# BAB II
# TINJAUAN PUSTAKA



Bab ini berisi tentang dasar teori dari penelitian yang dilakukan. Adapun yang menjadi landasan teori adalah Bekantan, *Artificial Intelligence*, Algoritma optimisasi, *Particle Swarm Optimization, Moth-Flame Optimization*, *Multi-Verse Optimizer*, *Artificial Bee Colony,* Metode Numerik *Newton Raphson, Levy Flight,* dan Fungsi Objektif.

## 2.1 Bekantan

Bekantan (Nasalis Iarvatus Wurmb) merupakan primata endemik Borneo (Arief, 2006). Primata yang digolongkan rentan oleh IUCN, dimasukkan ke dalam Apendiks I CITES, dan dilindungi peraturan perundang-undangan Pemerintah RI ini telah ditetapkan sebagai fauna identitas Provinsi Kalimantan Selatan berdasarkan Surat Keputusan (SK) Gurbernur Kepala Daerah Tingkat I Kalimantan Selatan No. 29 Tahun 1990 tanggal 16 Januari 1990 (Arief, 2006).

Bekantan mendapat perhatian sangat tinggi dalam upaya konservasinya. Jenis ini tergolong langka dan endemik, dengan habitat terbatas pada hutan bakau, hutan di sekitar sungai, dan habitat rawa gambut dimana sebagian telah terancam oleh berbagai aktivitas manusia. Menurut Mcneely et al. (1990), dari 29.500 km persegi habitat bekantan, saat ini telah berkurang seluas 40% sedangkan yang berstatus Kawasan konservasi hanya 4,1% pada tahun 2000. Akibat dari penurunan luas habitat tersebut maka populasi bekantan cenderung menurun karena primate ini kurang toleran terhadap kerusakan habitat (Wilson dan Wilson, 1975; yeager, 1992).

Secara morfologi, warna rambut bekantan bervariasi. Di bagian bahu dan punggung atas berwarna coklat kemerahan. Ujung-ujung rambutnya berwarna merah kecoklatan sedangkan dua pertiganya berwarna abu-abu. Punggung atas berwarna kuning keabuan, perut berwarna kekuningan atau abu-abu, kadang-kadang ada bagian yang berwana kuning kecoklatan. Tangan dan kaki putih

kekuningan, kepala berwarna coklat kemerahan, dan leher berwarna putih ke abuan. Ciri khas bekantan yang mudah dikenali adalah ukuran hidung yang besar dan Panjang pada bekantan jantan seperti yang ditunjukan pada gambar 2.1 dan runcing pada bekantan betina (Bismark, 2009).

Gambar 2. 1 Bekantan Jantan Dewasa (Bismark, 2009)

Di Kalimantan Selatan, bekantan diketahui menghuni tipe habitat, yang berupa hutan mangrove, hutan campuran di tepi pantai, disepanjang atau di muara sungai, hutan rawa gambut, hutan rawa yang didominasi gelam *Melaleuca cajuputi*, hutan yang tumbuh di atas bukit kapur, serta hutan karet *Hevea brasiliensis*. Di habitat tersebut primata endemik Borneo ini melakukan aktivitas harian, baik yang dilakukan secara individu maupun dalam kerangka interaksi social. Secara umum aktivitas harian mencakup berpindah dari tapak tertentu ke tapak lain atau menetap di tapak tertentu selama waktu tertentu dan selanjutnya melakukan kegiatan makan, istirahat, bermain, menyelisik, atau aktivitas lain. Tujuan utama beraktivitas adalah untuk mempertahankan hidup (Arief, 2006).

Pada struktur kelompok populasi bekantan dibangun oleh kelompok-kelompok bekantan dengan jumlah individu yang sangat bervariasi, yaitu antara 6-16 individu per kelompok, 3-17 individu, dan 17-25 individu (Bismark, 1994). Kelompok bekantan akan melakukan pembentukan sub-kelompok yang dimulai sejak bekantan meninggalkan pohon tempat beristirahat, pembentukan sub-kelompok ini berhubungan dengan efisiensi pemanfaatan waktu mencari makan dan

penghematan energi untuk aktivitas pergerakan dari kelompok bekantan. Sebaran sub-kelompok bekantan ini memiliki jarak antar kelompok 50-150 meter satu dengan yang lainnya juga bertujuan untuk mengurangi kemungkinan terjadinya konflik antara individu dalam kelompok dan bermanfaat dalam pengontrolan daerah jelajahnya. Persebaran sub-kelompok akan dipertemukan kembali pada saat akan kembali ke pohon yang berada dekat sungai untuk bermalam (Bismark, 2009).

Primata bekantan mempunyai sistem pencernaan mirip ruminansi yaitu sistem pencernaan melakukan proses fermentasi makanan oleh bakteri yang ada pada perut bekantan yang membuat bekantan dapat menetralisir pengaruh racun yang berasal dari tumbuhan yang dimakan. Bekantan merupakan primata folivorous (Pemakan daun) yang mendapatkan protein esensial dari daun yang dikonsumsi. Bekantan tidak hanya memakan daun, ada juga bunga dan buah yang berada di ujung cabang pohon, pada saat bekantan ingin memetik pakan bekantan akan berusaha meraih ranting di sekitarnya atau duduk di atas ranting pohon yang berada di ujung-ujung pohon (Bismark, 2009).

Pakan utama yang dikonsumsi bekantan adalah daun muda dengan urutan 1 sampai 3 dari ujung ranting. Pakan tersebut dapat diambil langsung oleh bekantan dengan mulut atau dengan cara memetik. Daun yang dimakan satu per satu atau dua lembar dengan cara menggigit hingga tiga kali. Setiap gigitan dikunyah 10 hingga 30 kali, sehingga dalam 5 menit bekantan dapat mengkonsumsi 7,5 lembar daun. Aktivitas makan bekantan tidak menetap pada satu pohon saja, dan selalu diikuti aktivitas berjalan diantara cabang atau dari pohon ke pohon lainnya. Dalam hal ini perpindahan lokasi makan bekantan terjadi antara 10 hingga 15 menit. Selain itu jumlah dan sebaran pohon pakan yang merata memungkinkan untuk menunjang aktivitas makan bekantan di setiap saat (Bismark, 2009).

## 2.2 *Artificial Intelligence*

Teknologi kecerdasan buatan atau *Artificial Intelligence* sekarang sangat mempengaruhi kehidupan manusia. Kecerdasan buatan tanpa disadari kini berada didalam kehidupan sehari-hari kita, bisa kita lihat bahwa peralatan-peralatan yang memanfaatkan teknologi yang ada disekitar kita kini telah menggunakan kercerdasan buatan, contohnya yang berada dirumah seperti radio, mesin cuci,

kulkas, ponsel pintar, komputer dan lainnya dilengkapi dengan *Integrated Circuit* komputer yang memiliki kecerdasan buatan didalamnya maka dari itu kecerdasan buatan merupakan ilmu dan rekayasa yang membuat mesin mempunyai intelligensi atau kecerdasan sendiri yang unik, khususnya program komputer yang cerdas (Suyanto, 2011). Soft computing merupakan inovasi baru dalam membangun sistem yang cerdas yaitu sistem yang memiliki keahlian seperti manusia pada domain tertentu, mampu beradaptasi dan belajar agar dapat bekerja lebih baik jika terjadi perubahan lingkungan. Soft computing mengeksploitasi adanya toleransi terhadap ketidaktepatan, ketidakpastian dan kebenaran parsial untuk dapat diselesaikan dan dikendalikan dengan mudah agar dapat disesuaikan dengan kenyataan (Zadeh, 1998).

*Artificial Intelligence* atau kecerdasan buatan merupakan salah satu hal yang dikembangkan oleh para ahli untuk memudahkan manusia untuk mengerjakan suatu hal menjadi lebih efektif dan efisien (Kusumadewi, 2003). *Artifical Intelligence* dapat didefinisikan sebagai *acting rationally* dengan pendekatan *rational agent* atau agent yang selalu bertindak memaksimalkan kinerja, karena pada pemikiran bahwa komputer dapat melakukan penalaran secara logis dan juga melakukan aksi secara rasional berdasarkan hasil penalaran tersebut (Suyanto, 2014). *Artifical Intelligence* banyak dikembangkan ke berbagai bidang, salah satu contohnya adalah permasalahan optimisasi dimana terdapat dua metode yang dikembangkan yaitu metode heuristik dan metaheuristik (Hasad, 2011).

Heuristik merupakan metode yang menemukan solusi dengan mencoba- coba atau *trial and eror* 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 terinspirasi dari alam yang dapat berupa *flora* dan *fauna*. Alam telah juga 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 agar menemukan solusi yang terbaik ke optimisasi. Sedangkan pengacakan yang terbaik menghindari atau mengatasi solusi yang masih terdiam atau terperangkap di optimal lokal dan pada waktu yang sama meningkatkan keragaman solusi. Kombinasi yang terbaik dari kedua komponen tersebut bahwa optimisasi global dapat tercapai (Yang, 2010).

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

## 2.3 Algoritma Optimisasi

Optimisasi menurut kamus besar Bahasa Indonesia adalah nilai tertinggi, terbaik, paling menguntungkan, mengoptimalkan berarti menjadi paling tinggi, menjadikan nilai paling maksimal. Optimaliasi adalah proses pencarian solusi terbaik, tidak selalu keuntungan yang paling tinggi yang bisa dicapai jika tujuan pengoptimalan adalah memaksimumkan keuntungan, atau tidak selalu biaya yang paling kecil yang bisa di tekan jika tujuan pengoptimalan adalah meminimumkan biaya.

Algoritma Optimisasi (*Optimization Algorithms*) dapat didefinisikan sebagai algoritma untuk mendapatkan nilai x dengan keanekaragaman sehingga dapat menghasilkan nilai fungsi f(x) yang bernilai sekecil atau sebesar mungkin untuk suatu permasalahan fungsi yang diberikan, yang dapat juga disertai dengan beberapa batasan pada nilai x. dimana nilai x bisa berupa skalar atau vektor dari nilai-nilai kontinu maupun nilai diskrit (Hasad, 2011).

Global Optimization didefinisikan sebagai suatu cabang ilmu dari matematika dan Analisa numerik yang membahas optimisasi dengan kriteria yang bersifat tunggal, ganda atau yang memiliki banyak kriteria. Kriteria diekspresikan sebagai

himpunan fungsi matematika dimana kriteria tersebut mencakup kesulitan-kesulitan tertentu atau disebut fungsi-fungsi objektif. Fungsi objektif atau fungsi tujuan (*Objective Function*) adalah fungsi linear yang digunakan untuk mendapatkan atau mencari nilai optimum dan memiliki batasan-batasan pertidaksamaan linear yang memiliki himpunan penyelesaian pada suatu permasalahan pada program liniear. Himpunan penyelesaian yang ada merupakan titik-titik koordinat yang disubstitusikan kedalam fungsi linear untuk mendapatkan nilai dari suatu fungsi tersebut (Hasad, 2011).

Sebagian besar algoritma metaheuristik terinspirasi dari alam karena dikembangkan berdasarkan abstraksi alam. Alam telah berevolusi jutaan tahun dan telah menemukan solusi sempurna untuk hamper semua masalah yang ada pada alam, dengan demikian kita dapat mempelajari keberhasilan pemecahan masalah dari alam dan mengembangkan algoritma heuristik dan atau metaheuristik yang terinspirasi dari alam. Terkhusus lagi, beberapa algoritma yang terinspirasi dari alam diinspirasi oleh teori evolusi Darwin, yang menyebabkan algoritma tersebut dikatakan terinspirasi dari biologi atau yang mendekatinya (Yang, 2010).

Dua aspek utama dari setiap algoritma metaheuristik adalah, pemilihan solusi terbaik dan pengacakan. Pemilihan yang terbaik memastikan bahwa solusi akan bernilai memusat dengan optimalitas. Sementara pengacakan bertujuan untuk menghindari solusi yang terjebak pada local optima dan, pada saat yang sama akan meningkatkan keragaman solusi. Kombinasi yang baik dari kedua aspek tersebut biasanya akan memastikan bahwa optimalitas secara keseluruhan dapat dicapai. algoritma metaheuristic dapat diklasifikasikan dalam beberapa cara. Salah satunya adalah mengklasifikasikan algoritma berdasarkan populasi dan berdasarkan lintasan. Misalnya algoritma genetika berbasis populasi karena menggunakan serangkaian lintasan, demikian juga *Particle Swarm Optimization* (PSO) yang menggunakan banyak agen atau partikel. PSO juga disebut algoritma berbasis agen (Yang, 2010).

## 2.4 *Particle Swarm Optimization*

*Particle Swarm Optimization* (PSO) merupakan algoritma yang didasarkan pada perilaku sekawanan dimana pada salah satu contohnya adalah sekawanan

burung. Algoritma PSO meniru perilaku *social organisme* ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu lain dalam waktu suatu kelompok. Kata partikel menunjukkan seekor burung dalam sebuah kawanan. Setiap individu berperilaku dengan cara menggunakan kecerdasannya atau pemikiran sendiri dan juga dipengaruhi kelompok kolektifnya (Santosa, 2011).

Pada *Particle Swarm Optimization* (PSO), kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak disuatu lokasi yang acak dalam ruang pencarian multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/space tertentu dan mengingat 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 (Santosa, 2011).

Pada algoritma PSO pencarian solusi dilakukan oleh suatu populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara random dengan Batasan nilai terkecil dan terbesar. Setiap partikel merepresentasikan posisi atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuaian terhadap posisi partikel terbaik dari partikel tersebut (*local best*) dan penyesuaian terhadap posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performansinya dengan cara memasukkan solusi tersebut dalam f*itness function* (Santosa, 2011).

Setiap partikel diperlakukan seperti sebuah titik pada suatu dimensi ruang tertentu. Kemudian terdapat dua faktor yang memberikan karakter terhadap status partikel pada ruang pencarian yaitu posisi partikel dan kecepatan partikel (Kennedy, 1995). Berikut formulasi matematika penggambaran posisi dan kecepatan partikel pada suatu dimensi ruang tertentu:

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

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

dimana *X* merepresentasikan posisi partikel, *V* merepresentasikan kecepatan partikel, *i* adalah indeks partikel, *t* menunjukan iterasi ke-*t,* dan *N* adalah ukuran dimensi ruang.

Berikut merupakan model matematika yang menggambarkan mekanisme pembaruan status partikel:

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

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

dimana

$X_i^L = x_{i1}^L, x_{i2}^L, \ldots, x_{iN}^L$ merepresentasikan local best dari partikel ke-i. sedangkan $X^G = x_{i1}^G, x_{i2}^G, \ldots, x_{iN}^G$ merepresentasikan global best dari seluruh kawanan. Sedangkan $c_1$ 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 (Santosa, 2011).

*Particle Swarm Optimization* memiliki *pseudocode* yang dapat dilihat pada gambar 2.2 berikut:

```
For each particle
   Initialize particle
END
Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in history
         set current value as the new pBest
   End

   Choose the particle with the best fitness value of all the particles as the gBest
   For each particle
      Calculate particle velocity
      Update particle position
   End
While maximum iterations or minimum error criteria is not attained
```

Gambar 2. 2 *Pseudocode Particle Swarm Optimization* (Kennedy, 1995)

## 2.5 *Moth-Flame Optimization*



*Moth-Flame Optimization* (MFO) *algorithm* merupakan algoritma optimisasi metaheuristik yang terinspirasi dari alam. Inspirasi utama pada algoritma optimisasi ini adalah metode navigasi dari serangga ngengat yang berada di alam, pergerakan navigasi ini disebut dengan orientasi melintang. Serangan ngengat terbang di malam hari dengan mempertahankan sudut tetap terhadap bulan, mekanisme ini sangat efektif untuk melakukan perjalanan dalam garis lurus untuk jarak yang jauh. Akan tetapi, serangga ngengat ini mudah terperangkap pada jalur spiral yang tidak berguna disaat berada disekitaran lampu buatan. Pada algoritma *Moth-Flame Optimization* ini memodelkan secara matematis perilaku serangga ini untuk melakukan optimisasi (Mirjalili, 2015).

Pada algoritma *Moth-Flame Optimization*, diasumsikan bahwa solusi kandidat adalah ngengat dan variabel permasalahan adalah posisi ngengat di dalam ruang penyelesaian yang memiliki satu, dua, atau lebih dimensi yang merupakan vektor posisi. Karena *Moth-Flame Optimization* merupakan algoritma yang berbasis populasi maka populasi ngengat di wakilkan sebagai matriks, dan untuk keseluruhan ngengat juga diasumsikan sebagai himpunan penyimpanan untuk nilai *fitness* (Mirjalili, 2015).

Algoritma *Moth-Flame Optimization* ini merupakan pendekatan yang menggunakan tiga baris pendekatan untuk nilai global optimal yang di representasikan sebagai persamaan berikut:

$$MFO = (I, P, T) \tag{2.5}$$

Dimana $I$ adalah sebuah fungsi yang menginisiasi *random* populasi dari hewan ngengat dan sama dengan nilai *fitness*. Permodelan persamaan dari fungsi ini ditunjukan sebagai berikut:

$$I: \emptyset \rightarrow \{M, OM\} \tag{2.6}$$

Dimana $M$ merupakan matriks yang merepresentasikan populasi, dan $OM$ merupakan *array* yang memuat nilai fitness yang sesuai dengan populasi. Fungsi $P$ merupakan fungsi utama pergerakan ngengat disekitar ruang pencarian, fungsi ini didapatkan dari matriks $M$ dan dikembalikan nilainya setelah di *Update*.

$$P: M \rightarrow M \tag{2.7}$$

Fungsi *T* merupakan pengembalian dan pengecekan kriteria dimana bernilai salah atau benar dari matriks *M*



$$T: M \rightarrow \{true, false\} \quad (2.8)$$

Dengan *I*, *P*, dan *T*, secara garis besar algoritma *Moth-Flame Optimization* memiliki kerangka dasar yang dapat dilihat sebagai *pseudocode* berikut:

```
M = I();
while T(M) is equal to false
  M = P(M);
end
```

Gambar 2. 3 Kerangka Dasar *Moth-Flame Optimization* (Mirjalili, 2015)

Fungsi *I* harus menginisiasi solusi dan menghitung nilai dari fungsi objektif, dan dimana pada fungsi ini dapat digunakan pendistribusian random. Dengan *j* adalah besar dimensi, Penggunaan dapat dilihat sebagai berikut:

```
for i = 1: n
  for j = 1: d
    M(i,j) = (ub(i) – lb(i)) * rand() + lb(i);
  end
end
OM = FitnessFunction(M);
```

Gambar 2. 4 *Pseudocode* Fungsi I *Moth-Flame Optimization* (Mirjalili, 2015)

Terdapat dua *array* yang disebut *ub* dan *lb*, kedua *array* tersebut merupakan definisi dari batas atas dan batas bawah variables (Mirjalili, 2015).

Setelah di inisiasi, iterasi fungsi *P* dijalankan hingga fungsi T mengembalikan nilai *true*. Fungsi *P* adalah fungsi utama yang merepresentasikan pergerakan ngenat mengelilingi ruang pencarian. Dalam persamaan matematis posisi untuk setiap ngengat di *Update* dengan mengarah ke cahaya dengan menggunakan persamaan berikut:

$$M_i = S(M_i, F_j) \quad (2.9)$$

Dimana $M_i$ mengindikasi ngengat ke-*i*, $F_j$ mengindikasi cahaya ke-*j,* dan *S* adalah fungsi spiral. Logaritmatik spiral didefinisikan sebagai berikut:

$$S(M_i, F_j) = D_i . e^{bt} . \cos(2\pi t) + F_j \quad (2.10)$$



Dimana $D_i$ mengindikasikan jarak, *b* adalah konstanta, dan *t* adalah nilai random antara -1 hingga 1. Nilai *D* dapat dihitung menggunakan persamaan berikut:

$$D_i = \left|F_j - M_i\right| \tag{2.11}$$

Langkah-langkah fungsi $P$ dapat dilihat sebagai berikut:

```
Update flame no using Eq. (3.14)
OM = FitnessFunction(M);
if iteration == 1
    F = sort(M);
    OF = sort(OM);
else
    F = sort(M_{t-1}, M_t);
    OF = sort(M_{t-1}, M_t);
end
for i = 1: n
    for j = 1: d
        Update r and t
        Calculate D using Eq. (3.13) with respect to the
  corresponding moth
        Update M(i,j) using Eqs. (3.11) and (3.12) with respect to
  the corresponding moth
    end
end
```

Gambar 2. 5 *Pseudocode* Fungsi *P Moth-Flame Optimization* (Mirjalili, 2015)

## 2.6 *Multi-Verse Optimizer*

*Multi-Verse Optimizer* (MVO) merupakan algoritma optimisasi metaheuristik yang terinspirasi dari alam. Alam telah menjadi inspirasi utama bagi sebagian besar teknik optimasi stokastik berbasis populasi. Teori big bang membahas bahwa alam semesta yang kita tempati awalnya dimulai dengan ledakan besar. Menurut teori ini, big bang adalah asal mula segala sesuatu yang ada di dunia ini, dan tidak ada yang sebelumnya. Teori *multi-verse* adalah teori baru yang terkenal di antara fisikawan. Diyakini dalam teori ini ada lebih dari satu big bang dan setiap big bang menyebabkan lahirnya sebuah alam semesta. Istilah *multi-verse* berlawanan dengan sebuah alam semesta yang mengacu pada kebenaran alam semesta selain alam semesta yang kita semua tempati (Mirjalili, 2015).

*Multi-Verse Optimizer* mengambil tiga konsep dari teori *multi-verse* yang dijadikan sebagai inspirasi algoritma, yaitu lubang putih (*white holes*), lubang hitam (*black holes*), dan lubang cacing (*wormholes*). Algoritma yang berbasis populasi membagi proses pencarian menjadi dua fase yaitu eksplorasi dan ekspoitasi. Pada *Multi-Verse Optimizer* menggunakan konsep lubang putih dan lubang hitam untuk menjelajahi ruang pencarian dan sebaliknya konsep lubang cacing membantu

algoritma dalam mengeksploitasi ruang pencarian. Pada algoritma ini mengasumsikan bahwa setiap solusi analog dengan *universe* dan setiap variabel merupakan objek yang ada di dalam *universe* tersebut, dan ditetapkan untuk setiap solusi adalah tingkat inflasi yang sebanding dengan nilai solusi fungsi fitness serta menggunakan istilah waktu untuk setiap iterasinya (Mirjalili, 2015).

Permodelan matematis untuk lubang putih dan lubang hitam dan juga pertukaran objek yang ada di *universe* pada algoritma MVO ini menggunakan mekanisme *roulette wheel*. Pada setiap iterasinya MVO mengurutkan *universe* berdasarkan tingkat inflasi mereka dan memilihnya dengan roulette wheel untuk mendapatkan lubang putih. Dapat diasumsikan permodelan diatas sebagai matriks berikut:

$$U = \begin{bmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_n^1 & \cdots & x_n^d \end{bmatrix} \tag{2.12}$$

Dimana d adalah jumlah parameter atau variabel dan n adalah jumlah *universe* dan didapat persamaan sebagai berikut:

$$x_i^j = \begin{cases} x_k^j & r1 < NI(Ui) \\ x_i^j & r1 \geq NI(Ui) \end{cases} \tag{2.13}$$

Dimana $x_k^j$ mengindikasi parameter alam ke-*i* dan ke-*j* semesta, *Ui* menunjukan *universe* ke-*i*, *NI*(*Ui*) adalah tingkat inflasi normal dari *universe* ke-*i*, *r*1 adalah bilangan random dari 0 hingga 1, dan $x_k^j$ dipilih menggunakan mekanisme *roulette wheel* dengan pseudocode seperi berikut:

```
SU=Sorted universes
NI=Normalize inflation rate (fitness) of the universes
for each universe indexed by i
        Black_hole_index=i;
        for each objeect indexed by j
                r1=random([0,1]);
                if r1<NI(Ui)
                        White_hole_index= RouletteWheelSelection(-NI);
                        U(Black_hole_index,j)= SU(White_hole_index,j);
                end if
        end for
end for
```

Gambar 2. 6 *Pseudocode* Pemilihan *Universe* MVO (Mirjalili, 2015)

Untuk memberikan perubahan lokal untuk setiap *universe* dan memiliki probabilitas yang tinggi untuk meningkatkan tingkat inflasi menggunakan lubang cacing, pada algoritma MFO mengansumsikan bahwa lubang cacing selalu berada di antara universe dan universe terbaik yang terbentuk sejauh ini, yang dapat direpresentasikan sebagai persamaan berikut:

$$x_i^j = \begin{cases} \begin{cases} X_j + TDR \times \left( \left( ub_j - lb_j \right) \times r4 + lb_j \right) & r3 < 0.5 \\ X_j - TDR \times \left( \left( ub_j - lb_j \right) \times r4 + lb_j \right) & r3 \geq 0.5 \end{cases} & r2 < WEP \\ x_i^j & \geq WEP \end{cases} \quad (2.14)$$

Dimana $X_j$ mengindikasi parameter ke-*j* dari *universe* terbaik saat itu, TDR dan WEP adalah koefisien, $lb_j$ menunjukan batas bawah variabel ke-*j*, dan $ub_j$ adalah batas atas variabel ke-*j*, dan *r*2, *r*3, *r*4 adalah nilai *random* antara 0 hingga 1. Sehingga didapat *pseudocode Update* nilai setiap *universe* sebagai berikut:

```
for each universe indexed by i
  for each object indexed by j
        r2=random([0,1]);
        if r2<Wormhole_existance_probability
              r3=random([0,1]);
              r4=random([0,1]);
              if r3<0.5
                    U(i,j)=Best_universe(j) + Travelling_distance_rate * (( ub(j) - lb(j)) *
                    r4 + lb(j));
              else
                    U(i,j)=Best_universe(j) - Travelling_distance_rate * (( ub(j) - lb(j)) *
                    r4 + lb(j));
              end if
        end if
        end for
end for
```

Gambar 2. 7 *Pseudocode Update* Nilai Lokal MVO (Mirjalili, 2015)

Nilai koefisien wormhole existence probability (WEP) dan travelling distance rate (TDR) didapatkan menggunakan persamaan (2.15) dan (2.16) berikut:

$$WEP = min + l \times \left( \frac{max - min}{L} \right) \quad (2.15)$$

Dimana nilai *min* adalah nilai ketetapan minimum yang ditentukan, dan *max* adalah nilai ketetapan maksimum yang ditentukan, *l* mengindikasi iterasi pada saat itu dan *L* menunjukan nilai maksimum iterasi.

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \quad (2.16)$$

Dimana nilai p merupakan nilai akurasi dari eksploitasi atas iterasi yang ditentukan, Semakin tinggi nilai p maka eksploitasi akan lebih cepat dan akurat (Mirjalili, 2015).



## 2.7 *Artificial Bee Colony*

*Artificial Bee Colony* (ABC) *algorithm* merupakan algoritma optimisasi metaheuristik yang dikembangkan oleh D. Karaboga pada tahun 2005. Sejak saat itu, Karaboga dan Basturk telah secara sistematis mempelajari kinerja dari Algoritma ABC mengenai masalah optimisasi yang tidak terstruktur dan perluasannya ke optimisasi yang tidak dibatasi. Dalam algoritma *Artificial Bee Colony,* lebah diinisiasikan dalam sebuah koloni yang terbagi menjadi tiga kelompok lebah yaitu: lebah pekerja (lebah pengumpul), lebah pengamat dan lebah pengintai. Dengan kata lain jumlah lebah pekerja sama dengan jumlah sumber makanan. Lebah pekerja yang sudah tidak berada dilokasi makanan akan menjadi lebah pengintai yang bertujuan untuk mencari sumber makanan baru secara acak. Lebah pekerja akan memberitahukan informasi kepada lebah pengamat yang berada disarang sehingga lebah pengamat dapat memilih sumber makanan untuk mencari makan. Perilaku tersebut yang menjadi inspirasi algoritma metaheuristik *Artificial Bee Colony* (Yang, 2010).

Lebah pengamat akan memilih sumber makanan bergantung pada nilai probabilitas yang terkait dengan sumber makanan itu sendiri, yang di representasikan sebagai persamaan berikut:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{2.17}$$

Dimana nilai $p_i$ merepresentasikan sumber makanan, $fit_i$ adalah nilai *fitness* untuk solusi *i* yang sebanding dengan jumlah makanan yang ada di sumber makanan pada posisi *i* dan *SN* adalah jumlah sumber makanan yang sama dengan jumlah lebah pekerja (*BN*). Untuk menghasilkan posisi kandidat makan dari yang lama di memori menggunakan persamaan berikut:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2.18}$$

Dimana nilai k adalah anggota nilai populasi dan j adalah anggota nilai dimensi yang dipilih secara random dan tidak sama dengan nilai i, $\phi_{ij}$ adalah nilai random

di antara -1 hingga 1. Persamaan posisi sumber makanan yang baru direpresentasikan sebagai persamaan berikut:

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j) \quad (2.19)$$

Setelah mendapatkan kandidat posisi sumber makanan yang baru $v_{i,j}$ didapat dan kemudian dievaluasi dan dibandingkan dengan yang lama (Karaboga, 2007).

Pseudocode untuk algoritma *Artificial Bee Colony* dapat dilihat sebagai gambar 2.8 berikut:

1. Initialize the population of solutions $x_{i,j}$, $i = 1 \ldots SN, j = 1 \ldots D$
2. Evaluate the population
3. Cycle = 1
4. **Repeat**
5. Produce new solutions $v_{i,j}$ for the employed bees and evaluate
6. Apply the greedy selection process
7. Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$
8. Produce the new solutions $v_{i,j}$ for the onlookers from the solutions $x_{i,j}$ Selected depending on $P_{i,j}$ and evaluate them
9. Apply the greedy selection process
10. Determine the abandonend solution fot the scout, if exists, and replace it with a new randomly produced solution $x_{i,j}$
11. Memorize the best solution achieved so far
12. Cycle = Cycle + 1
13. **Until** cycle = Maximum cycle number

Gambar 2. 8 *Pseudocode Artificial Bee Colony* (Karaboga, 2007)

## 2.8 Metode Numerik *Newton Raphson*

Metode numerik *Newton Raphson* merupakan metode penentuan akar-akar persamaan, metode *Newton Raphson* juga merupakan metode yang paling sering digunakan pada bidang keilmuan karena nilai konvergensinya paling cepat diantara metode numerik lainnya. Pada metode *Newton Raphson* terdapat dua pendekatan yang digunakan dalam penurunan persamaan yaitu penurunan metode secara geometri dan penurunan metode dengan deret Taylor. Langkah penyelesaian penentuan akar dari suatu persamaan dengan metode *Newton Raphson* ini dapat dilakukan sebagai langkah berikut (Afrianita, 2015).

1. Inisiasi nilai awal akar untuk fungsi $f(x)$. Untuk nilai awal dari akar fungsi dinyatakan dengan $x_0$.

2. Mencari turunan pertama ($f'(x)$) dari fungsi $f(x)$.
3. Evaluasi $f(x)$ dan $f'(x)$ dengan nilai $x_t = x_0$.
4. Hitung pendekatan akar yang baru dari fungsi $f(x)$ dimana $f(x_r) \neq 0$,dengan persamaan (2.20) berikut:

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)} \tag{2.20}$$

5. Periksaan nilai *Error* ($E_r$) hasil perhitungan dimana.
   a. Jika *Error* lebih besar dari tolerasi yang ditetapkan maka perhitungan diulang lagi Langkah ke 2 sampai langkah ke 4.
   b. Jika *Error* lebih kecil dari toleransi yang ditetapkan maka perhitungan selesai. Untuk perhitungan nilai *Error* digunakan persamaan

$$E_r = \left|\frac{x_{t(i+1)} - x_{t(i)}}{x_{t(i+1)}}\right| . 100\% \tag{2.21}$$

## 2.9 *Levy Flight*

*Levy Flight* adalah *random walk* yang mengambil langkah dari distribusi *Levy*, yang direpresentasikan dengan persamaan sederhana $L(S) \sim |S|^{-1-\beta}$ dengan nilai $\beta$ diantara 0 hingga 2 yang merupakan indeks. Distribusi *Levy* merupakan distribusi yang non negatif dan berekor berat (*heavy tail*). Pergerakan *Levy Flight* lebih efisien daripada gerak brown dalam mengeksplorasi ruang pencarian berskala besar yang tidak diketahui dari hal tersebut diketahui bahwa variansi dari *Levy Flight* meningkat jauh lebih cepat daripada variasi gerak *Brown* (Yang, 2010).

Pada perhitungan *Levy flight* digunakan persamaan sebagai berikut :

$$Levy(x) = 0{,}01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \tag{2.22}$$

Dimana nilai $r_1$ dan $r_2$ merupakan bilangan random dari 0 hingga 1, $\beta$ bernilai diantara 0 hingga 2, dan $\sigma$ didapatkan melalui perhitungan sebagai berikut :

$$\sigma = \left( \frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{1+\beta}{2}\right)}} \right)^{1/\beta} \tag{2.23}$$

Dengan nilai $\Gamma(x) = (x-1)!$.

## 2.10 Fungsi Objektif



Fungsi Objektif atau fungsi tujuan merupakan fungsi linear yang digunakan untuk memenuhi nilai optimum suatu permasalahan *linear*. Pada studi algoritma optimisasi baru dibutuhkan tes fungsi uji objektif dalam pengujiannya. Fungsi uji adalah masalah buatan, dan bisa jadi digunakan untuk mengevaluasi perilaku suatu algoritma dalam keragaman situasi yang sulit. Permasalahan buatan dapat mencakup minimum global tunggal atau lebih global minimum. Permasalahan tersebut dimodifikasi dan dimanipulasi untuk diuji kepada algoritma dalam berbagai situasi. Di sisi lain, permasalahan yang ada berasal dari berbagai bidang seperti fisika, kimia, teknik, matematika, dan lain-lain permasalahan tersebut sulit untuk dimanipulasi dan mungkin mengandung aljabar atau diferensial yang rumit (Jamil, 2013).

### 2.9.1 Fungsi Uji *Unimodal*

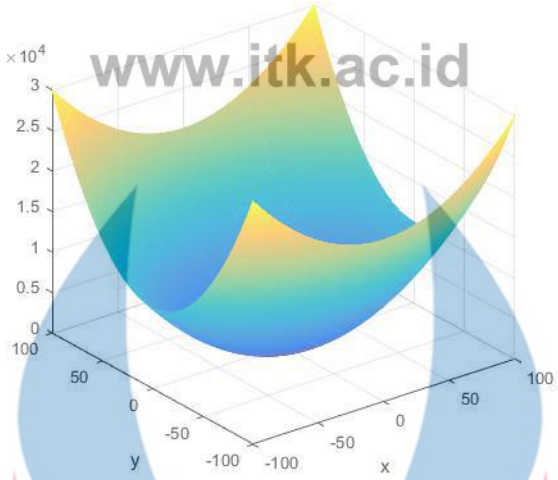
Fungsi uji *unimodal* merupakan fungsi uji yang memiliki satu titik nilai optimum dari segi minimum atau maksimum. Berikut contoh fungsi uji *unimodal* (Jamil, 2013).

a. Fungsi Uji *Bohachevsky N.1*

Fungsi uji *Bohachevsky N.1* merupakan fungsi uji unimodal hanya dapat di definisikan pada bidang 2 dimensi, yang memiliki persamaan sebagai berikut:

$$f(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7 \qquad (2.24)$$

Batasan atas dan bawah yang digunakan pada fungsi *Bohachevsky N.1* yaitu $-100 \leq x \leq 100$, dengan nilai Global minimum terdapat pada nilai $x = (0,0)$ dan nilai fungsi $f(x) = 0$. Berdasarkan persamaan fungsi, fungsi uji *Bohachevsky N.1* memiliki bentuk seperti berikut:



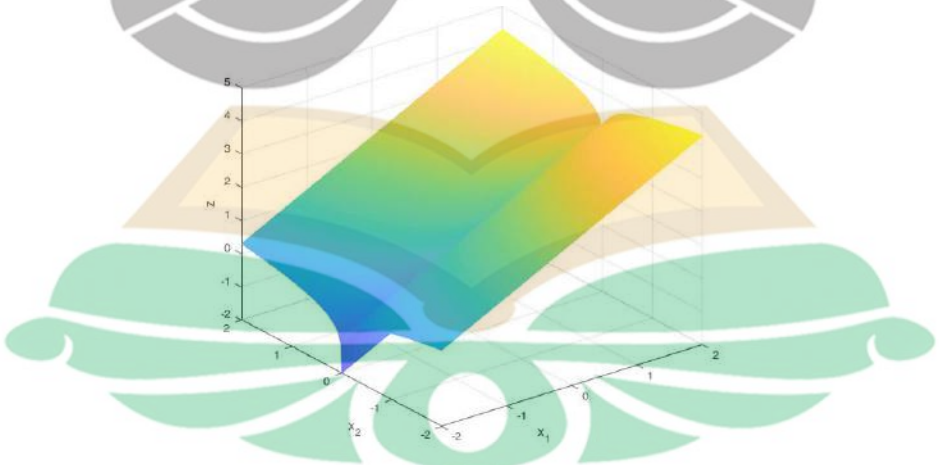

Gambar 2. 9 Fungsi Uji *Bohachevsky N.1*

b. Fungsi Uji *Ridge*

Fungsi uji *Ridge* merupakan fungsi uji *unimodal* yang memiliki n-dimensi. Fungsi uji *Ridge* memiliki persamaan sebagai berikut:

$$f(x) = x_1 + d\left(\sum_{i=2}^{n} x_i^2\right)^{\alpha} \quad (2.25)$$

Batasan atas dan bawah yang digunakan pada fungsi *Ridge* yaitu $-5 \leq x \leq 5$, nilai $d = 1$ dan $\alpha = 0.5$, dengan nilai Global minimum terdapat pada nilai $x$ = (-5,0,0…,0) dan nilai fungsi $f(x) = -5$. Berdasarkan persamaan fungsi, fungsi uji *Ridge* memiliki bentuk seperti berikut:




Gambar 2. 10 Fungsi Uji *Ridge*

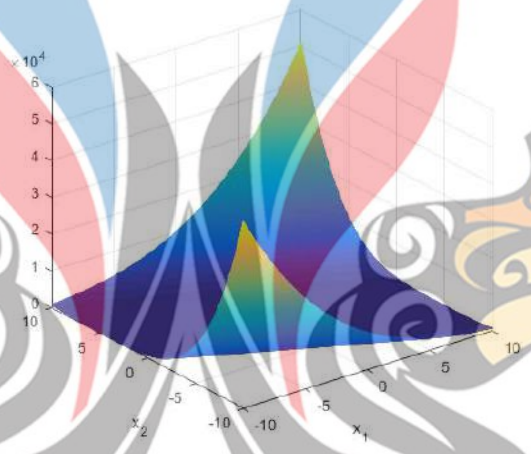
c. Fungsi Uji *Zakharov*

Fungsi uji *Zakharov* merupakan fungsi uji *unimodal* yang memiliki n-dimensi. Fungsi uji *Zakharov* memiliki persamaan sebagai berikut:

$$f(x) = f(x_1, \dots, x_n)$$

$$= \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4 \quad (2.26)$$

Batasan atas dan bawah yang digunakan pada fungsi *Zakharov* yaitu $-5 \leq x \leq 10$, dengan nilai Global minimum terdapat pada nilai $x = (0,\dots,0)$ dan nilai fungsi $f(x) = 0$. Berdasarkan persamaan fungsi, fungsi uji *Zakharov* memiliki bentuk seperti berikut:




Gambar 2. 11 Fungsi Uji *Zakharov*

## 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. Berikut contoh fungsi uji multimodal (Jamil, 2013).

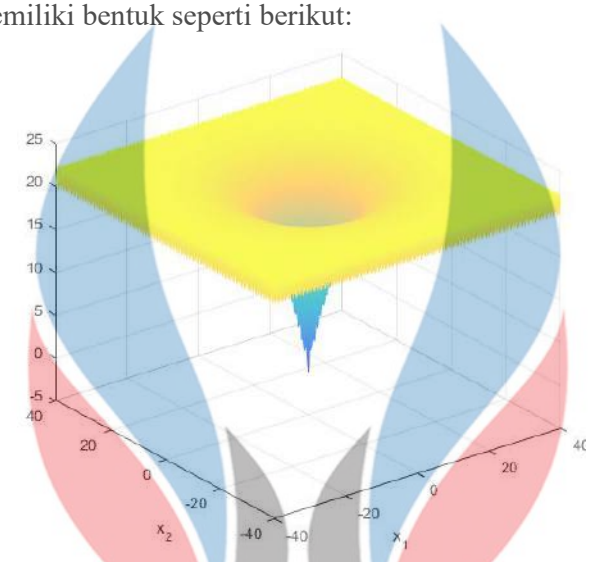
a. Fungsi Uji *Ackley*

Fungsi uji *Ackley* merupakan fungsi uji *multimodal* yang memiliki n dimensi. Fungsi uji *Ackley* memiliki persamaan sebagai berikut:

$$f(x) = f(x_1, \dots, x_n)$$

$$= -a.\, e^{-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)} + a + e \quad (2.27)$$

Batasan atas dan bawah yang digunakan pada fungsi *Ackley* yaitu -32 ≤ $x$ ≤ 32, nilai $a$ = 20, $b$ = 0.2 dan $c$ = 2π. Nilai Global minimum terdapat pada nilai $x$ = (0,…,0) dan nilai fungsi $f(x)$ = 0. Berdasarkan persamaan fungsi, fungsi uji *Ackley* memiliki bentuk seperti berikut:



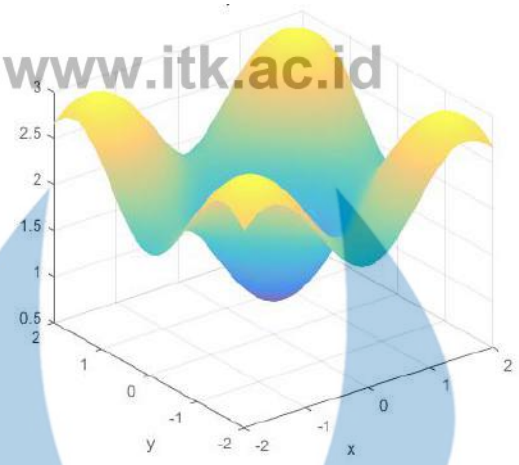

Gambar 2. 12 Fungsi Uji *Ackley*

b. Fungsi Uji *Periodic*

Fungsi uji *Periodic* merupakan fungsi uji *multimodal* yang memiliki n-dimensi. Fungsi uji *Periodic* memiliki persamaan sebagai berikut:

$$f(x) = f(x_1, \dots, x_n)$$
$$= 1 + \sum_{i=1}^{n} sin^2(x_i) - 0.1e^{\sum_{i=1}^{n} x_i^2} \quad (2.28)$$

Batasan atas dan bawah yang digunakan pada fungsi *Periodic* yaitu -10 ≤ $x$ ≤ 10, dengan nilai Global minimum terdapat pada nilai $x$ = (0,…,0) dan nilai fungsi $f(x)$ = 0.9. Berdasarkan persamaan fungsi, fungsi uji *Periodic* memiliki bentuk seperti berikut:



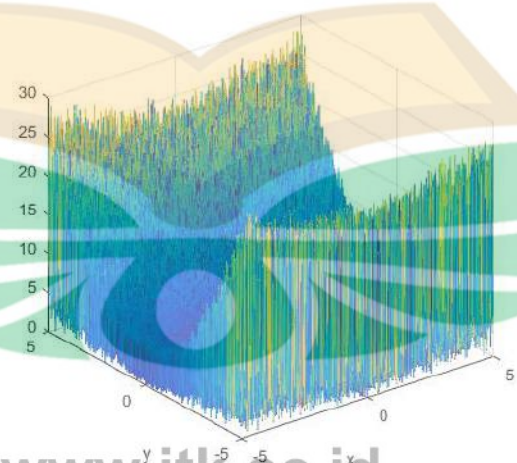

Gambar 2. 13 Fungsi Uji *Periodic*

c. Fungsi Uji *Xin-She Yang*

Fungsi uji *Xin-She Yang* merupakan fungsi uji *multimodal* yang memiliki n-dimensi. Fungsi uji *Xin-She Yang* memiliki persamaan sebagai berikut:

$$f(x) = f(x_1, \dots, x_n) = \sum_{i=1}^{n} \epsilon_i |x_i|^i \tag{2.29}$$

Dimana nilai ϵ adalah nilai random antara 0 sampai 1, batasan atas dan bawah yang digunakan pada fungsi *Xin-She Yang* yaitu $0 \leq x \leq 10$, dengan nilai Global minimum terdapat pada nilai $x = (0,\dots,0)$ dan nilai fungsi $f(x) = 0$. Berdasarkan persamaan fungsi, fungsi uji *Xin-She Yang* memiliki bentuk seperti berikut:






Gambar 2. 14 Fungsi Uji *Xin-She Yang*

## 2.11 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 | 1995 | *Particle Swarm Optimization* | Pemodelan algoritma didasarkan oleh perilaku sekawanan burung atau ikan. |
| 2 | Dervis Karaboga | 2005 | *Artificial Bee Colony* | Pemodelan algoritma didasarkan oleh perilaku kawanan lebah mencari madu |
| 3 | Seyedali Mirjalili | 2015 | *Multi-Verse Optimizer* | Pemodelan algoritma di inspirasi oleh konsep kosmik |
| 4 | Seyedali Mirjalili | 2015 | Moth-Flame Optimalization | Pemodelan algoritma yang terinspirasi dari perilaku pergerakan serangga ngengat |
| 5 | Tommy Artha | 2019 | Pemodelan Algoritma Optimisasi dari Perilaku Bekantan | Pemodelan perilaku bekantan sebagai pemodelan algoritma optimisasi |