BAB II
TINJAUAN PUSTAKA

Bab ini berisi tentang dasar teori dari penelitian yang dilakukan. Adapun yang menjadi landasan teori adalah *Artificial Intelligence*, Algortitma optimisasi, buaya muara, *Particle Swarm Optimization*, *Grey Wolf Optimizer*, *Dragonfly Alghorithm*, *Whale Optimization Alghorithm*, Metode Numerik Newton Raphson dan Fungsi objektif *Unimodal* dan *Multimodal*.

2.1 Artificial Intelligence

Istilah *Artificial Intelligence* pertama kali dikemukakan pada tahun 1956 di konferensi Darthmouth (Russel, 1995) *Artificial Intelligence* atau kecerdasan buatan merupakan suatu sistem yang digunakan untuk memudahkan manusia untuk bekerja agar lebih efektif dan efisien (Kusumadewi, 2003). *Artificial Intelligence* (AI) sekarang dapat didefinisikan sebagai *acting rationally*. *Acting rationally* merupakan suatu tindakan rasional yang dilakukan untuk mengetahui Tindakan yang sesuai dengan sistem kecerdasan yang ingin diteliti dengan pendekatan *rational agent*. Hal ini didasarkan pada pemikiran bahwa komputer dapat melakukan penalaran secara logis dan juga melakukan aksi secara rasional berdasarkan hasil penalaran tersebut (Suyanto, 2014).

Artificial Intelligence (AI) banyak dikembangakan ke berbagai bidang contohnya pada optimisasi ada dua metode yang dikembangkaan yaitu metode heuristik dan metaheuristik (Hasad, 2011). Heuristik merupakan metode sebelum dikembangkannya metaheuristik (Purnomo, 2014) Heuristik merupakan metode yang menemukan solusi dengan mencoba-coba atau *trial and error* sehingga mendapatkan solusi yang dapat menyelesaikan suatu permasalahan yang kompleks dengan pengerjaan waktu sesingkat-singkatnya. Kompleksnya suatu permasalahan yang ingin diselesaikan membuat suatu permasalah sulit dicari solusi atau kombinasi yang terbaik dalam skala waktu tertentu. Ide dasarnya bahwa suatu

algoritma dapat menemukan solusi yang terbaik dan efisien dalam jangka waktu yang singkat dan diharapkan dari beberapa solusi menemukan hasil yang optimal atau mendekati hasil (Yang, 2010).

Algoritma metaheuristik banyak terinsipirasi dari alam yang dapat berupa *flora* dan *fauna*. Alam telah berevolusi dari tahun ke tahun sehingga dapat menemukan solusi sempurna untuk hampir semua permasalahan. Sehingga dapat dipelajari menyelesaikan permasalahan dari alam dan berkembang algoritma heuristic dan metaheuristik yang terinspirasi dari alam (Yang, 2010).

Dua komponen utama dari setiap algoritma metaheuristik adalah pemilihan solusi dan pengacakan terbaik. Pemilihan solusi yang terbaik bertujuan untuk menemukan solusi terbaik untuk optimalisasi. Sedangkan pengacakan yang terbaik menghindari atau mengatasi solusi yang masih terdiam atau terperangkap di lokal optima dan pada waktu yang sama meningkatkan keragaman solusi. Kombinasi terbaik dari keduanya adalah tercapainya global optimalisasi (Yang, 2010).

Algoritma metaheuristik dapat diklasifikasikan dalam berbagai cara. Salah satunya adalah dengan mengklasifikasikan sebagai populasi dan lintasan. Contohnya *Genetic Alghorithm* (GA) yang berbasis populasi dan berbagai jalur dan juga *Particle Swarm Optimization* (PSO) yang menggunakan berbagai partikel atau disebut juga algoritma berbasis partikel (Yang, 2010).

Pada metaheuristik ada dua jenis populasi yaitu *Single-Based Population* dan *Population-Based* Perbedaan antara *population-based* metaheuristik dibanding dengan *single-based* metaheuristik adalah *population-based* metaheuristik akan memanipulasi sekelompok populasi kandidat penyelesaian untuk setiap iterasinya. *Population-based* metaheuristik memiliki divergensi yang tinggi sehingga memiliki sifat ekspolarasi/diversifikasi yang lebih tinggi dibanding dengan *single-based* metaheuristik yang bersifat eksploitasi/intensifikasi, intensifikasi dan diversifikasi. Intensifikasi merupakan suatu teknik untuk mengeksploitasi suatu daerah pencarian yang terbatas, disekitar kandidat solusi yang didapat untuk menemukan solusi yang lebih baik. Insentifikasi bertujuan untuk mempercepat proses pencarian. Sedangkan diversifikasi adalah suatu proses untuk menjelajahi berbagai daerah di ruang pencarian. Diversifikasi dimaksudkan untuk menghindari konvergen yang terlalu dini karena pencarian terjebak pada suatu daerah yang

terbatas. Untuk mendapatkan hasil yang baik. Intensifikasi yang terlalu berlebihan membuat proses pencarian rawan terjebak dalam nilai optimal *local*. Di sisi lain, diversifikasi yang berlebihan membuat proses pencarian kurang terarah sehingga konvergensi berjalan lambat. Metaheuristik juga ada dua jenis yaitu dengan *single-based* metaheuristik dan *population based* metaheuristik (Purnomo, 2014). bahwa kecepatan waktu tiap algoritma optimisasi berbeda dikarenakan faktor seperti pemodelan matematis tiap algoritma yang dikatakan tidak efisien dan proses yang rumit dapat menyebabkan proses komputasi menjadi lama (Santosa, 2011). Sedangkan pada hasil iterasi yang didapatkan terdapat pada algoritma optimisasi ada yang berhasil dan ada juga yang tidak berhasil mencapai titik optimal minimum dikarenakan tidak semua algoritma dapat menyelesaikan semua masalah optimasi (Mirjalili,2016).

2.2 Algoritma Optimisasi

Algoritma optimisasi didefinisikan sebagai algoritma untuk mencari nilai di suatu fungsi tujuan atau fungsi objektif yang dapat berupa titik nilai minimum atau maksimum. Pada fungsi tersebut dapat dibuat suatu batasan nilai yang dapat berupa skalar atau vektor dari nilai-nilai kontinu atau diskrit (Hasad, 2011).

Algoritma optimisasi yang bertujuan untuk melakukan proses tercepat dengan mendapatkan hasil yang optimum. Hasil yang optimum dapat berupa permasalahan yaitu meminimalkan biaya, mempersingkat waktu, mengecilkan resiko maupun memaksimalkan keuntungan dan kerugian (Purnomo, 2014).

Algoritma optimisasi memiliki perbedaan dengan algoritma pencarian (*search alghorithm*). Pada algoritma pencarian terdapat kriteria tertentu untuk menyatakan suatu elemen merupakan solusi atau bukan. Sedangkan, algoritma optimisasi tidak memiliki kriteria tersebut akan tetapi, algoritma optimisasi pada fungsi objektif menggambarkan apakah suatu konfigurasi dapat digambarkan secara bagus atau tidak. Dapat dikatakan bahwa algoritma pencarian merupakan kasus khusus dari algoritma optimisasi (Hasad, 2011).

2.3 Buaya Muara

Buaya muara (*Crocodylus porosus*) merupakan salah satu dari tujuh jenis buaya yang hidup di Indonesia. Buaya Muara diketahui mencapai kedewasaan pada ukuran panjang 3-3,6 meter. Panjang minimum buaya muara pada saat memijah adalah 2,2 meter untuk buaya betina dan 3 meter untuk buaya jantan atau umur minimum 10 tahun untuk buaya betina dan umur 15 tahun untuk buaya jantan (Direktorat Jendral PHPA, 1985). Pendapat ahli lainnya mengemukakan bahwa buaya muara jantan dewasa mencapai dewasa kelamin pada ukuran panjang tubuh 2,9-3,3 meter dengan berat badan 80-160 kg, sedangkan buaya betina mencapai dewasa pada ukuran panjang minimum 2,4-2,8 meter mencapai dewasa diperkirakan 8-12 tahun. Buaya muara sesekali ke daratan untuk berjemur (Grzimek, 1975).

Gambar 2. 1 Ukuran Standar Buaya Muara (Winarno, 2018)

2.3.1 Perilaku Sosial

Buaya Muara mempunyai daerah teritori yang luas. Buaya sering merendam hampir seluruh badannya dalam air, tanpa mengganggu pernapasan dan penglihatannya sebab lubang hidung dan mata terletak pada sisi atas kepala. Perilaku sosial yang paling besar frekuensinya dari Buaya Muara adalah dominansi. Hal ini ditunjukkan dengan adanya perkelahian. Perkelahian pada buaya dapat

terjadi ketika dalam wilayah kekuasaan buaya dominan tersebut dimasuki oleh buaya lain. Hanya individu tertentu yang boleh masuk wilayahnya. Dari hasil pengamatan ada saatnya buaya menyendiri, karena menjaga wilayahnya atau kalah dalam persaingan (Winarno, 2018).

2.3.2 Perilaku Makan

Buaya Muara (*Crocodylus porosus*) memakan beragam jenis ikan dan mamalia seperti kambing, rusa, sapi bahkan manusia. Buaya muara memiliki otak yang cerdas sehingga dapat mempelajari pola atau kebiasaan mangsa sehingga buaya dapat mengetahui jam makan untuk berburu mangsa (Morpurgo dkk, 1993). Buaya Muara dapat dengan mudah memakan hewan terbang contohnya kekelawar di sekitar habitatnya, ikan dan katak. Perilaku Buaya muara juga lebih sering dengan strategi menerkam tiba-tiba mangsanya di perairan. Buaya Muara mendekati mangsa di dalam air dengan cara berkamuflase dengan mata telinga dan nostril tetap di permukaan air lalu menerkam mangsa ketika lengah untuk kemudian ditarik masuk ke dalam air hingga tenggelam (Winarno, 2018).

2.3.3 Perilaku Pergerakan

Buaya muara mampu melompat keluar dari air untuk menyerang mangsanya. Bahkan bilamana kedalaman air melebihi panjang tubuhnya, buaya muara mampu melompat serta menerkam secara vertikal mencapai ketinggian yang sama dengan panjang tubuhnya (Winarno, 2018). Buaya Muara terkenal juga sebagai jenis buaya terganas di dunia dikarenakan memburu mangsa secara agresif dengan cara penyergapan secara mendekati mangsa secara perlahan lalu menerkam mangsa secara tiba-tiba (Winarno, 2018).

2.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) dikembangkan oleh Kennedy dan Eberhart di tahun 1995, terinspirasi dari sekelompok ikan dan burung yang berada di alam dalam mencari makan. Pada perilaku ini kawanan burung atau ikan akan bergerak secara berkelompok dalam mencari makan dengan menentukan posisi terbaik dan kecepatan terbaik dalam mencari makan (Yang, 2010).

Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/*space* tertentu dan mengetahui posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaik kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut. Sehingga PSO dikembangkan dengan berdasarkan pada model berikut :

1. Ketika seekor burung mendekati target atau makanan (atau *minimum* atau *maximum* suatu fungsi tujuan), maka burung tersebut akan mengirim informasi kepada burung lainnya dalam populasi tertentu.
2. Burung yang lain akan mengikuti arah menuju ke makanan tetapi tidak secara langsung.
3. Ada hal yang diperhatikan pada tiap burung yaitu memori mengenai jalur yang sudah dilewati pada waktu sebelumnya.

Pada algoritma PSO (*Particle Swarm Optimization*) setiap individu atau partikel berperilaku merata dalam menggunakan kecerdasannya sendiri dan mempengaruhi kelompok kolektifnya. Jika ada individu yang menemukan jalan terbaik dan jalan terpendek dalam mencari makanan maka kelompok lain dalam suatu kelompok dapat mengikuti jalan tersebut juga (Kennedy, 1995). Pada hal ini setiap partikel atau setiap burung yang memiliki posisi terbaik yang disebut dengan *local best* dan posisi partikel terbaik dari semua populasi disebut *global best*. Hal ini mencari lintasan terbaik dalam ruang pencarian. Pada tiap iterasi yang dilakukan akan direpresentasikan pada posisi partikel terbaik lalu dievaluasi ke dalam fungsi objektif atau *fitness function* (Yang, 2010)

Setiap partikel memiliki dual hal yaitu posisi partikel dan kecepatan partikel. (Kennedy, 1995). Berikut formulasi matematika penggambaran posisi dan kecepatan partikel pada suatu dimensi ruang tertentu:

$$X_i(t) = x_{i1}(t), x_{i2}(t), \dots, x_{iN}(t) \quad (2.1)$$

$$V_i(t) = v_{i1}(t), v_{i2}(t), \dots, v_{iN}(t) \quad (2.2)$$

Keterangan persamaan dari (2.1) dan (2.2) yaitu $X_i^L = x_{i1}^L, x_{i2}^L, \dots, x_{iN}^L$ merepresentasikan *local best* dari partikel ke- . i sedangkan $X_i^G = x_{i1}^G, x_{i2}^G, \dots, x_{iN}^G$ merepresentasikan *global best* dari seluruh kawanan X adalah posisi partikel, V

adalah kecepatan partikel, i adalah indeks partikel, t adalah iterasi ke- dan N adalah ukuran dimensi ruang

Berikut merupakan model matematika yang menggambarkan mekanisme pembaruan status partikel:

$$V_i^k(t) = V_i(t-1) + c_1 r_1 \left(X_i^L - X_i(t-1)\right) + c_2 r_2 \left(X^G - X_i(t-1)\right) \quad (2.3)$$

$$X_i^k(t) = V_i(t) + X_i(t-1) \quad (2.4)$$

Keterangan dari persamaan (2.3) dan (2.4) yaitu c_1 dan c_2 adalah suatu konstanta yang bernilai positif yang biasanya disebut sebagai *learning factor*. Kemudian r_1 dan r_2 adalah suatu bilangan random yang bernilai antara 0 sampai 1. (Yang, 2010)

Particle Swarm Optimization memiliki *pseudocode* sebagai berikut:

```
Initialize population of particles
Initialize parameter c1,c2,r1,r2,
Initialize each particle position and velocity randomly
Evaluate the fitness of each particle and initial position as PBest
While (t < iterationmax)
    Select the GBest particle in the swarm
    For i = 1:nParticle
        Calculate the velocity of particle (vi)
        Update the position of particle (xi)
    End for i
    For i = 1:nParticle
        Evaluate the fitness of updated particle xi
        If f(xi) < f (PBest)
        Then set current position as PBest
        End if
    End for i
    Find the best particle
End while
```

Gambar 2.2 *Pseudocode Particle Swarm Optimization* (Kennedy, 1995)

2.5 *Grey Wolf Optimizer*

Filosofi alam dari Algoritma *Grey Wolf Optimizer* (GWO) meniru hirarki kepemimpinan dan mekanisme perburuan serigala abu-abu di alam. Seperti empat jenis serigala abu-abu yaitu *Alfa, Beta, Delta* dan *Omega* digunakan untuk mensimulasikan hirarki kepemimpinan. Serigala *Alfa* akan menjadi pemimpin dan pembuat keputusan dalam kelompok untuk mencari mangsa dan menyerang mangsanya. Kelompok serigala *beta* adalah penasehat kelompok *alfa* dan bertanggung jawab untuk menjaga disiplin dalam kelompok. Serigala *delta* adalah penjaga perawatan kebutuhan serigala dan mengawasi wilayah serigala. Serigala *omega* dalam kelompok bertindak sebagai pendukung alpha untuk menemukan dan menyerang mangsa dan mereka juga mematuhi perintah kelompok lain (Mirjalili, 2014). Fase utama perburuan serigala abu- abu sebagai berikut:

- Melacak, mengejar dan mendekati mangsa
- Mengejar, melingkari dan menjatuhkan mangsa sampai berhenti bergerak
- Menyerang mangsa

Formulasi matematika yang digunakan pada algoritma ini sebagai berikut:

$$\vec{D} = \left|\vec{C}.\vec{X}p(t) - \vec{X}(t)\right| \quad (2.5)$$

$$\vec{X}(t+1) = \vec{X}p(t) - \vec{A}.\vec{D} \quad (2.6)$$

$$D_\alpha = \left|\vec{C}_1\vec{X}_\alpha - \vec{X}\right|, D_\beta = \left|\vec{C}_2\vec{X}_\beta - \vec{X}\right|, D_\delta = \left|\vec{C}_3\vec{X}_\delta - \vec{X}\right| \quad (2.7)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \quad (2.8)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (2.9)$$

Keterangan persamaan. (2.5), (2.6), (2.7) dan (2.8) yaitu D_α, D_β dan D_δ adalah jarak antara mangsa dengan serigala alpha, serigala beta dan serigala omega. $\vec{X}_1$, $\vec{X}_2$ dan $\vec{X}_3$ adalah posisi serigala dalam menyerang mangsa$t$ adalah indikasi iterasi saat ini. $\vec{A}$ dan $\vec{C}$ adalah koefisien vektor. $\vec{X}p$ adalah vektor posisi dari mangsa. $\vec{X}$ adalah indikasi vektor posisi dari serigala. Vektor-vektor $\vec{A}$ dan $\vec{C}$ dihitung sebagai berikut:

$\vec{A} = 2\vec{a}.\overrightarrow{r1} - \vec{a}$ (2.10)

$\vec{C} = 2.\overrightarrow{r2}$ (2.11)

Keterangan persamaan (2.7) dan (2.8) yaitu $\vec{a}$ adalah konstanta yang menurun secara linear dari 2 ke 0 selama iterasi dan $\overrightarrow{r1}$ dan $\overrightarrow{r2}$ adalah vector acak [0, 1] (Mirjalili, 2014).

Grey Wolf Optimizer memiliki *pseudocode* sebagai berikut:

Initialize the grey wolf population X_i *(i = 1, 2, ..., n)*
Initialize a, A, and C
Calculate the fitness of each search agent
X_α*=the best search agent*
X_β*=the second best search agent*
X_δ*=the third best search agent*
while *(t < Max number of iterations)*
  for *each search agent*
    Update the position of the current search agent
  end for
  Update a, A, and C
  Calculate the fitness of all search agents
  Update X_α, X_β, *and* X_δ
  t=t+1
end while
return X_α

Gambar 2. 2 *Pseudocode Grey Wolf Optimizer* (Mirjalili, 2014)

2.6 Whale Optimization Alghorithm

Whale Optimization Alghorithm (WOA) adalah salah satu algoritma optimasi yang dikembangkan oleh Sayedali Mirjalili dan Andew Lewis. *Whale* atau ikan paus merupakan salah satu mamalia terbesar di dunia. Terdapat 7 jenis ikan paus salah satunya adalah *humpack whale*/paus bungkuk. Ikan paus *humpback* memiliki metode berburu bernama *bubble-net feeding*. *Whale Optimization Alghorithm* memiliki beberapa parameter internal yaitu jumlah agen pencari (n), jumlah iterasi maksimum (t), dan vektor $\vec{A}$. Terdapat tiga model matematika yang digunakan pada teknik *bubble-net feeding* dari algoritma ini:

- *Encircling Prey/* Mengelilingi Mangsa
- *Bubble-Net Attack* / Fase Eksploitasi

- *Search for prey* / Pencarian mangsa (eksplorasi)

Pemodelan matematis dari *Whale Optimization Alghorithm* (WOA) dapat dirumuskan sebagai berikut:

1. *Encircling Prey/* Mengelilingi Mangsa

   Pada tahap ini *humpback whale* atau paus bungkuk mengenali lokasi dari target mangsa, setelah itu paus bungkuk ini akan mengelilingi target. Oleh karena itu posisi optimal dalam ruang pencarian tidak diketahui secara apriori, dengan demikian algoritma WOA mengasumsikan bahwa solusi terbaik saat ini adalah target mangsa yang dikelilingi atau dapat dikatakan posisi optimum saat ini adalah mendekati optimal. Setelah *search agent*/ikan pencari terbaik didefinisikan, maka ikan yang lain akan mencoba memperbarui posisi-posisi mereka mendekati posisi ikan terbaik. Perilaku ini dapat ditunjukkan oleh persamaan berikut

   $$\vec{D} = |\vec{C}\ X^*(t) - X(t)| \quad (2.12)$$

   $$X(t+1) = X^*(t) - \vec{A}\vec{D} \quad (2.13)$$

   Dimana t menunjukkan iterasi saat ini, $\vec{A}$ dan $\vec{C}$ adalah koefisien vektor, X^* merupakan vektor posisi dari solusi terbaik yang didapatkan saat ini, X adalah vektor posisi yang akan diperbaharui, sedangkan $||$ adalah nilai absolut. Vektor $\vec{A}$ dan $\vec{C}$ dihitung sebagai berikut.

$$= 2 \quad \vec{A} \quad \vec{a}.\overrightarrow{r1} - \quad \vec{a} \quad (2.14)$$

$$\vec{C} = 2.\overrightarrow{r2} \quad (2.15)$$

   Keterangan persamaan (2.10) dan (2.11) yaitu $\vec{a}$ adalah menurun secara linear dari 2 ke 0 selama iterasi dan $\overrightarrow{r1}$ dan $\overrightarrow{r2}$ adalah vector acak [0, 1] (Mirjalili,2016).

2. *Bubble-Net Attack* / Fase Eksploitasi

   Perlu diperhatikan bahwa ikan paus bungkuk berenang melingkar dalam lingkaran melengkung dan sepanjang jalur spiral secara bersamaan. Untuk memodelkan perilaku simultan ini, diasumsikan terdapat peluang 50% ikan paus ke *i* memilih antara mekanisme *shrinking encircling* atau *spiral*

updating position untuk memperbarui posisi selama iterasi. Sehingga dapat dimodelkan sebagai berikut:

$$X(t+1)=\begin{cases} X^*(t) - \vec{A}.\vec{D} & if\ P < 0.5 \\ \overrightarrow{D'}.e^{bl}.cos2\pi l + X^*(t) & if\ P \geq 0.5 \end{cases} \quad (2.16)$$

Dimana p adalah bilangan random pada interval $0 \leq p \leq 1$ atau [0,1] (Mirjalili,2016).

3. *Search for prey* / Pencarian mangsa (eksplorasi)/

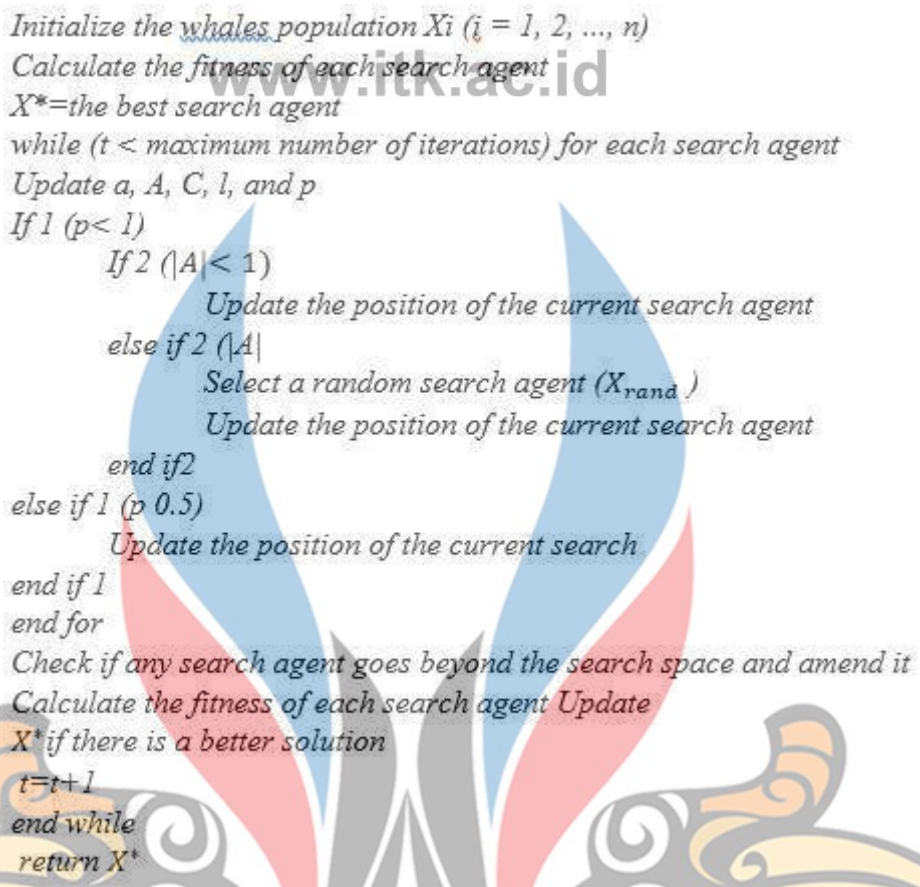
   Pendekatan yang sama berdasarkan variasi vektor $\vec{A}$ dapat digunakan pada tahap pencarian mangsa atau fase eksplorasi. Pada kenyataannya, ikan paus mencari mangsa secara random dengan memperhatikan posisi satu sama lain. Oleh karena itu, digunakan nilai random $\vec{A}$ lebih besar dari 1 atau kurang dari -1 untuk memaksa ikan pencari i bergerak menjauh dari ikan acuan. Fase ekplorasi ini akan dilakukan jika $|A|>1$, pada fase ekplorasi ini algoritma WOA akan melakukan pencarian global. Model matematisnya sebagai berikut:

   $$\vec{D} = |\vec{C}\ .\vec{X}_{rand} - X| \quad (2.17)$$

   $$X(t+1) = \vec{X}_{rand} - \vec{A}.\vec{D} \quad (2.18)$$

   Dimana $\vec{X}_{rand}$ adalah vektor posisi acak yang dipilih sesuai dengan jumlah populasi ikan (Mirjalili,2016).

Whale Optimization Alghorithm memiliki *Pseudocode* sebagai berikut:



Gambar 2. 3 *Pseudocode Whale Optimization Algorithm* (Mirjalili, 2016)

2.7 Dragonfly Alghorithm

Dragofly Algorithm (DA) adalah salah satu teknik optimasi yang dikembangkan berdasarkan perilaku statis dan dinamis capung di alam. Terdapat 5 gerakan capung terhadap lingkungannya. Pertama kohesi, yaitu pola gerak capung untuk mendekati kawanannya yang berada di sekitar capung tersebut. Kedua bergerak seirama, yaitu gerak capung yang beriringan dengan kawanannya. Ketiga pemisahan, gerak capung yang menjauhi kawanannya untuk mencari sesuatu. Keempat mencari makan, yaitu pergerakan capung dalam mencari makanannya. dan kelima menghindari musuh, gerakan capung dalam melindungi diri atau menjauhi musuh. Pola gerakan makhluk hidup tersebut dapat dimodelkan dalam persamaan matematis yang bisa digunakan untuk membuat kecerdasan buatan (*artificial intelligent*) (Mirjalili, 2016a).

` Tujuan utama dari setiap gerombolan capung adalah bertahan hidup, sehingga semua individu harus memiliki hasrat mencari makan atau menghindari

musuh. Dari 5 pola gerakan capung yang disebutkan diatas, persamaan dari model matematisnya pemisahan (*seperation*):

$$S_I = -\sum_{j=1}^{n} X - X_j \quad (2.19)$$

Dimana *X* adalah posisi individu saat ini, X_j menunjukkan posisi individu tetangga dan *n* adalah jumlah individu tetangga.

Gerak seirama dihitung sebagai berikut:

$$A_i = \frac{\sum_{j=1}^{N} Vj}{N} \quad (2.20)$$

Di mana V_j menunjukkan kecepatan individu tetangga. Gerak *cohesion* dihitung sebagai berikut:

$$C_i = \frac{\sum_{j=1}^{N} Xj}{N} - X \quad (2.21)$$

Dimana *X* adalah posisi individu saat ini, *N* adalah jumlah capung tetangga dan X_j menunjukkan posisi capung tetangga. Untuk gerak mencari makan:

$$F_i = X^+ - X \quad (2.22)$$

Dimana X dalah posisi individu saat ini, dan $\boldsymbol{X}^+$ adalah posisi makanan. Untuk gerak menghindari musuh:

$$E_i = X^- + X \quad (2.23)$$

Dimana $\boldsymbol{X}^-$ adalah posisi musuhnya. Vektor arah:

$$\nabla X_{t+1} = (sS_i + aA_i + cC_i + eE_i + fF_i + w\nabla X_t \quad (2.24)$$

Dan untuk vector posisinya:

$$X_{t+1} = X_t + \nabla X_{t+1} \quad (2.25)$$

Dan untuk *Levy flights* saat tidak ada capung lain disekitar capung tersebut, berikut model matematisnya:

$$X_{t+1} = X_t + Levy(d) \times X_t \quad (2.26)$$

Dragonfly Alghorithm memiliki *Pseudocode* sebagai berikut:

```
Initialize the dragonflies population Xi (i = 1, 2, ..., n)
Initialize step vectors ΔXi (i = 1, 2, ..., n)
while the end condition is not satisfied
    Calculate the objective values of all dragonflies
    Update the food source and enemy
    Update w, s, a, c, f, and e
    Calculate S, A, C, F, and E
    Update neighbouring radius
    if a dragonfly has at least one neighbouring dragonfly
        Update velocity vector
        Update position vector
    else
        Update position vector
    end if
    Check and correct the new positions based on the boundaries of variables
end while
```

Gambar 2. 4 *Pseudecode Dragonfly Algorithm* (Mirjalili, 2016a)

2.8 Metode Numerik *Newton Raphson*

Metode Newton Raphson yang mengambil nama dari Issac Newton dan Joseph Raphson Metode penentuan akar–akar persamaan, metode *Newton Raphson* merupakan metoda yang paling banyak dipakai dalam sains terapan dan rekayasa. Metode ini memiliki konvergensinya paling cepat diantara metoda lainnya. Pada *Newton Raphson* terdapat pendekatan yang digunakan dalam penurunan persamaan metode *Newton Raphson* yaitu penurunan metoda *Newton Raphson* secara geometri. Adapun langkah–langkah penentuan akar dari suatu persamaan dengan metode *Newton Raphson* ini dinyatakan sebagai berikut (Afrianita, 2015).

1. Tentukan taksiran awal akar untuk fungsi *f(x).* Untuk taksiran awal dari akar fungsi *f(x)* dinyatakan dalam bentuk xa
2. Tentukan turunan pertama dari fungsi *f(x).* Untuk turunan pertama dari *f(x)* dinyatakan dalam bentuk *f'(x)*
3. Lakukan evaluasi f(x) dan *f'(x)* untuk x = xa
4. Hitung pendekatan akar yang baru dari fungsi *f(x)* dengan menggunakan persamaan (2.27) berikut:

$$x_{t+1} = x_{t-1} - \frac{f(x_i)}{f'(x_i)} \text{ dimana } f'(x_i) \neq 0 \qquad (2.27)$$

5. Periksa kesalahan relatif (*Er*) hasil perhitungan dimana
   a. Jika kesalahan relatif (*Er*) besar dari tingkat ketelitian yang diizinkan maka perhitungan diulangi lagi langkah ke 2 sampai langkah ke 4.
   b. Jika kesalahan relatif (*Er*) kecil dari tingkat ketelitian yang diizinkan maka perhitungan selesai. Untuk perhitungan kesalahan relatif digunakan persamaan (2.28)

$$E_r = \left|\frac{X_{r(i+1)} - X_{r(i)}}{X_{r(i+1)}}\right| 100\% \quad (2.28)$$

Keterangan persamaan. (2.27) dan (2.28) yaitu x_{r+1} adalah variable iterasi selanjutnya x_r variable *x*, $f(x_r)$adalah varibel pada fungsi *x*, $f'(x_r)$ adalah variable turunan fungsi *x*. dan E_r adalah kesalahan relative (Afrianita, 2015)

2.9 Fungsi Objektif

Fungsi objektif adalah fungsi yang dapat dioptimumkan dari segi minimum atau maksimum (Kristanto, 2015). Fungsi objektif ini digunakan pada metaheuristik digunakan dalam dalam proses pencarian solusi dari suatu permasalahan. Ketepatan dan ketelitian dalam merumuskan fungsi objektif menentukan keberhasilan metode metaheuristik (Purnomo, 2014) Pada studi algoritma optimisasi baru dibutuhkan tes fungsi uji objektif *unimodal* dan *multimodal*. Fungsi uji adalah masalah buatan, dan bisa jadi digunakan untuk mengevaluasi perilaku suatu algoritma dalam terkadang beragam dan sulit situasi. Masalah *artificial* dapat mencakup minimum global tunggal, tunggal atau ganda minimum global di hadapan banyak minimum lokal, lembah sempit panjang, ruang kosong, dan permukaan rata. Masalah tersebut dapat dimodifikasi dan dimanipulasi untuk diuji algoritma dalam berbagai skenario. Di sisi lain, masalah dunia nyata berasal dari berbagai bidang seperti fisika, kimia, teknik, matematika, dll. Ini masalah sulit untuk dimanipulasi dan mungkin mengandung aljabar atau diferensial yang rumit ekspresi dan mungkin memerlukan sejumlah besar data untuk dikompilasi. Sebuah koleksi dari masalah optimisasi tidak terbatas (Jamil,2013).

2.9.1 Fungsi Uji *Unimodal*

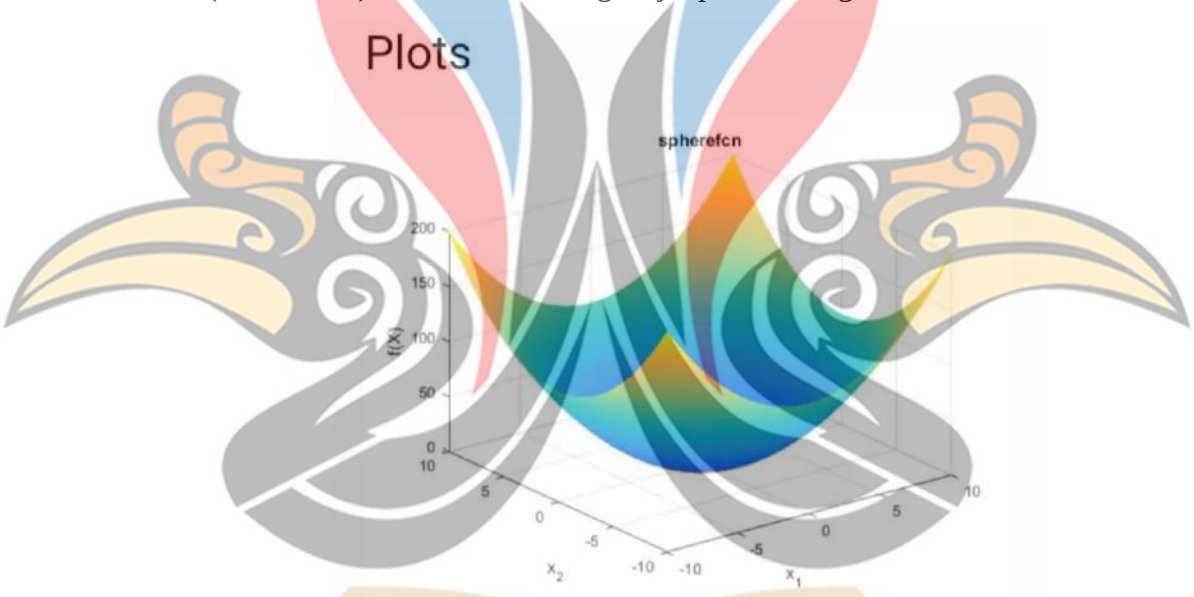
Fungsi uji *unimodal* merupakan fungsi uji yang memiliki satu titik nilai optimum dari segi minimum atau maksimum. Contoh fungsi uji *unimodal* (Jamil,2013).

A. Fungsi Uji *Sphere*

Fungsi *sphere* diperkenalkan pertama kali oleh Schumer and Steiglitz, pada tahun 1968 Berdasarkan fungsi uji *sphere* didapatkan persamaan matematis sebagai berikut:

$$f(x) = f(x_1, x_2, \dots\dots, x_n) = \sum_{i=1}^{n} x_i^2 \quad (2.29)$$

Batas bawah dan atas dari fungsi *sphere* yaitu dari titik $-10 \leq 10$. Global minimum terbaiknya yaitu $x = f(0,\cdots,0)$, dan nilai fungsi $f(x) = 0$ (Jamil,2013). Gambar dari fungsi uji *Sphere* sebagai berikut



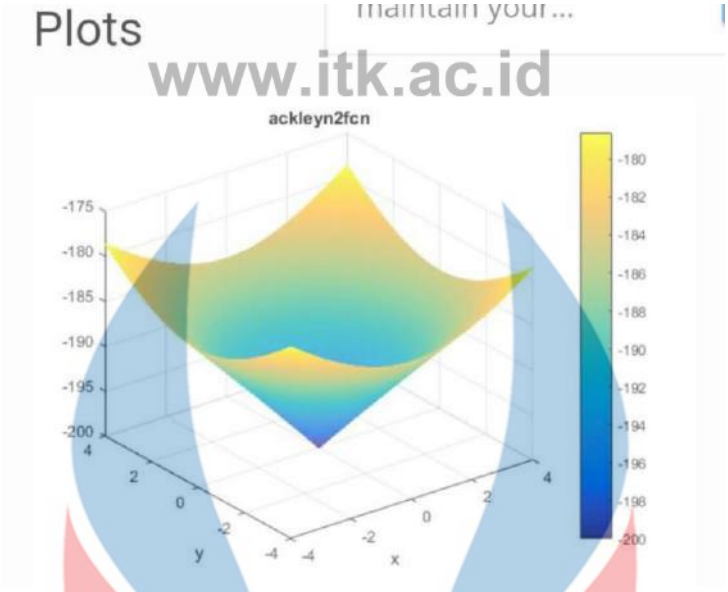
Gambar 2. 5 Fungsi Uji *Sphere* (Jamil, 2013)

B. Fungsi Uji *Ackley* N.2

Fungsi *Ackley* N.2 ditemukan oleh Ackley pada tahun 1987. Berdasarkan fungsi uji *Ackley* N.2 didapatkan persamaan matematis sebagai berikut:

$$f(x,y) = -200e^{-0.2\sqrt{x^2+y^2}} \quad (2.30)$$

Batas bawah dan atas dari fungsi *Ackley* N.2 yaitu $-32 \leq x \leq 32$. Global minimum terbaiknya yaitu $x = f(0,0)$, dan nilai fungsi $f(x) = -200$ (Jamil,2013). Gambar dari fungsi uji *Ackley* N.2 sebagai berikut



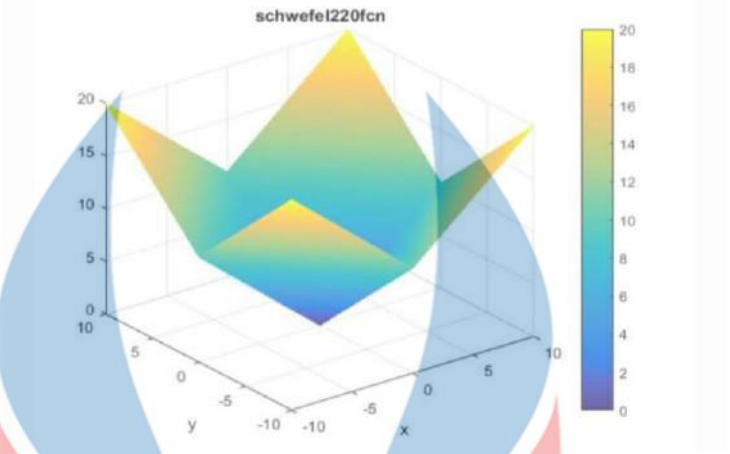
Gambar 2. 6 Fungsi Uji *Ackley* N.2 (Jamil, 2013)

C. Fungsi Uji *Schwefel* 2.20

Fungsi *Schwefel* 2.20 ditemukan oleh Schwefel pada tahun 1981. Berdasarkan fungsi uji *Schwefel* 2.20 didapatkan persamaan matematis sebagai berikut:

$$f(x) = f(x_1, \dots, x_n) = \sum_{i=1}^{n} |x_i| \quad (2.31)$$

Batas bawah dan atas dari fungsi *Schwefel* 2.20 yaitu $-100 \leq x \leq 100$. Global minimum terbaik yaitu $x = f(0, \dots, 0)$, dan nilai fungsi $f(x) = 0$ (Jamil,2013). Gambar dari fungsi uji *Schwefel* 2.20 sebagai berikut



Gambar 2. 7 Fungsi Uji *Schwefel* 2.20 (Jamil, 2013)

2.9.2 Fungsi Uji *Multimodal*

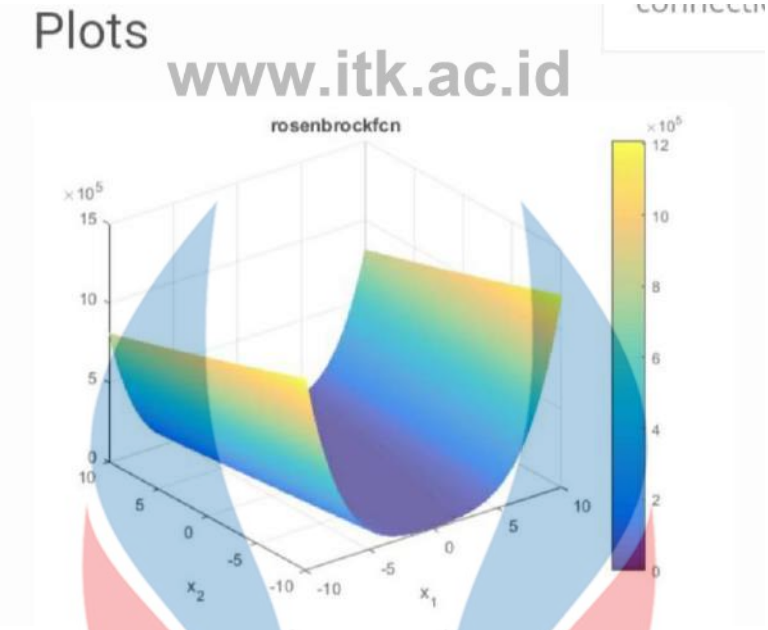
Fungsi uji *multimodal* merupakan fungsi uji yang memiliki lebih dari satu titik nilai optimum dari segi minimum atau maksimum. Contoh fungsi uji *multimodal* (Jamil,2013).

A. Fungsi Uji *Rosenbrock*

Fungsi uji *Rosenbrock* ditemukan oleh Rosenbrock pada tahun 1960 Berdasarkan fungsi uji *Rosenbrock* didapatkan persamaan matematis sebagai berikut:

$$f(x) = \sum_{i=1}^{n} [\mathrm{b}(x_{i+1} - x_i^2)^2 + (a - x_i)^2 \quad (2.32)$$

Batas bawah dan atas dari fungsi *Rosenbrock* yaitu $-5 \leq x \leq 10$. Global minimum terbaik yaitu $x = f(1,\ldots,1)$, dan nilai fungsi $f(x) = 0$. Sedangkan konstanta $a = 1$ dan konstanta $b = 100$ (Jamil,2013). Gambar dari fungsi uji *Rosenbrock* sebagai berikut



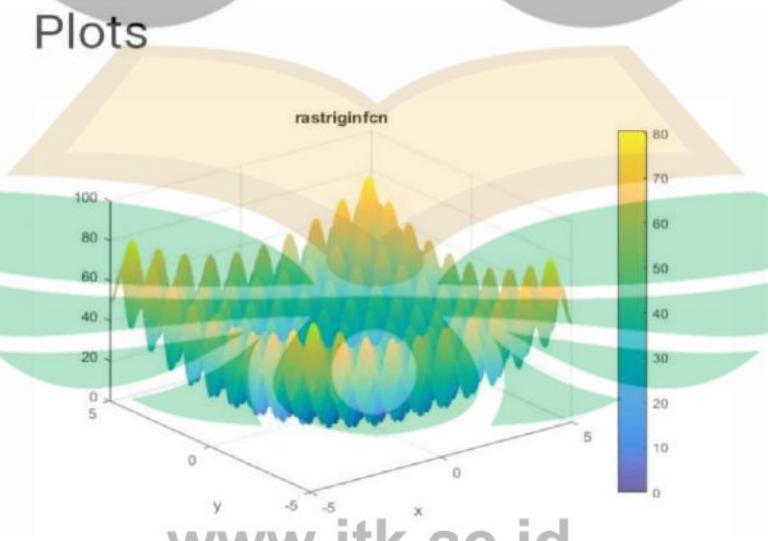
Gambar 2. 8 Fungsi Uji *Rosenbrock* (Jamil, 2013)

B. Fungsi Uji *Rastrigin*

Fungsi uji *Rastrigin* ditemukan oleh Rastrigin pada tahun 1961 Berdasarkan fungsi uji *Rastrigin* didapatkan persamaan matematis sebagai berikut:

$$f(x) = 10n + \sum_{i=0}^{n} x_i^2 - 10\cos(2\pi x_i)) \quad (2.33)$$

Batas bawah dan atas dari fungsi *Rastrigin* yaitu $-5.12 \leq x \leq 5.12$. Global minimum terbaik yaitu $x = f(0,\cdots,0)$, dan nilai fungsi $f(x = 0$ (Jamil,2013). Gambar dari fungsi uji *Rastrigin* sebagai berikut



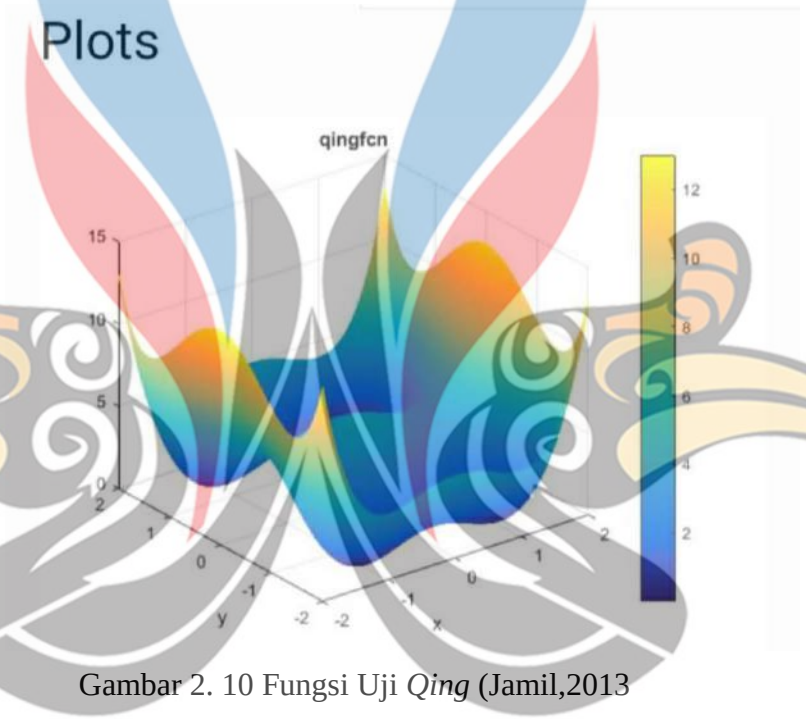
Gambar 2. 9 Fungsi Uji *Rastrigin* (Jamil,2013)

C. Fungsi Uji *Qing*

Fungsi uji *Qing* ditemukan oleh Qing pada tahun 2006. Berdasarkan fungsi uji *Qing* didapatkan persamaan matematis sebagai berikut:

$$f(x) = f(x_1, \dots, x_n) = \sum_{i=1}^{n} (x^2 - i)^2 \quad (2.34)$$

Batas bawah dan atas dari fungsi *Qing* yaitu $-500 \leq x \leq 500$. Global minimum terbaik $x = f(\pm\sqrt{i})$, dan nilai fungsi $f(x) = 0$ (Jamil,2013). Gambar dari fungsi uji *Qing* sebagai berikut



Gambar 2. 10 Fungsi Uji *Qing* (Jamil,2013

2.10 Gambaran Umum Posisi Penelitian

Berikut merupakan tabel gambaran umum penelitian dalam tugas akhir yang akan dilakukan.

Tabel 2.1 Gambaran Umum Posisi Penelitian

NO	Peneliti	Tahun	Judul	Keterangan
1	James Kennedy dan Eberhart	1995	*Particle Swarm Optimization*	Pemodelan algoritma didasarkan oleh perilaku sekawanan burung atau ikan.
2	Seyedali Mirjalili	2014	*Grey Wolf Optimizer*	Pemodelan algoritma di inspirasi oleh perilaku serigala abu-abu pada saat berburu
3	Seyedali Mirjalili	2016	*Whale Optimization Alghorithm*	Pemodelan algoritma di inspirasi oleh paus bungkuk berburu mangsa
4	Seyedali Mirjalili	2016	*Dragonfly Alghorithm*	Pemodelan algoritma di inspirasi oleh perilaku capung mencari makanan
5	Ibnu Afdhal	2020	Perilaku Buaya sebagai pemodelan algoritma optimisasi	Pemodelan perilaku buaya sebagai pemodelan algoritma optimisasi

Halaman ini sengaja dikosongkan