BAB 3
ANALISIS DAN KEBUTUHAN IMPLEMENTASI

3.1. Deskripsi Masalah

Desi Bakery adalah suatu industri rumah tangga yang bergerak dalam bidang produksi roti. Masalah yang dialami dan akan berdampak pada keuntungan yang didapat dari produksi di industri ini adalah sebagai berikut:

1. Keterbatasan pegawai
2. Keterbatasan alat produksi
3. Keterbatasan biaya produksi
4. Keterbatasan waktu produksi

3.2. Analisis Masalah Produksi

Analisis merupakan proses identifikasi serta evaluasi mengenai masalah-masalah yang timbul dari pengembangan *simulator*, baik mengenai aturan serta teknik-teknik yang digunakan dalam pengembangan *simulator* tersebut.

Dalam pembangunan *simulator* produksi ini yang akan dicari solusi optimalnya hanya dua permasalahan utama yaitu:

1. Keterbatasan biaya produksi
2. Keterbatasan waktu produksi

Adapun untuk keterbatasan pegawai dan alat produksinya akan diabaikan karena dengan mengoptimalkan biaya dan waktu akan membantu memaksimalkan penjualan dan meminimalkan biaya dan waktu produksinya walaupun dengan jumlah pegawai dan alat produksi yang terbatas.

Untuk membuat model program *linear* diperlukan data jenis, harga bahan baku dan harga jual setiap roti serta data waktu dalam proses pencetakan rotinya dengan diasumsikan kecepatan dalam membuat satu buah roti.

Dari data-data tersebut akan diperoleh beberapa fungsi tujuan yaitu fungsi tujuan pertama didapat dari harga jual roti untuk memaksimumkan penjualan dan fungsi tujuan kedua adalah jumlah produksi setuap roti untuk meminimumkan target minimal, serta akan mengalami beberapa kendala seperti kendala keterbatasan target produksi yang harus dipenuhi, keterbatasan biaya produksi dan keterbatasan waktu pencetakan roti.

Berikut ini adalah data bahan baku beserta harga-harganya untuk pembuatan roti satu kali produksi di CV. Desi Bakery pada tahun 2015 yang ditampilkan pada Tabel 3.1

Tabel 3.1 Daftar jenis dan harga bahan baku

Bahan Baku	ukuran	Harga
Terigu	20 kg	Rp 194000
Mentega	4 kg	Rp 52300
Gula	6 kg	Rp 70800
Telur	2 kg	Rp 44000
Cokelat	3 kg	Rp 35400
Susu	1 kg	Rp 16000
Keju	1 kg	Rp 22000
Kacang	1 kg	Rp 15000
Ragi	500 g	Rp 24000

Dari daftar harga diatas diperoleh total biaya untuk sekali produksi roti yaitu sebesar Rp 473500. total biaya ini akan digunakan untuk batasan biaya produksi dalam pembuatan model program *linear*.

Adapun prosedur tahapan dalam membuat roti yaitu pengadonan, pencetakan, fermentasi, pemanggangan dan pengemasan. Untuk waktu produksi yang akan dioptimasi yaitu hanya pada tahapan pencetakan roti.

Data pengamatan yang diambil pada penelitian ini adalah suatu contoh kasus dari pemrograman *linear* berupa optimalisasi produksi roti dari sebuah industri rumahan yang bernama Desi Bakery. Industri tersebut memproduksi tiga jenis roti yaitu Roti biasa, Roti kombinasi dan Pia basah.

Pembuatan masing-masing roti memerlukan bahan baku terigu, mentega, gula dan telur yang disajikan pada Tabel 3.2.

Tabel 3.2 Bahan baku pembuatan roti

Jenis / Bahan Baku	Roti Biasa	Roti Kombinasi	Pia Basah	Kapasitas Maksimum
	RB	RK	PB	
Terigu	8100g	6500g	4750g	≤ 20.000g
Mentega	1800g	1150g	1050g	≤ 4000g
Gula	2700g	1700g	1600g	≤ 6000g
Telur	810g	700g	450g	≤ 2000g
Jumlah produksi	≤ 600 buah	≥ 120 buah	≤ 700 buah	

Sedangkan untuk mengetahui biaya produksi satu buah rotinya didapat dari total biaya bahan baku yang digunakan untuk masing-masing jenis roti. Dan untuk proses pencetakan roti diasumsikan kecepatan pembuatan rotinya yaitu roti biasa 0,13 menit, roti kombinasi 0,5 menit dan pia basah 0,12 menit. Berikut ini adalah data biaya produksi, waktu pencetakan, target produksi dan harga jual roti disajikan pada Tabel 3.3.

Tabel 3.3 Biaya pembuatan roti sekali produksi

Jenis / Bahan Baku	Roti Biasa	Roti Kombinasi	Pia Basah	Kapasitas Maksimum
	RB	RK	PB	
Terigu	Rp 65475	Rp 24250	Rp 48500	≤ Rp 194000
Mentega	Rp 26730	Rp 13500	Rp 16875	≤ Rp 52300
Gula	Rp 40356	Rp 17700	Rp 20650	≤ Rp 70800
Telur	Rp 19800	Rp 11000	Rp 16500	≤ Rp 44000
Total	Rp 152361	Rp 66450	Rp 102525	
Biaya satu roti	Rp 300	Rp 600	Rp 200	≤ Rp 400000
Waktu pencetakan satu buah roti	0,13 menit	0,5 menit	0,12 menit	≤ 300 menit
Harga jual roti	Rp 1600	Rp 4000	Rp 800	

Masalahnya adalah menentukan berapa buah roti biasa, roti kombinasi dan pia basah yang seharusnya diproduksi agar bisa memaksimalkan penjualan dan meminimalkan target jumlah roti yang harus dipenuhi untuk produksinya.

3.3. Analisis Model Program Linear

Perencanaan produksi yang optimal diperoleh dengan menggunakan *linear programming* kemudian akan diimplementasikan dengan menggunakan algoritma *firefly*. Program *linear* dipilih karena memberikan penyelesaian atas permasalahan mengenai pengoptimalan alokasi sumberdaya berupa bahan baku, jam kerja pegawai, dan target produksi untuk setiap jenis roti. maka akan memberikan kombinasi produk yang optimal sehingga keuntungan yang maksimal dapat dicapai.

Berikut ini adalah tahapan dalam membuat program linear:

3.4. Membuat model matematika berdasarkan fungsi tujuan dan kendala

Dari hasil analisis masalah produksi diperoleh beberapa fungsi tujuan yang didapat dari data jenis, biaya, harga dan target produksi yaitu fungsi tujuan pertama diperoleh dari data harga jual dari setiap jenis roti, untuk harga jual ini ditentukan manual tanpa melihat biaya produksi per satu roti. yaitu roti biasa Rp 1600, roti kombinasi Rp 4000, dan pia basah 800. Sedangkan untuk fungsi tujuan kedua adalah target minimal produksi setiap jenis roti. yaitu roti biasa sebanyak $\leq$ 600 buah, roti kombinasi sebanyak $\geq$ 120 buah buah dan pia basah $\leq$ 700 buah buah roti. Maka formula fungsi tujuannya yaitu sebagai berikut :

Maksimumkan $F_1(x) = 1600RB + 4000RK + 800PB$

Minimumkan $F_2(x) = RB + RK + PB$

F1 merupakan fungsi tujuan pertama untuk penjualan yang dimaksimumkan, sedangkan F2 merupakan fungsi tujuan kedua untuk targat produksi yang diminimumkan, dalam mengubah dua fungsi tujuan menjadi satu fungsi tujuan, harus memiliki satu tujuan maksimum/minimum. Sehingga, dalam hal ini akan memaksimumkan fungsi tujuan . maka fungsi tujuan yang kedua diubah menjadi maksimum seperti berikut:

Minimumkan $F_2(x) = RB + RK + PB$

Diubah menjadi

Maksimumkan $-F_2(x) = RB + RK + PB$

$F_2(x) = -RB - RK - PB$

Kendala yang dipakai dalam permasalahan ini yaitu kendala keterbatasan target produksi, keterbatasan biaya dan keterbatasan waktu. Dari penjelasan yang telah diuraikan dapat disimpulkan bahwa fungsi tujuan berubah seperti pada Tabel 3.4.

Tabel 3.4 Fungsi tujuan dan kendala

Fungsi Tujuan	
Maksimumkan	$F_1(x)$ = 1600 RB + 4000 RK + 800 PB
Maksimumkan	$F_2(x)$ = –RB – RK – PB
Kendala	
Keterbatasan Target produksi	RB ≤ 600
	RK ≥ 120
	PB ≤ 700
Keterbatasan biaya produksi	300 RB + 600 RK + 200 PB ≤ Rp 400.000
Keterbatasan Waktu pencetakan	0,13 RB + 0,5 RK + 0,12 PB ≤ 300 menit
Jumlah roti	RB, RK, PB ≥ 0

Maka untuk menyelesaikan fungsi tujuan ganda merupakan penyelesaian dari beberapa fungsi objektif yang menghasilkan nilai yang paling optimal. Model matematis untuk *Multi-objective Optimization* menggunakan *weight sum method* yang diformulasikan sebagai berikut [**10**]:

$$F = w_1 * F_1(1600(RB) + 4000(RK) + 800(PB)) + w_2 * F_2(-RB - RK - PB) \quad (3.1)$$

Dimana :

w_k = bobot untuk fungsi tujuan ke-k k = 1,2,3, ... , m

$F_k(x)$ = fungsi tujuan ke k k = 1,2,3, ... , m

RB = Roti Biasa

RK = Roti Kombinasi

PB = Pia Basah

4. Penentuan kombinasi kandidat solusi

Proses inisialisasi untuk kandidat solusi dilakukan dengan cara menentukan jumlah produksi setiap jenis roti dengan memberikan nilai acak sesuai batasan yang telah ditentukan.

Tabel 3.5 Batasan produksi roti

Jenis Roti	Batas Bawah	Batas Atas
Roti Biasa	0	600
Roti Kombinasi	120	667
Pia Basah	0	700

Terdapat tiga jenis roti yang akan diproduksi sesuai dengan jenis hasil produksi yang ada pada **Error! Reference source not found.** dan lima buah kandidat solusi dengan nilai awal acak disesuaikan dengan batasan target minimal industri, seperti yang ditampilkan pada Tabel 3.6.

Tabel 3.6 Target jumlah roti yang diproduksi

Kandidat Solusi		Jenis Roti	
	Roti Biasa	Roti Kombinasi	Pia Basah
1	352	163	614
2	370	387	100
3	468	290	165
4	225	243	380
5	322	238	652
6	340	300	559
7	359	280	214
8	352	228	600
9	193	450	187
10	550	255	237
11	164	232	470
12	400	136	252
13	496	260	369
14	496	260	369
15	372	177	455

5. Perhitungan nilai fitness sebagai fungsi objektif awal

Permasalahan yang ingin diselesaikan adalah bagaimana memaksimalkan penjualan dalam produksi, maka hitung nilai fungsi objektif dari setiap kandidat solusi yang telah dibangkitkan dari langkah inisialisasi diatas dengan menghitung masing-masing bobotnya, untuk mendapatkan fungsi tujuan tunggal, substitisikan bobot masing-masing fungsi tujuan. Jika pembobotnya masing-masing sebesar 50% yaitu w1 = 0,5 dan w2 = 0,5. Pada langkah ini, masing-masing kandidat solusi akan dievaluasi dengan cara menghitung nilai *fitness* berdasarkan fungsi objektif yang diberikan, yaitu untuk memaksimalkan penjualan roti dan meminimalkan target produksi roti dalam satu kali produksi. Perhitungan nilai *fitness* masing-masing solusi ditentukan berdasarkan jenis roti yang akan diproduksi. kombinasi tersebut dievaluasi apakah menghasilkan produksi yang optimal dengan cara mengalikan kandidat solusi dengan harga jual setiap produk yang sudah ditentukan. maka proses perhitungan nilai fungsi objektif dari kandidat solusi tersebut seperti pada persamaan (3.1).

$F(x) = w_1 * (1600(RB) + 4000(RK) + 800(PB)) + w_2 * (-RB-RK-PB)$

Perhitungan fungsi objektif setiap kandidat solusi:

Kandidat Solusi 1 : $F(X)_1$

= 0,5 * (1600(352) + 4000(163) + 800(614)) + 0,5 (-352-163-614) = 852636

Kandidat Solusi 2 : $F(X)_2$

= 0,5 * (1600(378) + 4000(387) + 800(100)) + 0,5 (-378-387-100) = 1109572

Kandidat Solusi 3 : $F(X)_3$

= 0,5 * (1600(468) + 4000(290) + 800(165)) + 0,5 (-468-290-165) = 1019939

Kandidat Solusi 4 : $F(X)_4$

= 0,5 * (1600(225) + 4000(243) + 800(380)) + 0,5 (-225-243-380) = 817576

Kandidat Solusi 5 : $F(X)_5$

= 0,5 * (1600(322) + 4000(238) + 800(652)) + 0,5 (-322-238-652) = 993794

Kandidat Solusi 6 : $F(X)_6$

= 0,5 * (1600(340) + 4000(300) + 800(559)) + 0,5 (-340-300-559) = 1095001

Kandidat Solusi 7 : $F(X)_7$

= 0,5 * (1600(359) + 4000(280) + 800(214)) + 0,5 (-359-280-214) = 932374

Kandidat Solusi 8 : $F(X)_8$

= 0,5 * (1600(352) + 4000(228) + 800(600)) + 0,5 (-352-228-600) = 977010

Kandidat Solusi 9 : $F(X)_9$

= 0,5 * (1600(193) + 4000(450) + 800(187)) + 0,5 (-193-450-187) = 1128785

Kandidat Solusi 10 : $F(X)_{10}$

= 0,5 * (1600(550) + 4000(255) + 800(237)) + 0,5 (-550-255-237) = 1044279

Kandidat Solusi 11 : $F(X)_{11}$

= 0,5 * (1600(164) + 4000(232) + 800(470)) + 0,5 (-164-232-470) = 782767

Kandidat Solusi 12 : $F(X)_{12}$

= 0,5 * (1600(400) + 4000(136) + 800(252)) + 0,5 (-400-136-252) = 692406

Kandidat Solusi 13 : $F(X)_{13}$

= 0,5 * (1600(415) + 4000(240) + 800(256)) + 0,5 (-415-240-256) = 913945

Kandidat Solusi 14 : $F(X)_{14}$

= 0,5 * (1600(496) + 4000(260) + 800(369)) + 0,5 (-496-260-369) = 1063838

Kandidat Solusi 15 : $F(X)_{15}$

= 0,5 * (1600(372) + 4000(177) + 800(455)) + 0,5 (-372-177-455) = 833098

Dari dua perhitunga di atas dapat dilihat bahwa nilai *fitness* yang dihasilkan oleh kandidat solusi 1 adalah 852636, kandidat solusi 2 adalah 1109572, kandidat solusi 3 adalah 1019939, kandidat solusi 4 adalah 817576, kandidat solusi 5 adalah 993794, kandidat solusi 6 adalah 1095001, kandidat solusi 7 adalah 932374, kandidat solusi 8 adalah 977010, kandidat solusi 9 adalah 1128785, kandidat solusi 10 adalah 1044279, kandidat solusi 11 adalah 782767, kandidat solusi 12 adalah 692406, kandidat solusi 13 adalah 913945, kandidat solusi 14 adalah 1063838, sedangkan nilai *fitness* yang dihasilkan oleh kandidat solusi 15 adalah 833098.

6. 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.

3.5. Analisis Algoritma Firefly

Analisis algoritma dilakukan untuk mengetahui optimalitas suatu algoritma yang dapat diukur dari parameter-parameternya.

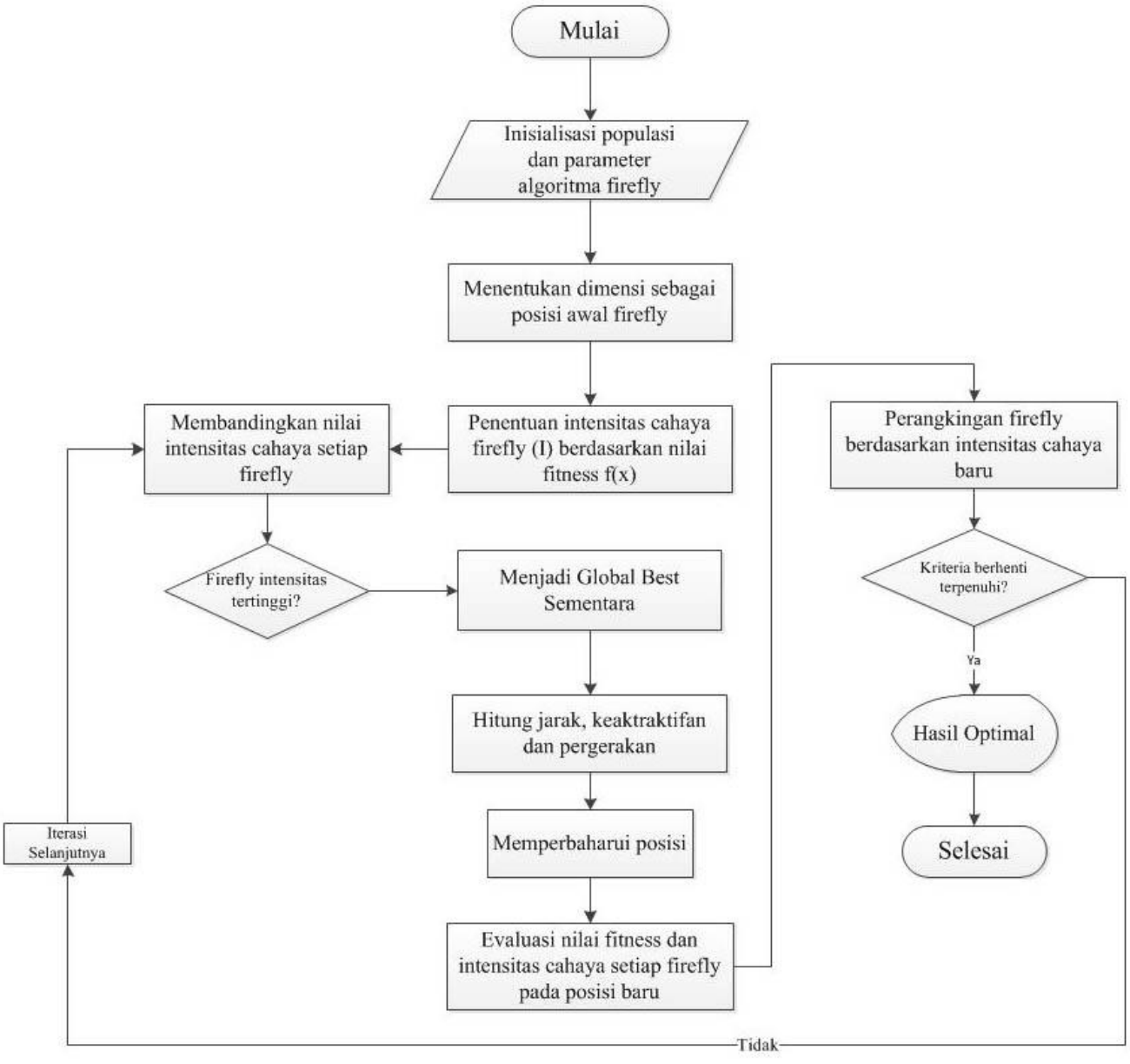
Mengingat keberhasilan Algoritma *Firefly* dalam memecahkan permasalahan optimasi, maka masuk akal jika algoritma *firefly* digunakan untuk permasalahan produksi roti. Algoritma *firefly* akan diterapkan untuk mencari kombinasi solusi optimal pada produksi roti.

3.5.1. Tahapan Proses Penyelesaian Dalam Algoritma Firefly

Pada tahapan optimasi target produksi, penerapan algoritma *firefly* digunakan untuk mengoptimalkan jumlah produksi yang akan dihasilkan dari kombinasi kandidat solusi optimal sehingga didapatkan solusi produksi yang lebih baik dibandingkan proses produksi yang dilakukan sebelumnya sesuai dengan batasan-batasan yang ada.

Tahapan awal pada algoritma *firefly* adalah inisialisasi untuk pembentukan populasi awal dan parameter-parameter algoritma *firefly*, kemudian nilai intensitas cahaya masing-masing *firefly* akan ditentukan sesuai dengan nilai *fitness* yang dihasilkan pada tahapan sebelumnya, setelah didapat nilai intensitas tertinggi maka dilakukan perhitungan jarak dan keaktraktifan baru setelah itu kunang-kunang akan bergerak menuju kunang-kunang yang memiliki intensitas cahaya paling terang, Pada tahapan pergerakan *firefly* ini masing-masing kunang-kunang dibandingkan dan diurutkan berdasarkan intensitas cahayanya, bagi yang memiliki intensitas cahaya terbesar akan ditandai sebagai *firefly* terbaik atau kunang-kunang solusi terbaik pada iterasi ini dan akan dipertahankan untuk iterasi berikutnya.

Tahapan penyelesaian algoritma *firefly* dalam optimalisasi produksi dapat dilihat pada *flowchart* di Gambar 3.1



Gambar 3.1 Tahapan Proses Penyelesaian dalam Algoritma Firefly

3.6. Penerapan Algoritma Firefly Dalam Masalah Produksi

Solusi untuk menyelesaikan masalah *multi-objective* dalam proses produksi adalah dengan menggunakan kunang-kunang yang bergerak menuju intensitas cahaya yang tinggi atau ke arah kunang-kunang yang lebih terang sehingga semua kunang-kunang terurut dari nilai fungsi tertinggi (terbaik) ke nilai terendah (terburuk). Solusi ini dapat disebut optimal bila solusi yang didapat tersebut merupakan kombinasi jumlah roti yang optimal tanpa melebihi batasan yang sudah ada pada masalah produksi di CV. Desi Bakery.

Algoritma *Firefly* yang diusulkan dapat dibagi menjadi empat tahap utama dalam proses perhitungannya pertama penentuan nilai intensitas cahaya *firefly* berdasarkan nilai *fitness*, menghitung keaktraktifannya, menghitung jarak dan melakukan pergerakan pada kunang-kunang berintensitas cahaya terang sehingga akan menghasilkan posisi baru yang akan dijadikan sebagai kandidat solusi baru.

3.7. Analisis Kasus

Setelah membuat model program linear dan menentukan nilai *fitness* dari masing-masing kandidat solusi yang sudah ditentukan sebelumnya, maka dilanjutkan menganalisisnya menggunakan algoritma *firefly*. Penjelasan dan perhitungan mengenai langkah-langkah penyelesaian kasus di atas menggunakan metode algoritma *firefly* adalah sebagai berikut:

1. Inisialisasi populasi firefly dan parameter algoritma firefly

Istilah populasi dalam algoritma *firefly* dapat diartikan sebagai solusi dimana solusi tersebut yang akan menyelesaikan permasalahan ini.

Pada tahapan ini parameter-parameter algoritma *firefly* ditentukan, seperti jumlah populasi *firefly* (n), jumlah iterasi (t), kostanta *alpha* (α), kostanta *basebeta* (β_0), konstanta *gamma* (γ), dan nilai *rand* untuk menghitung tingkat ketertarikan antar kunang-kunang. Nilai posisi awal dari masing-masing *firefly* juga ditentukan sesuai dengan bilangan acak yang dihasilkan pada tahapan sebelumnya sebanyak jenis roti yang akan diproduksi dan kandidat solusi yang diinginkan. Parameter awal yang akan digunakan diambil berdasarkan pada jurnal penelitian sebelumnya [**5**]. Penelitian yang membahas tentang analisis konvergensi dan pemilihan parameter untuk optimasi menggunakan algoritma *firefly*. Maka parameter awal

yang digunakan yaitu jumlah popolasi 15 *firefly* dengan nilai *basebeta* 1, *alpha* 0,1 dan *gamma* 0,01.

Tabel 3.7 Penentuan parameter algoritma firefly

Simbol	Keterangan	Nilai
n	Jumlah Populasi	15
t	Jumlah Iterasi	100
α	Konstanta Alpha	0,1
β_0	Konstanta Ketertarikan	1
γ	Konstanta Gamma	0,01
Rand	Bilangan Pengacak	0,8

2. Menentukan dimensi firefly yang akan digunakan sebagai posisi awal.

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 banyaknya jenis roti yang akan diproduksi. Setelah banyaknya dimensi ditentukan, pembangkitan bilangan acak yang sudah dilakukan sebelumnya digunakan untuk menentukan nilai posisi awal masing-masing *firefly*, kemudian diisikan ke dalam matriks solusi (*i x j*) dimana dimensi *i* akan mewakilkan kandidat solusi dan dimensi *j* akan mewakilkan roti yang akan diproduksi. Setiap satu *firefly i* akan memiliki nilai posisi sebanyak *j*. Sebagai contoh, jika ada 15 *firefly* dengan jumlah jenis roti sebanyak 3, maka akan ada 45 nilai posisi yang harus diisikan ke dalam matriks solusi. Adapun nilai posisi awal masing-masing *firefly* [i] untuk setiap jenis roti pada dimensi *j* dapat dilihat pada Tabel 3.8

Tabel 3.8 Nilai posisi awal setiap firefly

Firefly [i]	Dimensi (*j*)		
	pRB	pRK	pPB
Firefly [1]	352	163	614
Firefly [2]	370	387	100
Firefly [3]	468	290	165

Firefly [4]	225	243	380
Firefly [5]	322	238	652
Firefly [6]	340	300	559
Firefly [7]	359	280	214
Firefly [8]	352	228	600
Firefly [9]	193	450	187
Firefly [10]	550	255	237
Firefly [11]	164	232	470
Firefly [12]	400	136	252
Firefly [13]	496	260	369
Firefly [14]	496	260	369
Firefly [15]	372	177	455

3. Menghitung nilai intensitas cahaya dari setiap firefly.

Pada penelitian ini, nilai intensitas cahaya masing-masing *firefly* ditentukan oleh fungsi objektif, nilai intensitas cahaya awal *firefly* ditentukan berdasarkan nilai *fitness* yang dihasilkan oleh masing-masing kandidat solusi pada tahapan perhitungan *fitness* sebelumnya. Nilai fitness yang dihasilkan masing-masing kandidat solusi akan dijadikan nilai intensitas cahaya awal masing-masing *firefly*.

Tabel 3.9 Nilai intensitas cahaya awal setiap firefly

Firefly	Intensitas Cahaya (I)	Dimensi (*j*)		
		pRB	pRB	pRB
[1]	852636	352	163	614
[2]	1109572	370	387	100
[3]	1019939	468	290	165
[4]	817576	225	243	380
[5]	993794	322	238	652
[6]	1095001	340	300	559
[7]	932374	359	280	214
[8]	977010	352	228	600
[9]	1308740	193	450	187

[10]	1044279	550	255	237
[11]	782767	164	232	470
[12]	692406	400	136	252
[13]	913945	496	260	369
[14]	1063838	496	260	369
[15]	833098	372	177	455

4. Menentukan Gbest (Global Best).

Firefly Gbest merupakan *firefly* yang memiliki nilai intensitas cahaya paling tinggi dimana dalam kasus ini akan memaksimalkan penjualan roti. *Firefly* dengan intensitas cahaya paling terang merupakan *firefly* yang paling optimal dan akan dijadikan tujuan. Dengan melihat Intensitas cahaya setiap *firefly* diatas, didapatkan nilai intensitas tertinggi berasal dari *firefly* ke [9] pada iterasi pertama. Maka *firefly* ke [9] menjadi sebuah *firefly Gbest* sementara.

Tabel 3.10 Firefly global best sementara

Firefly	Intensitas Cahaya	Status	Firefly	Intensitas Cahaya	Status
[1]	852636	-	[9]	1128785	Gbest
[2]	1109572	-	[10]	1044279	-
[3]	1019939	-	[11]	782767	-
[4]	817576	-	[12]	692406	-
[5]	993794	-	[13]	913945	-
[6]	1095001	-	[14]	1063838	-
[7]	932374	-	[15]	833098	-
[8]	977010	-			

5. Hitung jarak posisi antar firefly.

Proses perhitungan jarak *firefly* terhadap *firefly Gbest* menggunakan metode *Euclidean*. Nilai posisi awal setiap *firefly* bisa dilihat pada Tabel 3.8 maka perhitungan jarak antara *firefly* dirumuskan menggunakan persamaan (2.3) dan ditampilkan pada Tabel 3.11.

Tabel 3.11 Perhitungan jarak setiap firefly

Firefly [i]	Nilai Posisi Awal		
	RB	RK	PB
Gbest [9]	193	450	187
Firefly [1]	$(193-352)^2 = 25281$	$(450-163)^2 = 82369$	$(187-614)^2 = 182329$
Firefly [2]	$(193-370)^2 = 31329$	$(450-387)^2 = 3969$	$(187-100)^2 = 7569$
Firefly [3]	$(193-468)^2 = 75625$	$(450-390)^2 = 25600$	$(187-165)^2 = 484$
Firefly [4]	$(193-225)^2 = 1024$	$(450-243)^2 = 42849$	$(187-380)^2 = 37249$
Firefly [5]	$(193-322)^2 = 16641$	$(450-238)^2 = 44944$	$(187-652)^2 = 216225$
Firefly [6]	$(193-340)^2 = 21609$	$(450-300)^2 = 22500$	$(187-559)^2 = 138384$
Firefly [7]	$(193-359)^2 = 27556$	$(450-280)^2 = 28900$	$(187-214)^2 = 729$
Firefly [8]	$(193-352)^2 = 25281$	$(450-228)^2 = 49284$	$(187-600)^2 = 170569$
Firefly [10]	$(193-550)^2 = 127449$	$(450-255)^2 = 38025$	$(187-237)^2 = 2500$
Firefly [11]	$(193-164)^2 = 841$	$(450-232)^2 = 47524$	$(187-470)^2 = 80089$
Firefly [12]	$(193-400)^2 = 42849$	$(450-136)^2 = 98596$	$(187-252)^2 = 4225$
Firefly [13]	$(193-415)^2 = 49284$	$(450-240)^2 = 44100$	$(187-256)^2 = 4761$
Firefly [14]	$(193-496)^2 = 91809$	$(450-260)^2 = 36100$	$(187-369)^2 = 33124$
Firefly [15]	$(193-372)^2 = 32041$	$(450-177)^2 = 74529$	$(187-455)^2 = 71824$

Jarak antar *firefly* dihitung dengan menjumlahkan selisih kuadrat dari posisi setiap *firefly* dengan *firefly Gbest* sebanyak dimensi yang ada. Pada perhitungan diatas dapat dilihat hasil dari selisih kuadrat posisi setiap *firefly* untuk masing-masing dimensi, kemudian hasil selisih kuadrat tersebut dijumlahkan dan dicari akarnya.

r Firefly [1] $= \sqrt{}(25281 + 82369 + 182329) = 538,4969823$

r Firefly [2] $= \sqrt{}(31329 + 3969 + 7569) = 207,0434737$

r Firefly [3] $= \sqrt{}(75625 + 25600 + 484) = 318,9184849$

r Firefly [4] $= \sqrt{}(1024 + 42849 + 37249) = 284,8192409$

r Firefly [5] $= \sqrt{}(16641 + 44944 + 216225) = 527,0768445$

r Firefly [6] $= \sqrt{}(21609 + 22500 + 138384) = 427,1919943$

r Firefly [7] = √ (27556 + 28900 + 729) = 239,1338537

r Firefly [8] = √ (25281 + 49284 + 170569) = 495,1100888

r Firefly [10] = √ (127449 + 38025 + 2500) = 409,8463127

r Firefly [11] = √ (841 + 47524 + 80089) = 358,4047991

r Firefly [12] = √ (42849 + 98596 + 4225) = 381,6673945

r Firefly [13] = √ (49284 + 44100 + 4761) = 313,281024

r Firefly [14] = √ (91809 + 36100 + 33124) = 401,2891725

r Firefly [15] = √ (32041 + 74529 + 71824) = 422,3671389

6. Hitung Nilai Keaktraktifan (β)

Proses perhitungan nilai keaktraktifan setiap *firefly* dirumuskan menggunakan persamaan (2.2) dengan parameter-parameter yang sudah ditentukan sebelumnya.

$Firefly\ [1]\ \beta_1 = 1e^{-0{,}01*538{,}4969823^2} = 0{,}000344734$

$Firefly\ [2]\ \beta_2 = 1e^{-0{,}01*207{,}0434737^2} = 0{,}002327368$

$Firefly\ [3]\ \beta_3 = 1e^{-0{,}01*318{,}9184849^2} = 0{,}000982231$

$Firefly\ [4]\ \beta_4 = 1e^{-0{,}01*284{,}8192409^2} = 0{,}001231194$

$Firefly\ [5]\ \beta_5 = 1e^{-0{,}01*527{,}0768445^2} = 0{,}000359829$

$Firefly\ [6]\ \beta_6 = 1e^{-0{,}01*427{,}1919943^2} = 0{,}000547666$

$Firefly\ [7]\ \beta_7 = 1e^{-0{,}01*239{,}1338537^2} = 0{,}001745658$

$Firefly\ [8]\ \beta_8 = 1e^{-0{,}01*495{,}1100888^2} = 0{,}000407774$

$Firefly\ [10]\ \beta_{10} = 1e^{-0{,}01*409{,}8463127^2} = 0{,}000594976$

$Firefly\ [11]\ \beta_{11} = 1e^{-0{,}01*358{,}4047991^2} = 0{,}000777883$

$Firefly\ [12]\ \beta_{12} = 1e^{-0{,}01*381{,}6673945^2} = 0{,}000686012$

$Firefly\ [13]\ \beta_{13} = 1e^{-0{,}01*313{,}281024^2} = 0{,}001017864$

$Firefly\ [14]\ \beta_{14} = 1e^{-0{,}01*401{,}2891725^2} = 0{,}000620605$

$Firefly\ [15]\ \beta_{15} = 1e^{-0{,}01*422{,}3671389^2} = 0{,}000560243$

7. Melakukan pergerakan pada firefly

Pergerakan *firefly* terjadi apabila ada satu *firefly* yang intensitas cahayanya lebih besar daripada intensitas cahaya *firefly* yang berada di sekitarnya. Setelah keaktraktifan antar *firefly* diperoleh, maka dilanjutkan dengan menggerakkan

firefly lain mendekati *firefly Gbest* dengan cara mencari posisi baru bagi *firefly* [*i*]. Posisi awal dari *Firefly* [*i*] diubah menjadi posisi baru (xF[*i*]'). Proses perhitungan pergerakan *firefly* menggunakan persamaan (2.4) dengan parameter *alpha* = 0,1 dan bilangan acak (*rand*) = 0,8. Seperti yang sudah ditentukan sebelumnya pada Tabel 3.7 tentang penentuan parameter algoritma *firefly*.

$xF[i]' = pRB[i] + (\beta) * (GbestRB - pRB[i]) + \alpha * (Rand - ½)$

xF[1]RB' = 352 + (0,000344734) * (193-352) + 0,1 * (0,8–0,5)
= 352 + 0,000344734* (-159) + 0,03
= 351,9751

xF[1]RK' = 163 + (0,000344734) * (450-163) + 0,1 * (0,8–0,5)
= 163 + 0,000344734* (287) + 0,03
= 163,1289

xF[1]PB' = 614 + (0,000344734) * (187-614) + 0,1 * (0,8–0,5)
= 614 + 0,000344734* (-427) + 0,03
= 613,8827

Perhitungan tersebut dilanjutkan sebanyak dimensi *j* yang ada pada semua *firefly*. Adapun hasil perhitungan pergerakan setiap *firefly* untuk setiap dimensi *j* ditampilkan pada Tabel 3.12.

Tabel 3.12 Posisi baru setiap firefly

Firefly (i)	Dimensi (*j*)		
	xF[i]RB'	xF[i]RK'	xF[i]PB'
xF[1]'	351,9751	163,1289	613,8827
xF[2]'	369,3667	387,1694	99,8484
xF[3]'	467,7579	289,9936	165,1154
xF[4]'	224,9906	243,2848	379,7923
xF[5]'	321,9475	238,1046	651,8626
xF[6]'	339,9144	300,1094	558,8262
xF[7]'	358,7402	280,3267	213,9828
xF[8]'	351,9243	228,1205	599,8208
xF[10]'	549,8175	255,1460	237,0002

xF[11]’	164,0525	232,1995	469,8098
xF[12]’	399,8879	136,2454	251,9854
xF[13]’	414,8040	240,2437	255,9597
xF[14]’	495,8740	260,1727	368,9170
xF[15]’	371,8997	177,1829	454,8798

Pada Tabel 3.12, nilai xf[i]’ merupakan posisi baru dari setiap *firefly* dan akan dijadikan posisi awal untuk proses selanjutnya. Perhitungan pergerakan *firefly* akan dilanjutkan sampai semua *firefly* pada dimensi *i* selesai dibandingkan intensitas cahayanya sehingga masing-masing *firefly* akan menghasilkan posisi baru kecuali *firefly* dengan intensitas cahaya terbesar yaitu *firefly* [9] yang merupakan sebuah *firefly Gbest* sementara untuk iterasi pertama.

8. Memperbaharui posisi firefly yang sudah bergerak.

Perbaharuan posisi setiap *firefly* untuk dimensi *j* akan diurutkan dari yang terkecil hingga terbesar melalui implementasi aturan *Smallest Position Value* (*SPV*) untuk mengubah setiap dimensi dalam nilai posisi *firefly* menjadi bentuk urutan permutasi sehingga terbentuk kandidat solusi yang baru, seperti yang ditampilkan pada Tabel 3.13.

Tabel 3.13 Hasil memperbaharui posisi firefly

Kandidat Solusi		Dimensi(*j*)		
		RB	RK	PB
Firefly [1]	Nilai Posisi Baru	351,9751	163,1289	613,8827
	Urutan Permutasi	2	1	3
Firefly [2]	Nilai Posisi Baru	369,3667	387,1694	99,8484
	Urutan Permutasi	2	3	1
Firefly [3]	Nilai Posisi Baru	467,7579	289,9936	165,1154
	Urutan Permutasi	3	2	1
Firefly [4]	Nilai Posisi Baru	224,9906	243,2848	379,7923
	Urutan Permutasi	1	2	3

Firefly [5]	Nilai Posisi Baru	321,9475	238,1046	651,8626
	Urutan Permutasi	2	1	3
Firefly [6]	Nilai Posisi Baru	339,9144	300,1094	558,8262
	Urutan Permutasi	2	1	3
Firefly [7]	Nilai Posisi Baru	358,7402	280,3267	213,9828
	Urutan Permutasi	3	2	1
Firefly [8]	Nilai Posisi Baru	351,9243	228,1205	599,8208
	Urutan Permutasi	2	1	3
Firefly [10]	Nilai Posisi Baru	549,8175	255,1460	237,0002
	Urutan Permutasi	3	2	1
Firefly [11]	Nilai Posisi Baru	164,0525	232,1995	469,8098
	Urutan Permutasi	1	2	3
Firefly [12]	Nilai Posisi Baru	399,8879	136,2454	251,9854
	Urutan Permutasi	3	1	2
Firefly [13]	Nilai Posisi Baru	414,8040	240,2437	255,9597
	Urutan Permutasi	3	1	2
Firefly [14]	Nilai Posisi Baru	495,8740	260,1727	368,9170
	Urutan Permutasi	3	1	2
Firefly [15]	Nilai Posisi Baru	371,8997	177,1829	454,8798
	Urutan Permutasi	2	1	3

Dapat dilihat bahwa kandidat solusi awal yang dimiliki oleh masing-masing *firefly* mengalami perubahan. Hal ini disebabkan oleh berubahnya nilai posisi yang dimiliki oleh setiap *firefly*. Sedangkan kandidat solusi yang dimiliki oleh *firefly Gbest* yaitu *firefly* [9] tidak berubah karena nilai posisi *firefly* [9] tidak mengalami pergerakan.

9. Evaluasi nilai fitness firefly pada posisi baru

Setelah kandidat solusi baru masing-masing *firefly* didapat, maka proses selanjutnya adalah mengevaluasi kembali intensitas cahaya yang dihasilkan oleh solusi baru setiap *firefly* tersebut.

Tabel 3.14 Target produksi pada posisi baru

Kandidat Solusi Baru		Jenis Roti	
	Roti Biasa	Roti Kombinasi	Pia Basah
1	163	352	614
2	100	370	387
3	165	290	468
4	225	243	380
5	238	322	652
6	300	340	559
7	214	280	359
8	228	352	600
9	193	450	187
10	237	255	550
11	164	232	470
12	136	252	400
13	240	256	415
14	260	369	496
15	177	372	455

Perhitungan nilai *fitness* setiap *firefly* pada posisi baru adalah sebagai berikut:

Kandidat Solusi 1 : $F(X)_1$

= 0,5 * (1600(163) + 4000(352) + 800(614)) + 0,5 (-163-352-614) = 1079436

Kandidat Solusi 2 : $F(X)_2$

= 0,5 * (1600(100) + 4000(370) + 800(387)) + 0,5 (-100-370-387) = 974372

Kandidat Solusi 3 : $F(X)_3$

= 0,5 * (1600(165) + 4000(290) + 800(468)) + 0,5 (-165-290-468) = 898739

Kandidat Solusi 4 : $F(X)_4$

= 0,5 * (1600(255) + 4000(243) + 800(380)) + 0,5 (-255-243-380) = 817576

Kandidat Solusi 5 : $F(X)_5$

= 0,5 * (1600(238) + 4000(322) + 800(652)) + 0,5 (-238-322-652) = 1094594

Kandidat Solusi 6 : $F(X)_6$

= 0,5 * (1600(300) + 4000(340) + 800(559)) + 0,5 (-300-340-559) = 1143001

Kandidat Solusi 7 : $F(X)_7$

= 0,5 * (1600(214) + 4000(280) + 800(359)) + 0,5 (-214-280-359) = 874374

Kandidat Solusi 8 : $F(X)_8$

= 0,5 * (1600(228) + 4000(352) + 800(600)) + 0,5 (-228-352-600) = 1125810

Kandidat Solusi 9 : $F(X)_9$

= 0,5 * (1600(193) + 4000(450) + 800(187)) + 0,5 (-193-450-187) = 1128785

Kandidat Solusi 10 : $F(X)_{10}$

= 0,5 * (1600(237) + 4000(255) + 800(550)) + 0,5 (-237-255-550) = 919079

Kandidat Solusi 11 : $F(X)_{11}$

= 0,5 * (1600(164) + 4000(232) + 800(470)) + 0,5 (-164-232-470) = 782767

Kandidat Solusi 12 : $F(X)_{12}$

= 0,5 * (1600(136) + 4000(252) + 800(400)) + 0,5 (-136-252-400) = 772406

Kandidat Solusi 13 : $F(X)_{13}$

= 0,5 * (1600(240) + 4000(256) + 800(415)) + 0,5 (-240-256-415) = 869545

Kandidat Solusi 14 : $F(X)_{14}$

= 0,5 * (1600(260) + 4000(369) + 800(496)) + 0,5 (-260-369-496) = 1143838

Kandidat Solusi 15 : $F(X)_{15}$

= 0,5 * (1600(177) + 4000(372) + 800(455)) + 0,5 (-177-372-455) = 1067098

10. Perangkingan firefly berdasarkan intensitas cahaya baru.

Nilai *fitness* yang dihasilkan oleh kandidat solusi yang baru tersebut akan digunakan sebagai nilai intensitas cahaya *firefly* yang baru. Adapun nilai intensitas cahaya baru masing-masing *firefly* dapat dilihat pada Tabel 3.15.

Tabel 3.15 Perubahan nilai intensitas cahaya setelah mengalami pergerakan

Firefly (solusi)	Intensitas Cahaya Awal	Jenis Roti			Intensitas Cahaya Baru	Jenis Roti		
		Roti Biasa	Roti Kombinasi	Pia Basah		Roti Biasa	Roti Kombinasi	Pia Basah
[1]	852636	352	163	614	1079436	163	352	614
[2]	1109572	370	387	100	1109572	370	387	100
[3]	1019939	468	290	165	898739	165	290	468
[4]	817576	225	243	380	817576	225	243	380

[5]	993794	322	238	652	1094594	238	322	652
[6]	1095001	340	300	559	1143001	300	340	559
[7]	932374	359	280	214	874374	214	280	359
[8]	977010	352	228	600	1125810	228	352	600
[9]	1128785	193	450	187	1128785	193	450	187
[10]	1044279	550	255	237	919079	237	255	550
[11]	782767	164	232	470	782767	164	232	470
[12]	692406	400	136	252	772406	136	252	400
[13]	913945	415	240	256	869545	240	256	415
[14]	1063838	496	260	369	1143838	260	369	496
[15]	833098	372	177	455	1067098	177	372	455

Dapat dilihat bahwa nilai intensitas cahaya *firefly* tertentu mengalami perubahan dari sebelumnya, intensitas cahaya tertinggi baru dimiliki oleh *firefly* [14] dengan nilai intensitas sebesar 1143838. Peningkatan nilai intensitas cahaya disebabkan oleh pergerakan yang dilakukan oleh *firefly* lain mendekati *firefly Gbest* sehingga nilai intensitas cahaya baru yang dihasilkan oleh *firefly* lain akan mendekati nilai intensitas cahaya *firefly Gbest*.

11. Mengecek kombinasi kandidat solusi apakah konvergen atau belum.

Kandidat solusi [14] yang merupakan *firefly* dengan nilai intensitas tertinggi baru pada iterasi pertama dicek kedalam fungsi keterbatasan biaya dan waktu.
Dikatakan optimal apabila hasil yang didapat mendekati batasan biaya dan waktu yaitu biaya produksi ≤ Rp 400.000 dan waktu pencetakan roti ≤ 300 menit. apabila nilai posisi baru melebihi batas nilai minimal dan maksimal pada masing-masing dimensi, maka dilakukan kembali perhitungan langkah ke 5 pada iterasi selanjutnya dengan mencari kembali *firefly* dengan nilai intensitas paling tinggi yang akan dijadikan *Global Best* berikutnya dan mengembalikan nilai posisinya agar kembali dalam rentang nilai pada dimensi tersebut.

3.8. Hasil Penyelesaian dan Kesimpulan

Setelah dilakukan pengecekan maka dipilih *firefly* terbaik yaitu *firefly* yang memiliki nilai intensitas cahaya terbesar dari seluruh *firefly* yang ada, *firefly* tersebut akan dianggap sebagai *firefly* yang memiliki solusi terbaik.

Nilai intensitas tertinggi dari perhitungan di atas dimiliki oleh *Firefly* [14] dengan nilai intensitas cahaya yaitu 1143838.

Maka dapat di simpulkan kombinasi optimal untuk perkiraan jumlah target produksi setiap roti berdasarkan hasil perhitungan menggunakan algoritma *firefly* ditampilkan pada Tabel 3.16.

Tabel 3.16 Kombinasi optimal target produksi roti

Roti Biasa (RB)	Roti Kombinasi (RK)	Pia Basah (PB)
260	369	496

1. Perhitungan Perkiraan Total Penjualan

Untuk mendapatkan perkiraan total penjualan diperlukan mensubstitusi kombinasi jumlah target produksi setiap roti kedalam fungsi objektif pertama untuk memaksimalkan penjualan, yaitu sebagai berikut :

TPJ = 1600(RB) + 4000(RK) + 800(PB)

TPJ = 1600(260) + 4000(369) + 800(496)

TPJ = 416000 + 1476000+ 396800 = Rp 2.288.800,-

Keterangan :

TPJ = Total Penjualan

2. Perhitungan Perkiraan Biaya Produksi

Untuk mendapatkan perkiraan optimasi biaya diperlukan asumsi penggunaan biaya produksi untuk satu buah roti dan untuk biaya produksi tidak boleh melebihi modal produksi yang sudah ditentukan sebelumnya yaitu sebesar Rp 400.000,- seperti yang telah disajikan pada Tabel 3.3

Maka untuk proses perhitungannya yaitu dengan mensubtitusikan kombinasi optimal jumlah target produksi kedalam fungsi kendala keterbatasan biaya, sehingga didapatkan total biaya produksi untuk satu hari produksi adalah sebagai berikut :

TBP = 300(RB) + 600(RK) + 200(PB) $\leq$ 400.000

TBP = 300(260) + 600(369) + 200(496)

TBP = 78000 + 221400 + 99200 = Rp 398.600,-

Keterangan :

TBP = Total Biaya Produksi

3. Perhitungan Perkiraan Waktu pencetakan roti

Untuk mengetahui waktu pencetakan setiap jenis roti di CV. Desi Bakery dapat dilihat pada Tabel 3.3 proses pencetakan roti diperlukan waktu kerja selama 300 menit. Untuk mengetahui berapa buah roti yang bisa dihasilkan selama proses pencetakan diperlukan asumsi kecepatan membuat satu buah roti, Untuk proses perhitungannya dengan mensubtitusikan kombinasi optimal jumlah target produksi kedalam fungsi kendala keterbatasan waktu produksi adalah sebagai berikut:

WPP = 0,13(RB) + 0,5(RK) + 0,12(PB) $\leq$ 300 menit

WPP = 0,13(260) + 0,5(369) + 0,12(496)

WPP = 33,8 + 184,5 + 59,52 = 277,82 menit

Keterangan :

WPP = Waktu Pengerjaan Produksi

Pada kasus ini, solusi yang ingin dicapai adalah perkiraan jumlah target produksi setiap jenis roti yang harus dipenuhi untuk mendapat keuntungan maksimal berdasarkan keterbatasan biaya produksi dan waktu pencetakan roti sehingga produksi roti menjadi lebih optimal. Maka, setelah menerapkan algoritma *firefly* dalam masalah produksi roti ini telah berhasil memperoleh solusi optimal untuk kombinasi jumlah roti yang harus diproduksi CV. Desi Bakery yaitu kombinasi roti biasa 260 buah, roti kombinasi 369 buah dan pia basah 496 buah, berhasil memaksimalkan penjulannya sebesar Rp 2.288.800,- Dengan memimimumkan biaya produksinya menjadi Rp 398.600,- Dan waktu pencetakan roti selama 278 menit.

3.9. Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk *simulator*. Spesifikasi ini meliputi elemen atau komponen-komponen apa saja yang akan dibutuhkan *simulator* yang akan dibangun, sampai simulator tersebut diimplementasikan. Adapun kebutuhan non fungsionalnya yaitu meliputi analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

1. Analisis Kebutuhan Perangkat Lunak (*software*)

   Perangkat lunak (*software*) yang dibutuhkan dalam penerapan algoritma *firefly* ini menggunakan sistem operasi *Windows* 7.

   Berikut ini merupakan spesifikasi perangkat lunak yang digunakan dalam membangun aplikasi ini, yaitu: Sistem Operasi *Microsoft Window* 7 dan *Microsoft Visual Studio* 2010.

2. Analisis Kebutuhan Perangkat Keras (*hardware*)

   Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Spesifikasi minimun perangkat keras yang dibutuhkan dapat dilihat pada Tabel 3.17.

Tabel 3.17 Spesifik Kebutuhan Perangkat Keras

Perangkat Keras	Spesifikasi
Prosessor	2,0 GHz
Harddisk	120 GB
RAM	2 GB
Video Card	1 GB
Monitor	1366 x 768 *Pixel*
Mouse	*Standard*
Keyboard	*Standard*

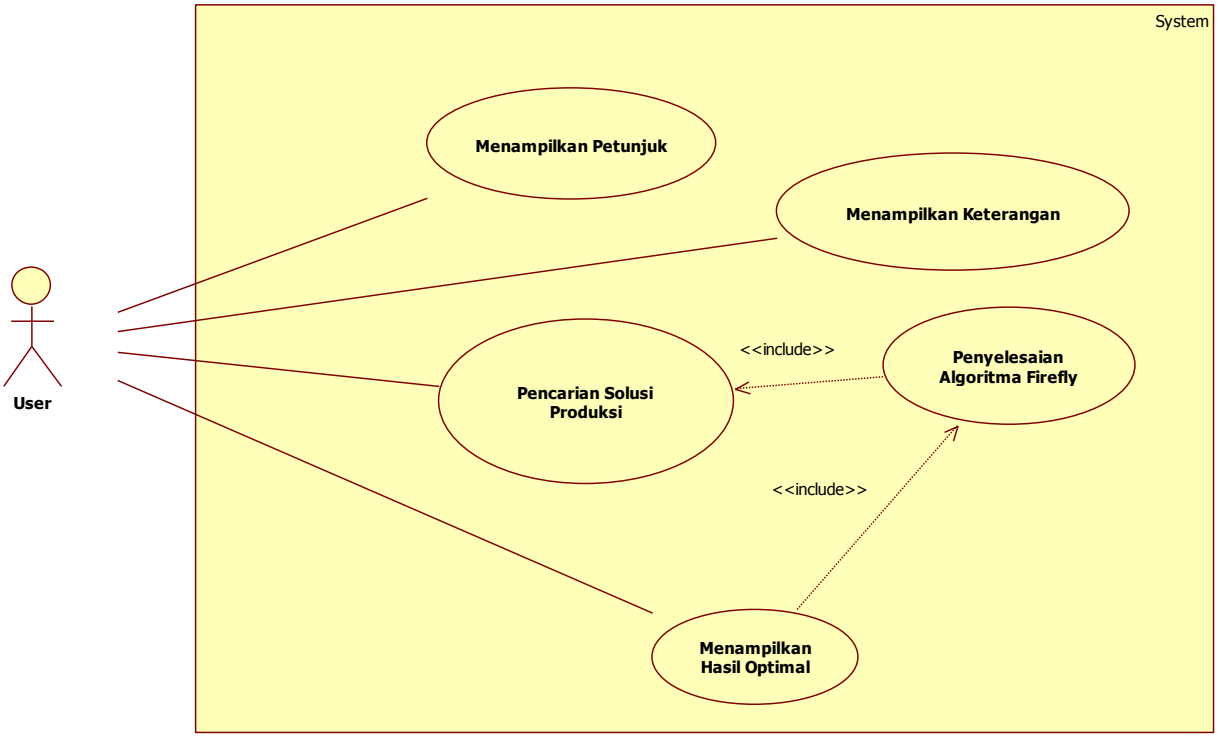
3.10. Analisis Kebutuhan Fungsional

Pemodelan sistem yang akan dibangun dilakukan dengan metode analisis perancangan dan pengembangan perangkat lunak berorientasi objek dan menggunakan *tools Unified Modeling Language* (*UML*) sebagai alat bantu dalam

menggambarkan rancangan sistem. Pemodelan ini mencangkup *use case* diagram, *activity* diagram, class diagram dan *sequence* diagram.

3.10.1. Use Case Diagram Implementasi FA pada proses produksi

Use case diagram untuk memodelkan pengimplementasian algoritma *firefly* dalam mengoptimalkan produksi dapat dilihat pada Gambar 3.2.



Gambar 3.2 Use Case Diagram Simulasi Produksi

1. Definisi Aktor

Pada simulasi produksi ini, yang dimaksud dengan aktor adalah pihak yang mengakses *use case*. Sehingga aktor yang terlibat dalam simulasi proses produksi ini adalah *user*.

2. Definisi Use Case Diagram

Use case diagram adalah suatu konstruksi untuk mendeskripsikan sistem. Deskripsi dari setiap *use case* yang ada pada simulasi proses produksi ini dapat dilihat pada Tabel 3.18.

Tabel 3.18 Deskripsi Use Case

No.	Nama Use case	Deskripsi
UC-01	Pencarian Solusi Produksi	Fungsional untuk melakukan pembangkitan kandidat solusi roti, menghitung fungsi objektif serta total waktu, total biaya dan total penjualan dalam simulasi produksi.
UC-02	Penyelesaian Algoritma Firefly	Fungsional untuk pencarian solusi atau menyelesaikan permasalahan optimasi pada simulasi produksi menggunakan algoritma *firefly.*
UC-03	Menampilkan Hasil Optimal	Fungsional untuk melihat kombinasi jumlah produksi yang optimal berdasarkan keterbatasan biaya dan waktu produksi.
UC-04	Menampilkan Petunjuk	Fungsional untuk melihat petunjuk penggunaan simulasi produksi.
UC-05	Menampilkan Keterangan	Fungsional untuk melihat keterangan data masukkan tentang keterbatasan atau kendala dalam simulator

3. Skenario Use Case Diagram Simulasi Produksi.

Skenario *use case* adalah penjelasan *use case* diagram yang lebih detail dari awal sampai akhirnya diperoleh suatu *output*.

a. Skenario dari *use case* pencarian solusi produksi dapat dilihat pada Tabel 3.19.

Tabel 3.19 Skenario Use Case Pencarian Solusi Produksi

Identifikasi	
Nomor	UC-01
Nama Use Case	**Pencarian Solusi Produksi**
Tujuan Dalam Konteks	Mendapatkan kandidat solusi dalam menentukan kombinasi target jumlah produksi roti dan memperkirakan total waktu, biaya dan total penjualan

	di CV. Desi Bakery.	
Deskripsi	Fungsional untuk melakukan pembangkitan kandidat solusi roti, menghitung fungsi objektif, intensitas cahaya serta total waktu, total biaya dan total penjualan dalam simulasi produksi.	
Aktor	User	
Skenario Utama		
Kondisi Awal	User berada di tampilan awal simulasi produksi	
Aksi Aktor		**Reaksi Sistem**
1. Mengatur parameter: populasi firefly, jumlah iterasi, alpha, basebeta, gamma		
2. Mengisi fungsi tujuan: harga jual, mengisi kendala: min/maks setiap roti, biaya produksi dan waktu pencetakan		
		3. Melakukan pembangkitan kandidat solusi roti biasa, roti kombinasi dan pia basah.
		4. Melakukan perhitungan nilai fungsi objektif.
		5. Menghitung waktu pencetakan, biaya produksi dan total penjualan.
Kondisi Akhir	Simulator menampilkan hasil optimasi berupa kombinasi kandidat solusi roti RB, RK dan PB, nilai fungsi objektif, total waktu pencetakan, total waktu produksi dan total penjualan	

b. Skenario dari *use case* penyelesaian algoritma *firefly* dapat dilihat pada Tabel 3.20.

Tabel 3.20 Skenario Use Case Penyelesaian Algoritma Firefly

<table>
<tr><th colspan="3">Identifikasi</th></tr>
<tr><td>Nomor</td><td colspan="2">UC-02</td></tr>
<tr><td>Nama Use Case</td><td colspan="2">Penyelesaian Algoritma Firefly</td></tr>
<tr><td>Tujuan Dalam Konteks</td><td colspan="2">Mencari solusi menggunakan algoritma firefly</td></tr>
<tr><td>Deskripsi</td><td colspan="2">Fungsional untuk pencarian solusi atau menyelesaikan permasalahan optimasi pada simulasi produksi menggunakan algoritma firefly</td></tr>
<tr><td>Aktor</td><td colspan="2">User</td></tr>
<tr><th colspan="3">Skenario Utama</th></tr>
<tr><td>Kondisi Awal</td><td colspan="2">User berada di tampilan awal simulasi produksi</td></tr>
<tr><th colspan="2">Aksi Aktor</th><th>Reaksi Sistem</th></tr>
<tr><td colspan="2">1. Mengatur parameter:
populasi firefly, jumlah iterasi, alpha, basebeta, gamma</td><td></td></tr>
<tr><td colspan="2">2. Mengisi fungsi tujuan: harga jual, mengisi kendala: min/maks setiap roti, biaya produksi dan waktu pencetakan</td><td></td></tr>
<tr><td colspan="2"></td><td>3. Evalusai nilai fitness
4. Mengitung intensitas cahaya
5. Membangkitkan dimensi firefly
6. Menghitung jarak
7. Menghitung keaktraktifan
8. Menghitung pergerakan
9. Melakukan perankingan firefly</td></tr>
<tr><td>Kondisi Akhir</td><td colspan="2">Simulator menampilkan kombinasi kandidat solusi roti, fungsi objektif, nilai intensitas cahaya serta total waktu pencetakan, biaya dan total penjualan</td></tr>
</table>

c. Skenario dari *use case* menampilkan hasil optimal dapat dilihat pada Tabel 3.21.

Tabel 3.21 Skenario Use Case Menampilkan Hasil Optimal

Identifikasi	
Nomor	UC-03
Nama Use Case	**Menampilkan Hasil Optimal**
Tujuan Dalam Konteks	Menseleksi kandidat solusi yang optimal
Deskripsi	Fungsional untuk melihat kombinasi jumlah produksi yang optimal berdasarkan keterbatasan biaya dan waktu produksi.
Aktor	User
Skenario Utama	
Kondisi Awal	User berada di tampilan awal simulasi produksi
Aksi Aktor	**Reaksi Sistem**
	1. Menseleksi kandidat solusi yang sudah dioptimasi
	2. Menampilkan hasil optimasi dari kandidat solusi roti yang paling optimal.
Kondisi Akhir	Simulator menampilkan kombinasi kandidat solusi roti, fungsi objektif, nilai intensitas cahaya serta total waktu pencetakan, biaya dan total penjualan

d. Skenario dari use case menampilkan petunjuk dapat dilihat pada Tabel 3.22.

Tabel 3.22 Skenario Use Case Menampilkan Petunjuk

Identifikasi	
Nomor	UC-04
Nama Use Case	**Menampilkan Petunjuk**
Tujuan Dalam Konteks	Memberikan informasi tentang petunjuk penggunaan simulator
Deskripsi	Fungsional untuk melihat petunjuk penggunaan simulasi produksi.
Aktor	User
Skenario Utama	
Kondisi Awal	User berada di tampilan awal simulasi produksi
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan petunjuk penggunaan simulator
Kondisi Akhir	Simulator menampilkan informasi tentang petunjuk penggunaan simulator.

e. Skenario dari *use case* keterangan variabel analisis dapat dilihat pada Tabel 3.23.

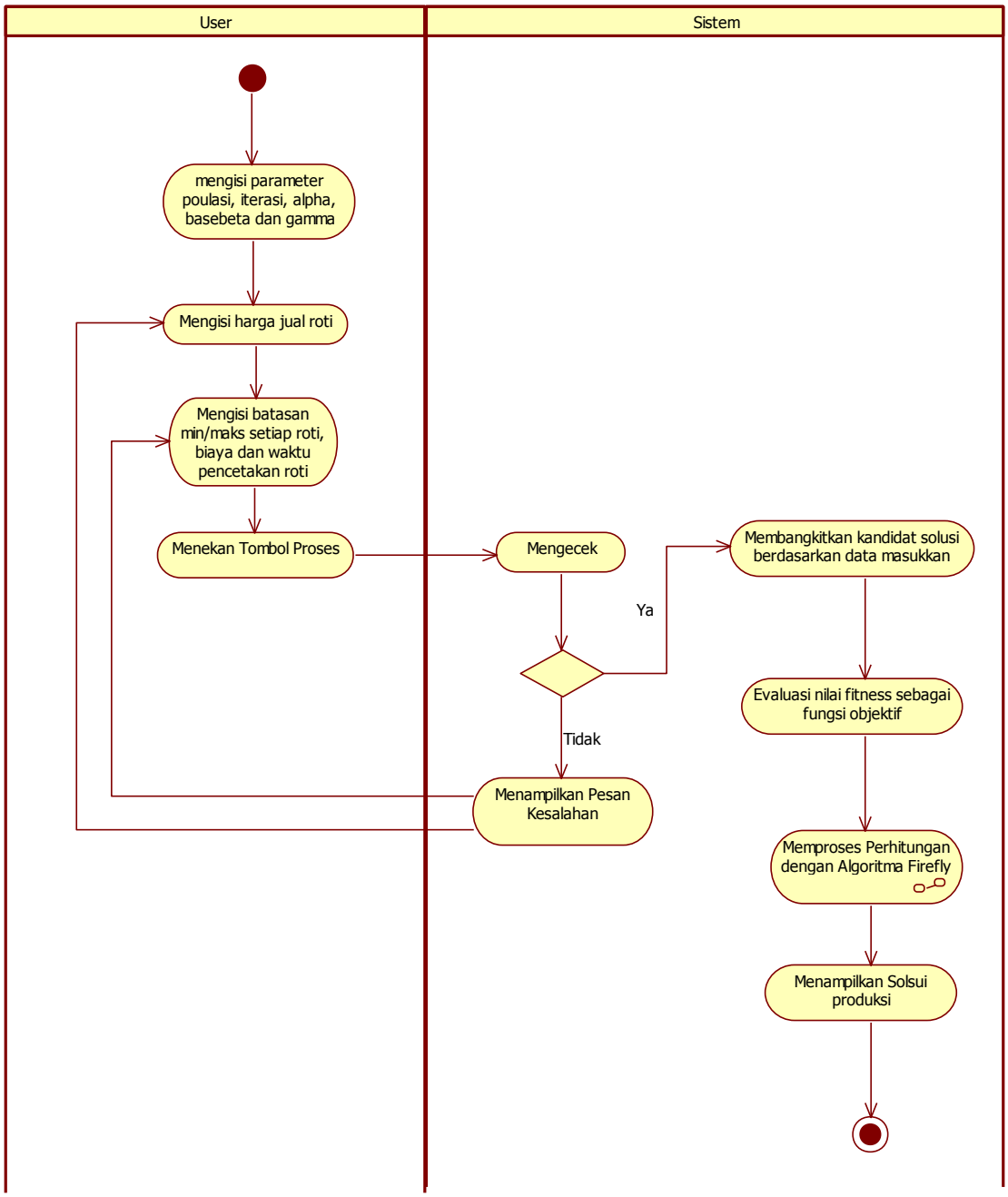
Tabel 3.23 Skenario Use Case Menampilkan Keterangan

Identifikasi	
Nomor	UC-05
Nama Use Case	**Menampilkan Keterangan**
Tujuan Dalam Konteks	Memberikan informasi tentang keterangan data masukkan pada simulator
Deskripsi	Fungsional untuk melihat keterangan data masukkan tentang keterbatasan atau kendala dalam simulator
Aktor	User
Skenario Utama	
Kondisi Awal	User berada di tampilan awal simulasi produksi
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan keterangan data masukkan yang digunakan pada simulator
Kondisi Akhir	Simulator menampilkan informasi tentang keterangan data masukkan pada simulator.

4. Activity Diagram Simulasi Produksi

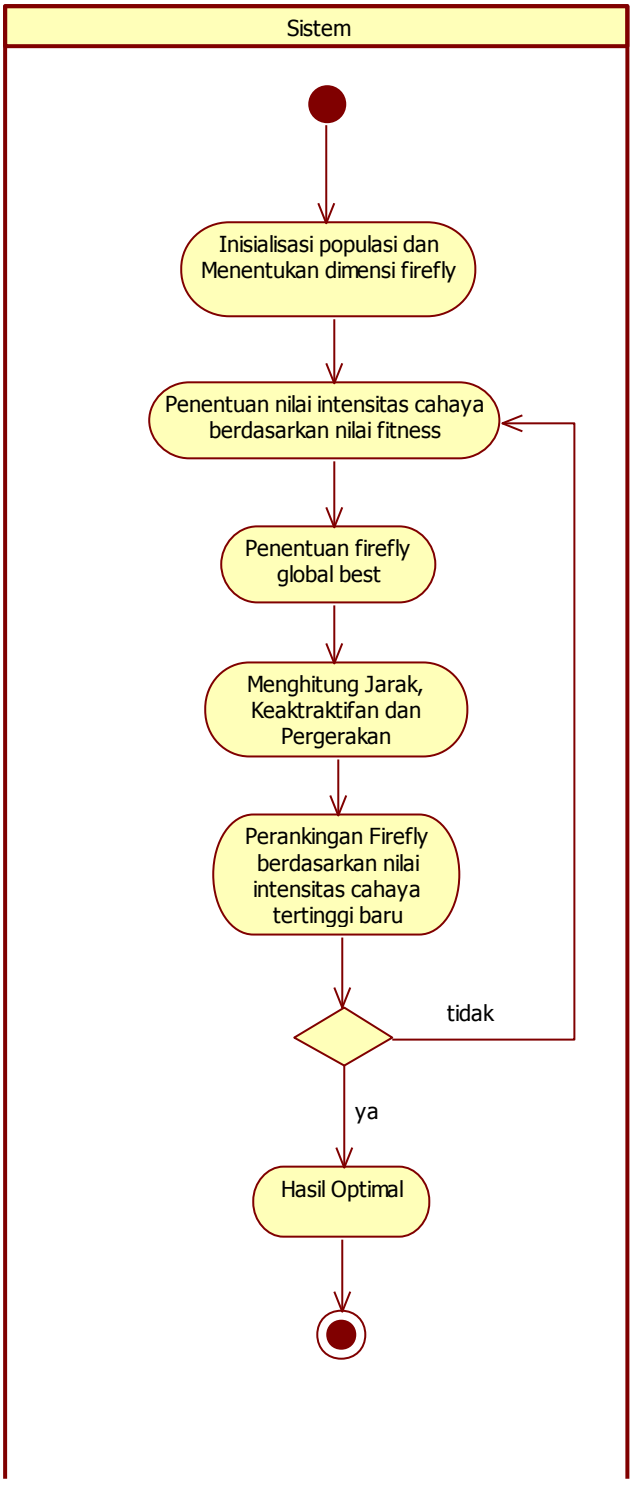
Activity diagram menggambarkan berbagai aliran sistem yang sedang dirancang, bagaimana masing-masing aliran berawal dan percabangan (*decision*) yang mungkin terjadi dari setiap *use case*.

a. *Activity* diagram untuk *use case* pencarian solusi produksi dapat dilihat pada Gambar 3.3



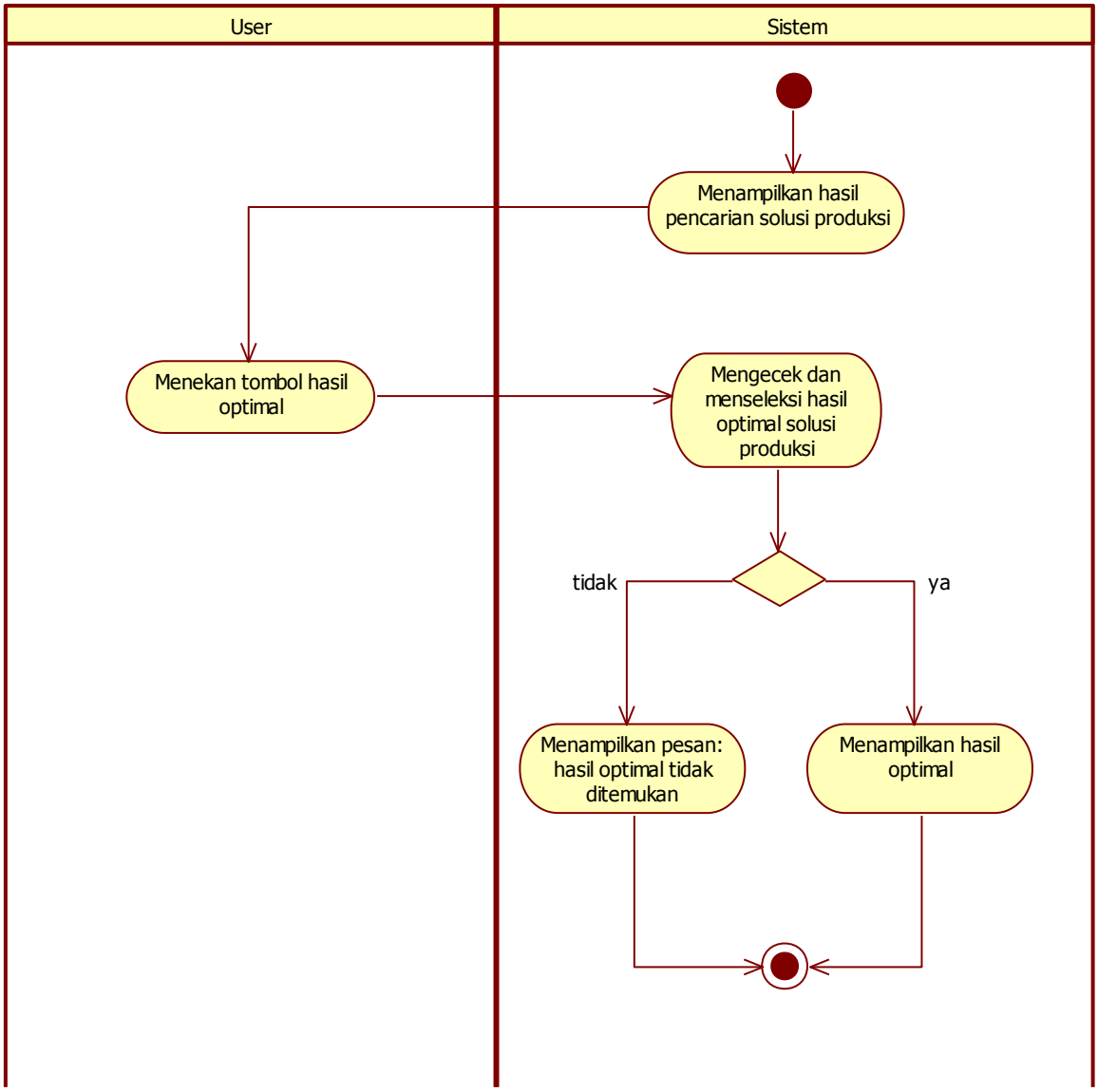
Gambar 3.3 Activity Diagram Pencarian Solusi Produksi

b. *Activity* diagram untuk *use case* penyelesaian algoritma *firefly* dapat dilihat pada Gambar 3.4.



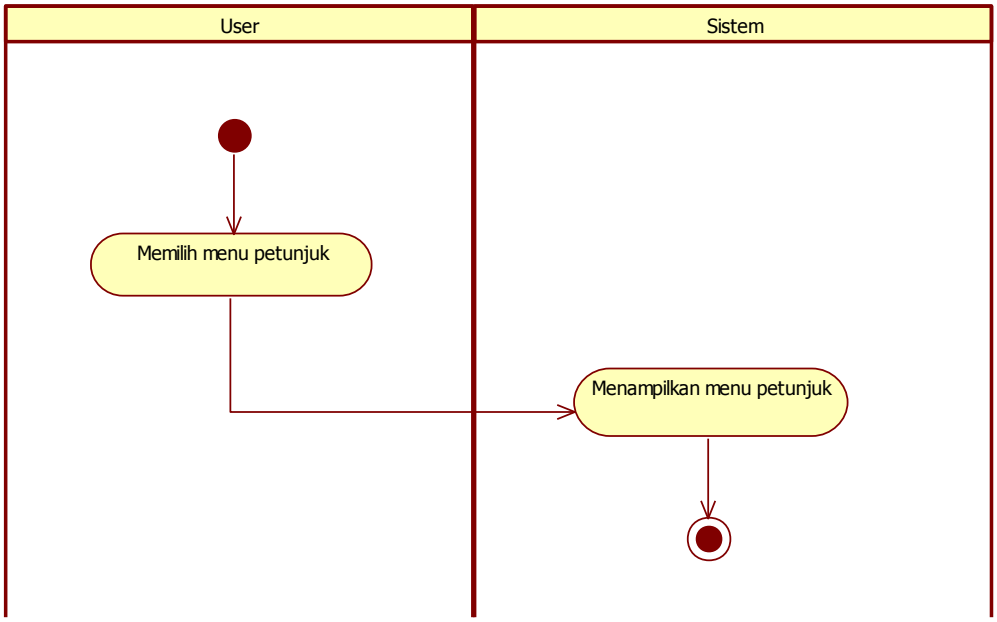
Gambar 3.4 Activity Diagram Penyelesaian Algoritma Firefly

c. *Activity* diagram untuk *use case* menampilkan hasil optimal dapat dilihat pada Gambar 3.5.



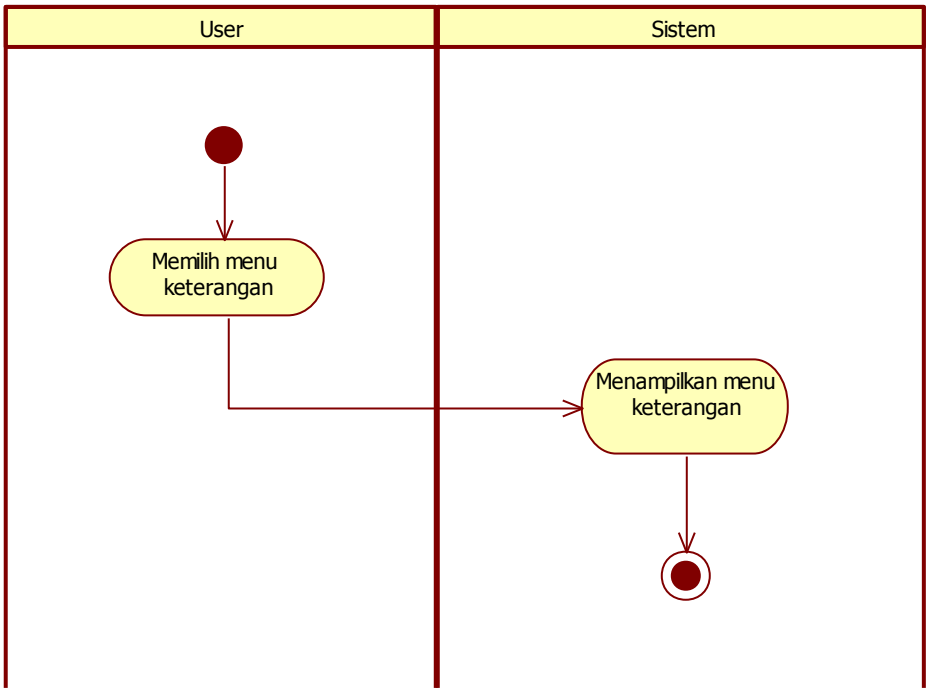
Gambar 3.5 Activity Diagram Menampilkan Hasil Optimal

d. *Activity* diagram untuk *use case* menampilkan petunjuk dapat dilihat pada Gambar 3.6.



Gambar 3.6 Activity Diagram Menampilkan Petunjuk

e. *Activity* diagram untuk *use case* menampilkan keterangan dapat dilihat pada Gambar 3.7.

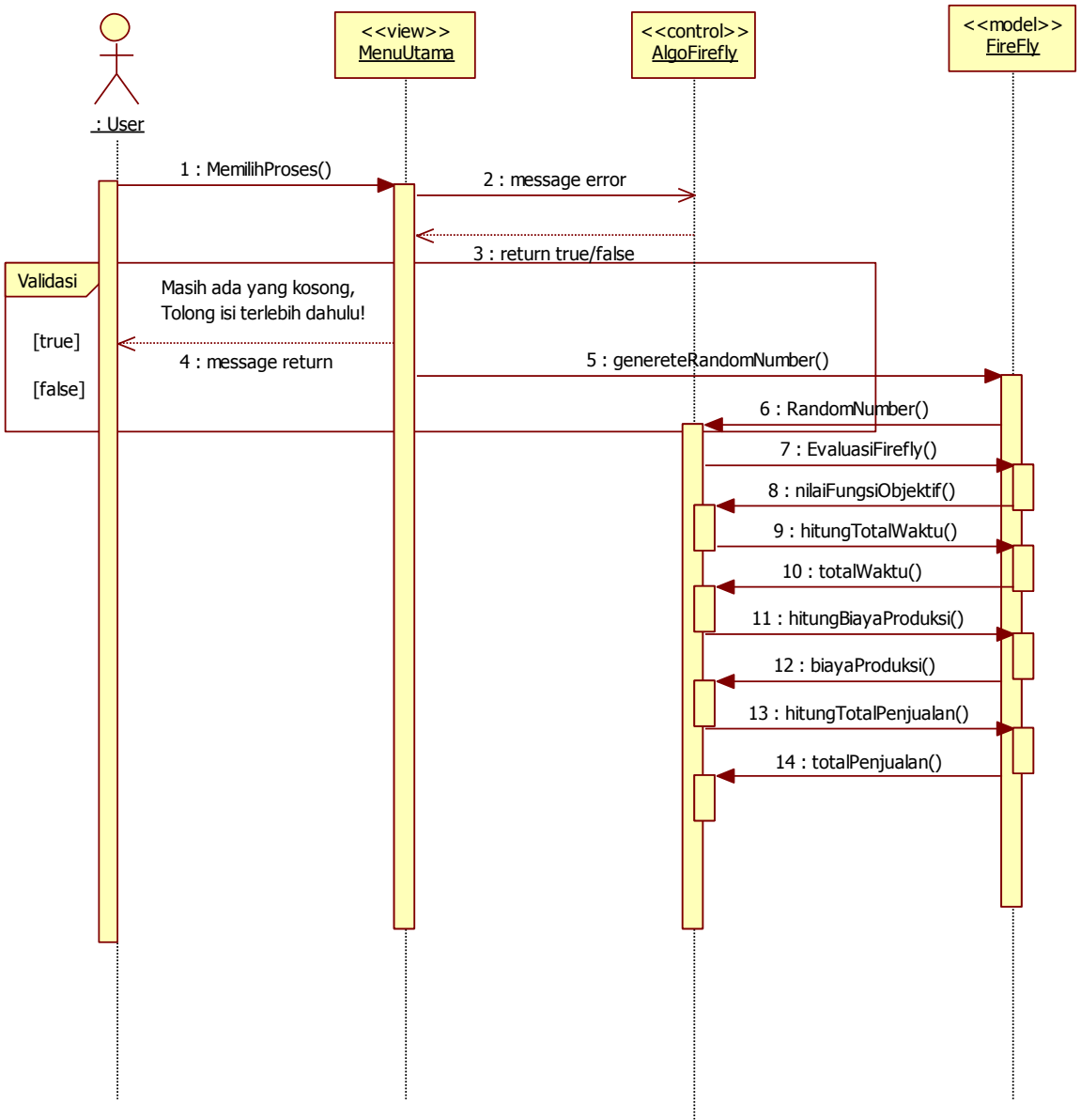


Gambar 3.7 Activity Diagram Menampilkan Keterangan

5. Sequence Diagram

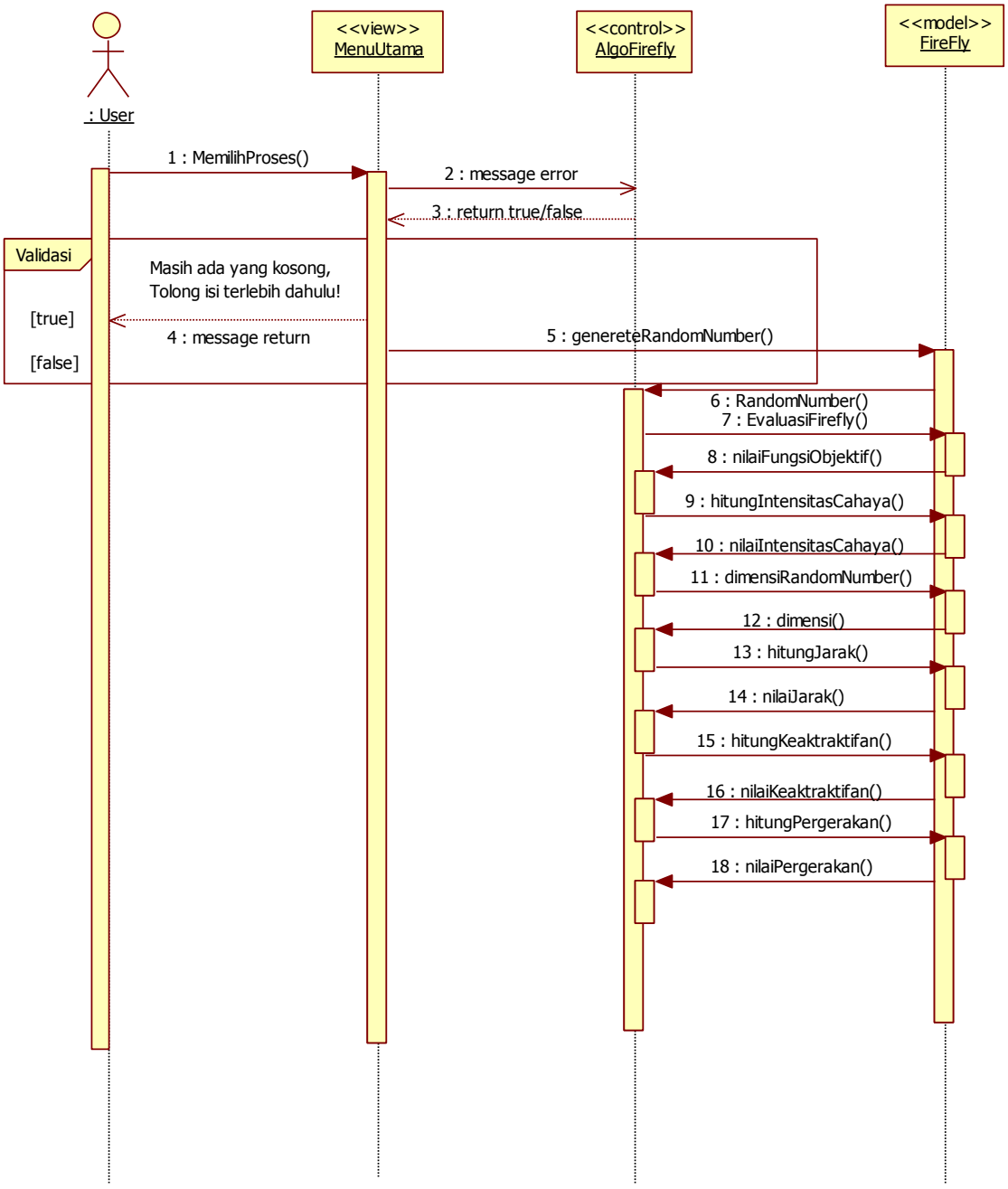
Sequence diagram digunakan untuk menggambarkan prilaku pada sebuah skenario. Diagram ini menunjukan sejumlah contoh obyek dan *message* pesan yang diletakan diantara obyek-obyek ini dalam *use case*.

a. *Sequence diagram* pencarian solusi produksi diperlihatkan pada Gambar 3.8



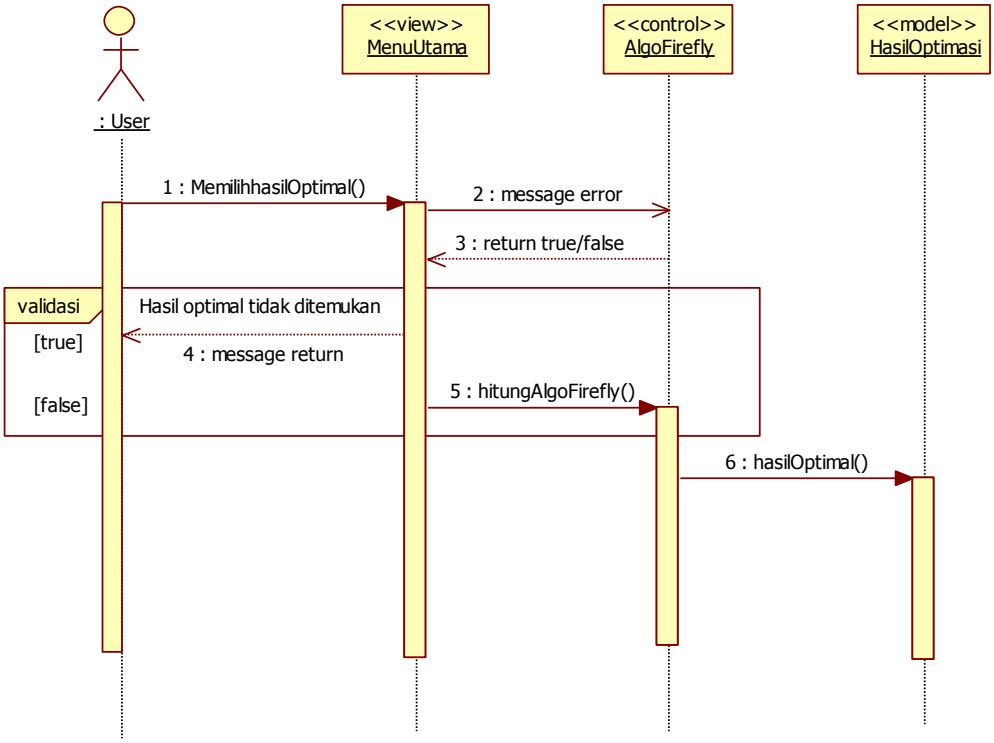
Gambar 3.8 Sequence Diagram Pencarian Solusi Produksi

b. *Sequence diagram* penyelesaian algoritma *firefly* diperlihatkan pada Gambar 3.9.



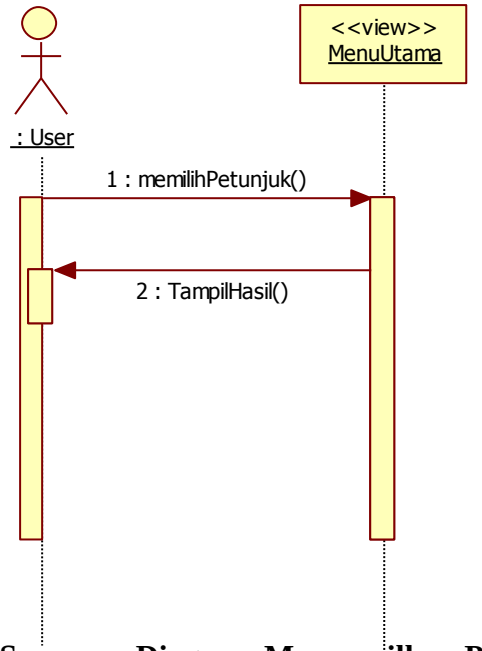
Gambar 3.9 Sequence Diagram Penyelesaian Algoritma Firefly

c. *Sequence diagram* menampilkan hasil optimasi diperlihatkan pada Gambar 3.10.



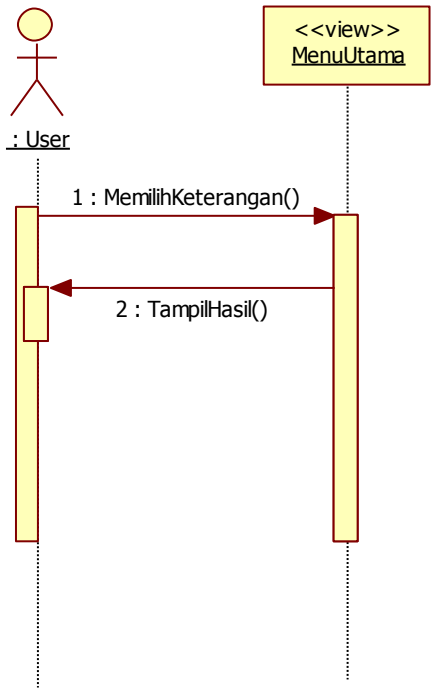
Gambar 3.10 Sequence Diagram Menampilkan Hasil Optimasi

d. *Sequence diagram* menampilkan petunjuk diperlihatkan pada Gambar 3.11.



Gambar 3.11 Sequence Diagram Menampilkan Petunjuk

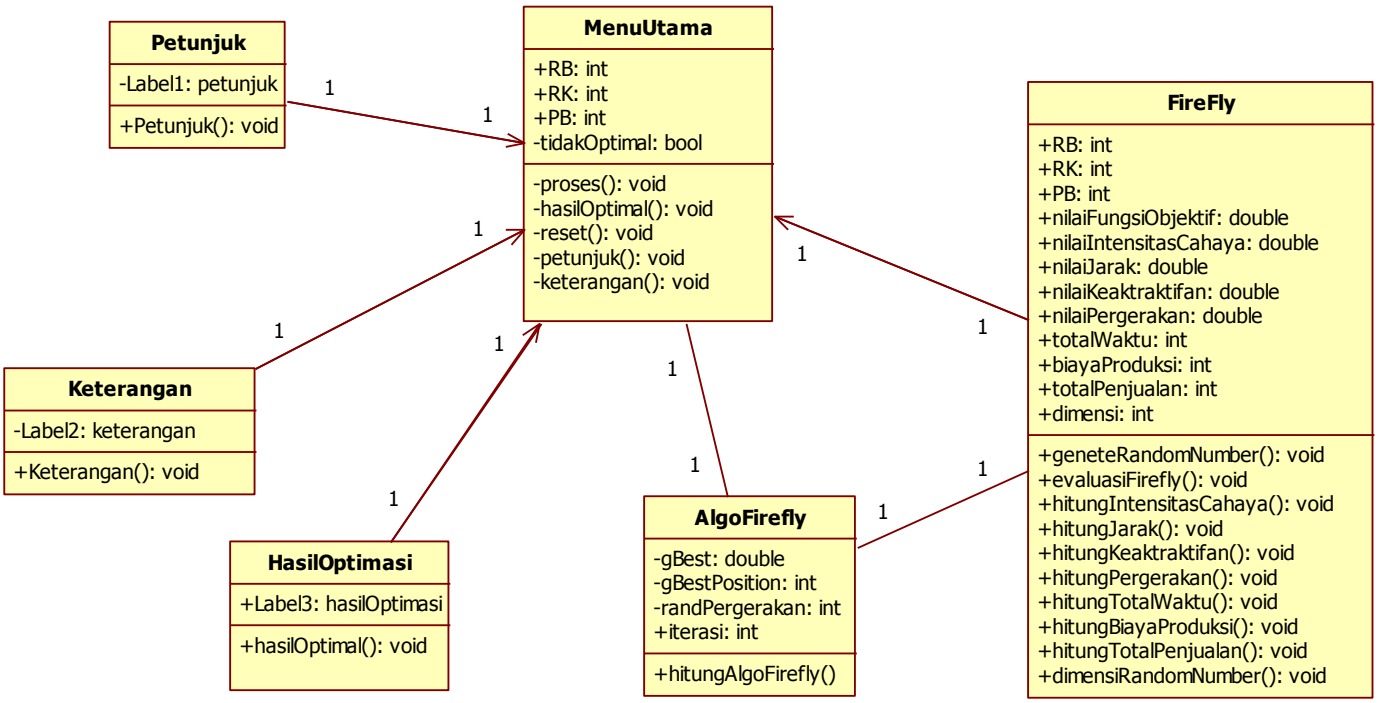
e. *Sequence diagram* menampilkan keterangan diperlihatkan pada Gambar 3.12.



Gambar 3.12 Sequence Diagram Menampilkan Keterangan

6. Class Diagram Diagram Implementasi Algoritma Firefly pada proses produksi

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi obyek. *Class* menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut. *Class* Diagram dari implementasi algoritma *firefly* pada proses produksi diperlihatkan pada Gambar 3.13.



Gambar 3.13 Class Diagram Simulasi Produksi

3.11. Perancangan Arsitektur

Perancangan arsitektur merupakan perancangan yang dibuat sebelum program aplikasi dibuat. Dengan perancangan arsitektur akan mempermudah dalam proses pembangunan perangkat lunak. Perancangan perangkat lunak ini mengikuti konsep yang telah dilakukan pada tahap analisis sistem. Perancangan arsitektur terdiri dari perancangan class, perancangan struktur menu, perancangan antar muka, perancangan pesan dan jaringan semantik.

3.11.1. Perancangan Class

Perancangan *class* merupakan kumpulan class yang ada pada sistem yang dirancang dan berfungsi untuk membangun sistem. Dan berikut ini adalah penjelasan nya.

Tabel 3.24 Deskripsi Class

Class	Jenis Class	Deskripsi
MenuUtama	Boundary	Class yang berisikan method untuk menampilkan hasil perhitungan algoritma
AlgoFirefly	Control	Class yang berisikan method pencarian solusi produksi
FireFly	Entity	Class yang berisikan method algoritma firefly
HasilOptimasi	Boundary	Class yang berisikan method untuk menampilkan solusi optimal
Petunjuk	Boundary	Class yang berisikan method untuk menampilkan petunjuk penggunaan simulasi
Keterangan	Boundary	Class yang berisikan method untuk menampilkan keterangan

Dalam *class* diagram terdapat beberapa atribut didalamnya. Adapun penjelasannya adalah sebagai berikut.

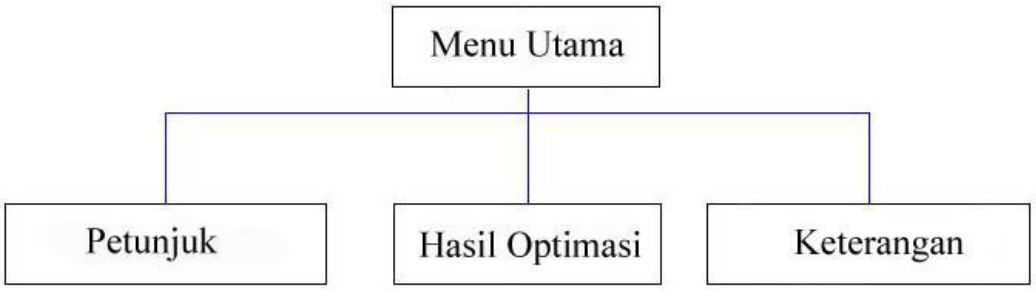
Tabel 3.25 Perancangan Class

1	**Class MenuUtama**	**Tipe Data**	**Visibility**
	Nama Atribut		
	RB	*integer*	*public*
	RK	*integer*	*public*
	PB	*integer*	*public*
	tidakOptimal	*boolean*	*private*
2	**Class AlgoFirefly**	**Tipe Data**	**Visibility**
	Nama Atribut		
	gBest	*double*	*private*
	gBestPostition	*integer*	*private*

	randPergerakan	*double*	*private*
	iterasi	*integer*	*public*
3	**Class FireFly**	**Tipe Data**	**Visibility**
	Nama Atribut		
	RB	*integer*	*public*
	RK	*integer*	*public*
	PB	*integer*	*public*
	nilaiFungsiObjektif	*double*	*public*
	nilaiIntensitasCahaya	*double*	*public*
	nilaiJarak	*double*	*public*
	nilaiKeaktraktifan	*double*	*public*
	nilaiPergerakan	*double*	*public*
	totalWaktu	*integer*	*public*
	biayaProduksi	*integer*	*public*
	totalPenjualan	*integer*	*public*
	dimensi	*integer*	*public*
4	**Class Petunjuk**	**Tipe Data**	**Visibility**
	Nama Atribut		
	Label1	petunjuk	*private*
5	**Class Keterangan**	**Tipe Data**	**Visibility**
	Nama Atribut		
	Label2	keterangan	*private*
6	**Class Petunjuk**	**Tipe Data**	**Visibility**
	Nama Atribut		
	Label3	hasilOptimasi	*public*

3.11.2. Perancangan Struktur Menu

Perancangan struktur menu adalah gambaran jalur dari setiap menu-menu yang disusun berstruktur pada simulasi, dapat berfungsi sebagai navigasi halaman-halaman yang ada dalam simulasi yang dibangun. Perancangan struktur pada simulasi ini ditunjukan pada Gambar 3.14.



Gambar 3.14 Struktur Menu Simulasi

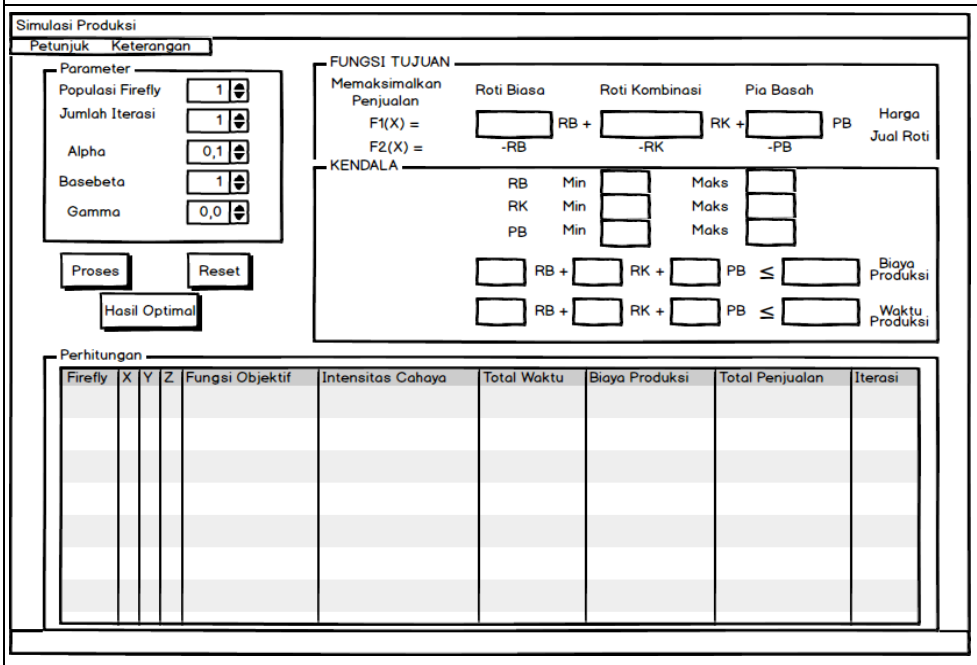
3.11.2. Perancangan Antarmuka

Perancangan antarmuka adalah rancangan tampilan dari simulasi yang akan dibuat. Rancangan antarmuka dari implemntasi algoritma *firefly* pada proses produksi adalah sebagai berikut:

1. Perancangan Antarmuka Menu Utama

Perancangan antarmuka menu utama simulasi produksi dari implementasi algoritma *firefly* dalam proses produksi dapat dilihat pada Gambar 3.15.

A01



- Klik Petunjuk Menuju A02
- Klik Keterangan menuju A03
- Klik Hasil Optimasi menuju A04

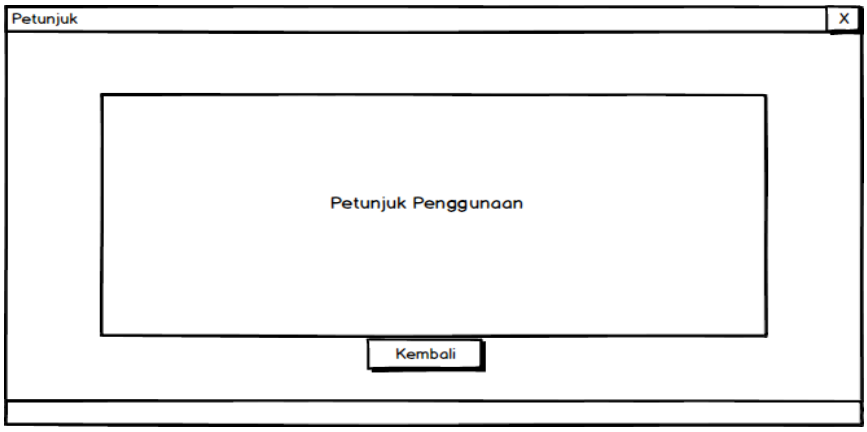
Font Color	: Hitam
Font Size	: Menyesuaikan
Background	: Abu-abu

Gambar 3.15 Antarmuka Menu Utama

2. Perancangan Antarmuka Petunjuk

Perancangan antarmuka petunjuk simulasi produksi dari implementasi algoritma *firefly* pada proses produksi dapat dilihat pada Gambar 3.16.

A02



- Klik x Menuju A01

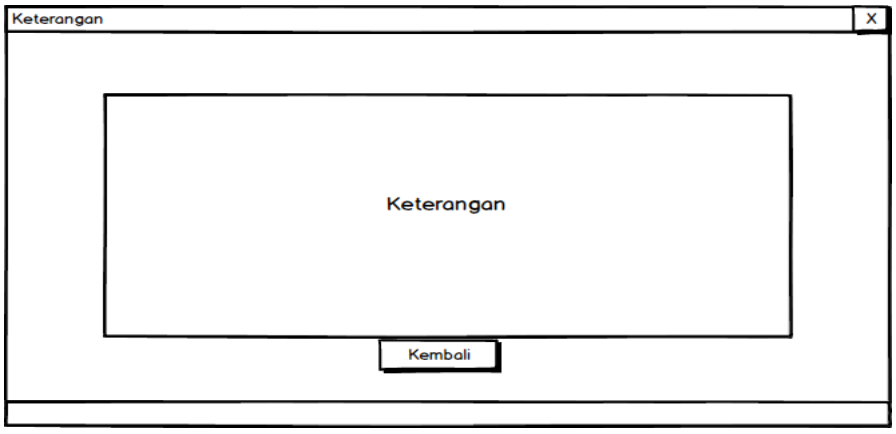
Font Color	: Hitam
Font Size	: Menyesuaikan
Background	: Abu-abu

Gambar 3.16 Antarmuka Petunjuk

3. Perancangan Antarmuka Keterangan

Perancangan antarmuka keterangan simulasi produksi dari implementasi algoritma *firefly* pada proses produksi dapat dilihat pada Gambar 3.17.

A03



- Klik x Menuju A01

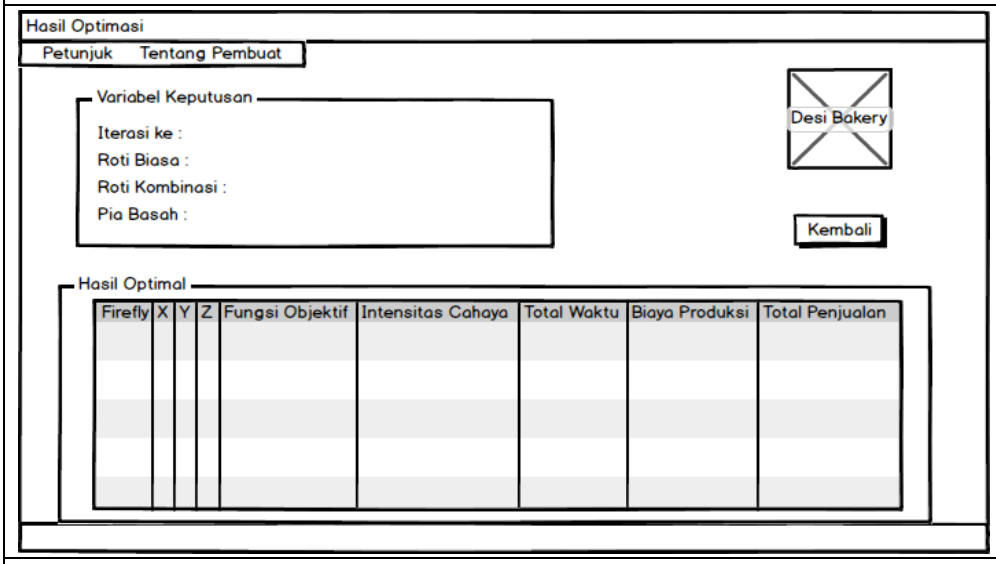
Font Color	: Hitam
Font Size	: Menyesuaikan
Background	: Abu-abu

Gambar 3.17 Antarmuka Keterangan

4. Perancangan Antarmuka Hasil Optimasi

Perancangan antarmuka Hasil Optimasi simulasi produksi dari implementasi algoritma *firefly* pada proses produksi dapat dilihat pada Gambar 3.18.

A04



- Klik Kembali Menuju A01

Font Color	: Hitam
Font Size	: Menyesuaikan
Background	: Abu-abu

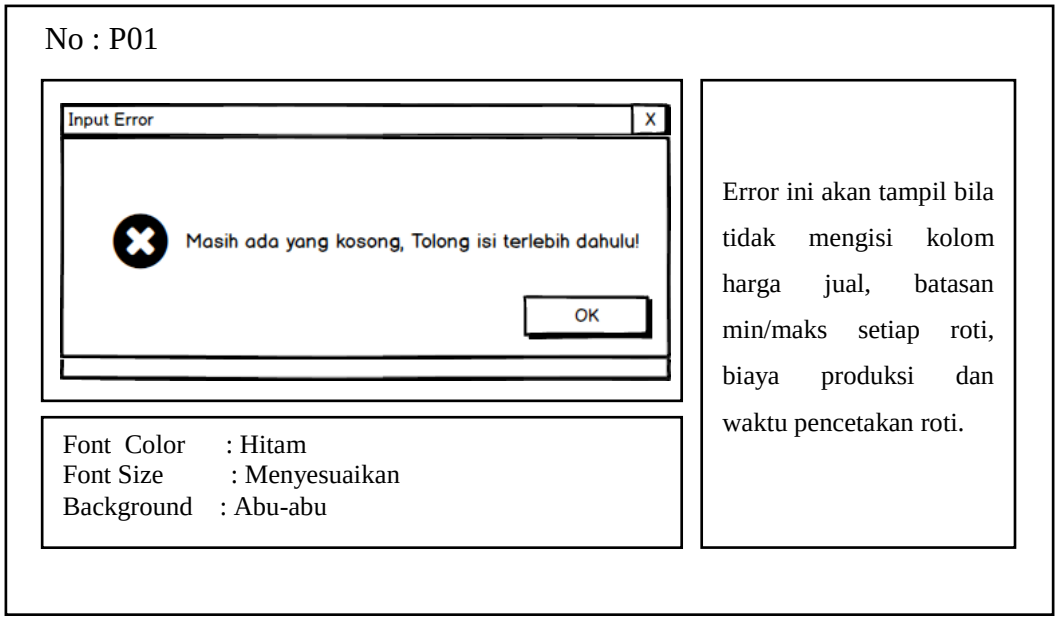
Gambar 3.18 Antarmuka Hasil Optimasi

3.11.3. Perancangan Pesan

Perancangan pesan mendeskripsikan rencana tampilan setiap pesan yang digunakan pada simulasi produksi. Berikut ini gambaran dari perancangan pesan pada simulasi produksi seperti terlihat pada Gambar 3.19 dan Gambar 3.20.

1. Perancangan pesan kesalahan input data

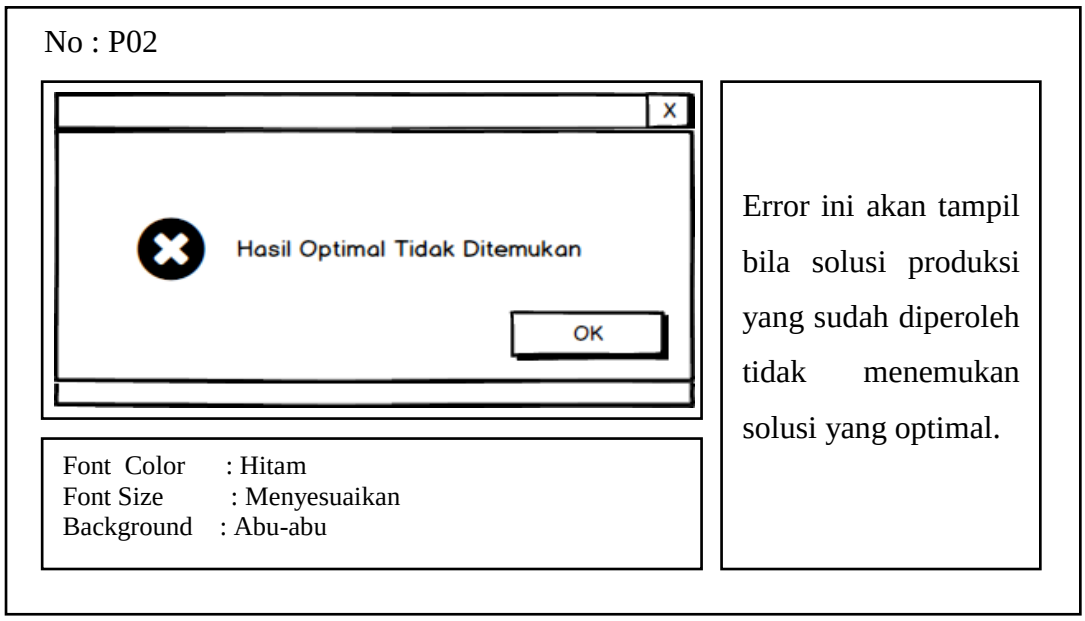
Pesan kesalahan input data harga jual, batasan min/maks setiap roti, biaya produksi dan waktu pencetakan bahwa informasi yang diberisikan data masukkan pada kolom tersebut masih kosong. Gambaran umum dari perancangan pesan kesalahan *input* data pada simulasi produksi dapat dilihat pada Gambar 3.19.



Gambar 3.19 Pesan Kesalahan Input Data

2. Perancangan pesan kesalahan hasil optimal

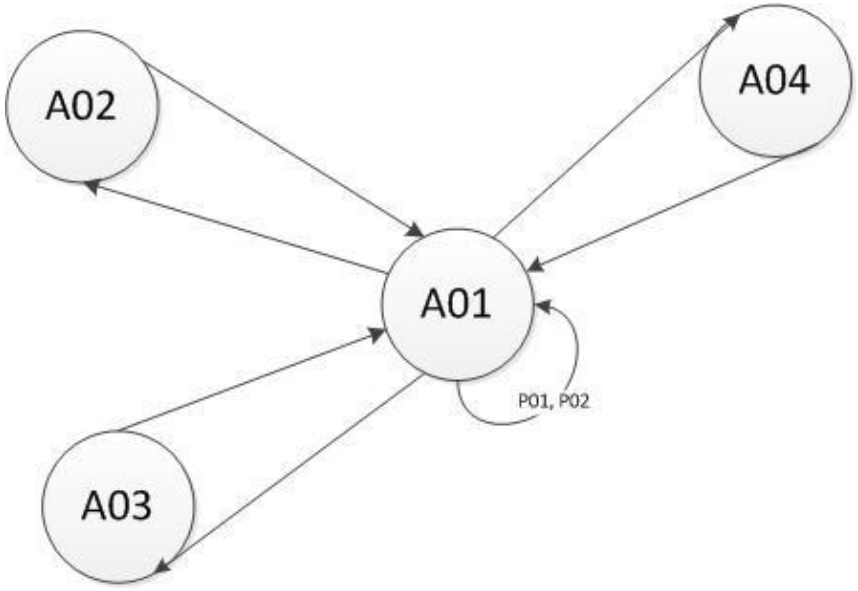
Pesan kesalahan hasil optimal terjadi bila solusi produksi yang sudah diperoleh dari pembangkitan dan perhitungan menggunakan algoritma *firefly* tidak menemukan solusi yang optimal, dalam arti tidak ada yang mendekati batasan biaya produksi dan waktu pencetakan yang sudah ditentukan sebelumnya. Pesan kesalahan ini dapat dilihat pada Gambar 3.20.



Gambar 3.20 Pesan Kesalahan Hasil Optimal

3.11.4. Jaringan Semantik Simulasi Produksi

Jaringan semantik bertujuan untuk menggambarkan alur informasi dari setiap aplikasi. Jaringan semantik dari simulasi produksi dapat dilihat pada Gambar 3.21.



Gambar 3.21 Jaringan Semantik Simulasi Produksi

A01 : Halaman untuk menampilkan menu utama
A02 : Halaman untuk menampilkan petunjuk penggunaan
A03 : Halaman untuk menampilkan keterangan
A04 : Halaman untuk menampilkan hasil optimasi
P01 : Pesan kesalahan input data
P02 : Pesan kesalahan hasil optimal