BAB 2
TINJAUAN PUSTAKA

2.1 DOTA 2

Defend of the Ancients 2 atau yang lebih dikenal dengan DOTA 2 adalah sebuah permainan dengan genre MOBA (*Multiplayer Online Battle Arena*) yang dikembangkan oleh *Valve Corporation.* DOTA 2 merupakan sekuel dari *Defend of the Ancients* (DotA), yang merupakan *mod* (*modifying*) yang dikembangkan oleh sebuah komunitas pada permainan *Warcraft III: Reign of Chaos - The Frozen Throne.* Pengembangan dari DOTA 2 dimulai pada tahun 2009 ketika *IceFrog*, pengembang dari *mod Defense of the Ancients*, dipekerjakan oleh *Valve* sebagai desainer utama untuk membuat *remake* dari DotA yang lebih dimodernisasi dan secara resmi DOTA 2 telah dirilis pada Juli 2013 serta dapat dimainkan secara eksklusif melalui *Steam* yang merupakan distributor resmi *Valve* [1].

Dalam permainan DOTA 2, tim dibagi menjadi 2 bagian, yaitu DIRE dan RADIANT yang masing-masing beranggotakan 5 orang pemain dengan setiap pemainnya memainkan karakter (*hero*) yang berbeda-beda dengan memilih salahsatu dari 117 *hero* yang tersedia. Para pemain dapat memperkuat *hero* yang dimainkannya dengan membeli item yang juga dapat digunakan untuk mendukung tim selama permainan berlangsung. Terdapat 3 jalur (*lane*) utama dalam permainan ini, yaitu *toplane*, *midlane*, dan *bottomlane* yang menghubungkan markas (*base*) dari kedua tim dan akan memunculkan sekumpulan pasukan (*creeps*) setiap 60 detik sekali pada setiap *lane*. Disetiap *lane* tersebut, terdapat 3 buah menara (*tower*) yang akan menyerang setiap *hero* ataupun *creeps* yang berada didekatnya. Pada *tower* terakhir disetiap *lane*, terdapat 2 buah *barrack*, serta terdapat 1 buah *ancient* pada masing-masing *base* tim. *Barrack* merupakan sebuah bangunan dimana *creeps* nanti akan bermunculan, sedangkan *ancient* adalah bangunan utama yang harus dilindungi dan merupakan inti dalam permainan ini. Kedua tim akan saling menghancurkan *base* satu sama lain dan mempertahankan *base* nya masing-masing

agar tidak hancur. Kemenangan akan ditentukan oleh tim mana yang lebih dulu dapat menghancurkan *ancient* lawan.

Dengan kompleksitas permainan yang tidak terbatas baik dari segi *hero* ataupun item, DOTA 2 tidak memberikan batasan cara bermain, pemain bebas menentukan dan mengekspresikan gaya bermainnya sendiri [1].

2.2 *The International*

The International adalah salah satu turnamen terbesar dari sekian banyak turnamen yang diadakan dalam permainan DOTA 2. Turnamen ini adalah kompetisi yang diselenggarakan oleh *Valve Corporation* setiap satu tahun sekali. *The International* pertama kali diselenggarakan pada bulan Agustus tahun 2011 di Cologne, Jerman. Tim-tim terbaik dari berbagai macam negara akan berkumpul disatu tempat dan bersaing untuk mendapatkan gelar *The International*, uang tunai, serta piala *Aegis of the Champion.* Untuk menarik para pemain dari seluruh dunia, *Valve* menyiapkan total hadiah yang tidak sedikit. Secara bertahap total hadiah yang ditawarkan terus bertambah setiap tahunnya, dimulai dari $1,6 juta (*The International* 2011 dan 2012), $2,8 juta (*The Interational* 2013), $10,9 juta (*The International* 2014), $18,4 juta (*The International* 2015), $20,7 juta (*The International* 2016), $24,7 juta (*The International* 2017), dan $25,5 juta (*The International* 2018) atau setara dengan Rp361 miliar dan merupakan total hadiah terbesar dalam turnamen *The International* saat ini [2]. Total hadiah dari tahun ke tahun semakin bertambah, hal ini dikarenakan adanya produk yang dikeluarkan oleh *Valve* yang bernama *Compendium* saat *The International* berlangsung, dimana 0,25 sen dari hasil penjualan produk tersebut dimasukkan ke total hadiah *The International*.

2.3 Prediksi

Prediksi atau peramalan merupakan suatu proses yang dilakukan untuk memperkirakan suatu kejadian pada masa yang akan datang berdasarkan pada kejadian ataupun data yang telah terjadi pada masa lalu [12]. Data yang telah terjadi pada masa lalu tersebut kemudian diolah menggunakan suatu metode tertentu untuk

dapat memperkirakan kejadian pada masa yang akan datang. Terdapat 2 pendekatan yang dapat dilakukan dalam peramalan, yaitu sebagai berikut [13]:

1. Peramalan Kuantitatif

   Peramalan kuantitatif merupakan metode yang didasarkan pada data/informasi kuantitatif pada masa lalu. Metode ini dapat diterapkan jika terdapat 3 kondisi kuantitatif berikut:

   a. Terdapat data/informasi yang telah terjadi pada masa lalu (data historis).
   b. Data/informasi dapat dikuantifikasi dalam bentuk data berupa angka.
   c. Dapat diasumsikan bahwa beberapa aspek dari pola dimasa lalu akan berlanjut dimasa yang akan datang.

2. Peramalan Kualitatif

   Peramalan kualitatif merupakan metode yang bersifat intuitif berdasarkan pengetahuan serta pengalaman. Metode ini dapat dilakukan jika data/informasi kuantitatif yang dimiliki hanya sedikit atau bahkan tidak ada sama sekali, dan memiliki pengetahuan kualitatif yang cukup.

2.4 Normalisasi Data

Normalisasi data adalah sebuah proses yang dilakukan untuk mengubah data awal (data mentah) yang didapatkan ke dalam bentuk lain dengan tujuan untuk mendapatkan data yang lebih tepat untuk dilakukan pemodelan dan analisis [14]. Proses normalisasi berfokus pada penskalaan data, baik dalam hal skala/rentang ataupun distribusi. Normalisasi data dapat membantu untuk mencegah atribut dengan rentang awalan yang besar agar tidak melebihi atribut dengan rentang awalan yang lebih kecil dengan memberikan semua atribut bobot yang sama. Terdapat beberapa cara yang dapat dilakukan untuk melakukan normalisasi data, diantaranya adalah sebagai berikut [15]:

1. *Min-max Normalization*

   Min-max normalization merupakan metode normalisasi yang dilakukan dengan cara memproyeksikan rentang data asli ke rentang data yang baru.

Rentang data yang baru setelah dilakukan normalisasi umumnya adalah [0, 1] atau [-1, 1]. Metode normalisasi ini dapat dilakukan dengan melakukan perhitungan dengan persamaan matematis berikut [15]:

$$v' = \frac{v - min_x}{max_x - min_x} (new\_max_x - new\_min_x) + new\_min_x \quad (2.1)$$

Keterangan :

v' = Nilai baru hasil normalisasi.
v = Nilai sebelum normalisasi.
min_x = Nilai terkecil pada atribut x.
max_x = Nilai terbesar pada atribut x.
new_min_x = Nilai terkecil baru untuk atribut mix_x.
new_max_x = Nilai terbesar baru untuk atribut max_x.

Metode normalisasi ini memiliki keuntungan dalam menjaga hubungan antara nilai-nilai data asli, akan tetapi jika inputan untuk normalisasi turun diluar rentang data asli dari *x*, metode ini dapat menyebabkan kesalahan *out of bound*.

2. *Standardization*

Z-score normalization atau yang dapat juga disebut dengan *Standardization* adalah metode normalisasi yang mengubah tidak hanya besarnya data, tetapi juga mengubah penyebaran data (dispersi). Dalam metode ini, nilai untuk atribut yang akan diubah, dinormalisasi berdasarkan mean dan standar deviasi. Berikut persamaan matematis yang dapat dilakukan untuk menggunakan metode ini [15]:

$$v' = \frac{v - \bar{x}}{\sigma_x} \quad (2.2)$$

Keterangan :

v' = Nilai baru hasil normalisasi.
v = Nilai sebelum normalisasi.
$\bar{x}$ = Mean dari atribut x.
σ_x = Standar deviasi dari atribut x.

Untuk mendapatkan nilai mean dapat dilakukan dengan menggunakan persamaan berikut [24]:

$$\bar{x} = \frac{\sum_{i=0}^{n} x_i}{n} \quad (2.12)$$

Keterangan :

$\bar{x}$ = Nilai mean.

x_i = Nilai x ke-i.

n = Jumlah data.

Untuk mendapatkan nilai dari standar deviasi dapat dilakukan dengan menggunakan persamaan berikut [24]:

$$\sigma_x = \sqrt{\frac{\sum(x_i-\mu)^2}{N}} \quad (2.13)$$

Keterangan :

σ_x = Nilai standar deviasi.

x_i = Nilai x ke-i.

μ = Mean dari data keseluruhan.

N = Jumlah data.

Sedangkan jika perhitungan standar deviasi tidak dilakukan pada keseluruhan data atau berupa data sampel, perthitungan standar deviasi dapat dilakukan dengan persamaan berikut [24]:

$$s_x = \sqrt{\frac{\sum(x_i-\bar{x})^2}{n-1}} \quad (2.14)$$

Keterangan :

s_x = Nilai standar deviasi.

x_i = Nilai x ke-i.

$\bar{x}$ = Mean dari data sampel.

n = Jumlah data.

Metode *standardization* sangat berguna ketika nilai minimum dan maksimum dari atribut *x* tidak diketahui, atau ketika terdapat *outlier* dalam metode *min-max normalization*.

3. *Decimal Scaling*

Decimal Scaling adalah metode normalisasi yang dilakukan dengan cara memindahkan titik desimal dari suatu nilai atribut. Jumlah titik desimal yang dipindahkan bergantung pada nilai absolut maksimum atribut tersebut.

Metode ini dapat dilakukan dengan menggunakan persamaan matematis berikut [15]:

$$v' = \frac{v}{10^j} \tag{2.3}$$

Keterangan :

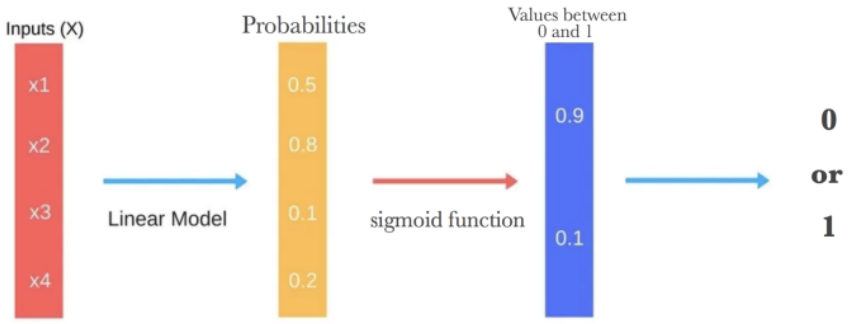
v' = Nilai baru hasil normalisasi.

v = Nilai sebelum normalisasi.

j = Bilangan bulat terkecil, $max(|v'|) < 1$

2.5 *Logistic Regression*

Logistic Regression merupakan salahsatu algoritma dalam *machine learning* yang digunakan dalam melakukan klasifikasi. Metode ini dikembangkan oleh seorang ahli statistik bernama David Cox pada tahun 1958. Tujuan dari klasifikasi *Logistic Regression* adalah untuk membuat sebuah model, dimana model tersebut dapat menganalisis suatu dataset yang memiliki satu atau lebih variabel *independent* yang akan mempengaruhi hasil dari variabel *dependent*, hasil dari variabel *dependent* tersebut dapat berupa 1/0, ya/tidak, benar/salah [5]. Seperti kebanyakan metode pada *machine learning* lainnya, metode ini dipinjam dari bidang statistik dan terlepas dari namanya, metode ini bukanlah algoritma yang digunakan untuk masalah regresi yang digunakan untuk memprediksi hasil yang berkelanjutan. Sebaliknya, *Logistic Regression* adalah sebuah metode yang digunakan untuk melakukan klasifikasi biner yang memberikan hasil biner diskrit antara 0 dan 1. Berikut merupakan gambaran dari tahapan-tahapan dalam metode *Logistic Regression*:



Gambar 2.1 Tahapan *Logistic Regression*

Logistic Regression mengukur hubungan antara variable *dependent* (hasil prediksi) dengan satu atau lebih variabel *independent*, dengan memperkirakan probabilitas menggunakan fungsi logistik yang mendasarinya. Probabilitas ini kemudian diubah menjadi nilai-nilai biner (0 dan 1) untuk melakukan prediksi. Hal tersebut merupakan tugas dari fungsi logistik yang dapat disebut dengan fungsi sigmoid. Dengan kata lain, fungsi sigmoid digunakan untuk menjaga agar nilai prediksi selalu berada diantara 0 dan 1. Berikut merupakan persamaan linear dan fungsi sigmoid:

$$z = \alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k \quad (2.4)$$

Keterangan :

z = Persamaan linear.

β_0 = Bobot ke-0 terbaik sementara pada inisialisasi partikel.

β_1 = Bobot ke-1 terbaik sementara pada inisialisasi partikel.

β_k = Bobot ke-k terbaik sementara pada inisialisasi partikel.

X_1 = Data hasil normalisasi pada baris ke-1.

$$f(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\alpha + \sum \beta_i X_i)}} \quad (2.5)$$

Keterangan :

$f(z)$ = Fungsi Sigmoid.

e^{-z} = Nilai eksponen dari hasil persamaan nilai z.

Pada persamaan (2.4), α, β_1, β_2, dan β_k adalah *best weights* yang merupakan nilai numerik yang akan didapatkan melalui tahap pelatihan pada dataset. Sedangkan X_1, X_2, dan X_k adalah variabel independen yang didapatkan dari dataset. Setelah mendapatkan nilai dari *z*, hasil tersebut kemudian dimasukkan kedalam persamaan (2.5). Hasil dari *f(z)* akan selalu menghasilkan nilai antara 0 atau 1. Jika hasil dari *f(z)* kurang dari 0,5, maka dapat disimpulkan nilai yang dihasilkan adalah 0, dan jika hasil dari *f(z)* lebih besar dari 0,5, maka dapat disimpulkan nilai yang dihasilkan adalah 1 seperti berikut:

$p >= 0{,}5$, *class = 1*,

$p < 0{,}5$, *class = 0* (2.6)

2.6 Multi-Swarm Optimization

Multi-Swarm Optimization (*MSO*) adalah salah satu metode optimasi dalam *machine learning* yang digunakan untuk memperkirakan suatu solusi untuk masalah numerik yang rumit [16]. MSO merupakan bagian dari *Swarm Algorithm*, dimana *swarm algorithm* berfungsi untuk memecahkan suatu masalah dengan mensimulasikan pergerakan sekelompok objek dalam ruang kemungkinan solusi didasarkan pada perilaku kolektif dan dapat mengatur dirinya sendiri [25]. Metode ini merupakan variasi dari metode *Particle Swarm Optimization* (*PSO*). Metode *MSO* memperluas *particle swarm optimization* dengan menggunakan beberapa *swarm* dalam partikel simulasi, berbeda dengan *PSO* yang hanya menggunakan 1 *swarm. PSO* pertama kali di perkenalkan pada tahun 1995 oleh R. Eberhart dan J. Kennedy. Metode ini dikembangkan dengan mengidentifikasi perilaku dari kawanan hewan yang bekerja secara berkelompok, seperti burung dan ikan. Umumnnya sekawanan burung tidak mempunyai pemimpin. Mereka menemukan makanan secara acak mengikuti salah satu anggota dalam kawanan. Anggota-anggota tersebut harus berada paling dekat dengan posisi dari sumber makanan. Anggota yang memiliki posisi terdekat ini disebut sebagai solusi potensial untuk kawanan tersebut. Para anggota kawanan akan saling berkomunikasi satu sama lain dengan anggota yang sudah memiliki situasi yang terbaik, yang dalam hal ini adalah posisi. Sehingga anggota lainnya akan segera bergerak ke posisi tersebut. Hal ini terjadi secara berulang kali hingga kondisi terbaik atau sumber makanan ditemukan.

MSO dapat diterapkan pada beberapa permasalahan dalam *machine learning*, seperti memperkirakan bobot dan nilai bias pada jaringan saraf tiruan, ataupun dalam memperkiran nilai bobot dalam permasalahan prediksi dan klasifikasi [16]. Terdapat beberapa komponen dalam algoritma *MSO*, diantaranya adalah *swarm,* setiap *swarm* terdiri dari beberapa partikel, setiap partikel dari *swarm* akan mencari posisi terbaik pada ruang pencariannya masing-masing yang posisi awalnya ditentukan secara acak, pencarian tersebut akan terus dilakukan hingga iterasi maksimum terpenuhi. Selain iterasi maksimum, terdapat kondisi-kondisi tertentu yang dapat digunakan untuk mengakhiri proses iterasi pencarian

dalam *PSO*. Berikut adalah kondisi berhenti yang dapat digunakan untuk mengakhiri proses iterasi pencarian [23]:

1. Ketika jumlah iterasi maksimum telah terpenuhi.
2. Ketika solusi yang memenuhi kriteria telah ditemukan.
3. Ketika tidak ada peningkatan yang dipantau dalam jumlah iterasi.
4. Ketika nilai radius *swarm* yang dinormalisasi, yang didapatkan dari nilai maksimum radius *swarm* dibagi dengan diameter *swarm* awal, mendekati nol.
5. Ketika grafik fungsi objektif mendekati nol.

Pergerakan dari setiap partikel akan ditentukan oleh perhitungan *velocity* pada setiap pencarian posisi. Dalam proses pencarian, disetiap iterasi akan dilakukan pengacakan partikel disetiap swarm sehingga perhitungan tidak selalu dimulai dari indeks terkecil. Hal tersebut dilakukan untuk mencegah algoritma agar tidak terjebak dalam solusi yang tidak optimal.

Berikut merupakan penjelasan dari setiap langkah-langkah yang dilakukan dalam algoritma *MSO* :

1. Inisialisasi posisi partikel.

   Ini merupakan tahapan awal dalam *MSO*, pada tahap ini, penggunaan jumlah swarm dan partikel akan ditentukan, serta penempatan setiap partikelnya pada posisi acak. Inisialisasi posisi partikel dapat dilakukan dengan menggunakan persamaan berikut:

   $$x_i = minX + r_i(maxX\text{-}minX) \quad (2.7)$$

   Keterangan :

   x_i = Posisi partikel ke-i.

   r_1 = Nilai acak dengan rentang (0, 1).

   $minX$ = Batas minimal posisi partikel.

   $maxX$ = Batas maksimal posisi partikel.

   Setelah melakukan inisialisasi untuk setiap partikel, selanjutnya adalah memberikan kecepatan pada setiap partikel tersebut secara acak. Inisialisasi kecepatan pada setiap partikel dapat dilakukan dengan menggunakan persamaan berikut:

$$v_j = minV + r_j(maxV\text{-}minV) \quad (2.8)$$

Keterangan :

v_j = Kecepatan partikel ke-j.

r_j = Nilai acak dengan rentang (0, 1).

minx = Batas minimal kecepatan partikel.

maxX = Batas maksimal kecepatan partikel.

2. Mengevaluasi optimisasi nilai fungsi pada setiap partikel.

   Melakukan evaluasi nilai fungsi pada setiap partikel dilakukan untuk mendapatkan partikel dengan nilai fungsi terbaik. Salah satu metode yang dapat digunakan untuk mendapatkan nilai fungsi adalah dengan menggunakan *MSE* (*Mean Squared Error*). Berikut adalah persamaan dari *MSE* yang digunakan [15]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \check{Y}_i\right)^2 \quad (2.9)$$

   Keterangan :

   n = Jumlah data latih.

   Y_i = Hasil yang diamati dari variabel yang diprediksi ke-i.

   $\hat{Y}_i$ = Nilai hasil prediksi pada data ke-i.

3. Menentukan *bestPartikel.*

   bestPartikel adalah posisi terbaik dari setiap partikel. Posisi ini merupakan posisi yang ditempati oleh partikel saat ini.

4. Menentukan *bestSwarm*.

   bestSwarm adalah posisi terbaik dari setiap *bestSwarm*. Posisi ini merupakan posisi yang ditempati oleh *bestSwarm* saat ini.

5. Menentukan *bestMulti-Swarm.*

   *bestMulti-Swarm.*adalah posisi terbaik dari setiap *Multi-Swarm.* Posisi ini merupakan posisi yang ditempati oleh *Multi-Swarm* saat ini.

6. Memperbarui kecepatan partikel (*velocity*).

   Perhitungan nilai *velocity* dilakukan untuk menentukan posisi baru untuk setiap partikel. Berikut merupakan persamaan yang digunakan untuk menentukan nilai *velocity* [8]:

   $v\ (t+1) = w.v(t) + (c_1.r_1)(p(t)\text{-}x(t)) +$

$$(c_2.r_2)(s(t)-x(t)) + (c_3.r_3)(g(t)-x(t)) \quad (2.10)$$

Keterangan :

$v\ (t+1)$ = *Velocity* pada saat *(t+1)*, *velocity* yang baru.

w = Faktor inertia (konstanta).

$v(t)$ = Posisi *velocity* saat ini.

$x(t)$ = Posisi partikel saat ini.

$p(t)$ = Posisi terbaik dari partikel saat ini.

$s(t)$ = Posisi terbaik dari setiap partikel pada *swarm* saat ini.

$g(t)$ = Posisi *gbest* terbaik dari setiap partikel disemua *swarm.*

c_1, c_2, c_3 = Bobot kognitif, sosial, atau global *weights* (konstanta).

r_1, r_2, r_3 = Nilai acak antara 0 dan 1 yang memberikan efek pengacakan untuk setiap pembaruan kecepatan.

7. Memperbarui posisi partikel.

   Setelah melakukan perhitungan *velocity* yang baru pada persamaan (2.10), selanjutnya adalah melakukan perhitungan untuk menentukan posisi partikel yang baru dengan persamaan berikut [16]:

   $$x(t+1) = x(t) + v(t+1) \quad (2.11)$$

   Keterangan :

   $x(t+1)$ = Posisi partikel pada saat *(t+1)*, posisi partikel yang baru.

   $x(t)$ = Posisi partikel saat *t*.

   $v(t+1)$ = Posisi *velocity* yang baru (persamaan 2.10).

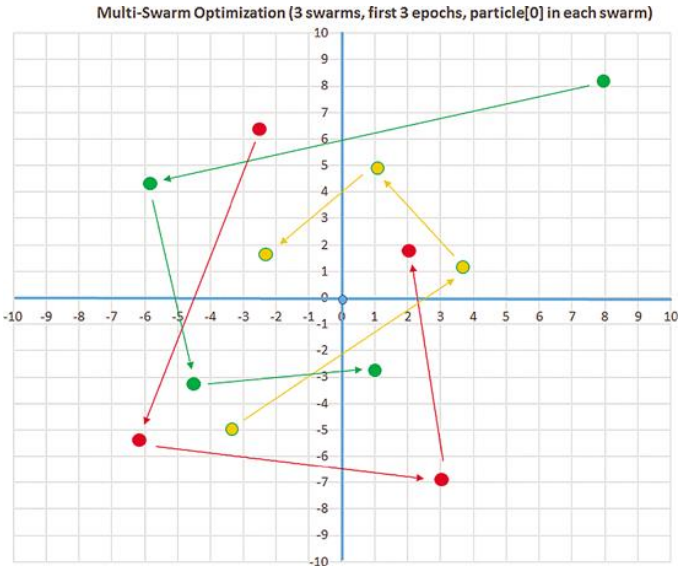
8. Melakukan perulangan dari langkah ke-2 hingga kriteria terpenuhi, kriteria dikatakan terpenuhi jika telah mendapatkan nilai fungsi yang cukup baik atau apabila telah mencapai batas iterasi maksimum.

Bagian terpenting dalam algorima *MSO* adalah menghitung kecepatan partikel, dimana kecepatan partikel tersebut yang nantinya akan mengatur kemana selanjutnya partikel akan bergerak. Berikut merupakan *pseudo-code* dari algoritma *MSO* yang dapat dilihat pada gambar 2.2 [16]:

```
loop maxLoop times
  for each swarm
    for each particle
      compute new velocity
      use velocity to update position
      check if a new best cost has been found
    end for
  end for
end loop
```

Gambar 2.2 *Multi-Swarm Optimization Pseudo-code*

Gerakan spiral adalah karakteristik dari algoritma *MSO*. Gambaran grafik dari gerakan spiral dalam algoritma *MSO* dapat dilihat pada gambar 2.3 [8]:



Gambar 2.3 Gerakan Spiral Algoritma *MSO*

Grafik pada gambar 2.3 menggambarkan proses dari *MSO* untuk masalah dengan dua nilai *x*, tiga *swarm* dengan masing-masing empat partikel, dan dimana posisi optimal berada pada (0, 0). Grafik menunjukkan bagaimana partikel pertama disetiap *swarm* menutup pada posisi optimal. Jika dibandingkan dengan algoritma *PSO*, optimasi *MSO* sedikit lebih kompleks, dan cenderung menghasilkan hasil yang lebih berkualitas, meskipun lebih lambat [16]. *MSO* adalah optimasi *meta-heuristic* numerik yang biasanya digunakan dalam *machine learning* untuk menemukan serangkaian *weights* yang meminimalkan beberapa jenis fungsi *error*.

2.7 Diagram Konteks (*Context Diagram*)

Diagram Konteks atau *Context Diagram* merupakan sebuah diagram yang menggambarkan suatu sistem yang mencakup *input* dan *output* dari sistem tersebut secara umum. Diagram ini merupakan tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses yang menunjukan sistem secara keseluruhan. Semua entitas eksternal yang ditunjukkan oleh diagram konteks beserta aliran-aliran data utama menuju ke sistem dan berasal dari sistem. Diagram konteks dimulai dengan penggambaran entitas, aliran data, dan proses tunggal yang menunjukkan keseluruhan sistem. Aliran dalam diagram konteks menggambarkan masukan ke sistem dan keluaran dari sistem. Aliran data digambarkan hanya jika diperlukan untuk tujuan mendeteksi suatu aktivitas yang terjadi dalam sistem, dimana sistem harus memberikan respon atau membutuhkan data untuk menghasilkan respon. Selain itu, aliran data juga dibutuhkan untuk menggambarkan transportasi antara sistem dan entitas [17].

2.8 Diagram Alir Data (*Data Flow Diagram*)

Diagram Alir Data atau *Data Flow Diagram* (DFD) merupakan suatu model proses yang digunakan untuk menggambarkan aliran data yang melalui suatu sistem serta menggambarkan pekerjaan atau pemrosesan yang dilakukan oleh sistem [18]. Aliran data pada suatu sistem digambarkan menggunakan notasi-notasi tertentu sehingga membentuk suatu diagram. Notasi-notasi yang digunakan dalam DFD menggambarkan komponen-komponen yang terdapat dalam sebuah sistem, serta aliran-aliran data dari komponen yang ada, dimulai dari aliran data asal, aliran data tujuan, dan penyimpanan data dari sistem tersebut.

DFD digunakan untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem yang ada atau untuk menyusun dokumentasi untuk sistem yang baru. DFD merupakan alat bantu dari pengembangan sebuah sistem yang dibangun secara terstruktur atau *procedural*. Terdapat beberapa keunggulan yang dimiliki oleh DFD, diantaranya adalah DFD mudah dipahami baik oleh orang teknik maupun non-teknik, memberikan gambaran sistem secara menyeluruh lengkap

dengan lingkup sistem dan hubungan ke sistem lainnya, serta memberikan tampilan komponen-komponen sistem secara detail [17].

2.9 *Flowchart*

Flowchart atau diagram alir merupakan sebuah diagram dalam bentuk grafis atau simbolik yang merepresentasikan langkah-langkah dari suatu algoritma, alir kerja atau suatu proses [19]. Deskripsi tekstual dari suatu algoritma mungkin tidak dapat dipahami dengan mudah dan cepat oleh sebagian orang, karena itulah representasi bergambar dalam bentuk flowchart dapat digunakan sebagai pengganti dari suatu algoritma agar dapat dipahami dengan mudah. *Flowchart* digunakan untuk mendesain dan mendokumentasi proses atau program sederhana, seperti diagram lainnya, diagram ini membantu menggambarkan apa yang sedang terjadi dan membantu untuk memahami sebuah proses. Dalam *flowchart,* notasi-notasi dalam berbagai bentuk digunakan untuk menunjukkan berbagai operasi. Notasi-notasi tersebut kemudian akan dihubungkan oleh garis-garis dengan panah yang menentukan aliran atau arah yang harus dilanjutkan untuk mengetahui langkah selanjutnya. Garis-garis yang menghubungkan antara notasi yang satu dengan notasi lainnya disebut garis aliran. Simbol yang berbeda digunakan dalam *flowchart* untuk menunjukan operasi yang berbeda yang terjadi dalam sebuah program.

Flowchart dapat diklasifikasikan ke dalam dua kategori, yaitu *flowchart* program dan *flowchart* sistem [19]. *Flowchart* program bertindak sperti cermin dari suatu program komputer dalam hal simbol *flowcharting*, *flowchart* ini berisi langkah-langkah penyelesaian suatu unit masalah untuk hasil tertentu. Sedangkan *flowchart* sistem adalah kebalikan dari *flowchart* program, *flowchart* ini mengandung solusi dari banyak unit masalah secara bersama yang terkait erat dan berinteraksi satu sama lain untuk mencapai suatu tujuan. *Flowchart* dapat digunakan untuk menggambarkan/menunjukkan suatu urutan langkah-langkah untuk melakukan pekerjaan apapun, serangkaian operasi sederhana yang melibatkan penerimaan input, melakukan operasi aritmatika dalam input dan menunjukkannya kepada pengguna untuk menunjukkan struktur urutan logika dari suatu program. Tujuan dasar dari *flowchart* adalah untuk menentukan urutan

langkah-langkah yang digunakan untuk menunjukkan solusi masalah melalui aritmatik dan/atau manipulasi logis yang dapat diinstuksikan pada komputer.

2.10 XML

Extensible Markup Language atau yang biasa disingkat dengan XML adalah sebuah bahasa *markup* yang digunakan untuk keperluan umum dan didesain untuk menyimpan serta mengantarkan data [20]. XML mulai dikembangkan pada tahun 1996 dan menjadi standar resmi *World Wide Web Consortium* (W3C) pada tahun 1998. XML didesain untuk mampu menyimpan data secara ringkas dan mudah diatur. Secara umum, sistem komputer dan basis data berisi data dalam format yang tidak kompatibel, mengubah data ke dalam bentuk XML dapat sangat mengurangi kompleksitas dan membuat data dapat dibaca oleh berbagai jenis aplikasi. Penggunaan XML lebih berfokus pada bagaimana informasi harus disusun ataupun digunakan diantara banyaknya aplikasi (termasuk *web browser*) yang berjalan di internet.

2.11 Visual Basic

Microsoft Visual Basic atau yang biasa disebut VB merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman. Visual Basic adalah pengembangan dari bahasa komputer BASIC (*Beginner's All-purpose Symbolic Instruction Code*). Bahasa BASIC diciptakan oleh Professor John Kemeny dan Thomas Eugene Kurtz dari Perguruan Tinggi Dartmouth pada pertengahan tahun 1960-an [21]. Bahasa program tersebut tersusun mirip dengan bahasa Inggris yang biasa digunakan oleh para programmer untuk menulis program-program komputer sederhana yang berfungsi sebagai pembelajaran bagi konsep dasar pemrograman komputer.

Visual Basic 1.0 dikenalkan pada tahun 1991. Konsep pemrograman dengan metode *drag-and-drop* untuk membuat tampilan aplikasi. Metode pada Visual Basic ini diadaptasi dari *prototype generator form* yang dikembangkan oleh Alan

Cooper dan perusahaannya, dengan nama Tripod. Microsoft kemudian mengontrak Cooper dan perusahaannya untuk mengembangkan Tripod menjadi sistem *form* yang dapat diprogram untuk Windows 3.0, di bawah kode nama Ruby. Visual Basic telah berkembang menjadi beberapa versi, sampai yang terbaru, yaitu Visual Basic 2010. Visual Basic 6.0 merupakan versi yang paling populer karena kemudahan yang ditawarkannya dalam membuat program dan tidak menghabiskan banyak memori. Pemrograman dengan Visual Basic sangat mudah dipahami bagi pemula karena dapat menghemat waktu pemrograman dengan tersedianya komponen-komponen yang siap pakai.