BAB IV

PEMBAHASAN

Pada bab ini akan dijelaskan penyelesaian masalah *Resource Constrained Project Scheduling Problem* menggunakan *Firefly Algorithm*.

4.1 *Resource Constrained Project Sceduling Problem (RCPSP)*

Resource Constrained Project Scheduling Problem atau Penjadwalan Proyek dengan Sumber Daya Terbatas merupakan suatu permasalahan penjadwalan yang bertujuan untuk mencari durasi tercepat dalam penyelesaian beberapa aktivitas dalam suatu proyek. Suatu proyek dikatakan selesai jika semua aktivitas dalam proyek selesai dijadwalkan. Dalam proses pengerjaan aktivitas dalam proyek membutuhkan beberapa tipe sumber daya dan banyaknya tipe sumber daya yang dibutuhkan setiap aktivitas adalah sama sedangkan sumber daya yang dibutuhkan terbatas.

Dalam penjadwalan proyek terdapat kendala yang harus dipenuhi yaitu aktivitas yang mempunyai aktivitas pendahulu bisa dikerjakan setelah semua pendahulunya selesai dikerjakan dan sumber daya yang digunakan untuk mengerjakan suatu aktivitas tidak boleh melebihi sumber daya yang tersedia.

4.2 *Firefly Algorithm*

Firefly Algorithm dalam skripsi ini merupakan algoritma yang digunakan untuk menyelesaikan permasalahan *Resource Constrained Project Scheduling Problem*. Proses penyelesaian permasalahan tersebut diawali dengan membangkitkan populasi awal kunang-kunang (*firefly*) yang merepresentasikan

solusi-solusi dari permasalahan. Setelah itu dilakukan langkah yang sesuai pada prosedur *Firefly Algorithm* hingga didapatkan populasi baru *firefly*. Dalam populasi baru tersebut terdapat solusi terbaik yang nantinya akan menjadi solusi dari permasalahan yang dicari.

Untuk mempermudah ilustrasi penyelesaian *Resource Constrained Project Scheduling Problem* dengan menggunakan *Firefly Algorithm*, maka prosedur di atas diuraikan dalam *flowchart* pada **Gambar 3.1**. Proses *Firefly Algorithm* ini akan berhenti setelah mencapai *maks_iterasi*.

Prosedur *Firefly Algorithm* dapat dilihat pada **Gambar 4.1**.

Firefly Algorithm

begin
input data();
inisialisasi parameter();
bangkitkan populasi awal *fireflies* x_i (i = 1,2,…,n);
evaluasi fungsi tujuan $f(x)$;
hitung intensitas cahaya I_i pada x_i berdasarkan $f(x)$;
while (t < maks_iterasi)
for i = 1 to m banyak *fireflies*
for j = 1 to m banyak *fireflies*
if ($I_i < I_j$) *firefly* i bergerak menuju j;
hitung jarak r;
hitung *attractiveness* β;
evaluasi solusi baru dan update intensitas cahaya; **endif**
end for j
end for i
tentukan global best sementara g*
lakukan *movement* pada *firefly* terbaik dengan $\beta = 0$
end while
end

Gambar 4.1 Prosedur *Firefly Algorithm*

Berikut ini dijelaskan langkah-langkah dalam prosedur *Firefly Algorithm*.

4.2.1 Inisialisasi Parameter

Parameter yang digunakan dalam proses *Firefly Algorithm* ini adalah jumlah populasi awal *firefly* yang disimbolkan dengan m, koefisien penyerapan cahaya pada *firefly* (γ), koefisien parameter random (α), keaktraktifan *firefly* awal (β_0), dan maks_iterasi merupakan iterasi maksimum. Untuk *input* data dan inisialisasi paramater prosedurnya disajikan pada **Gambar 4.2**.

Prosedur inisialisasi parameter

begin

jumlah populasi awal *fireflies* : *m*;
banyak iterasi : *maks_iterasi*;
koefisien parameter random : $\alpha = 0.2$;
koefisien keatraktifan awal : $\beta_0 = 1$
koefisien penyerapan cahaya oleh udara : $\gamma = 1$;

end

Gambar 4.2 Prosedur inisialisasi parameter

Proses berikutnya adalah membangkitkan populasi awal *firefly* sebanyak m *firefly*.

4.2.2 Pembangkitan Populasi Awal

Pembangkitan populasi awal *firefly* yang merepresentasikan penjadwalan proyek dilakukan sebanyak *m firefly*. Tiap *firefly* terdiri dari lokus *firefly* sebanyak n aktivitas yang berisi nilai prioritas dari aktivitas yang diwakili oleh lokus *firefly* tersebut. Oleh karena itu, untuk membangkitkan populasi awal perlu dibangkitkan

bilangan random antara 0 dan 1 untuk mewakili prioritas tiap *firefly* dalam tiap untaian *firefly*. Nilai prioritas tiap untaian *firefly* nantinya akan digunakan untuk menentukan aktivitas mana yang akan dijadwalkan terlebih dahulu. Untuk mempermudah pembangkitan populasi awal, dibuat matrix dengan ukuran *m* x *n aktivitas*. Baris ke-*i* dalam matrix menyatakan *firefly* ke-*i,* sedangkan kolom ke-*j* menyatakan *firefly* ke-*j* dari tiap *firefly*. **Gambar 4.3** merupakan prosedur pembangkitan populasi awal yang disajikan dalam sebuah matrix .

```
Prosedur Pembangkitan Populasi awal

Begin
    for i=0 to m-1
        for j=0 to naktivitas
            matrix_i,j = random (0,1);
        end
    end
end
```

Gambar 4.3 Prosedur Pembangkitan Populasi Awal

Langkah selanjutnya yaitu mengevaluasi fungsi tujuan dengan menghitung durasi tiap *firefly*.

4.2.3 Evaluasi Fungsi Tujuan atau Menghitung Durasi

Setelah nilai prioritas diperoleh dari pembangkitan populasi awal, maka aktivitas-aktivitas dalam suatu proyek sudah bisa dijadwalkan. Durasi proyek akan diperoleh jika semua aktivitas sudah dijadwalkan. Penjadwalan tiap aktivitas dilakukan dengan cara menentukan waktu mulai atau *start time (st)* dan waktu selesai atau *finish time (ft)* dari tiap aktivitas tersebut. Prosedur yang digunakan

untuk menghitung durasi proyek dengan cara menjadwalkan tiap aktivitasnya dengan menggunakan *priority scheduling method* terdapat pada **Lampiran 1**.

Setelah menghitung durasi, langkah selanjutnya adalah menghitung intensitas cahaya tiap kunang-kunang.

4.2.4 Menghitung Intensitas Cahaya Tiap *Firefly*

Dalam kasus memaksimumkan intensitas cahaya pada *firefly* sebanding dengan fungsi tujuan sehingga perhitungan intensitas cahaya tiap *firefly* bergantung pada nilai dari fungsi tujuan. Dalam kasus meminimumkan intensitas cahaya berbanding terbalik dengan fungsi tujuan. Prosedur yang digunakan untuk menghitung intensitas cahaya disajikan pada **Gambar 4.4**.

```
Prosedur menghitung intensitas cahaya firefly ke-i

begin
    intensitascahaya[i]=1/fungsitujuan[i];
end
```

Gambar 4.4 Prosedur menghitung intensitas cahaya *firefly* ke-*i*

Langkah selanjutnya, intensitas cahaya tiap *firefly* dibandingkan dengan intensitas cahaya *firefly* lainnya.

4.2.5 Membandingkan Intensitas Cahaya Tiap *Firefly*

Pada proses ini, jika terdapat *firefly* yang intensitas cahayanya lebih besar, maka *firefly* dengan intensitas cahaya kecil akan bergerak menuju *firefly* dengan

intensitas cahaya yang lebih besar. Prosedur membandingkan intensitas cahaya tiap *firefly* disajikan pada **Gambar 4.5**

Prosedur membandingkan intensitas cahaya tiap *firefly*

```
begin
  for i = 1 to m
      for j = 1 to m
          if intensitascahaya[i] < intensitascahaya[j]
             firefly i bergerak menuju j; end
        end
    end
end
```

Gambar 4.5 Prosedur membandingkan intensitas cahaya tiap *firefly*

Langkah selanjutnya yaitu menghitung *distance, attractiveness,* dan *movement* apabila terdapat pergerakan *firefly*.

4.2.6 Menghitung *Distance, Attractiveness,* dan *Movement*

Apabila terdapat *firefly i* yang bergerak menuju *firefly j*, maka posisi *firefly i* akan berubah atau dengan kata lain terjadi perubahan solusi dari *firefly i*. Untuk mengetahui posisi baru pada *firefly i*, maka langkah pertama yaitu menghitung *distance* (jarak) antara *firefly i* dengan *firefly j* menggunakan persamaan (2.7), kemudian langkah kedua yaitu menghitung *attractiveness* (β) *firefly i* menggunakan persamaan (2.6).

Prosedur menghitung *distance* (jarak) dan *attractiveness* (keatraktifan) disajikan pada **Gambar 4.6**.

Prosedur menghitung *distance* dan *attractiveness*

```
begin
jumlah = 0
for k = 1 to n
    a = firefly1[i][k];
    b= pow(a,2);
    jumlah = jumlah + b;
end
jarak = sqrt(jumlah);
β = β0 *exp(-γ* jarak);
end
```

Gambar 4.6 Prosedur menghitung *distance* dan *attractiveness*

Setelah dihitung *distance* dan *attractiveness* dari *firefly i* ke *j*, maka terjadi perubahan posisi baru *firefly i* karena bergerak menuju *firefly j* sehingga langkah ketiga yaitu menghitung posisi baru pada *firefly i* dengan menggunakan persamaan *movement* seperti pada persamaan (2.9). Untuk prosedur persamaan *movement* disajikan pada **Gambar 4.7**

Prosedur persamaan *movement*

```
begin
rand = random (0,1);
  for i=1 to m
     for k = 1 to n
      w = (1-β)*firefly1[i][k]+β*firefly1[j][k] + α*(rand-0.5);
      firefly1[i][k] = w;
      end
   end
end
```

Gambar 4.7 Prosedur persamaan *movement*

Setelah dilakukan persamaan *movement*, maka solusi baru diperoleh. Langkah selanjutnya yaitu menentukan *global best* (g^*).

4.2.7 Menentukan *Global Best* (*g**)

Setelah membandingkan intensitas cahaya tiap *firefly* selesai dilakukan, dicari *firefly* dengan intensitas cahaya tertinggi (*firefly* terbaik). *Firefly* terbaik tersebut akan dibandingkan dengan *Global Best* (*g**). Apabila *firefly* terbaik saat itu lebih besar intensitas cahayanya dari pada *g**, maka *firefly* tersebut menjadi *g**. Hal ini bertujuan agar solusi terbaik yang pernah didapatkan tidak hilang. Prosedur menentukan *Global Best* disajikan pada **Gambar 4.8**.

```
Prosedur menentukan Global Best

begin
best = intensitascahaya[1];
for i = 2 to m
    if best < intensitascahaya[i]
        best = intensitascahaya[i];

      end

end
if g* < best
    g* = best; end
end
```

Gambar 4.8 Prosedur menentukan *Global Best*

Setelah menentukan *Global Best*, langkah berikutnya adalah melakukan *movement* pada *firefly* terbaik.

4.2.8 Melakukan *Movement* Pada *Firefly* Terbaik

Firefly terbaik tiap iterasi akan bergerak random sesuai persamaan *movement* (persamaan 2.9) dengan memasukkan nilai $\beta = 0$. Prosedur untuk melakukan *movement* pada *firefly* terbaik dapat dilihat pada **Gambar 4.9**.

```
Prosedur melakukan movement pada firefly terbaik

begin
rand = random(0,1);
   for i=1 to jumbest
      for k = 1 to n
       w = firefly1[best][k]+ α*(rand-0.5);
       firefly1[best][k] = w;
      end
   end
end
```

Gambar 4.9 Prosedur melakukan *movement* pada *firefly* terbaik.

Selanjutnya yang akan dibahas adalah data yang akan digunakan dalam proses manual dan implementasi program.

4.3 Data

Data yang digunakan dalam skripsi ini ada 3, antara lain:

1. Data proyek yang terdiri dari 10 aktivitas dan 4 sumber daya.

   Data ini diperoleh dari data 30 aktivitas dengan 4 sumber daya tetapi hanya diambil 10 aktivitas pertama. Data ini diambil dari www.om-db.wi.tum.de/psplib/data.html dan nantinya akan digunakan sebagai contoh penyelesaian *resource constrained project scheduling problem* secara manual. Pada data proyek yang terdiri dari 10 aktivitas dan 4 sumber daya, diketahui

nilai durasi setiap aktivitasnya, jumlah sumber daya yang dibutuhkan dalam setiap aktivitasnya, pendahulu setiap aktivitas, serta jumlah tipe sumber daya dan ketersediaan sumber daya untuk menyelesaikan sebuah proyek. Data selengkapnya bisa dilihat pada **Tabel 4.1**.

2. Data proyek yang terdiri dari 25 aktivitas dan 3 sumber daya.

Data ini terdapat pada jurnal (**Zhang dkk, 2006).** Pada data proyek yang terdiri dari 25 aktivitas dan 3 sumber daya, diketahui nilai durasi setiap aktivitasnya, jumlah sumber daya yang dibutuhkan dalam setiap aktivitasnya, pendahulu setiap aktivitas, serta jumlah tipe sumber daya dan ketersediaan sumber daya untuk menyelesaikan sebuah proyek. Data selengkapnya bisa dilihat pada **Lampiran 2.**

3. Data proyek yang terdiri dari 60 aktivitas dan 4 sumber daya

Data ini diambil dari www.om-db.wi.tum.de/psplib/data.html. Pada data proyek yang terdiri dari 60 aktivitas dan 4 sumber daya, diketahui nilai durasi setiap aktivitasnya, jumlah sumber daya yang dibutuhkan dalam setiap aktivitasnya, pendahulu setiap aktivitas, serta jumlah tipe sumber daya dan ketersediaan sumber daya untuk menyelesaikan sebuah proyek. Data selengkapnya bisa dilihat pada **Lampiran 3.**

4.4 Contoh Penyelesaian *Resource Constrained Project Scheduling Problem (RCPSP)* 10 Aktivitas Secara Manual

Pada sub bab ini akan ditunjukkan bagaimana mengimplementasikan *Firefly Algorithm* pada contoh kasus *Resource Constrained Project Scheduling Problem* (RCPSP) dengan 10 aktivitas dan 4 sumber daya. Durasi aktivitas menggunakan satuan waktu dan sumber daya menggunakan satuan unit.

Data proyek dengan 10 aktivitas dan 4 tipe sumber daya bisa dilihat pada **Tabel 4.1**

Tabel 4.1 Data Proyek dengan 10 Aktivitas

Aktivitas(i)	**Dur_i**	**$R_{i,1}$**	**$R_{i,2}$**	**$R_{i,3}$**	**$R_{i,4}$**	**$Pendahulu_{i,1}$**
1	9	3	7	0	10	0
2	5	0	2	10	6	0
3	9	7	5	7	0	0
4	10	4	6	10	8	1
5	8	8	4	7	8	3
6	1	9	7	1	3	2
7	10	3	0	2	4	5
8	6	4	0	3	8	3
9	3	0	7	4	0	1
10	9	4	0	4	8	1

Sumber daya yang tersedia (R_1, R_2, R_3, R_4) = (20,19,23,23)

Keterangan :

Dur_i : Durasi aktivitas i.

$R_{i,1}$: Sumber daya tipe 1 yang dibutuhkan oleh aktivitas *i*.

$Pendahulu_{i,1}$: Pendahulu aktivitas i ke 1.

R_i : Sumber daya tipe *i* yang tersedia.

t_{now} : Waktu sekarang.

Fns : Kumpulan aktivitas yang selesai dijadwalkan

Langkah-langkah yang harus dilakukan untuk menyelesaikan *Resource Constrained Project Scheduling Problem (RCPSP)* sesuai dengan prosedur *Firefly Algorithm* adalah sebagai berikut :

Langkah 1 : Menentukan parameter

Parameter yang digunakan yaitu *m* nilainya harus lebih dari 2, karena dalam mendapatkan solusi baru terdapat proses membandingkan intensitas cahaya antar kunang-kunang 1 dengan kunang-kunang lain, γ adalah bilangan real yang nilainya antara 0 hingga ∞, α adalah bilangan real antara 0 hingga 1, β_0 adalah bilangan real yang nilainya antara 0 hingga ∞, dan *maks_iterasi* adalah bilangan integer yang nilainya lebih dari 1. Berikut ini nilai parameter yang digunakan dalam perhitungan manual :

a. Ukuran populasi *firefly* = 3
b. Koefisien penyerapan cahaya (γ) = 1
c. Parameter random (α) = 0,2
d. Keaktraktifan *firefly* awal = 1
e. Banyaknya iterasi yang akan dijalankan (*maks_iterasi)* = 1

Setelah menentukan parameter yang akan digunakan, langkah selanjutnya adalah membangkitkan populasi awal.

Langkah 2 : Membangkitkan populasi awal

Populasi awal yang dibangkitkan adalah sebanyak *m* yaitu 3 *firefly*. Dibangkitkan secara random bilangan real dari 0 sampai 1 yang menyatakan nilai prioritas untuk setiap untaian *firefly* dalam tiap *firefly*.

Pembangkitan populasi awal *firefly* dapat dilihat pada **Tabel 4.2.**

Tabel 4.2 Populasi Awal

m	Aktivitas									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x_1	0,0487	0,9027	0,9448	0,4909	0,4893	0,3377	0,9001	0,3692	0,1112	0,7803
x_2	0,3897	0,2417	0,4039	0,0965	0,1320	0,9421	0,9561	0,5752	0,0598	0,2348
x_3	0,4419	0,6765	0,1161	0,3308	0,6868	0,4988	0,6666	0,3436	0,0107	0,4818

Setelah nilai prioritas diperoleh, proyek sudah bisa dijadwalkan.

Langkah 3 : Menghitung fungsi tujuan (f(x)) dan Intensitas Cahaya (I(x))

i. $t_{now} = 0$

Pada saat $t_{now} = 0$, belum ada aktivitas yang selesai dijadwalkan sehingga *Fns* = Ø. Sumber daya yang tersedia disajikan dalam bentuk vektor sebagai berikut : $(R_1, R_2, R_3, R_4) = (20, 19, 23, 23)$

Pada **Tabel 4.1**, aktivitas 1, aktivitas 2, dan aktivitas 3 aktivitas pendahulunya 0 artinya aktivitas tersebut tidak mempunyai aktivitas pendahulu. Periksa sumber daya yang dibutuhkan oleh aktivitas tersebut. Jika lebih kecil atau sama dengan sumber daya yang tersedia maka aktivitas tersebut termasuk dalam *eligible activity (E).*

a. Aktivitas 1

R_i	**Tanda**	$R_{1,i}$
20	>	3
19	>	7
23	>	0
23	>	10

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 1 termasuk dalam *E*.

b. Aktivitas 2

R_i	Tanda	$R_{2,i}$
20	>	0
19	>	2
23	>	10
23	>	6

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 2 termasuk dalam *E*.

c. Aktivitas 3

R_i	Tanda	$R_{3,i}$
20	>	7
19	>	5
23	>	7
23	>	0

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 1 termasuk dalam *E*.

Aktivitas 1, aktivitas 2, dan aktivitas 3 memenuhi kendala aktivitas pendahulu langsung dan memenuhi kendala sumber daya, maka aktivitas tersebut termasuk dalam himpunan *E*.

E = (aktivitas 1, aktivitas 2, aktivitas 3)

Untuk menentukan aktivitas mana yang akan dijadwalkan terlebih dahulu, dilihat dari nilai prioritas dari aktivitas yang terdapat dalam *E*. Aktivitas yang mempunyai nilai prioritas terbesar yang akan dijadwalkan.

$\max\{x_{1,1}; x_{1,2}; x_{1,3}\} = \max\{0{,}0497; 0{,}9027; 0{,}9448\} = 0{,}9448$

Karena nilai prioritas aktivitas 3 paling besar, jadwalkan aktivitas 3 terlebih dahulu.

$Dur_3 = 9$

$StartTime_3 = t_{now} = 0$

$FinishTime_3 = t_{now} + Dur_3 = 0 + 9 = 9$

$(R_{3,1}, R_{3,2}, R_{3,3}, R_{3,4}) = (7, 5, 7, 0)$

Perbarui vektor sumber daya dengan cara mengurangkan vektor sumber daya yang tersedia dengan sumber daya aktivitas 3.

$(R_1, R_2, R_3, R_4) = (R_1, R_2, R_3, R_4) - (R_{3,1}, R_{3,2}, R_{3,3}, R_{3,4})$

$= (20, 19, 23, 23) - (7, 5, 7, 0)$

$= (13, 14, 16, 23)$

Perbarui himpunan *E* dengan memeriksa kembali sumber daya yang tersedia apakah lebih besar atau sama dengan sumber daya yang dibutuhkan.

a. Aktivitas 1

R_i	Tanda	$R_{1,i}$
20	>	3
17	>	7
13	>	0
17	>	10

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 1 termasuk dalam *E*.

b. Aktivitas 2

R_i	Tanda	$R_{2,i}$
20	>	0
19	>	2
23	>	10
23	>	6

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 1 dan aktivitas 2 termasuk dalam *E*.

E = {aktivitas 1, aktivitas 2}

$\max\{x_{1,1}; x_{1,3}\} = \max\{0{,}0497; 0{,}9027\} = 0{,}9027 = x_{1,2}$

Nilai prioritas aktivitas 2 lebih besar dari pada nilai prioritas aktivitas 1, maka jadwalkan aktivitas 2 lebih dulu.

$Dur_2 = 5$

$StartTime_2 = t_{now} = 0$

$FinishTime_2 = t_{now} + Dur_2 = 0 + 5 = 5$

$(R_{2,1}, R_{2,2}, R_{2,3}, R_{2,4}) = (0, 2, 10, 6)$

$(R_1, R_2, R_3, R_4) = (R_1, R_2, R_3, R_4) - (R_{2,1}, R_{2,2}, R_{2,3}, R_{2,4})$

$= (13, 14, 16, 23) - (0, 2, 10, 6)$

$= (13, 12, 6, 17)$

Perbarui himpunan *E* dengan memeriksa kembali sumber daya yang tersedia apakah lebih besar atau sama dengan sumber daya yang dibutuhkan.

a. Aktivitas 1

R_i	Tanda	$R_{1,i}$
13	>	3
12	>	7
6	>	0
17	>	10

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 1 termasuk dalam *E*.

E = {aktivitas 1}

Jadwalkan aktivitas 1.

$Dur_1 = 9$

$StartTime_1 = t_{now} = 0$

$FinishTime_1 = t_{now} + Dur_1 = 0 + 9 = 9$

$(R_{1,1}, R_{1,2}, R_{1,3}, R_{1,4}) = (3, 7, 0, 10)$

(R_1, R_2, R_3, R_4) $= (R_1, R_2, R_3, R_4) - (R_{1,1}, R_{1,2}, R_{1,3}, R_{1,4})$

$= (13, 12, 6, 17) - (3, 7, 0, 10)$

$= (10, 5, 6, 7)$

Karena sudah tidak ada aktivitas yang bisa dijadwalkan maka $E = Ø$. langkah selanjutnya adalah perbarui t_{now} menjadi *finish time* (*ft*) terawal.

$t_{now} = \min \{ft_2, ft_3, ft_1\} = \min\{5,9,9\} = 5 = ft_2$

t_{now} ini akan menjadi t_{now} selanjutnya.

ii. $t_{now} = 5$

Pada saat $t_{now} = 5$ aktivitas 2 selesai dijadwalkan maka sumber daya yang digunakan oleh aktivitas 2 bisa digunakan oleh aktivitas lain, oleh karena itu perbarui vektor sumber daya dengan menambahkan sumber daya yang tersedia dengan sumber daya yang digunakan oleh aktivitas 2.

$(R_1, R_2, R_3, R_4) = (R_1, R_2, R_3, R_4) + (R_{2,1}, R_{2,2}, R_{2,3}, R_{2,4})$

$= (10, 5, 6, 7) + (0, 2, 10, 6)$

$= (10, 7, 16, 13)$

Setelah aktivitas 2 selesai dikerjakan, aktivitas yang belum terjadwal dan aktivitas pendahulu langsungnya hanya aktivitas 2 akan bisa dijadwalkan jika memenuhi kendala sumber daya. Aktivitas tersebut adalah aktivitas 6.

a. Aktivitas 6

R_i	Tanda	$R_{6,i}$
10	>	9
7	>	7
16	>	1
13	>	3

Semua tipe sumber daya yang dibutuhkan lebih kecil dari sumber daya yang tersedia, maka aktivitas 6 termasuk dalam *E*.

E = {aktivitas 6}

Jadwalkan aktivitas 6.

$Dur_6 = 1$

$StartTime_6 = t_{now} = 5$

$FinishTime_6 = t_{now} + Dur_6 = 5 + 1 = 6$

$(R_{6,1}, R_{6,2}, R_{6,3}, R_{6,4}) = (9, 7, 1, 3)$

$(R_1, R_2, R_3, R_4) = (R_1, R_2, R_3, R_4) - (R_{6,1}, R_{6,2}, R_{6,3}, R_{6,4})$

$= (10, 7, 16, 13) - (9, 7, 1, 3)$

$= (1, 0, 15, 10)$

$E = Ø$

Perbarui t_{now}.

$t_{now} = \min \{ft_3, ft_1, ft_6\} = \min\{9, 9, 6\} = 6 = ft_6$

t_{now} ini akan menjadi t_{now} selanjutnya.

iii. $t_{now} = 6$

Fns = {aktivitas 2, aktivitas 6}

Perbarui vektor sumber daya.

$(R_1, R_2, R_3, R_4) = (R_1, R_2, R_3, R_4) + (R_{6,1}, R_{6,2}, R_{6,3}, R_{6,4})$

$= (1, 0, 15, 10) + (9, 7, 1, 3)$

$= (10, 7, 16, 13)$

Tidak ada aktivitas yang memenuhi kendala aktivitas pendahulu langsung sehingga tidak ada aktivitas dalam himpunan *E*.

$E = Ø$

Perbarui t_{now}.

t_{now} = min $\{ft_3, ft_1\}$

=min{9, 9}

= 9 = ft_3, ft_1

Proses diatas diulangi sampai semua aktivitas sudah dijadwalkan. Diperoleh *finish time* tiap aktivitas bisa dilihat pada **Tabel 4.3.** Durasi *firefly* 1 diperoleh dari waktu *finish time* paling besar diantara *finish time* semua aktivitas.

Durasi$_1$ = max$\{ft_1, ft_2, ft_3, ft_4, ft_5, ft_6 ft_7, ft_8, ft_9 ft_{10}\}$

= max{9, 5, 9, 19, 26, 6, 36, 25, 12, 18 } = 36

Jadi durasi *firefly* 1 adalah 36.

Tabel 4.3 *Finish Time* Aktivitas

ft_i	**waktu**
Aktivitas 1	9
Aktivitas 2	5
Aktivitas 3	9
Aktivitas 4	19
Aktivitas 5	26
Aktivitas 6	6
Aktivitas 7	36
Aktivitas 8	25
Aktivitas 9	12
Aktivitas 10	18

Dengan cara yang sama, diperoleh durasi untuk setiap *firefly* yang disajikan pada **Tabel 4.4.**

Tabel 4.4 Nilai durasi

x_i	D_i
1	36
2	33
3	28

Setelah selesai menghitung durasi tiap *firefly*, langkah berikutnya adalah menghitung intensitas cahaya tiap *firefly*. Semakin kecil nilai durasi yang dihasilkan suatu *firefly*, maka semakin besar intensitas cahaya *firefly* tersebut. Begitu pula sebaliknya, semakin besar nilai durasi yang dihasilkan suatu *firefly*, maka semakin kecil intensitas cahaya *firefly* tersebut.

Untuk menghitung intensitas cahaya tiap *firefly* menggunakan rumus: $I(x_i) = \frac{1}{f(x_i)}$. Hasil perhitungan intensitas cahaya tiap *firefly* disajikan pada **Tabel 4.5**.

Tabel 4.5 Intensitas cahaya tiap *firefly*

x_i	$I(x_i)$
x_1	0,0278
x_2	0,0303
x_3	0,0357

Setelah menghitung intensitas cahaya tiap *firefly*, langkah berikutnya adalah membandingkan intensitas cahaya tiap *firefly*.

Langkah 4 : Membandingkan intensitas cahaya tiap *firefly*

Tiap *firefly* dibandingkan intensitas cahayanya dengan *firefly* lain. Apabila terdapat *firefly* yang intensitas cahayanya lebih tinggi, maka *firefly* dengan intensitas cahaya lebih rendah akan bergerak menuju *firefly* dengan intensitas cahaya lebih tinggi kemudian menggunakan persamaan *movement* sehingga akan diperoleh solusi baru dari tiap pergerakan *firefly*.

Jika dilihat pada **Tabel 4.5** nilai $I(x_1) < I(x_2)$ artinya nilai intensitas cahaya *firefly* 1 lebih kecil dari intensitas cahaya *firefly* 2 sehingga *firefly* 1 bergerak menuju *firefly* 2. Langkah berikutnya yaitu menghitung *distance, attractivenes,* dan *movement* untuk mendapatkan posisi baru pada *firefly* 1.

Langkah 5 : Menghitung *distance, attractivenes,* dan *movement*

Untuk mendapatkan posisi baru pada *firefly* 1 maka langkah selanjutnya yaitu menghitung *distance* atau jarak antara *firefly* 1 dengan *firefly* 2 menggunakan rumus (2.7), yaitu:

$$R_{1,2} = \sqrt{(x_1^1 - x_2^1)^2 + (x_1^2 - x_2^2)^2 + \ldots + (x_1^{10} - x_2^{10})^2}$$

$$= \sqrt{(0.0497 - 0.3897)^2 + (0.9027 - 0.2417)^2 + \ldots + (0.7803 - 0.2348)^2}$$

$$= \sqrt{0.1156 + 0.4369 + \ldots + 0.2976}$$

$$= 1{,}34635$$

Diperoleh $r_{2,1}$= 1,34635. Kemudian menghitung nilai *attractiveness* menggunakan persamaan (2.6) dengan memasukkan nilai $\beta_0 = 1$, $\gamma = 1$, dan r = 1,34635.

$$\beta = \beta_0 e^{-\gamma r^2}$$

$$= e^{-1{,}34635^2} = 0{,}1632$$

Sehingga diperoleh nilai β = 0,1632. Langkah terakhir adalah melakukan proses *movement* terhadap *firefly* 1 menggunakan rumus (2.9), yaitu:

$$x_{1_{new}} = x_i^k + \beta\left(x_j^k - x_i^k\right) + \alpha\left(rand - \frac{1}{2}\right)$$

$$x_{1_{new}} = x_1^k + 0{,}1632\left(x_2^k - x_1^k\right) + 0.2\left(0{,}3092 - \frac{1}{2}\right)$$

Untuk k = 1,

$$x_{1_{new}} = x_1^1 + 0{,}1632(x_2^1 - x_1^1) + 0.2\left(0{,}3092 - \frac{1}{2}\right)$$

$$= 0.0497 + 0{,}1632(0.3897 - 0.0497) + 0.2\left(0{,}3092 - \frac{1}{2}\right)$$

$$= 0{,}067028$$

Demikian juga untuk k = 2, . . .,10, dan didapatkan *firefly* 1 baru, yaitu:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$x_{1_{new}}$	0,0670	0,8948	0,8929	0,4386	0,5240	0,4879	0,8709	0,3989	0,1954	0,7712

Setelah mendapatkan posisi *firefly* 1 baru atau $x_{1_{new}}$, maka langkah selanjutnya yaitu menghitung nilai intensitas cahaya pada $x_{1_{new}}$ dengan cara yang sama seperti pada **Langkah 3**.

Hasil lengkap proses membandingkan intensitas cahaya tiap *firefly* yaitu sebagai berikut :

1. i = 1 j = 2

Karena $I(x_1) < I(x_2)$, maka terjadi pergerakan antara *firefly* 1 menuju *firefly* 2.

$\boldsymbol{x_1}$	0,0487	0,9027	0,9448	0,4909	0,4893	0,3377	0,9001	0,3692	0,1112	0,7803
$\boldsymbol{x_2}$	0,3897	0,2417	0,4039	0,0965	0,1320	0,9421	0,9561	0,5752	0,0598	0,2348

Menghitung *distance*, *attractiveness*, dan *movement* untuk mendapatkan posisi baru pada *firefly* 1 yaitu sebagai berikut :

$r_{1,2}$= 1,34635

β = 0,1632

rand	0,3092	0,9999	0,6820	0,5604	0,9650	0,7582	0,3143	0,4805	0,9631	0,8996

Dengan persamaan *movement* maka posisi *firefly* 1 baru yaitu :

$x_{1_{new}}$	0,0670	0,8948	0,8929	0,4386	0,5240	0,4879	0,8709	0,3989	0,1954	0,7712

Intensitas cahaya ($x_{1_{new}}$) = 0,0370

2. i = 1 j = 3 (Tidak bergerak)

3. i = 2 j = 1

Karena $I(x_2) < I(x_1)$, maka terjadi pergerakan antara *firefly* 2 menuju *firefly* 1.

$x_{1_{new}}$	0,0670	0,8948	0,8929	0,4386	0,5240	0,4879	0,8709	0,3989	0,1954	0,7712
x_2	0,3897	0,2417	0,4039	0,0965	0,1320	0,9421	0,9561	0,5752	0,0598	0,2348

Menghitung *distance*, *attractiveness*, dan *movement* untuk mendapatkan posisi baru pada *firefly* 1 yaitu sebagai berikut :

$r_{2,1}$= 1,2613

β = 0,2037

rand	0,5772	0,6180	0,1351	0,8773	0,3002	0,3198	0,3147	0,1533	0,7878	0,0164

Dengan persamaan *movement* maka posisi *firefly* 2 baru yaitu :

$x_{2_{new}}$	0,3394	0,3983	0,4305	0,2416	0,1719	0,8135	0,9019	0,4699	0,1449	0,2473

Intensitas cahaya ($x_{2_{new}}$) = 0,0278

4. i = 2 j = 3

Karena $I(x_2) < I(x_3)$, maka terjadi pergerakan antara *firefly* 2 menuju *firefly* 1.

$x_{2_{new}}$	0,3897	0,2417	0,4039	0,0965	0,1320	0,9421	0,9561	0,5752	0,0598	0,2348
x_3	0,4419	0,6765	0,1161	0,3308	0,6868	0,4988	0,6666	0,3436	0,0107	0,4818

Menghitung *distance*, *attractiveness*, dan *movement* untuk mendapatkan posisi baru pada *firefly* 2 yaitu sebagai berikut :

$r_{2,3}$= 0,8386

β = 0,4949

rand	0,1275	0,1728	0,8758	0,4132	0,9999	0,9667	0,4631	0,0001	0,9632	0,0031

Dengan persamaan *movement* maka posisi *firefly* 2 baru yaitu :

$x_{2_{new}}$	0,3156	0,4706	0,3501	0,2684	0,5267	0,7511	0,7781	0,3074	0,1712	0,2640

Intensitas cahaya ($x_{2_{new}}$) = 0,0357

5. i = 3 j = 1

Karena $I(x_3) < I(x_1)$, maka terjadi pergerakan antara *firefly* 1 menuju *firefly* 1.

x_3	0,4419	0,6765	0,1161	0,3308	0,6868	0,4988	0,6666	0,3436	0,0107	0,4818
$x_{1_{new}}$	0,0670	0,8948	0,8929	0,4386	0,5240	0,4879	0,8709	0,3989	0,1954	0,7712

Menghitung *distance*, *attractiveness*, dan *movement* untuk mendapatkan posisi baru pada *firefly* 2 yaitu sebagai berikut :

$r_{3,1}$= 0,9965

β = 0,3704

rand	0,9776	0,8241	0,7031	0,2244	0,5755	0,3307	0,1845	0,5632	0,7729	0,1249

Dengan persamaan *movement* maka posisi *firefly* 3 baru yaitu :

$x_{3_{new}}$	0,3986	0,9665	0,5457	0,1074	0,2092	0,1469	0,2599	0,1604	0,1269	0,2109

Intensitas cahaya ($x_{3_{new}}$ = 0,0357)

6. i = 3 j = 2 (Tidak Bergerak)

Langkah 6 : Menentukan *global best* sementara

Global best adalah solusi terbaik yang pernah didapatkan saat proses berlangsung selama *maks_iterasi*. Untuk iterasi 1, *global best* adalah *firefly* terbaik saat itu. Sedangkan untuk iterasi berikutnya, jika *firefly* terbaik saat itu intensitas cahayanya lebih besar dari *global best*, maka *firefly* tersebut menjadi *global best*. Dari contoh kasus diatas, nilai intensitas cahaya yang tertinggi yaitu terdapat pada $x_{1_{new}}$, sehingga didapatkan *global best* untuk contoh kasus di atas adalah $g^* = x_{1_{new}}$. Dengan demikian solusi terbaik yang didapatkan adalah:

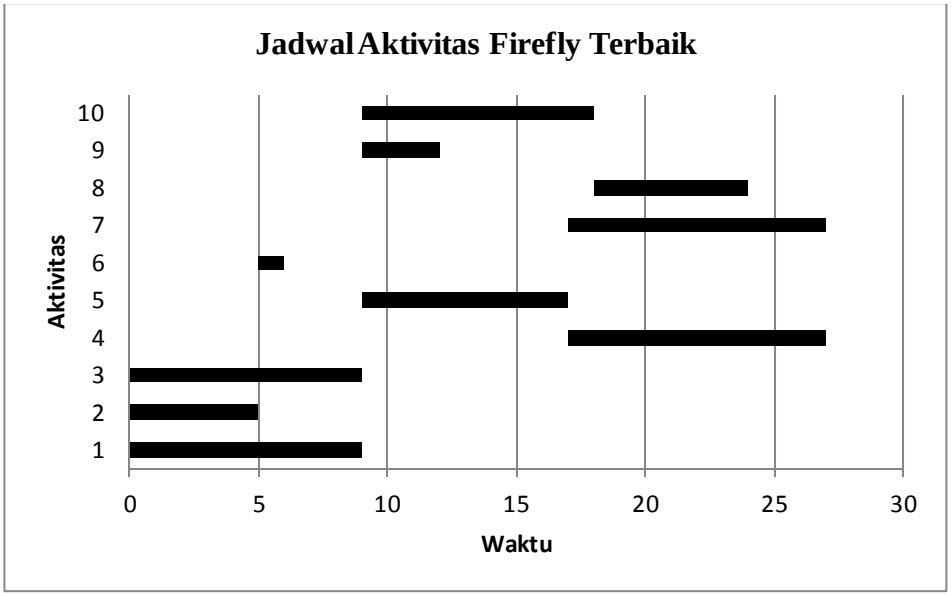
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$x_{1_{new}}$	0,0670	0,8948	0,8929	0,4386	0,5240	0,4879	0,8709	0,3989	0,1954	0,7712

Dengan menggunakan *priority scheduling method* didapatkan durasi dari *firefly* terbaik atau $x_{1_{new}}$ adalah 27. Hasil *finish time* dari solusi terbaik yang didapat hingga iterasi maksimal tercapai dapat disajikan pada **Tabel 4.6**.

Tabel 4.6 *Finish Time* Solusi Terbaik

fti	**Waktu**
1	9
2	5
3	9
4	27
5	17
6	6
7	27
8	24
9	12
10	18

Penjadwalan tiap aktivitas dari *firefly* dengan durasi tercepat bisa dilihat pada **Gambar 4.10.**



Gambar 4.10 Jadwal Aktivitas

Dari **Gambar 4.10** terlihat bahwa durasi dari $x_{1_{new}}$ yaitu 27. Aktivitas yang dapat dijadwalkan terlebih dahulu yaitu aktivitas 1, 2, dan 3 dengan *start time* dan *finish time* dari aktivitas 1 dan 3 yaitu 0 dan 9, sedangkan aktivitas 5 yaitu 0 dan 5. Kemudian aktivitas yang dapat dijadwalkan yaitu aktivitas 6 dengan *start time* dan *finish time* yaitu 5 dan 6, aktivitas 6 bisa dijadwalkan ketika aktivitas 2 selesai dikerjakan. Selanjutnya, aktivitas yang dapat dijadwalkan yaitu aktivitas 5, 9, 10 dengan *start time* dan *finish time* dari aktivitas 5 yaitu 9 dan 17, aktivitas 9 yaitu 9 dan 12, aktivitas 10 yaitu 9 dan 18. Aktivitas selanjutnya yang dapat dijadwalkan yaitu aktivitas 4, 7 dan 8 dengan *start time* dan *finish* time dari aktivitas 4 dan 7 yaitu 17 dan 27, sedangkan aktivitas 8 yaitu 18 dan 24.

Durasi dari $x_{1_{new}}$ ditentukan dari waktu *finish time* terbesar dari semua aktivitas, karena *finish time* terbesar 27 maka durasi dari $x_{1_{new}}$ yaitu 27.

Langkah 7 : Melakukan *movement* pada *firefly* terbaik

Langkah terakhir adalah melakukan proses *movement* (persamaan 2.9) pada *firefly* terbaik saat itu dengan memasukkan nilai $\beta = 0$. Pada langkah ke-6 diketahui *firefly* terbaik saat itu adalah x_1, maka:

$$x_{best}^{new} = x_{best}^{k} + \alpha\left(rand - \frac{1}{2}\right)$$

$$x_{1_{new}} = x_1^k + 0.2\left(0.0670 - \frac{1}{2}\right)$$

Untuk k = 1,

$$x_{1_{new}} = x_1^1 + 0.2\left(0.0670 - \frac{1}{2}\right)$$

$$x_{1_{new}} = 0.6351 + 0.1\left(0.0670 - \frac{1}{2}\right)$$

$$= 0.0941$$

Demikian juga untuk $k = 2, \ldots, 10$, sehingga diperoleh x_1 baru untuk iterasi berikutnya atau $x_{1_{new}}'$ yaitu:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$x_{1_{new}}'$	0,0941	0,8285	0,8091	0,4186	0,4979	0,5318	0,8661	0,3899	0,1085	0,7343

4.5 Program

Untuk mempermudah penyelesaian masalah *resource constrained project scheduling problem (RCPSP)* dengan *firefly algorithm* maka dibuat program menggunakan *software* Borland C++ 5.0.2.

4.6 Implementasi Program pada Contoh Kasus *Resource Constrained Project Scheduling Problem* (RCPSP)

Program *firefly algorithm* yang telah dibuat, diimplementasikan pada kasus RCPSP 10 aktivitas dan 4 tipe sumber daya, RCPSP 25 aktivitas dan 3 tipe sumber daya, dan RCPSP 60 aktivitas dan 4 tipe sumber daya. *Source code* program bisa dilihat di **Lampiran 4**.

4.6.1 Menggunakan Proyek Dengan 10 Aktivitas Dan 4 Sumber Daya

Berikut ini perbandingan solusi terbaik dari proyek dengan 10 aktivitas dan 4 tipe sumber daya yang dihasilkan dengan mengganti nilai *α*, *maks_iterasi,* dan *m*.

Tabel 4.7 Perbandingan Solusi Terbaik Proyek dengan 10 Aktivitas

Maks_Iterasi	***m***	***α***		
		0,1	**0,3**	**0,5**
5	10	27	27	27
	30	27	27	27
	50	27	27	27
100	10	27	27	27
	30	27	27	27
	50	27	27	27
300	10	27	27	27
	30	27	27	27
	50	27	27	27
500	10	27	27	27
	30	27	27	27
	50	27	27	27

Dari **Tabel 4.7** dapat dilihat bahwa saat nilai *maks_iterasi* dan *m* kecil sudah diperoleh solusi terbaik yaitu 27 dan saat parameter dari nilai α diubah, solusi yang diperoleh juga tidak berubah. Dari sini dapat diambil kesimpulan

bahwa perubahan parameter tidak menunjukkan pengaruh yang signifikan. Penjadwalan solusi terbaik dari data proyek sebanyak 10 aktivitas dengan 4 tipe sumber daya dapat dilihat pada **Lampiran 5**.

4.6.2 Menggunakan Proyek dengan 25 Aktivitas dan 3 Sumber Daya

Berikut ini perbandingan solusi terbaik dari proyek dengan 25 aktivitas dan 3 sumber daya yang dihasilkan oleh *firefly algorithm* dengan mengganti nilai *Maks_iterasi*, *m,* dan *α*.

Tabel 4.8 Perbandingan Solusi Terbaik Proyek dengan 25 Aktivitas

Maks_Iterasi	***m***	***α***		
		0,1	**0,3**	**0,5**
5	10	69	69	68
	30	67	67	67
	50	67	68	69
100	10	67	67	67
	30	66	67	67
	50	66	66	66
300	10	67	67	67
	30	67	67	66
	50	66	66	66
500	10	67	67	67
	30	66	66	66
	50	66	66	66

Dari **Tabel 4.8** dapat dilihat bahwa seiring bertambahnya nilai *m* dan maka durasi yang dihasilkan cenderung semakin baik. Kemudian dengan bertambahnya parameter α solusi yang dihasilkan bisa semakin baik, tetapi juga bisa semakin buruk seiring bertambahnya nilai *m*. Terlihat saat *maks_iterasi* =100 dan nilai *m* yaitu 30 dan 50, solusi yang dihasilkan menuju solusi terbaiknya yaitu 66. Dari sini dapat diambil kesimpulan bahwa perubahan parameter dari nilai α dan

maks_iterasi tidak terlalu menunjukkan pengaruhnya. Parameter yang berpengaruh yaitu *m*. Penjadwalan solusi terbaik dari data proyek sebanyak 25 aktivitas dengan 3 tipe sumber daya dapat dilihat pada **Lampiran 5**.

4.6.3 Menggunakan Proyek dengan 60 Aktivitas dan 4 Sumber Daya

Berikut ini perbandingan solusi terbaik dari proyek dengan 60aktivitas dan 4 sumber daya yang dihasilkan oleh *firefly algorithm* dengan mengganti nilai *Maks_iterasi*, *m,* dan *α*.

Tabel 4.9 Perbandingan Solusi Terbaik Proyek dengan 60 Aktivitas

Maks_Iterasi	***m***	***α***		
		0,1	**0,3**	**0,5**
5	10	82	82	82
	30	82	82	82
	50	82	82	82
100	10	82	82	82
	30	82	82	82
	50	82	82	82
300	10	82	82	82
	30	82	82	82
	50	82	82	82
500	10	82	82	82
	30	82	82	82
	50	82	82	82

Dari **Tabel 4.9** dapat dilihat bahwa saat nilai *maks_iterasi* dan *m* kecil sudah diperoleh solusi terbaik yaitu 82 dan saat parameter dari nilai α diubah, solusi yang diperoleh juga tidak berubah. Dari sini dapat diambil kesimpulan bahwa perubahan parameter tidak menunjukkan pengaruh yang signifikan. Penjadwalan solusi terbaik dari data proyek sebanyak 60 aktivitas dengan 4 tipe sumber daya dapat dilihat pada **Lampiran 5**.

4.7 Perbandingan Solusi dengan Algoritma Lain

Solusi yang dihasilkan dari proyek yang terdiri dari 25 aktivitas dan 3 sumber daya dibandingkan dengan solusi yang dihasilkan dari beberapa algoritma lain seperti yang ditulis oleh **Zhang, dkk, 2006**. Perbandingan solusi telah ditambahkan dengan solusi yang dihasilkan oleh *firefly algorithm*. Disajikan pada **Tabel 4.11**.

Tabel 4.11 Perbandingan Solusi Antar Algoritma

Algoritma	**Durasi Proyek**
Minimum Total Float (MITF)	74
Shortest Aktivity Duration (SAD)	71
Minimum Late Finish Time (MILFT)	67
Firefly Algorithm (FA)	66

Dari **Tabel 4.11** dapat kita lihat bahwa solusi yang dihasilkan oleh *Firefly Algorithm* merupakan solusi terbaik dibandingkan dengan Algoritma yang lain.