BAB 2
LANDASAN TEORI

2.1. Swarm Intelligence

Swarm Intelligence adalah salah satu teknik kecerdasan buatan yang berlandaskan kepada perilaku kolektif (*collective behaviour*) pada sistem yang terdesentralisasi dan dapat mengatur dirinya sendiri (*self-organizing*). Sistem yang memanfaatkan *Swarm Intelligence* biasanya merupakan sebuah populasi yang terdiri atas anggota berupa agen-agen yang sederhana, yang berinteraksi secara lokal dengan sesama anggota, dan juga berinteraksi dengan lingkungan. Walaupun pada umumnnya tidak ada struktur kendali secara terpusat (*centralized*) yang mendikte bagaimana masing-masing individu bertindak, namun interaksi secara lokal (di antara anggota) seringkali menuju pada pembentukan (*emergence*) perilaku global [**3**].

2.1.1. Sifat Dasar Swarm Intelligence

Sifat sifat dari *Swarm Intelligence* diantaranya [**3**]:

1. Terdiri dari banyak individu.
2. Individu yang relatif bersifat homogen (Baik yeng memiliki sifat yang sama maupun yang memiliki sifat atau tipe yang berbeda).
3. Interaksi antar individu didasari dari aturan perilaku sederhana yang memanfaatkan informasi lokal baik secara langsung atau melalui lingkungannya.

Kemampuan setiap individu melakukan interaksi dengan sesama individu maupun dengan lingkungannya dalam *swarm intelligence* disebut dengan *self-organizes*. Ini menunjukkan bahwa setiap individu dapat mengatur dirinya sendiri.

2.1.2. Prinsip Dasar Swarm Intelligence

Prinsip dasar *Swarm Intelligence* :

1. Prinsip kedekatan : setiap individu harus dapat menyesuaikan ruang dan waktu dengan perhitungan sederhana.
2. Prinsip kualitas : setiap individu harus dapat menanggapi faktor lingkungan.

3. Prinsip respon beragam : setiap individu tidak seharusnya melakukan kegiatannya pada jalur yang terlalu sempit.
4. Prinsip stabilitas : populasi tidak harus mengubah modus perilakunya setiap kali perubahan lingkungan.
5. Prinsip adaptasi : populasi harus mampu mengubah modus perilaku ketika komputasi bernilai penting atau berharga [**3**].

2.1.3. Algoritma Swarm Intelligence

Adapun beberapa contoh algoritma atau metode yang populer dari *swarm intelligence* diantaranya [**3**]:

a. Particle Swarm Optimization (PSO)

   PSO adalah suatu metode pencarian acak berbasis populasi yang didasarkan pada perilaku perpindahan individu dalam kawanan, seperti kawanan serangga, semut, rayap, lebah atau burung.

b. Ant Colony System (ACS)

   ACS adalah Algoritma semut merupakan algoritma yang digunakan untuk menyelesaikan masalah-masalah optimasi yang terinspirasi dari perilaku semut.

c. Artificial Bee Colony (ABC)

   ABC adalah kecerdasan buatan yang menirukan koloni lebah dalam mencari sumber *nektahr* (sari bunga).

d. Firefly Algorithm (FA)

   FA adalah sebuah algoritma metaheuristik yang terinspirasi dari perilaku kedip cahaya kunang-kunang.

2.2. Algoritma Firefly

Algoritma kunang-kunang atau *firefly algorithm* (*FA*) adalah algoritma metaheuristik yang terinspirasi dari perilaku kedip cahaya kunang-kunang. Terdapat dua fungsi dasar kedip cahaya tersebut, yaitu untuk menarik perhatian kunang-kunang yang lain (komunikasi) dan untuk menarik mangsa.

Algoritma ini dikembangkan oleh Xin-She Yang pada 2007, yang menggunakan tiga peraturan sebagai berikut [**1**].

1. Semua kunang-kunang berjenis kelamin satu sehingga seekor kunang-kunang akan tertarik pada kunang-kunang lain terlepas dari jenis kelamin.
2. Daya tarik sebanding dengan tingkat kecerahan cahaya kedip kunang-kunang. Oleh karena itu kunang-kunang dengan tingkat kecerahan lebih rendak akan tertarik dan bergerak ke kunang-kunang dengan tingkat kecerahan lebih tinggi. kecerahan dapat berkurang seiring dengan bertambahnya jarak dan adanya penyerapan cahaya akibat faktor udara. Jika tidak ada yang paling terang dari populasi tersebut, semua kunang-kunang akan bergerak acak.
3. Kecerahan atau intensitas cahaya dari seekor kunang-kunang dipengaruhi atau ditentukan oleh nilai fungsi tujuan dari masalah yang diberikan. Untuk masalah maksimisasi, intensitas cahaya sebanding dengan nilai fungsi tujuan.

Formulasi umum dari algoritma ini disajikan bersama-sama dengan pemodelan matematika analisis untuk memecahkan masalah dengan tujuan ekuivalen fungsi. Hasilnya dibandingkan dengan yang diperoleh dengan teknik alternatif yang diusulkan oleh literatur untuk menunjukkan bahwa ia mampu menghasilkan solusi optimal baik dan benar [**1**].

2.2.1. Perilaku kunang-kunang

Kunang-kunang merupakan keluarga serangga yang mampu menghasilkan cahaya alami untuk menarik pasangan atau mangsanya. Terdapat sekitar 2000 spesies kunang-kunang dan sebagian besar kunang-kunang hidup berkelompok, populasi kunang-kunang yaitu lebih dari 40 ekor tiap kelompoknya dan memproduksi cahaya yang berirama dan pendek. Kedipan cahaya yang dihasilkan kunang-kunang sering tampak dalam pola yang unik, kilatan cahaya yang diproduksi oleh proses *bioluminescene* juga berfungsi sebagai penarik perhatian kunang-kunang lain dan bisa digunkan sebagai mekanisme perlindungan sehingga membuat pemandangan menakjubkan didaerah tropis selama musim panas karna kilatan cahaya dari kunang-kunang tersebut [**1**].

Kunang-kunang betina merespon pola kilatan yang unik dari kunang-kunang jantan yang sejenis, kunang-kunang betina jenis *Photuris* dapat mendengar sinyal *bioluminescene* dan bahkan meniru pola perkawinan dari spesies lain sehingga dapat memancing dan memakan kunang-kunang jantan yang mungkin melakukan kesalahan dalam memberi sinyal. Beberapa kunang-kunang

tropis bahkan dapat mensinkronkan cahaya mereka, sehingga membentuk perilaku *biological selft-organized* [**1**].

2.2.2. Intensitas Cahaya Dan Keatraktifan

Pada algoritma *firefly*, ada dua hal yang berkaitan dan sangat penting dalam algoritma *firefly* yaitu itensitas cahaya dan fungsi keatraktifan. Dalam hal ini banyak dari kita berasumsi bahwa keatraktifan dipengaruhi oleh tingkat itensitas cahaya. daya tarik seekor kunang-kunang ditentukan oleh seberapa terang cahaya (*brightness*) yang dimilikinya yang dipengaruhi oleh fungsi objektif. Kunang-kunang yang memiliki hasil fungsi objektif paling maksimal pada algoritma *firefly* dilambangkan sebagai kunang-kunang yang mempunyai intensitas cahaya paling terang sehingga kunang-kunang lain bergerak mendekat. Berikut ini adalah persamaan intensitas cahaya [**4**].

$$I_i = f(x) \tag{2.1}$$

Dimana : I_i = Intensitas Cahaya Firefly i (i = 1, 2, ... ,n)

$f(x)$ = Fungsi Objektif

Intensitas cahaya menyatakan tingkat kecerahan dari kunang-kunang, semakin cerah seekor kunang-kunang maka dikatakan semakin bagus solusi kunang-kunang tersebut. Pada kunang-kunang yang berdekatan maka akan timbul daya tarik menarik yang dilambangkan dengan *beta* (β). Daya tarik yang diindikasikan dengan cerah tidaknya cahaya yang dihasilkan oleh kunang-kunang. Besar kecilnya daya tarik bersifat relatif yang tergantung pada seberapa jauh kunang-kunang, Karena daya tarik kunang-kunang sebanding dengan intensitas cahaya yang dilihat oleh kunang-kunang di dekatnya dapat dirumuskan daya tarik (*attractiveness*) menggunakan persamaan (2.2).

$$\beta = \beta_0 e^{-\gamma r2} \tag{2.2}$$

Dimana : β = Keaktraktifan

β_0 = Nilai daya tarik kunang-kunang (*basebeta*)

γ = Koefisien penyerapan cahaya (*gamma*)

r = Jarak

Persamaan pertama yang terdiri dari variabel β_0 dengan variasi nilai antara 0 sampai 1, namun disini variabel β_0 ini bernilai 1 karna merupakan nilai

keaktratifan awal pada kunang-kunang [**4**], Selain itu, variabel γ = 0,01 adalah koefisien penyerapan cahaya awal yang mengontrol penurunan intensitas cahaya sehingga intensitas cahaya semakin berkurang atau cahaya semakin lemah apabila jarak semakin jauh, nilai gamma bervariasi dari 0 sampai 10. [**5**] [**6**].

2.2.3. Jarak dan Pergerakan Kunang-kunang

Dalam menghitung sebuah jarak dibutuhkan 2 buah objek atau lebih, begitu juga dengan kunang-kunang akan memerlukan kunang-kunang lain yang lebih terang untuk bisa bergerak mendekat dan kunang-kunang akan bergerak secara acak jika tidak ada kunang-kunang yang memiliki intensitas cahaya paling terang diantara kelompoknya. Cahaya yang dihasilkan kunang-kuanang akan berkurang seiring dengan bertambahnya jarak maka semakin jauh jarak kunang-kunang akan semakin redup cahaya kunang-kunang yang berkedip. Jarak antara dua kunang-kunang *i* dan *j*, diwakili oleh x_i dan x_j adalah jarak *euclidean,* dimana selisih koordinat lokasi kunang-kunang *i* terhadap kunang-kunang *j* merupakan jarak diantara keduanya, yang dirumuskan menggunakan Persamaan (2.3).

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{2.3}$$

Pergerakan dari kunang-kunang *i* yang bergerak ke arah kunang-kunang *j* yang lebih terang atau menuju ke tingkat intensitas cahaya yang terbaik, ditentukan dengan persamaan (2.4).

$$x_i' = x_i + \beta = \beta_0 e^{-\gamma r 2}(x_j - x_j) + \alpha\left(rand - \frac{1}{2}\right) \tag{2.4}$$

Dimana : r_{ij} = Jarak kunang-kunang *i* dan *j*

x_i' = Posisi kunang-kunang *i* yang baru

x_i = Posisi kunang-kunang *i* sekarang (*i* = 1, 2, ... n)

x_j = Posisi kunang-kunang *j* sekarang (*j* = 1, 2, ... n)

d = Jumlah koordinat dimensi

$x_{i,k}$ = Posisi kunang-kunang *i* pada dimensi k

$x_{j,k}$ = Posisi kunang-kunang *j* pada dimensi k

β_0 = Daya tarik terhadap r = 0

γ = Koefisien penyerapan cahaya [0-10]

α = Parameter pengacak [0-1]

rand = Nilai random [0-1]

Pergerakan terjadi jika intensitas cahaya antara satu kunang-kunang lebih besar dibanding kunang kunang yang lain. Pada persamaan (2.4) penggalan rumus kedua menyatakan daya tarik (*attractiveness)* dari kunang-kunang, Sedangkan pada penggalan rumus ketiga adalah pergerakan acak kunang-kunang. Konstanta *alpha* dan *rand* adalah sebuah vector dari nilai acak yang memeiliki kisaran antara 0 sampai 1 yang diambil dari distribusi *gaussian* atau distribusi seragam. [**7**] [**6**].

2.2.4. Penyelesaian Dalam Algoritma Firefly

Setelah membuat model program linear dan menentukan nilai fitness dari masing-masing kandidat solusi yang sudah ditentukan maka dilanjutkan dengan menganalisisnya menggunakan algoritma *firefly*. Adapun gambaran umum tahapan metode algoritma *firefly* dalam menyelesaikan permasalahan optimasi adalah sebagai berikut [**8**]:

1. Inisialisasi populasi dan parameter algoritma kunang-kunang.

   Proses inisialisasi dilakukan dengan cara menentukan jumlah populasinya dan iterasi yang akan dilakukan serta menentukan nilai untuk variabel β_0 (*base beta*), γ (*gamma*) dan α (*alpha*) dan *rand* untuk menghitung tingkat ketertarikan, jarak dan pergerakan antar kunang-kunang. dimana jumlah populasi yaitu jumlah calon kandidat solusi yang sudah ditentukan di penentuan kombinasi kandidat solusi pada tahapan pembuatan model program *linear*.

2. Menentukan dimensi *firefly* sebagai posisi awal kunang-kunang

   Pembangkitan bilangan acak sebagai nilai posisi awal kunang-kunang, nilai ini adalah kombinasi kandidat solusi yang sudah dibangkitkan sebelumnya, dimana terdapat dua dimensi yaitu dimensi *i* dan dimensi *j*. Banyaknya dimensi *i* ditentukan oleh banyaknya populasi *firefly* yang merupakan jumlah kandidat solusi yang diinginkan sedangkan banyaknya dimensi *j* ditentukan oleh nilai dari kandidat solusi tersebut.

3. Menghitung nilai intensitas cahaya dari setiap kunang-kunang.

   Untuk memperoleh nilai intensitas cahaya dibutuhkan nilai hasil dari evaluasi fungsi objektif. Maka nilai *fitness* yang sudah didapat sebelumnya dijadikan

sebagai nilai intensitas cahaya setiap kunang-kunang. Karena tujuan permasalahan adalah mencari nilai maksimal, jadi semakin tinggi nilai fungsi, maka semakin tinggi nilai intensitasnya.

4. Menentukan *Gbest* (*Global best*)

   Nilai *Gbest* merupakan nilai intensitas cahaya paling terang dari semua kunang-kunang. Setelah didapat nilai intensitasnya setiap kunang-kunang dibandingkan untuk mencari kunang-kunang dengan intensitas cahaya paling tinggi. kemudian nilai posisi dari kunang-kunang yang memiliki intensitas cahaya paling tinggi tersebut akan dijadikan *Gbest*.

5. Menghitung jarak posisi antar kunang-kunang.

   Perhitungan jarak setiap kunang-kunang terhadap kunang-kunang yang memiliki intensitas cahaya paling tinggi (*Gbest*) dengan menggunakan metode *euclidean*.

6. Menghitung nilai keaktraktifan antar kunang-kunang.

   Lakukan perhitungan nilai beta menggunakan konstanta *base beta*, *gamma* dan jarak yang sudah diketahui sebelumnya.

7. Melakukan pergerakan pada kunang-kunang.

   Posisi masing-masing kunang-kunang bergerak ke kunang-kunang dengan intensitas cahaya yang lebih besar sehingga kunang-kunang akan menghasilkan posisi baru, kecuali untuk *Gbest* posisinya tidak berubah karna tidak melakukan pergerakan.

8. Memperbaharui posisi kunang-kunang yang sudah bergerak.

   Perbaharuan posisi setiap kunang-kunang akan diurutkan dari yang terkecil hingga terbesar melalui implementasi aturan *Smallest Position Value* (*SPV*) untuk mengubah setiap dimensi dalam nilai posisi kunang-kunang menjadi bentuk urutan permutasi sehingga terbentuk kandidat solusi yang baru.

9. Mengevaluasi nilai fitness kunang-kunang pada posisi baru.

   Setelah ditempatkan pada posisi baru, kunang-kunang akan dievaluasi kembali dan nilai *fitness* yang dihasilkan oleh kandidat solusi yang baru.

10. Perangkingan kunang-kunang berdasarkan intensitas cahaya baru.

    Nilai *fitness* yang dihasilkan oleh kandidat solusi yang baru tersebut akan digunakan sebagai nilai intensitas cahaya kunang-kunang yang baru.

11. Mengecek kandidat solusi apakah sudah konvergen atau belum.
    Nilai dikatakan konveregen jika posisi kunang-kunang mencapai posisi tujuan atau posisi terbaik. apabila nilai posisi baru melebihi batas nilai minimal dan maksimal pada masing-masing dimensi atau belum mencapai iterasi maksimum, maka kembalikan nilai posisi nya agar kembali dalam rentang nilai pada dimensi tersebut dan ulangi langkah awal dengan mencari kunang-kunang *Gbest* baru sampai menemukan solusi terbaik atau iterasi selesai.

2.3. Tentang Perusahaan

Desi bakery adalah sebuah industri rumah tangga yang memproduksi roti di Bandung, tepatnya di Jl. Paralon Gg sawargi rt 02/11 Cijerah, Bandung Barat. Dengan pemiliknya bernama Bapak Supriatna. industri tersebut dibangun oleh Bapak Supriatna di rumahnya sendiri yang dijadikan tempat usaha pengolahan dan pembuatan roti dengan dasar dari keterampilan dan pengalaman pada saat bekerja pada industri roti milik saudaranya. pada industri tersebut Bapak Supriatna bekerja sebagai karyawan di bagian produksi sehingga mengetahui resep adonan-adonan roti yang menjadi bekal ilmu dan keterampilannya. Kemudian pada tahun 1998 mempunyai inisiatif untuk mendirikan usaha sendiri.

Usaha Bapak Supriatna dalam menjalankan usahanya dilakukan sendiri bersama keluarga baik dalam pengolahan maupun pemasaran produknya. Nama industri Desi Bakery diambil dari nama anak kedua mereka namun sebelum memakai nama Desi Bakery, Bapak Supriatna memakai nama DR putra sebagai label produk roti mereka. Adapun pengiriman roti biasanya dikirim ke beberapa toko pengecer di lima daerah di Bandung yaitu Dago, Cimahi, Buah Batu, Lewi Panjang dan Stasiun Kiara Condong. Seiring berjalannya waktu mereka juga berusaha untuk meningkatkan kualitas dan kuantitas hasil produksi agar dapat memenuhi kebutuhan dan keinginan dari konsumen.

2.4. Tenaga Kerja dan Jam Kerja

Jumlah tenaga kerja di Desi Bakery saat ini adalah lima orang. yang terdiri dari satu orang pekerja di bagian pengadonan, dua orang di bagian pemotongan dan pencetakan, satu orang bagian frementasi dan pemanggangan, dan terakhir satu orang dibagian pengemasan. Setiap pekerja bertanggung jawab

atas pekerjaannya masing-masing. Hari kerja di Desi Bakery sebanyak enam hari kerja, dari senin sampai sabtu. Jam kerja perhari dari pukul 03.00 WIB sampai pukul 14.00 WIB.

2.5. Proses Produksi

Produksi merupakan bagian yang sangat penting di dalam suatu industri, terutama dalam industri roti, dimulai dari perencanaan penggunaan bahan baku hingga waktu yang dibutuhkan dalam proses produksi, dengan pengolahan bahan dan pemanfaatan waktu produksi seperti pada tahapan pengadonan, pencetakan, pemanggangan dan pengemasan yang efektif dan efisien akan menghasilkan produk yang sesuai dengan standar mutu yang telah diterapkan. Proses produksi adalah urut-urutan pekerjaan mulai dari bahan mentah sampai menjadi bahan siap dijual atau barang jadi. Selain bahan mentah yang berupa bahan baku juga diperlukan peralatan dalam proses produksi. Bahan yang digunakan dalam pembuatan produk antara lain adalah terigu, mentega, gula, garam, telur, air serta bahan tambahan yang digunakan dan ditambahkan pada proses produksi untuk membantu meningkatkan kualitas produk, bahan tambahan ini yaitu pasta cokelat, susu, keju, kacang. dan plastik yang digunakan untuk pengemasan produk.

2.5.1. Tahapan Membuat roti

Adapun tahapan dalam pembuatan roti dilakukan sebagai berikut :

1. Pengadonan

   Proses pertama yang dilakukan adalah memasukan bahan-bahan yang telah ditimbang takarannya, seperti tepung, mentega, gula, garam, telur dan air yang kemudian dicampur, diaduk dan diuleni hingga menjadi adonan. Proses pengadonan ini berlangsung 20 menit.

2. Pencetakan

   Masing-masing adonan yang telah diadon ini selanjutnya ditipiskan dengan menggunakan rol. Proses selanjutnya adalah mengisi adonan yang telah dipres dengan isi yang diinginkan seperti cokelat, susu, keju dan kacang. Adoanan yang sudah diisi tersebut kemudian dibentuk menjadi 3 jenis roti menggunakan cetakan yang sudah tersedia. Sesudah dibentuk adonan tersebut diletakkan dalam loyang, proses pencetakan ini dilakukan selama 300 menit.

3. Fermentasi

   Di dalam ruang fermentasi ini, adonan-adonan yang tersusun dalam loyang diuapkan agar mengembang. Proses pengembangan ini berlangsung sekitar 1,5 jam. Kecuali pia basah tidak memerlukan proses fermentasi.

4. Pemanggangan

   Setelah adonan-adonan sudah mengembang hasil dari proses fermentasi. Adonan ini selanjutnya dimasukkan kedalam oven. Pemanggangan berlangsung sekitar 90 menit.

5. Pengemasan

   Setelah pemanggangan selesai maka roti tersebut sudah siap untuk dikemas dan diberi label untuk selanjutnya di pasarkan ke toko-toko pengecer.

2.5.2. Jenis Hasil Produksi

Industri rumah tangga Desi Bakeri merupakan usaha yang bergerak dalam pembuatan roti. Roti yang diproduksi oleh industri ini beraneka ragam rasa dan jenisnya, yaitu sebagai berikut:

Tabel 2.1 Hasil produksi cv. Desi Bakery

Jenis Roti	Varian Rasa
Roti Biasa	Cokelat
	Susu
	Keju
	Kacang
Roti Kombinasi	Cokelat – Kacang
	Keju – Susu
Pia Basah	Cokelat

2.6. Program Linear

Untuk dapat menyelesaikan permasalahan produksi roti perlu digunakan sebuah cara untuk membuat model matematika dari permasalahan tersebut yaitu dengan menggunakan program *linear*. Program *linear* adalah suatu cara yang digunakan untuk menyelesaikan masalah optimasi suatu model *linear* dengan

berbagai tujuan dan kendala yang dihadapinya. Pemrograman *linear* berkaitan dengan penjelasan suatu kasus dalam dunia nyata sebagai suatu model matematika yang meliputi perencanaan aktivitas untuk mendapatkan hasil optimal, yaitu sebuah hasil yang mencapai tujuan terbaik diantara semua kemungkinan alternatif yang ada. Tujuan utama dari program *linear* adalah menentukan nilai optimum (maksimal/minimal) dari fungsi tujuan yang telah ditetapkan. Sesuai dengan sifat dari program ini, maka hubungan antara variabel *input/output* serta fungsi objektif yang harus dicapai harus merupakan hubungan linier baik yang berbentuk persamaan linier atau pertidaksamaan linier sedangkan kuantitas *input/output* merupakan kuantitas yang non negatif [**9**].

Dalam membangun model dari persoalan *linear programming* digunakan karakteristik-karakteristik sebagai berikut :

1. Variabel keputusan

   Variabel keputusan adalah variabel yang menguraikan secara lengkap keputusan-keputusan yang akan dibuat. Yang dimaksud disini adalah X1, X2, X3, ..., Xn.

2. Fungsi tujuan

   Fungsi tujuan merupakan fungsi dari variabel keputusan yang menggambarkan tujuan atau sasaran didalam program *linear* yang dimaksudkan untuk menentukan nilai optimum dari fungsi tersebut yaitu nilai maksimal untuk keuntungan dan nilai minimal untuk masalah biaya.

   Fungsi tujuan : Max/Min z = $f(x)$

3. Pembatas-pembatas

   Merupakan kendala-kendala yang dihadapi sehingga kita tidak bisa menentukan harga variabel keputusan secara sembarang namun bentuk penyajian secara matematika yang diperlukan berkenaan dengan adanya keterbatasan sumber daya yang tersedia. nilai dari variabel keputusan tersebut dibatasi oleh pembatas yaitu: $gi(x)$ $(\leq)(=)$ $(\geq)$ bi i=1,2,....,m $x \geq 0$

4. Pembatas tanda

   Pembatas tanda adalah pembatas yang menjelaskan apakah variabel keputusannya diasumsikan hanya berharga non negatif atau variabel keputusan tersebut.

Pada setiap masalah ditentukan variabel keputusan, fungsi tujuan, dan sistem kendala, yang bersama-sama membentuk suatu model matematika dari dunia nyata. Bentuk umum dari program linear adalah [**9**]:

Maksimumkan :

$$F(x) = \sum_{j=1}^{n} C_j X_j \tag{2.5}$$

Dengan Batasan :

$$\sum_{j=1}^{n} a_j X_j \{<, =, >\} b_i \tag{2.6}$$

Keterangan :

F(x) : Nilai fungsi tujuan.

C_j : Sambangan per unit kegiatan.

X_j : Banyaknya kegiatan j (j = 1, 2, 3, … n)

a_{ij} : Banyaknya sumberdaya i yang dikonsumsi sumber daya j.

b_i : Jumlah sumberdaya i (i = 1, 2, …, m)

2.6.1. Multi-Objective Optimization

Optimalisasi merupakan suatu proses untuk mengoptimalkan suatu solusi agar ditemukannya solusi terbaik dari sekumpulan alternatif solusi yang ada dengan menggunakan formulasi matematika. Optimalisasi dilakukan dengan memaksimalkan atau meminimalkan suatu fungsi objektif dengan tidak melanggar batasan yang ada (Sianturi, 2012). Dengan adanya optimalisasi, suatu sistem dapat meningkatkan efektifitasnya seperti meminimalisir biaya, meningkatkan keuntungan, meminimalisir waktu proses, dan sebagainya [**10**].

Optimasi Tujuan Ganda (*Multi-Objective*) Dalam perencanaan dan pengendalian produksi, yang bertujuan meminimalkan keterlambatan, lama aliran produk, kapasitas yang dibutuhkan, dan lain-lain. *Multi-Objective Optimization* merupakan metode penyelesaian persoalan optimasi yang terdiri lebih dari satu fungsi tujuan secara simultan (bersamaan). Dalam pemecahannya, seringkali terjadi konflik diantara tujuan dan solusinya bukanlah solusi tunggal tetapi berupa himpunan solusi. Cara yang umum untuk menyelesaikan masalah tujuan ganda

adalah menetapkan bobot untuk tujuan-tujuan yang berbeda sesuai dengan arti relatip pentingnya, dan kemudian mengembangkan metode yang tepat untuk menyelesaikan masalah tujuan tunggal (*single-objective*) yang diperoleh [**10**].

Metode yang dapat digunakan dalam optimasi *multi-objective* terdiri dari metode pembobotan, metode jarak, pendekatan kompromi dan pendekatan *goal programming*. Setiap metode memiliki caranya masing-masing dalam penyelesaiannya. Sedangkan penulis akan membahas dengan menggunakan metode pembobotan. Metode pembobotan ini salah satu metode yang digunakan untuk mengubah fungsi *multi-objective* menjadi fungsi tujuan tunggal. Ada tiga metode penyelesaian dalam pembobotan, yaitu *weight sum method*, *random-weight* dan *adaptive-weight* [**10**].

2.6.2. Pembuatan Model Program Linear

Gambaran umum tahapan pembuatan model dan penentuan nilai *fitness* awal dari kandidat solusi yang akan dicari solusi optimalnya yaitu sebagai berikut:

1. Membuat model matematika dari kasus yang diteliti

   Model matematika dibuat berdasarkan data jenis, harga bahan baku dan waktu pencetakan setiap roti sehingga akan diperoleh fungsi tujuan serta kendala keterbatasan seperti target produksi yang harus dipenuhi, biaya produksi serta waktu pencetakan rotinya.
2. Penentuan kombinasi kandidat solusi

   Setelah model dibuat maka proses penentuan ini akan dilakukan dengan menginisialisasi kandidat solusi dengan cara menentukan jumlah produksi setiap roti dengan memberikan nilai acak sesuai batasan yang telah ditentukan.
3. Perhitungan nilai fitness sebagai fungsi objektif awal

   Setiap kandidat solusi yang sudah ditentukan sebelumnya kemudian dihitung masing-masing nilainya yang akan dijadikan nilai fitness awal.
4. Proses penyelesaian menggunakan algoritma firefly.

   Setelah didapat nilai fitness dari masing-masing kandidat solusi maka dilakukan penyelesaian menggunakan algoritma *firefly* melalui tahapan menginisialisasi populasi dan parameternya serta menghitung jarak, keaktraktifan dan pergerakannya sehingga akan memperoleh solusi optimal.

2.7. Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena *UML* menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi dan mengkomunikaskan rancangan mereka dengan yang lain [**11**].

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh *Booch*, *Object Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan *design* ke dalam empat tahapan *iterative*, yaitu: identifikasi kelas-kelas dan obyek-obyek, identifikasi *semantic* dari hubungan obyek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan *Object Modeling Technique* yang dikembangkan *Rumbaugh* didasarkan pada analisis terstruktur dan permodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, *design* sistem, *design* obyek dan implementasi. Keungulan metode ini aladah dalam penotasian yang mendukung semua konsep *Object Oriented Software Engineering*. Metode *Object Oriented Software Engineering Jacobson* lebih memberikan penekanan pada *use case*. *Object Oriented Software Engineering* memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, *design* dan implementasi, dan model pengujian. Keungulan *metode* ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak [**11**].

Dengan *UML*, metode *Booch*, *Object Modeling Technique* dan *Object Oriented Software Engineering* digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga *UML* lebih *ekspresif* dan seragam dari pada *metode* lainnya.

2.7.1. Diagram-diagram UML

Didalam *UML* terdapat beberapa macam diagram yang dapat menggambarkan suatu sistem, berikut adalah beberapa diagram yang terdapat di dalam *UML*.

1. Use Case dan Use Case Diagram

   Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal antara pengguna sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan, *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna [**11**].

   Use case diagram adalah penggambaran interaksi pengguna sistem (*actor*) dengan kasus (*use case*) yang telah disesuaikan dengan langkah-langkah (scenario). *Diagram use case* menunjukan 3 aspek dari *system* yaitu *actor*, *use case* dan sistem, sub sistem *boundary*. *Actor* mewakili perang orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*.

1. Class Diagram

   Class dalam notasi *UML* digambarkan dengan kotak. *Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek. *Class* menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). *Class* mempunyai beberapa bagian yang menjelaskan isi *class* [**11**]:

   a. *Attribute* adalah *property* dari sebuah *class*. *Attribute* ini melukiskan batas nilai yang mungkin ada pada obyek *class*. Sebuah *class* mungkin mempunyai nol atau lebih *attribute*.
   b. *Operation* adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau *class* yang lain dapat lakukan untuk sebuah *class*.

c. *Responsibility* adalah keterangan tentang apa yang akan dilakukan *class* yaitu apa yang akan dicapai oleh *attribute* dan *operation*.

2. Statechart diagram

Menelusuri individu-individu obyek melalui keseluruhan daur hidupnya, menspesifikasikan semua urutan yang mungkin dari pesan-pesan yang akan diterima obyek tersebut, bersama-sama dengan tanggapan atas pesan-pesan tersebut. *Statechart* diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu obyek pada sistem sebagai akibat dari stimulasi yang diterima. Pada umunya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*). [**11**].

3. Activity Diagram

Activity Diagram adalah teknik untuk mediskripsikan logika *procedural*, proses bisnis dan aliran kerja dalam banyak kasus. *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *parallel* sedangkan *flowchart* tidak bisa. *Activity* diagram tidak menunjukan apa yang terjadi, tetap tidak menunjukan siapa yang melakukan apa. *Activity* diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir, *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Simbol-simbol yang sering digunakan pada pembuatan *activity diagram* [**11**].

4. Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan pesan yang diletakan diantara obyek-obyek ini dalam *use case*.

Sequence diagram menggambarkan interaksi antar obyek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi *vertical* (waktu) dan dimensi *horizontal* (obyek-obyek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan *scenario* atau rangkaian langkah-langkah yang dilakukan

sebagai *respons* dari sebuah *event* untuk menghasilkan output tertentu. Komponen utama *sequence* diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan *progress vertical* [**11**].

Penjelasan dari komponen utama *sequence diagram* yaitu :

a. Obyek

Obyek diletakan di dekat bagian atas *diagram* dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan *diagram*. Setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi *operation* dari *participiant*. Panjang kotak ini berbanding lurus dengan durasi *activation*.

b. Message

Sebuah *message* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

c. Time

Time adalah *diagram* yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas kebawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibandingkan *message* yang lebih dekat ke bawah.

5. Collaboration diagram

Collaboration diagram adalah perluasan dari obyek *diagram*. Obyek *diagram* menunjukan obyek-obyek dan hubungannya satu dengan yang lain. *Collaboration diagram* menunjukan *message-message* obyek yang dikirimkan satu sama lain. *Collaboration* diagram juga menggambarkan interaksi antar obyek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing obyek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, dimana *message* dari *level* tertinggi memiliki nomor satu *Message* dari *level* yang sama memiliki *prefix* yang sama. [**11**].

6. Component diagram

   Component diagram merepresentasikan dunia nyata item yaitu *component software*. *Component* diagram menggambarkan struktur dan hubunganantar komponen piranti lunak, termasuk ketergantungan (*dependency*) diantaranya. *Component software* adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada compile time, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk *komponen* lain. *Component* diagram mengandung *component, interface* dan *relationship* [**11**].

7. Deployment diagram

   Deployment diagram menunjukan tata letak sebuah sistem secara fisik, menampakan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*. *Deployment* diagram menyediakan gambaran bagaimana sistem secara fisik akan terlihat. Sistem terdiri dari node-node dimana setiap node diwakili untuk sebuah kubus. Garis yang menghubungkan antara dua kubus menunjukan hubungan diantara kedia node tersebut. Tipe node bisa berupa *device* yang berwujud *hardware* dan bisa juga *processor* (yang mengeksekusi *component*) atau *execution environment* (*software* yang menjadi *host* atau mengandung *software* yang lain) [**11**].

2.8. Microsoft Visual C# 2010 Express

Microsoft Visual Studio Express adalah kumpulan *freeware* lingkungan terpadu (*IDE*) yang dikembangkan oleh *Microsoft* yang merupakan versi ringan dari *Microsoft* Visual Studio. Gagasan edisi ekspres menurut *Microsoft* adalah untuk menyediakan efisiensi, mudah digunakan dan mudah dipelajari bagi pengguna *IDE* selain pengembang perangkat lunak profesional.

Microsoft Visual C# 2010 atau Visual C#(C *Sharp*) adalah bahasa pemrograman yang bekerja dalam sistem operasi *Windows* yang merupakan bahasa pemrograman yang mudah dipahami karena menggunakan struktur

pemrograman bahasa C. Visual C# juga menyediakan berbagai *tools* yang dapat mempermudah dalam membuat suatu aplikasi, sehingga akan mempermudah dalam pembuatan *web browser*. Visual C# mendukung pemrograman berbasis objek atau biasa di sebut *OOP* (*Object Oriented Programing*) dan memudahkan programmer untuk membuat aplikasi berbasis objek serta mendukung aplikasi berbasis *.NET*. Selain itu, Visual C# juga menyediakan fasiltas koneksi ke *database MySQL* dengan menggunakan konektor tambahan sebagai penghubungnya dan mendukung perintah – perintah *MySQL*.

2.8.1. Kelebihan Microsoft Visual Studio

Microsoft Visual Studio Express merupakan versi ringan dari *Microsoft visual* studio. Kedua *freeware* tersebut merupakan pusat pengembangan terpadu adatu *IDE* yang didirikan oleh *Microsoft*. Pembuatan aplikasi *visual studio* dalam format ekspres sesungguhnya bertujuan menyajikan sebuah layanan yang lebih efisien, cepat, dan mudah dipahami oleh orang yang bahkan baru saja belajar menggunakan *IDE*.

Berikut ini adalah kelebihan dari *Microsoft Visual Studio*:

1. *Visual Studio IDE* dukungan untuk pemrograman paralel.
2. Pola Perpustakaan Paralel: *Native* C + + perpustakaan yang menggunakan lambda fungsi dan menyelaraskan baik dengan *STL*.
3. Paralel *Extensions* ke *Framework*. NET: Perpustakaan-berdasarkan dukungan untuk data pentingdan paralelisme tugas melalui Perpustakaan Tugas Paralel, dukungan untuk data deklaratif paralelisme melalui *PLINQ*, dan bekerja koordinasi dan mengelola bersama negara melalui Koordinasi Data Struktur.
4. *Concurrency Runtime*: Manajemen Sumber Daya *hardware multicore* dan penjadwalan tugas.
5. *Debugging windows* Paralel yang akan memberikan pandangan tugas berjalan di aplikasi, dan tampilan grafis dari tumpukan panggilan paralel.
6. *Profiling* dilihat paralel yang akan memberikan pemahaman pengembang pemanfaatan aplikasi inti.