80798125
Value : [-0.8388318189202453, 10.622823709951088, 60.93237615063024, -
0.8156151034400956,        0.5870367059938393,        0.526004792630099,
110.1808048086998]
Fitness : -1.0022393
Value : [23.363291058998364, 3.7734798793757403, 28.23985904378788,
0.8264340108869328,    -0.6945407891514497,    0.6101309048426216,    -
171.1153427797363]
Fitness : -1.438127
Value : [26.392580043802496, 10.980040738612402, 30.310774921545907, -
0.1753559003784806,    0.3827702523655816,    -0.07888108177706687,    -
161.1597417082758]
Fitness : -1.5751408
Value : [4.458553274891514, -1.0864579535389076, 50.24884511180139, -
0.7302145157817816,      0.11297282074903925,      0.35637109133402256,
81.58896505156997]
Fitness : -1.5934348
```

```
Value : [15.176585326534592, -0.6712761016940565, 28.8799761309815, -
0.5418363238104529, 0.8888248321194847, 0.9661516249987223,
145.6012809626671]
Fitness : -0.8249572
Value : [5.117338581122684, -1.432109324702708, 30.14581065036788, -
0.9577753565466895, 0.7407897251478828, 0.41589718330046277, -
87.25566113643212]
Fitness : -1.0612307
Value : [29.591966431969723, 19.466911474206164, 57.27982438469098,
0.8935226030101362, -0.8380164483060295, -0.3751775828485333, -
134.39527727904263]
Fitness : -1.1250979
Value : [5.803427114243329, 15.506986902145115, 62.01268846712004,
0.5231334474219005, -0.4756295830921706, -0.9738054626638051, -
137.01545812513072]
Fitness : -1.0501765
Value : [1.9772298777261295, 3.221193692656115, 57.81125503450572, -
0.21208702164122917, -0.5398865237869455, 0.9996993201353104, -
44.910459510762536]
Fitness : -1.2111524
Value : [-1.3184762679872124, -1.9481702051019205, 39.54478858729594,
0.5409694449731803, -0.7658120170169429, -0.2358507114583439,
72.65345676072371]
Fitness : -0.93498296
Value : [28.08294120717273, 10.045005254718376, 33.514676642171835, -
0.9282520474649709, -0.37407318132188183, -0.27388443704255927, -
7.4232403071666795]
Fitness : -1.1222608
Value : [10.686362137360202, -3.184554904104407, 62.857120668541675,
0.5383125498338248, 0.5186692794150252, 0.11369258491363499, -
51.31406713484071]
Fitness : -0.75656414
```

Trial 4

```
Value : [32.87520228170659, 5.710574668253734, 32.19390065299683,
0.0984253299420037, -0.5310127009519687, -0.12791630342952098, -
99.45903227511775]
Fitness : -1.4091579
Value : [10.937704002501043, -0.7956457005774205, 35.70086850299072,
0.8060191094896756, -0.04145623466534531 5, 0.03251645885848853, -
10.648993144320144]
Fitness : -1.6306947
Value : [9.97081850379405, -1.8908862787179341, 35.88727684511622, -
0.6615373341096225, -0.9426288923890882, 0.49459367032173995, -
21.740581454130734]
Fitness : -1.4905553
Value : [17.959757628807196, 13.227232600154327, 59.85634962793264, -
0.3501923946628023, 0.4958508204871126, -0.6198486894683384, -
66.21330663883468]
Fitness : -1.2378522
Value : [15.259474686982198, 19.702671396900094, 60.55099355804259, -
0.6719046718044948, -0.6034121835419526, 0.8360793976108458,
83.46054496955895]
```

```
Fitness : -0.6839804
Value : [17.8956180065043, 19.006060438874204, 51.883373286251036, -
0.8798573025107401, -0.07078341707991354, -0.12696823536522595, -
75.0026000089155]
Fitness : -1.2697407
Value : [27.744419739032615, 22.134416700042813, 45.76917424502505,
0.9942982281515056, -0.5546263635902506, -0.8979622633729663, -
53.14115994425066]
Fitness : -1.1784558
Value : [4.250387463458132, -4.43200874515678, 51.62227343308185,
0.4929323472007823, 0.9835286604131737, 0.2930598111048055,
5.233291101288415]
Fitness : -1.3986253
Value : [8.005958602348386, -1.3487599472041527, 32.966419670802445, -
0.8048815142662109, -0.029356415751513376, 0.33637411648539906, -
103.5100342898417]
Fitness : -2.2165177
Value : [26.952882829763418, 12.133835307184208, 28.319614338363742, -
0.7266696293516861, -0.9978692461303615, -0.30427454748288696, -
125.08137061316413]
Fitness : -1.5837493
Value : [10.363220564148328, 2.911709773449445, 37.37249770749948, -
0.6063630231396777, 0.25832554032571675, 0.20761521119444382,
76.2177609628348]
Fitness : -1.7031751
Value : [29.08336826489402, 16.41494267713874, 34.85786857979737,
0.9495366969840424, -0.41309942914571507, 0.44128630378361433, -
159.39335514329835]
Fitness : -1.3036252
Value : [6.851463980975218, -4.542168237625431, 36.8375062925208, -
0.6830482813008065, 0.01365706622322893, -0.24354167814824512,
48.90377739337072]
Fitness : -1.1239396
Value : [13.50393869218414, 17.9009634975759, 61.17339880650678, -
0.6066964776116617, -0.7138584276994748, 0.3205592931332679,
56.598506672143884]
Fitness : -1.325837
Value : [15.14126018939831, 13.353559456212071, 62.896131290839534, -
0.23463716702375392, -0.8134868056231457, 0.43189535035316085,
50.517508882743584]
Fitness : -1.2490892
Value : [0.48960432553026134, 1.9421882357677305, 52.620077629523685, -
0.4056078581667435, -0.19804567605506662, 0.2709593992081736, -
12.253015520923356]
Fitness : -1.0691457
Value : [6.470206282212182, -2.042330346377298, 33.40416433507261, -
0.30367944755429654, -0.7011826648670356, 0.7345619547958737, -
144.11942893910063]
Fitness : -0.8492727
Value : [-0.07443741768211432, 0.47514120496037826, 42.82559266615115,
0.286268248216198, 0.1588339448160736, 0.5892169607596167, -
59.40027756352781]
Fitness : -0.91852826
Value : [18.89184154878637, 23.440779174601943, 56.298902340579474,
0.26849635544083394, -0.03920042434641524, -0.14733702836655405, -
53.405161308554256]
```

```
Fitness : -1.0988394
Value : [12.340136113008942, 21.925427586777516, 54.061473758816526,
0.4831521891975221, 0.7458463432116429, -0.8420792608897991, -
69.77981305056569]
Fitness : -0.78007287
```

Trial 5

```
Value : [2.510941213875178, -0.7546847039603914, 47.405799208730144,
0.5346962018449708, -0.29066276748014297, -0.8073669958155945, -
9.71187207790544]
Fitness : -1.0463846
Value : [30.754876334792748, -1.4244079170647623, 37.43303538451225, -
0.46598381283349855, 0.5917915332280723, 0.8204935277138885,
49.89276830417987]
Fitness : -1.3050292
Value : [31.4619667373222, 20.534178656842588, 42.935677114068554, -
0.7996689313810315, 0.806279147926432, -0.537356218049887,
135.2912502606393]
Fitness : -1.5523905
Value : [12.456005366622982, 20.596349438688076, 51.584320270265025, -
0.3897100942359872, -0.18296627236977514, 0.55392063735237, -
102.70457061099243]
Fitness : -0.79411286
Value : [12.62526179116336, 26.66367953012461, 55.463470776566886, -
0.16450873554712064, -0.14011387367113515, 0.5730626935953702,
120.55396754451141]
Fitness : -0.12565197
Value : [8.105923452343326, -1.348759472041527, 32.91612312919135, -
0.7948815142662109, -0.1293564151513376, 0.2312338111539906, -
103.51123999121]
Fitness : -2.3009868
Value : [5.193925993373078, 5.110822341424829, 60.43564933760621,
0.12757394923200405, 0.9343107597314926, 0.6448361540185654,
25.61668383272695]
Fitness : -1.1431154
Value : [23.672504570824216, 21.704901305127837, 30.515517569412136, -
0.9521119584046807, 0.9113329922878277, -0.9306227157390639,
64.99253073746956]
Fitness : -0.739512
Value : [5.774920708947333, -7.775568257905088, 37.492803681920435,
0.7510303799383833, -0.6886932553657032, -0.8359330084703449,
24.333617177253643]
Fitness : -1.508003
Value : [29.98163052390037, 25.86536789904126, 47.1790647356391, -
0.12992193973761745, 0.6078820273956924, 0.8687434335573299,
94.05682956943372]
Fitness : -1.0914686
Value : [-0.19151811215012238, 9.299583714486051, 61.24124392182699, -
0.700434827836379, -0.1601473557038704, 0.7738665945278125, -
118.5866637429294]
Fitness : -0.7955834
```

```
Value : [25.514627655721327, 3.469937308147866, 30.972664236370576,
0.01275568871348498, 0.9235058692414471, 0.29461018668904604, -
72.90232809220123]
Fitness : -0.89856535
Value : [-0.4145352932025075, 5.965219965436749, 47.2563938406474, -
0.5247101219008135, -0.19357411401392688, -0.8837374064166688,
119.60938497468794]
Fitness : -1.3104028
Value : [23.949093189024044, 25.187700092817224, 43.497560226440264, -
0.7574566267026497, -0.835275960959887, -0.4891036523409009,
121.29105858468142]
Fitness : -0.87173575
Value : [-3.3217090651603813, 3.7912467296350947, 60.688724768477265, -
0.42801233948810236, -0.6432924092102583, 0.5073272219106255, -
84.39825559994104]
Fitness : -1.6317123
Value : [7.7608318505155935, -5.807535437748668, 52.01249589087192,
0.9107150843414993, -0.9649737620338876, -0.12608377260734183, -
98.50481909624443]
Fitness : -1.4214903
Value : [7.314198049137406, 27.54671210719703, 51.1497649542181, -
0.27633228759178197, -0.15274169440694974, -0.15680447256520735, -
156.75627774197713]
Fitness : -0.7709697
Value : [21.738361587431328, 1.9805490416645029, 26.473204394522377,
0.37032523228054415, -0.26599710764845597, 0.056979705044267526,
135.50665173717778]
Fitness : -1.324274
Value : [4.104461863784888, -6.4562348090170945, 53.44623194821452, -
0.8794671464507182, 0.21862398355591406, -0.9325567548566185,
14.124830457228654]
Fitness : -1.1970253
Value : [18.640151756990186, 21.968403988987575, 55.56958858741881,
0.40729351488606125, -0.12862373623948153, 0.0986134189366088, -
32.89676909382834]
Fitness : -1.148783
```

Trial 6

```
Value : [15.92470873370471, 19.536458286895378, 61.34180336793476, -
0.7261764868039526, -0.7626074482486841, 0.277865095476742,
61.914026864012584]
Fitness : -0.34206995
Value : [15.938604120709526, 19.536463821869646, 61.33929509572418, -
0.7279714020179319, -0.7633991212332203, 0.27796622854169106,
62.11237369023623]
Fitness : -0.34351876
Value : [15.938547772403295, 19.540590615722277, 61.347222055866716, -
0.7285171988129889, -0.7604112799167415, 0.2789828701326229,
62.14432801880676]
Fitness : -0.34357095
Value : [15.907888336198063, 19.528641182914093, 61.34251837909923, -
0.7272114662626908, -0.7642341682794124, 0.27921852065336666,
61.95184115511952]
Fitness : -0.34316152
```

```
Value : [15.887799522655568, 19.54449172400426, 61.35106510334029, -
0.7285216495529456, -0.7613735818413412, 0.2769414170727629,
61.82826627710573]
Fitness : -0.34130812
Value : [15.920998440587148, 19.527312909489627, 61.33770503299544, -
0.7268707488154106, -0.7620934542735025, 0.27684186969531027,
61.738566378927175]
Fitness : -0.34691563
Value : [15.908558082136755, 19.530150586035916, 61.352932199206535, -
0.7264084818279952, -0.7628942439747962, 0.27793709167560005,
61.73027017494782]
Fitness : -0.3413152
Value : [15.864383043403615, 19.52518812661491, 61.34707216202466, -
0.7258661486561176, -0.7618499978384898, 0.2766376378719709,
61.69879374869483]
Fitness : -0.34433642
Value : [15.910017236875333, 19.546891504751777, 61.338447168676474, -
0.7286469973127745, -0.7606733841632148, 0.2782515561539938,
62.129369359142935]
Fitness : -0.34377092
Value : [15.927957634329795, 19.530108075288044, 61.33770097251375, -
0.7282113820274876, -0.7605342156782767, 0.2787840959606466,
61.746123283272325]
Fitness : -0.3485336
Value : [15.944195860980962, 19.531941385263217, 61.34888920131316, -
0.7288393084669327, -0.7604128245469703, 0.2795979279842148,
61.64788435435241]
Fitness : -0.3463012
Value : [15.910885158268083, 19.540484440452925, 61.33975804500349, -
0.7269349127289917, -0.7621449661946094, 0.27670911513405977,
61.69643013883944]
Fitness : -0.34302685
Value : [15.864669625693415, 19.525075449555114, 61.340806425393275, -
0.7272696857374761, -0.7608069108210037, 0.27804399515341244,
61.6699469781239]
Fitness : -0.3478686
Value : [15.922951300196642, 19.53191595876632, 61.34017748528538, -
0.7284492184902265, -0.7626569844783218, 0.2785729345345151,
61.996392752508775]
Fitness : -0.34557334
Value : [15.881862215418778, 19.521302967364, 61.34143963374019, -
0.7254012281050661, -0.764242381236902, 0.27672193648696775,
61.73029359656551]
Fitness : -0.34414166
Value : [15.916157776110861, 19.523727354207956, 61.33959343027737, -
0.7273426692399564, -0.7633949044591946, 0.2789159409350855,
61.84186866168646]
Fitness : -0.34622025
Value : [15.89916930875253, 19.54524484909011, 61.34050883155553, -
0.7283096988888533, -0.7610488367325512, 0.27737059003748005,
61.88818459053186]
Fitness : -0.34348825
Value : [15.8662519551972, 19.544192187806726, 61.346822761180654, -
0.7281734595701166, -0.7621225626498953, 0.2792028550872105,
61.81539795685817]
Fitness : -0.34083378
```

```
Value   :   [16.46689726698264,   7.569168819812788,   30.29252070545231,
0.8626167155723983,     -0.48302310022904105,     0.262140035820531,     -
121.68447875833962]
Fitness : -1.8400162
Value : [-3.9634383444363612, 2.413080010795918, 60.493837978054316, -
0.6184202263416079,     -0.38164844722873226,     0.7926605650672356,     -
160.29430917669728]
Fitness : -0.77836627
```

Trial 7

```
Value   :   [1.053171828090143,   27.038804858586545,   49.93412994927574,
0.8427436147916343,        0.7830035726398168,        0.1262847597156629,
157.0091293502195]
Fitness : -1.113389
Value : [15.841041224004549, -1.2830769941426396, 28.313718376030295, -
0.17078756285487118,       0.5253131464356735,       0.06338379684572204,
134.06457398504682]
Fitness : -1.1255225
Value   :   [11.887088029680896,   3.518850311391464,   29.47253472945366,
0.7979580184465305,       -0.7675936638842507,       -0.7783494019245363,
75.0576757167427]
Fitness : -1.3217893
Value   :   [29.66182903770001,   24.39864225477693,   41.09068714425462,   -
0.3366717319460619,       -0.41321250847962143,       -0.4596411191601113,
178.30947245242783]
Fitness : -1.0414875
Value : [-6.0816570545830855, 0.08187032571855468, 61.61821531168208, -
0.6174307393372513,       -0.7885483347307507,       -0.7049504539206775,
37.78524710948301]
Fitness : -0.9200125
Value   :   [24.24092536285484,   26.133530227140294,   43.247994810502874,   -
0.7710494112567344,       -0.23495276753317007,       0.018376151821048747,
145.03023879902645]
Fitness : -1.0500259
Value   :   [9.296107496557436,   -2.2269422913116426,   35.37429048186152,
0.3805638029066638,     -0.13748450733025708,     0.609493924027326,     -
51.3103631492404]
Fitness : -1.114679
Value   :   [5.901217030305329,   -7.989772316537021,   55.28543181052035,   -
0.9091743805739092,     -0.9149659180351257,     0.6840636281998504,     -
6.953968249767797]
Fitness : -1.1613073
Value   :   [24.257831490510352,   26.10868891828788,   43.22235343995962,   -
0.77269593990448,        -0.2369486139587179,        0.02059502132853665,
144.95530008116071]
Fitness : -1.052526
Value : [-3.5628983785743866, 3.8794813084831787, 47.127111699391286, -
0.4138854201114912,    -0.05804494509279179,    -0.13654981267193156,    -
105.20568270883172]
Fitness : -1.0733503
Value   :   [-4.0476377762717135,   5.531420505348692,   59.53792269710992,   -
0.7637998414550036,     -0.8492856362012178,     0.6665999406720506,     -
104.55600053114848]
Fitness : -1.5642102
```

```
Value : [27.365923012911168, 20.784418619530307, 46.29059653496189, -
0.1364115186822048, 0.8984016445948264, -0.5980420063338139,
11.028186256203355]
Fitness : -1.2384403
Value : [12.02572195549866, 5.825322780736535, 28.017611396836976, -
0.6887354817929265, 0.31017905281943237, 0.12546680085620876,
177.53195070371197]
Fitness : -1.5771536
Value : [7.277068897869057, -0.972612923085741, 31.246158948548892, -
0.5932840417335532, 0.3151031173898611, 0.17014961897571612, -
168.91783097422103]
Fitness : -1.2610571
Value : [14.410895384606825, 20.348416547677967, 54.07759735641585,
0.2449055691611146, -0.614311747618248, -0.41323318976587387,
40.00795903211986]
Fitness : -1.0344161
Value : [8.184319392291892, 6.133378858855018, 60.1012339213758,
0.8021623987186937, -0.030297345463363934, 0.6311223609864549,
96.37859900086954]
Fitness : -0.9189059
Value : [0.6677349083877688, 4.593390971760353, 48.277879470108275,
0.13820679902491606, -0.633696130427662, -0.21450234325596984, -
120.77019500736697]
Fitness : -0.8807646
Value : [18.636677672477767, 21.169743113947895, 58.604836132472826,
0.6949254066175854, 0.6556744367060545, 0.8634382157843461, -
108.3054192698762]
Fitness : -0.96093774
Value : [22.159643731144428, 22.68492292638045, 40.685858320293804,
0.7131341376858276, -0.20953759078708245, 0.7556165191385211, -
87.13026855543859]
Fitness : -1.5460595
Value : [15.14417782760026, -6.1552208596859685, 27.834044643745443, -
0.03170285345726298, 0.7649864375211877, 0.6261510561004318,
167.06983650966606]
Fitness : -1.094102
```

Trial 8

```
Value : [14.526679068225715, 24.02456406536747, 51.03446602939565,
0.020504797350473503, -0.2562663419191944, -0.9237370782413454,
65.90179325856113]
Fitness : -0.7833326
Value : [23.028050260000253, 5.387796924116348, 29.598956115747086, -
0.5637910347169428, 0.3209928020395967, 0.5059700517886221,
97.62789861870067]
Fitness : -0.9478925
Value : [-3.0962123685598524, 8.761167377606906, 59.56694036783635, -
0.9539044104671286, -0.7100829060579321, 0.702595641342064, -
153.74976951999122]
Fitness : -0.8777632
Value : [27.145743817132733, 13.153583059481871, 34.255685087152486,
0.29753452879121256, 0.183088795936323, -0.6910526679985596, -
48.88885973641948]
Fitness : -0.8794618
```

```
Value : [15.55682683966005, 19.883630723625902, 59.301140134835585,
0.1205929296923775, -0.8701435767310587, -0.6288286130110843, -
43.04282389215753]
Fitness : -0.72941804
Value : [29.43709702270604, 5.213823042753841, 30.992408158854598,
0.018524834108419652, 0.5761001993529551, 0.9437280735591012,
164.65861461669334]
Fitness : -0.7500572
Value : [24.476665740719834, 17.207417821870592, 58.36369484665681, -
0.21272831163435302, -0.8009543404171438, -0.1723122111492934, -
50.79061417511164]
Fitness : -1.270539
Value : [3.9745240512905973, 2.39858802821802, 59.42853115809153,
0.8925171364851074, 0.6220490909096699, 0.0702747242964441, -
28.499218056478696]
Fitness : -0.93913037
Value : [9.702885493911884, 3.3471259365839963, 33.23984017967721, -
0.6137518581569642, 0.4277946210073716, 0.611314732685101,
138.14064841555398]
Fitness : -1.5109951
Value : [0.7612412243738493, 2.2857668249727343, 55.51803807133248, -
0.5576978399784707, 0.7998421741098294, 0.2780385388095399, -
72.50518144728417]
Fitness : -1.114154
Value : [11.721244406518132, 0.25401613617927765, 60.15375671458495,
0.5811160261460508, -0.8003325088697306, -0.8785860226148645, -
97.45313852323974]
Fitness : -1.2236922
Value : [-4.506349004757791, 7.491213845879702, 62.88763865009908,
0.7380994414086253, -0.9273418067096997, -0.19054003122835805, -
13.083865962597486]
Fitness : -1.5645019
Value : [-1.3353198676685656, 2.611626695799064, 44.653201333555046, -
0.3648695798569821, 0.7478995073918748, -0.12140683111098283, -
139.66129638715037]
Fitness : -1.2130345
Value : [26.275488738252335, 19.648313021028844, 45.2513216951386,
0.026028499690071705, 0.3762016782418065, -0.32158539282952603,
66.55498081416508]
Fitness : -1.4641868
Value : [19.886008509578808, 22.031903804733126, 38.427616917767295, -
0.8033651414016991, -0.8107496581788782, 0.14749549771839932,
37.45230799968459]
Fitness : -0.9008059
Value : [-4.240864664763662, 20.92037421464967, 55.033979512489296,
0.38788943829121303, -0.7521440994595716, -0.7847947011417524, -
2.2453783798154063]
Fitness : -1.4465935
Value : [3.5739808267165927, 0.4560287270130985, 30.489990601268786, -
0.5805416625044155, -0.8109461006539112, -0.8226997283358151, -
136.05567665288083]
Fitness : -0.90250623
Value : [10.65434123695244, -0.5713544963779995, 36.91899149170203,
0.7299141974377215, 0.9820001997188708, -0.14482641894620785,
13.600690544732402]
Fitness : -2.0153031
```

```
Value : [5.323437718897207, -6.1779912994937884, 52.092252466779385,
0.3151860185104802, -0.028870600418340997, 0.8202809207602142,
161.9126823969246]
Fitness : -0.74358565
Value : [8.987345182720532, 23.824601780843558, 51.96398152577993, -
0.1825715653080724, 0.6649618333574931, -0.5343891988577611, -
67.55624287273353]
Fitness : -1.1788652
```

Trial 9

```
Value : [12.095813581834623, 0.01643135170594512, 62.00873439497846, -
0.6893090288354793, 0.1188746161763008, -0.49579465055636684,
123.25345365237018]
Fitness : -1.0084409
Value : [24.595532559164354, 18.52127360313002, 31.533457425119742, -
0.45459051240671555, -0.18295471323548695, -0.5554648414209187,
79.01027082318188]
Fitness : -1.1566936
Value : [20.389764150770908, 20.757779403330876, 57.457015841521624, -
0.9317936681563774, 0.5411998657945747, 0.9232471846701866,
135.09534888727342]
Fitness : -1.3474711
Value : [11.185382765221462, 25.55648769483893, 48.96016872298745, -
0.2464655544895451, 0.8669886844356074, 0.7606782819531375,
133.15536556959322]
Fitness : -1.8877426
Value : [26.81619909442238, 8.210164439554376, 30.288428349081066,
0.37735792559701653, 0.4938879987736349, -0.8263847515932903,
135.77540920464634]
Fitness : -0.9175989
Value : [2.7632659037598923, -7.838802442809232, 48.134759468865205,
0.44299588824917024, -0.8237747079733928, 0.8045592574165887, -
48.35942286880828]
Fitness : -0.84657186
Value : [19.914369721862165, 22.50627101178901, 53.22765852523144,
0.2610080910952901, -0.4930324922316185, 0.4238349837349553, -
37.65742131485271]
Fitness : -0.8894469
Value : [12.951893787033399, -6.037178572550562, 57.973636448345765,
0.45285063739434483, 0.6606477971844913, 0.9258706887418897,
57.10920354404482]
Fitness : -1.205361
Value : [6.597534712758591, 24.1838919259639, 55.27156167420556, -
0.2483047365361879, -0.11377504062815569, 0.7625370071193618,
133.3112699506893]
Fitness : -1.1188569
Value : [29.606596509735667, 12.137851229582282, 33.95767575645738, -
0.24861995246344004, 0.9274926080152199, 0.3541350954612399,
59.66969549592312]
Fitness : -1.3343434
Value : [20.35076680202095, 24.78387845996464, 45.59612073967126,
0.8393287506784239, -0.8541103453359471, 0.6311987124301577, -
114.44462770814577]
Fitness : -1.2072058
```

```
Value : [15.732237988686862, 24.11711031014665, 50.18203487141569, -
0.487793495542173, -0.8491779732012614, 0.5181387264317081,
51.89843824181261]
Fitness : -0.76480764
Value : [18.181048785965903, 26.642676960660552, 34.61409554728192,
0.023364201022544107, 0.47587558224014326, 0.45450213411397855, -
158.9333858559464]
Fitness : -1.248122
Value : [5.789358283865022, -3.6822667275194725, 51.4400389154741, -
0.9358551969515112, 0.9643897370030345, -0.3344738503063762,
32.229816957226404]
Fitness : -1.6084081
Value : [14.222688538187064, 25.416987650457294, 54.03758995788887, -
0.24423757907341426, 0.2549780102982593, 0.760360860829477,
132.73970636954908]
Fitness : -0.78811246
Value : [12.107587001492515, 23.697310206800168, 55.26861980420589,
0.747035416567412, -0.6859403153730081, -0.36849597506678133, -
35.5713368949221]
Fitness : -0.7678296
Value : [25.452088980311913, 14.017909966240747, 36.485059291191654,
0.20383563057671505, 0.901128637145624, -0.8347977023552979,
177.64018357751684]
Fitness : -1.1867914
Value : [27.359058312143127, -4.854771455779144, 32.21911041323461, -
0.9130942773572128, 0.7460794714926318, -0.5117573422300132,
127.57860828885663]
Fitness : -0.8849146
Value : [19.913897561134466, 17.94642987549788, 56.95582400256166,
0.4632814693682763, 0.008516825698767994, 0.3531752713084324,
150.33757615369205]
Fitness : -1.1951776
Value : [8.204636177491707, -5.943710696949672, 31.668114219955022, -
0.5207729079560954, -0.7879634863125569, 0.3353485195325343, -
144.85514435094413]
Fitness : -1.4396135
```

Trial 10

```
Value : [8.114165918552127, -1.3424347911584629, 32.92515728766453, -
0.7927681429197808, -0.12499005290791686, 0.23413184037656398, -
103.50664045126643]
Fitness : -2.2996628
Value : [3.2672620137446877, -0.7677222857034298, 52.05827122591985,
0.9729514560443437, 0.1214962055999873, -0.8197371474077106, -
62.849897895283874]
Fitness : -0.90140533
Value : [13.994321949353218, 19.615762955997813, 53.98739699893798, -
0.5728237485354577, 0.33175203329143765, 0.6764100870385186, -
113.79834074448965]
Fitness : -1.5591221
Value : [29.67611472257944, 7.127817045219643, 33.03975557369641, -
0.3211330178934275, -0.0364670095532309, -0.4765080742527483,
157.8849120360535]
Fitness : -1.2563529
```

```
Value : [28.106431105788253, 10.014628163676953, 33.79295782167699,
0.046173611953934435, 0.5994122031334999, 0.17004077948045215,
18.730994652000334]
Fitness : -1.5903726
Value : [17.936471333887983, 25.278976522338027, 47.47314500606615, -
0.9117055023572984, -0.6967005347642463, -0.8822891665149639, -
161.94304133454236]
Fitness : -1.0515919
Value : [27.324117555530776, 14.092293884165848, 35.98239921603209,
0.6687288404533114, 0.5887519207010274, 0.1369319584559261,
73.45292539223931]
Fitness : -0.88626397
Value : [1.3995641749732286, 22.7015923581478, 61.823836779149865, -
0.9636545023153886, -0.6795701455532037, 0.28825707493270025,
51.97215495699186]
Fitness : -1.0975434
Value : [8.903683981653177, -5.862128132389517, 55.515691463424204,
0.25729104630176947, 0.5011021714480155, 0.19370614965342203, -
131.11746429970495]
Fitness : -1.0525874
Value : [17.266043739766694, 14.647811377346947, 62.95804364474142,
0.9215889737686089, 0.5684105169325913, 0.778945232613165, -
36.26993011381168]
Fitness : -1.0837208
Value : [16.883600519782014, 18.59074458751897, 60.64226550215873, -
0.2231503540098918, 0.25507774469471167, -0.030513942241351844,
110.8705403278218]
Fitness : -0.9649575
Value : [21.66391785080516, 15.155254838197685, 57.61628837816485, -
0.7705093134472336, -0.6737075748586678, 0.8101130158095413, -
85.31585031425853]
Fitness : -1.4834111
Value : [23.032187063570724, -8.786411475757465, 30.71611378851032,
0.16060884847474943, -0.5087006375848733, 0.29667422028684465,
178.35817856017871]
Fitness : -1.0691019
Value : [11.690251029275839, -7.32369754567374, 59.93585858734699,
0.0377482686343662, 0.06422727391475336, -0.47550481169978753, -
159.78839931789338]
Fitness : -0.8927131
Value : [7.080653825002628, -7.180626639365111, 53.113538810002275, -
0.7825238399534205, -0.29426370275351243, 0.27247999239184306, -
138.8753849516273]
Fitness : -1.5511171
Value : [20.792445341682917, 27.8258011712577, 39.329275077495225,
0.7394400524263263, 0.04354419333630455, 0.24739873291575187, -
127.04888700118741]
Fitness : -0.9338341
Value : [22.880888725012376, 23.16093878532616, 37.377371290615415,
0.5079009925766225, -0.06576709322276808, 0.8811019468514423, -
148.51839405449857]
Fitness : -1.1887782
Value : [9.660372353825437, 5.516800019398126, 60.48585543292893,
0.6879690343782685, -0.9446227155923186, -0.6864480595871427, -
90.58005128068258]
Fitness : -1.0333257
```

```
Value : [13.067056648355809, 25.340998886541414, 50.667332811781975, -
0.6492353482261042,      -0.12887661614129997,      -0.7107127889221228,
41.622910362777986]
Fitness : -1.1768689
Value  :  [19.790757309910795,  21.74369730108188,  37.15953867136561,
0.24832882265055045,     0.460437485634287,     0.7964797946285063,     -
4.530953475280938]
Fitness : -1.1227437
```

LAMPIRAN K

Posisi x, Posisi y, dan Posisi z Ligan SA2014

Percobaan ke-	Atom	Posisi x	Posisi y	Posisi z
1	C	7.230	26.088	52.484
	C	6.727	24.708	52.946
	N	7.193	23.434	52.217
	C	8.163	23.541	51.025
	C	8.667	24.922	50.563
	N	8.200	26.195	51.292
	N	9.636	25.028	49.371
	C	10.140	26.408	48.909
	C	9.673	27.682	49.638
	C	8.703	27.575	50.830
	C	8.630	22.268	50.296
	C	9.600	22.374	49.104
	C	10.103	23.754	48.642
	C	11.073	23.861	47.450
2	C	10.706	-1.526	36.252
	C	10.759	-2.840	35.450
	N	10.728	-2.805	33.911
	C	10.644	-1.455	33.174
	C	10.591	-0.141	33.975
	N	10.622	-0.176	35.514
	N	10.507	1.208	33.238
	C	10.455	2.522	34.039
	C	10.486	2.486	35.579
	C	10.570	1.137	36.316
	C	10.613	-1.420	31.634
	C	10.529	-0.070	30.897
	C	10.476	1.243	31.698
	C	10.392	2.593	30.961
3	C	4.459	-1.086	50.249

	C	5.321	-2.021	51.119
	N	5.619	-3.455	50.645
	C	5.056	-3.956	49.302
	C	4.194	-3.022	48.432
	N	3.895	-1.587	48.906
	N	3.631	-3.523	47.090
	C	2.768	-2.588	46.220
	C	2.470	-1.154	46.693
	C	3.033	-0.653	48.037
	C	5.354	-5.391	48.829
	C	4.791	-5.891	47.486
	C	3.929	-4.957	46.617
	C	3.366	-5.458	45.273
4	C	8.006	-1.349	32.966
	C	7.767	-0.336	31.830
	N	8.392	1.069	31.908
	C	9.256	1.462	33.122
	C	9.494	0.450	34.258
	N	8.869	-0.956	34.180
	N	10.357	0.843	35.470
	C	10.595	-0.170	36.606
	C	9.971	-1.575	36.529
	C	9.107	-1.968	35.315
	C	9.880	2.868	33.199
	C	10.744	3.260	34.413
	C	10.982	2.248	35.548
	C	11.845	2.641	36.762
5	C	8.106	-1.349	32.916
	C	7.640	-0.326	31.863
	N	7.906	1.176	32.078
	C	8.638	1.654	33.346
	C	9.104	0.631	34.399
	N	8.838	-0.870	34.184

	N	9.836	1.110	35.666
	C	10.302	0.086	36.719
	C	10.036	-1.415	36.504
	C	9.303	-1.893	35.236
	C	8.905	3.156	33.561
	C	9.637	3.634	34.829
	C	10.102	2.611	35.881
	C	10.834	3.089	37.149
6	C	16.467	7.569	30.293
	C	16.423	7.406	31.823
	N	15.296	6.585	32.478
	C	14.214	5.927	31.602
	C	14.257	6.091	30.070
	N	15.384	6.912	29.416
	N	13.175	5.434	29.195
	C	13.219	5.597	27.664
	C	14.345	6.418	27.010
	C	15.428	7.076	27.886
	C	13.087	5.106	32.256
	C	12.004	4.449	31.380
	C	12.049	4.613	29.850
	C	10.966	3.955	28.974
7	C	12.026	5.825	28.018
	C	13.220	6.588	28.620
	N	13.647	6.324	30.076
	C	12.879	5.297	30.929
	C	11.685	4.534	30.327
	N	11.258	4.798	28.871
	N	10.918	3.508	31.180
	C	9.723	2.745	30.577
	C	9.297	3.009	29.121
	C	10.064	4.036	28.268
	C	13.306	5.033	32.385

	C	12.539	4.006	33.239
	C	11.345	3.244	32.636
	C	10.577	2.217	33.489
8	C	10.654	-0.571	36.919
	C	10.457	-1.782	35.988
	N	10.171	-1.564	34.491
	C	10.083	-0.134	33.924
	C	10.280	1.077	34.855
	N	10.566	0.858	36.353
	N	10.192	2.506	34.289
	C	10.389	3.717	35.220
	C	10.675	3.498	36.717
	C	10.763	2.068	37.284
	C	9.797	0.084	32.427
	C	9.709	1.514	31.860
	C	9.907	2.724	32.791
	C	9.818	4.154	32.225
9	C	11.185	25.556	48.960
	C	11.920	24.523	48.086
	N	11.444	23.059	48.060
	C	10.233	22.628	48.909
	C	9.498	23.662	49.784
	N	9.975	25.126	49.809
	N	8.288	23.231	50.632
	C	7.553	24.264	51.506
	C	8.030	25.729	51.532
	C	9.241	26.159	50.683
	C	9.756	21.164	48.884
	C	8.546	20.733	49.733
	C	7.812	21.766	50.607
	C	6.601	21.336	51.456
10	C	8.114	-1.342	32.925
	C	7.658	-0.318	31.869

	N	7.939	1.182	32.080
	C	8.675	1.657	33.347
	C	9.132	0.632	34.402
	N	8.851	-0.867	34.192
	N	9.868	1.107	35.668
	C	10.324	0.083	36.724
	C	10.044	-1.417	36.513
	C	9.307	-1.892	35.247
	C	8.956	3.156	33.557
	C	9.693	3.631	34.824
	C	10.149	2.607	35.879
	C	10.885	3.082	37.145

BIODATA PENULIS

Fedric Fernando lahir di Mojokerto, 16 Februari 1994. Penulis telah menempuh pendidikan di SD Taruna Nusa Harapan SMP Taruna Nusa Harapan dan SMA Negeri 1 Sooko. Tahun 2017, Penulis mendapatkan gelar Sarjana Sains dari Departmen Matematika Institut Teknologi Sepuluh Nopember (ITS). Setelah lulus, penulis sempat bekerja di Dwidasa Samsara Indonesia sebagai Java Programmer, dan saat penulis berhasil mendapatkan beasiswa Fresh Graduate dari ITS. Penulis memutuskan untuk berkuliah terlebih dahulu.

Dalam masa studi Magister di Departmen Matematika ITS, penulis berkesempatan melakukan intership di Sirindhorn International Institute of Technology, Thammasat University, Thailand. Dan berhasil menyelesaikan tesis setelah kembali dari Thailand.

Jika ingin memberikan saran, kritik, dan diskusi mengenai tesis ini, bisa melalui email fedricfernando43@gmail.com. Semoga bermanfaat.