Implementasi Algoritma *Bee Colony Optimization* Dalam Pencarian Langkah Solusi Terpendek Pada *Puzzle* Rubik's Cube



Oleh : Michael Alexander Syauta - 10109791

Rubik's Cube

- 1974 – Erno Rubik.
- 2010 – over 350 million sold.
- Computer Science – Combinatorial
- Mekanisme



Rubik's Cube

- Intuisi & metode yang dihafalkan > 40 langkah
- 18 operator
  - Face turn(180°) & quarter turn(90°).
- Ruang Pencarian
  - $((8! \times 3^8) \times (12! \times 2^{12})) / 12$
  - 43×10^{18} konfigurasi



Rubik's Cube

- 1981 – Morwen Thistlethwaite.
  - Group theory
- 1992 – Herbert Kociemba.
  - Group theory – IDA* like
- 1997 – Richard Korf.
  - IDA* with Pattern Database
- 2010 – Herbert Kociemba & Thomas Rokicky.
  - Google idle Computers
- Kelemahan: waktu dan ruang

Bee Colony Optimization

- 2005 - Dusan Teodorovic
- Metode Swarm Intelligence yang mensimulasikan perilaku mencari makan pada koloni lebah madu.
- Constructive heuristic.
  - Rubik's Cube
- Asumsi

Perumusan Masalah

- Bagaimana mengimplementasikan algoritma *Bee Colony Optimization* pada suatu simulasi Rubik's Cube.
- Bagaimana performansi algoritma *Bee Colony Optimization* dalam menyelesaikan masalah *puzzle* Rubik's Cube.

Tujuan

- Mengimplementasikan algoritma *Bee Colony Optimization* untuk mencari langkah solusi terpendek pada simulasi Rubik's Cube.
- Menguji performansi algoritma *Bee Colony Optimization* dalam menyelesaikan masalah *puzzle* Rubik's Cube.

Algoritma *Bee Colony Optimization*

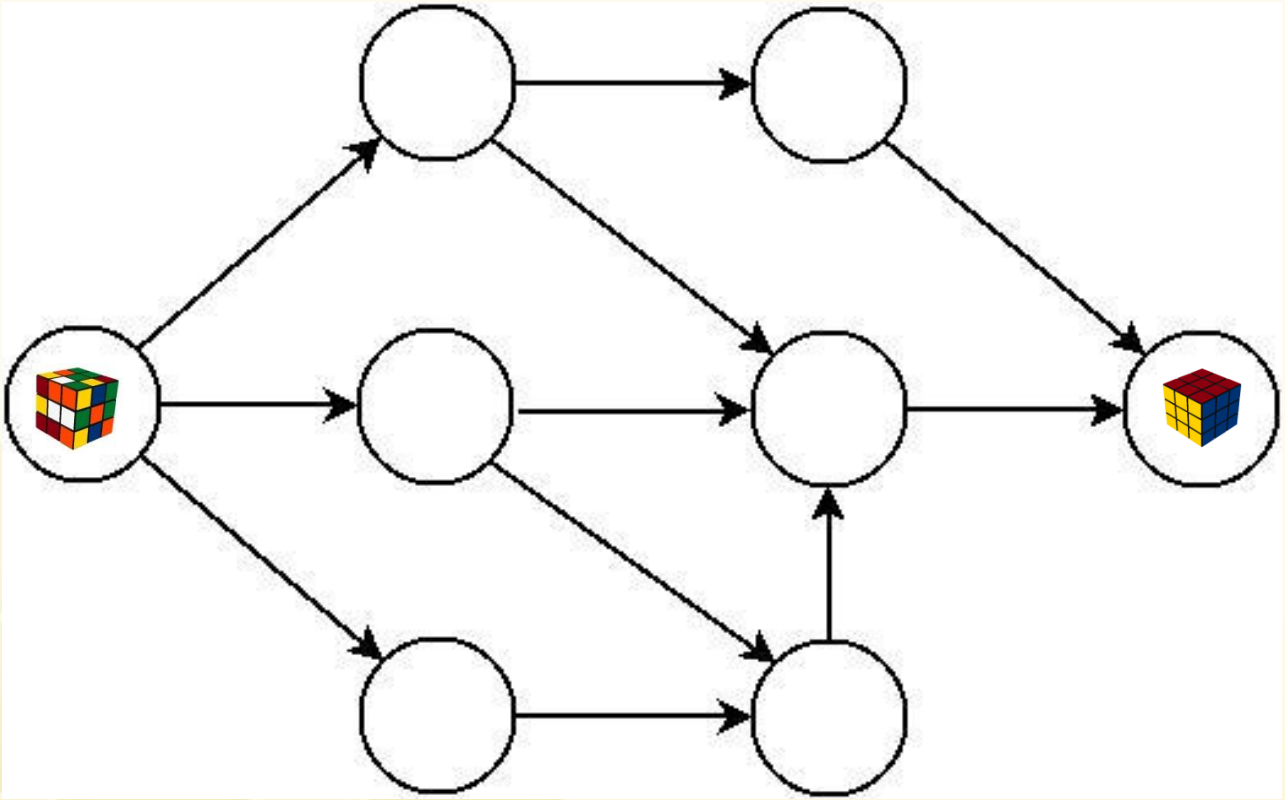
- Parameter
  - Jumlah Lebah
  - Gerakan Konstruktif Dalam 1 FP
  - Jumlah iterasi
- Tahapan
  - Forward Pass
  - Backward Pass

Pseudocode Algoritma BCO

```
begin
   Every bee is set to an empty solution;
   while (stopping criterion not met) do
      foreach (bee) do
         k ← 1; //counter for constructive moves in the forward pass
         while (k ≤ NC) do
            Evaluate all possible constructive moves;
            According to evaluation, choose one move using Roulette wheel;
            k ← k+1;
         end while;
        Sort bees by their objective function value;
        foreach (bee) do
           Choose to be recruiter or follower, according to objective function value;
        foreach (follower)do
           Choose a new solution from recruiters using roulette wheel;
   end while;
   Output the best result;
end.
```

Analisis

- Langkah Solusi Terpendek Rubik's Cube – Jalur Terpendek pada graf berarah.

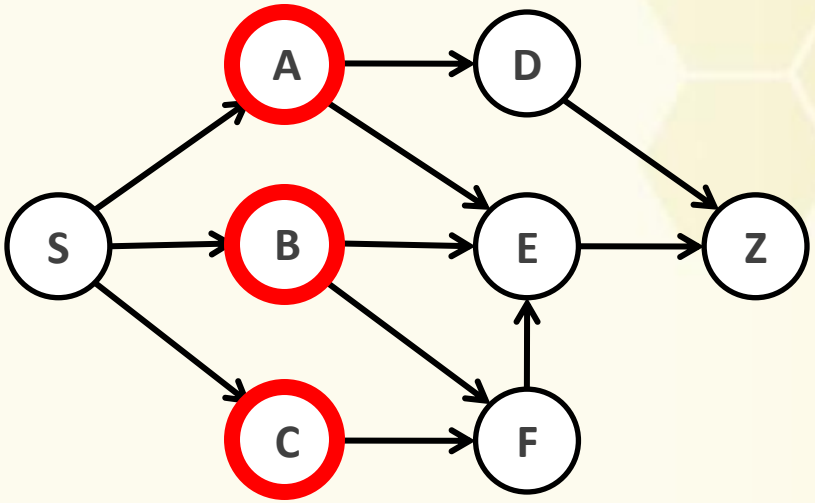


Forward Pass ke 1

Simpul *i*	*Li*	*Pfi*	*Pi*
A	1.20	0.8333	0.4386
B	1.5	0.6667	0.3509
C	2.5	0.40	0.2105
Sum		1.90	1

$$Pi = \frac{Pfi}{\sum_{i=1}^{N} Pfi}$$

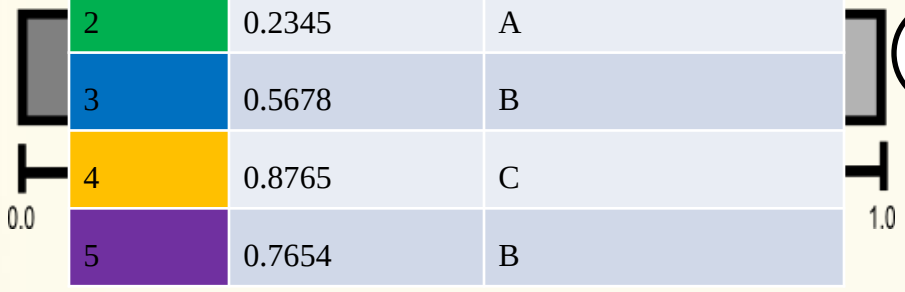
Parameter:
- Jumlah Jenis Moves = 1
- Constructive Moves = 1

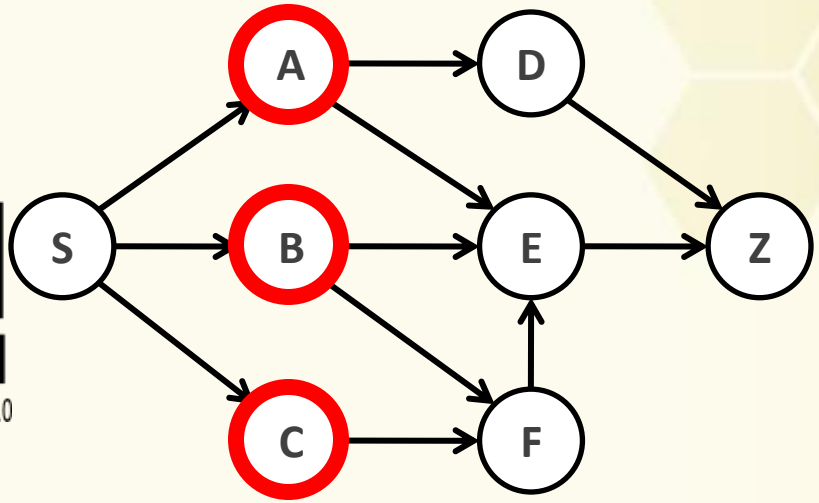


n	S	A	B	C	D	E	F	Z
h(n)	2.5	1.20	1.5	2.5	1.0	1.0	1.5	0.0

Roulette Wheel

Lebah *i*	Bilangan acak	Simpul yang dipilih
1	0.1234	A
2	0.2345	A
3	0.5678	B
4	0.8765	C
5	0.7654	B

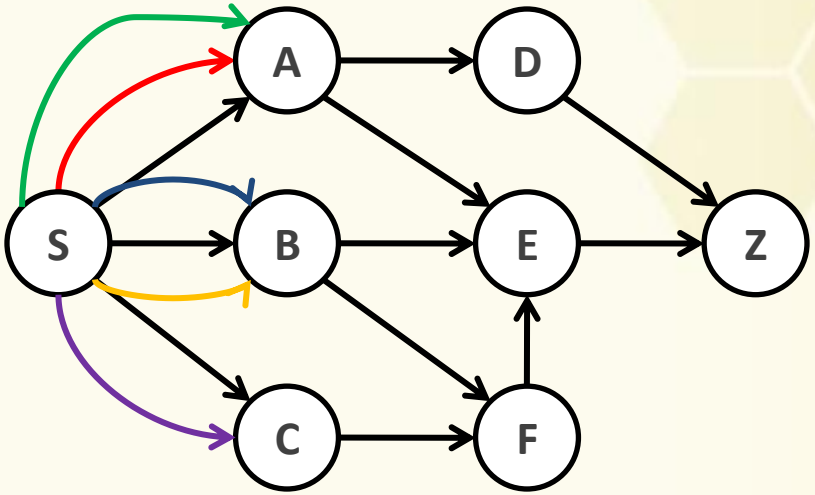




n	S	A	B	C	D	E	F	Z
P*fi*	-	0.8333	0.6667	0.40	-	-	-	-
P*i*	-	0.4386	0.3509	0.2105	-	-	-	-

Akhir *Forward Pass* ke 1

Lebah *i*	Bilangan acak	Simpul yang dipilih
1	0.1234	A
2	0.2345	A
3	0.5678	B
4	0.8765	C
5	0.7654	B



n	S	A	B	C	D	E	F	Z
P*fi*	-	0.8333	0.6667	0.40	-	-	-	-
P*i*	-	0.4386	0.3509	0.2105	-	-	-	-

Backward Pass ke 1

Pf_{colony} = (*Pf*1 + *Pf*2+ *Pf*3+ *Pf*4+ *Pf*5)/5

Pf_{colony} = (0.8333+0.8333+0.6667+0.40+0.6667)

Pf_{colony} = 0.68

Profitability Scores	*Pfollow*
$Pfi < 0.90*Pf\text{colony}$	0.60
$0.90*Pf\text{colony} \leq Pfi < 0.95 * Pf\text{colony}$	0.20
$0.95*Pf\text{colony} \leq Pfi < 1.15 * Pf\text{colony}$	0.02
$1.15*Pf\text{colony} \leq Pfi$	0.00

n	S	A	B	C	D	E	F	Z
P*fi*	-	0.8333	0.6667	0.40	-	-	-	-

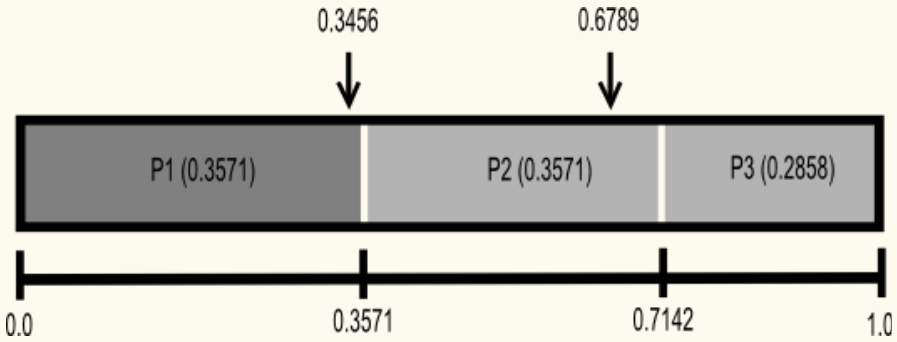


Seleksi *Recruiter/Follower*

Lebah *i*	Simpul	*Pfi*	Aturan *Profitability Score*	*Pfollow*	Random	Status
1	A	0.8333	**1.15****Pf*colony ≤ *Pf*i 0.782 ≤ P*fi*	0.00	0.6543	R
2	A	0.8333	**1.15****Pf*colony ≤ *Pf*i 0.782 ≤ *Pfi*	0.00	0.5678	R
3	B	0.6667	**0.95****Pf*colony ≤ *Pfi* < **1.15** * *Pf*colony 0.646 ≤ P*f*i < 0.782	0.02	0.4321	R
4	C	0.40	***Pfi* < 0.90****Pfcolony*** P*f*i < 0.612	0.60	0.2345	F
5	B	0.6667	**0.95****Pf*colony ≤ *Pfi* < **1.15** * *Pf*colony 0.646 ≤ P*f*i < 0.782	0.02	0.0123	F



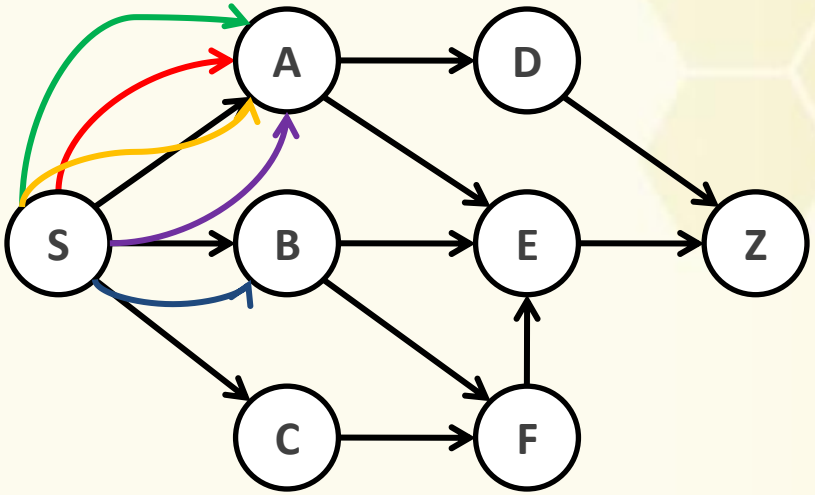
Roulette Wheel
Follower ke Recruiter



Lebah *i*	Simpul	*Pfi*	*Pi*
1	A	0.8333	0.3571
2	A	0.8333	0.3571
3	B	0.6667	0.2858
Sum		2.3333	1

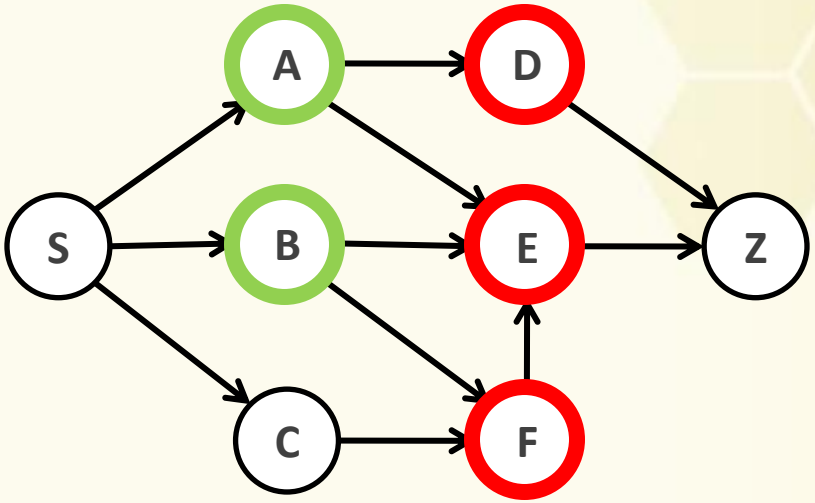
Akhir *Backward Pass* ke 1

Lebah *i*	Simpul yang dipilih
1	A
2	A
3	B
4	**A**
5	**A**



Forward Pass ke 2

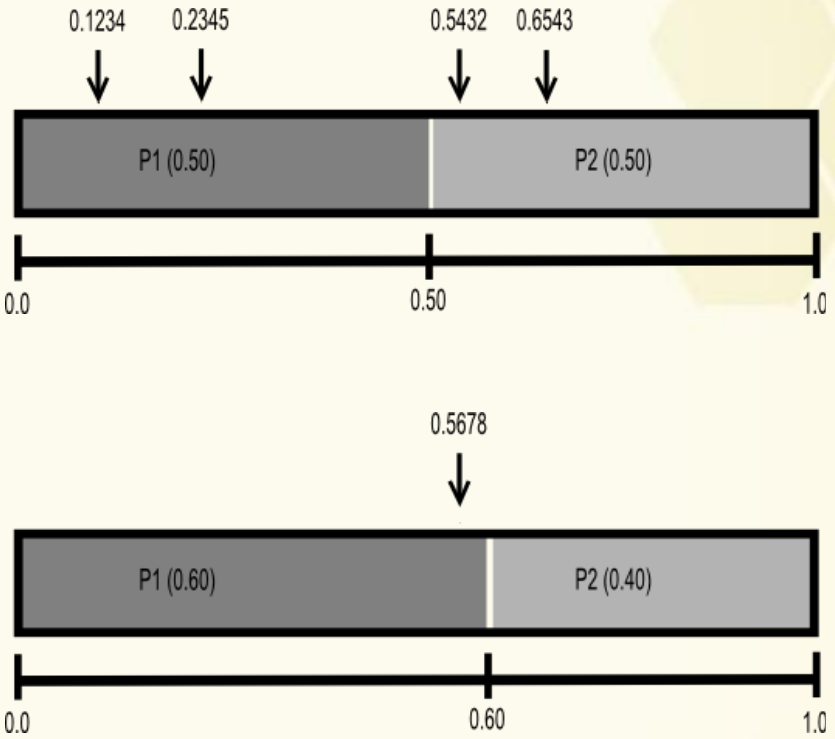
Simpul *i*	Lebah	Simpul Berikutnya	*Li*	*Pfi*	*Pi*
A	1,2,4,5	D	1.0	1	0.50
		E	1.0	1	0.50
Sum				2	1
B	3	E	1.0	1	0.60
		F	1.5	0.6667	0.40
Sum				1.6667	1



n	S	A	B	C	D	E	F	Z
h(n)	2.5	1.20	1.5	2.5	1.0	1.0	1.5	0.0

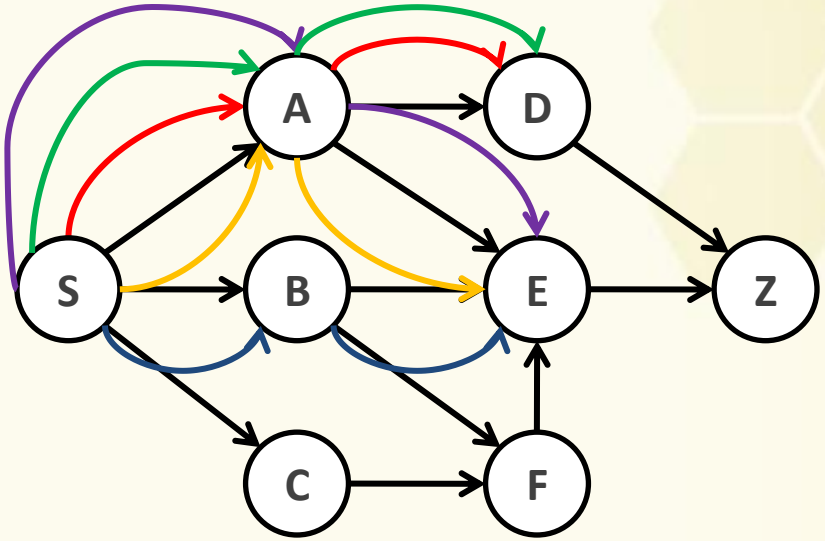
Roulette Distinct Nodes

Simpul *i*	Lebah	Simpul Berikutnya	*Li*	*Pfi*	*Pi*
A	1,2,4,5	D	1.0	1	0.50
		E	1.0	1	0.50
Sum				2	1
B	3	E	1.0	1	0.60
		F	1.5	0.6667	0.40
Sum				1.6667	1



Akhir *Forward Pass* ke 2

Lebah *i*	Bilangan acak	Simpul yang dipilih
1	0.1234	D
2	0.2345	D
3	0.5678	E
4	0.5432	E
5	0.6543	E



Backward Pass ke 2

Pf_{colony} = (*Pf*1 + *Pf*2+ *Pf*3+ *Pf*4+ *Pf*5)/5

Pf_{colony} = (1+1+1+1+1)/5

Pf_{colony} = 1

Profitability Scores	*Pfollow*
$Pfi < 0.90*Pf colony$	0.60
$0.90*Pf colony \leq Pfi < 0.95 * Pf colony$	0.20
$0.95*Pf colony \leq Pfi < 1.15 * Pf colony$	0.02
$1.15*Pf colony \leq Pfi$	0.00

n	S	A	B	C	D	E	F	Z
P*fi*	-	-	-	-	1	1	-	-

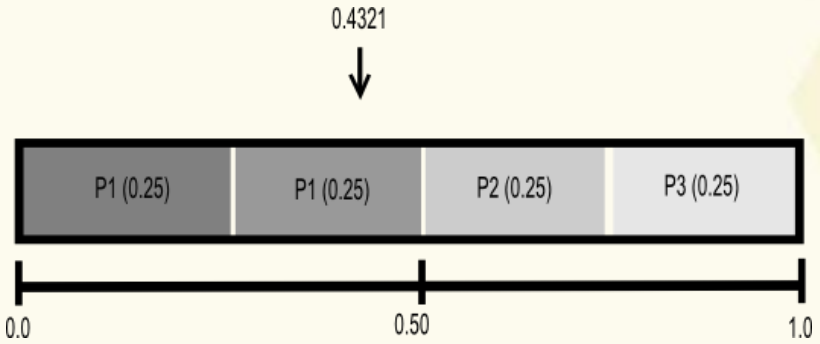
Seleksi *Recruiter/Follower*

Lebah *i*	Bilangan acak	Status
1	0.6543	R
2	0.5432	R
3	0.4321	R
4	0.2345	R
5	0.0123	F

Profitability Scores	*Pfollow*
$Pfi < 0.90*Pfcolony$	0.60
$0.90*Pfcolony \leq Pfi < 0.95 * Pfcolony$	0.20
$0.95*Pfcolony \leq Pfi < 1.15 * Pfcolony$	0.02
$1.15*Pfcolony \leq Pfi$	0.00

Roulette Wheel
Follower ke Recruiter

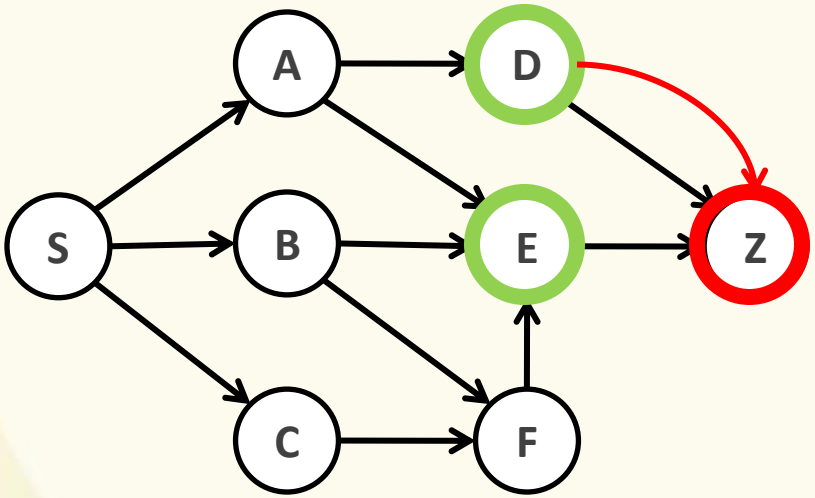
Lebah *i*	Bilangan acak	Status
1	0.6543	R
2	0.5432	R
3	0.4321	R
4	0.2345	R
5	0.0123	F



Akhir *Backward Pass* ke 2

Lebah *i*	Simpul yang dipilih
1	D
2	D
3	E
4	E
5	**D**

Forward Pass ke 3



Puzzle Rubik's Cube dengan *BCO*

- Heuristik
  - jumlah maksimum *3D Manhattan Distance* dari *corner cubies* dan *edge cubies* masing masing dibagi 4.
- Seleksi *Roulette Wheel*
  - Fungsi Objektif – Invers heuristik
- Gerakan Konstruktif = 1
- 12 Operator
  - U, U', D, D', F, F', R, R', L, L', B, dan B'.
  - Koloni bergerak bersama

Puzzle Rubik's Cube dengan *BCO*

- Setelah tahap forward pass pertama, tahap backward pass dan forward pass ke 2 diiterasikan sampai solusi ditemukan, atau bila nilai k melebihi 40 (*god's number*).
- Karena bersifat probabilistik maka terdapat kemungkinan solusi tidak ditemukan dalam sekali eksekusi sehingga keseluruhan algoritma dapat diiterasikan sebanyak jumlah iterasi yang ditentukan user.

Kebutuhan Non Fungsional

- Spesifikasi kebutuhan perangkat keras

Perangkat Keras	Spesifikasi
Prosesor	Prosesor 1.2 GHz
Resolusi Layar	1366 x 768 *Pixel*
Memory	*Memory* 500MB
Hard Disk	*Hard Disk* 20MB
Keyboard dan *Mouse*	-

Kebutuhan Non Fungsional

- Spesifikasi kebutuhan perangkat lunak

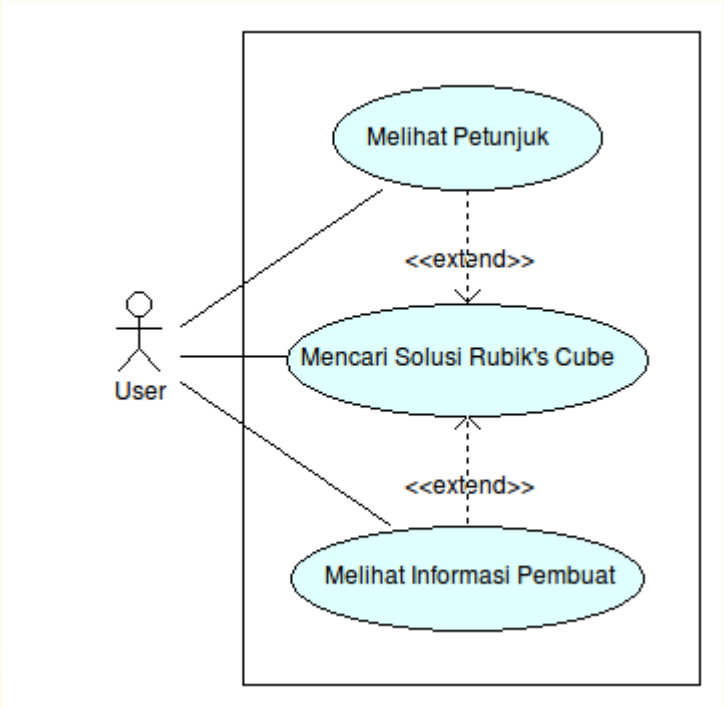
Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows XP, Windows 7, Ubuntu 10.4.
Java Runtime	JRE jre1.7.0_05

Kebutuhan Non Fungsional

- Spesifikasi kebutuhan pengguna
  - Peneliti yang dapat mengoperasikan komputer sehingga dapat menggunakan aplikasi yang akan dibangun.
  - Pengguna juga diharapkan mengerti gerakan-gerakan dasar yang diperlukan untuk memainkan Rubik's Cube

Kebutuhan Fungsional

- Use Case Diagram



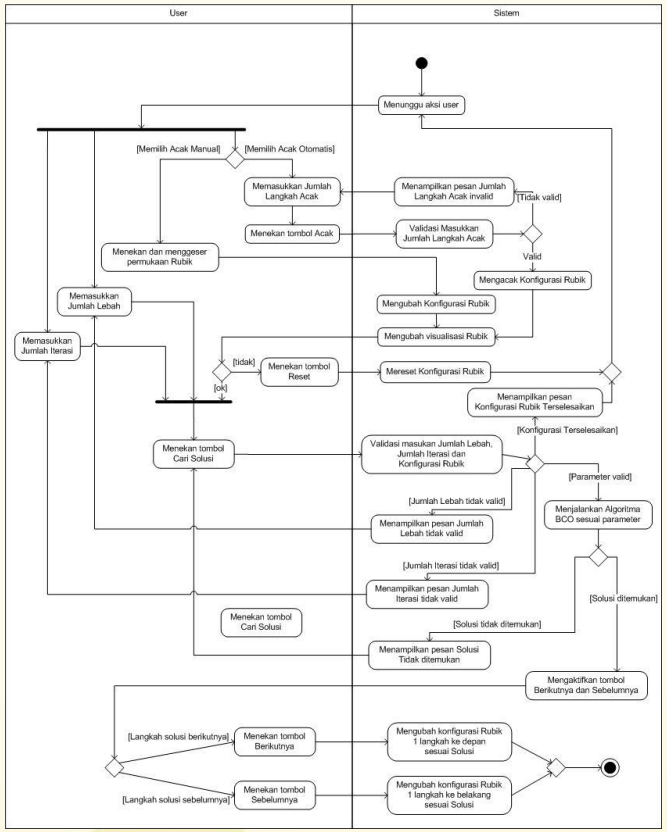
Kebutuhan Fungsional

- Definisi Use Case

Use Case	**Definisi**
Mencari Solusi Rubik's Cube	Proses berinteraksi dengan simulasi Rubik's Cube dan mencari solusinya
Melihat Petunjuk	Proses untuk melihat petunjuk penggunaan aplikasi
Melihat Informasi Pembuat	Proses untuk melihat informasi mengenai pembuat aplikasi

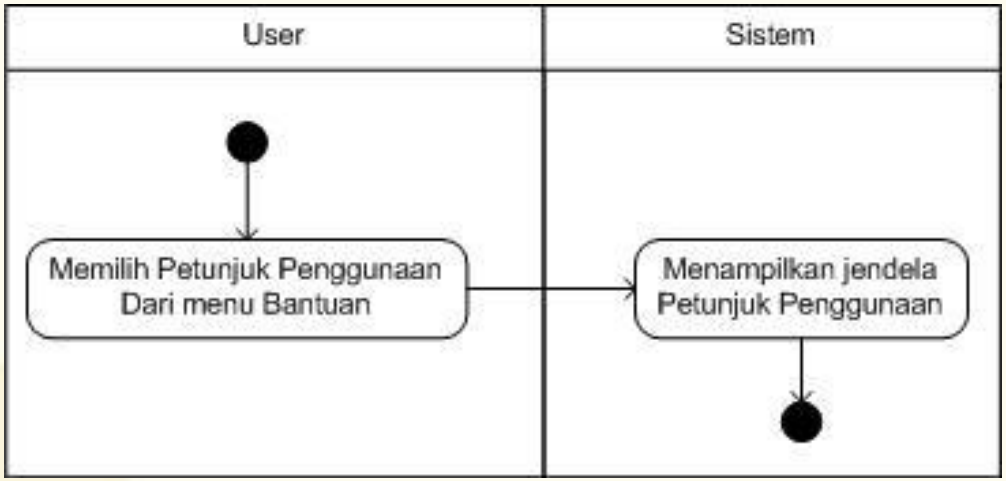
Kebutuhan Fungsional

- Activity Diagram Mencari Solusi Rubik's Cube



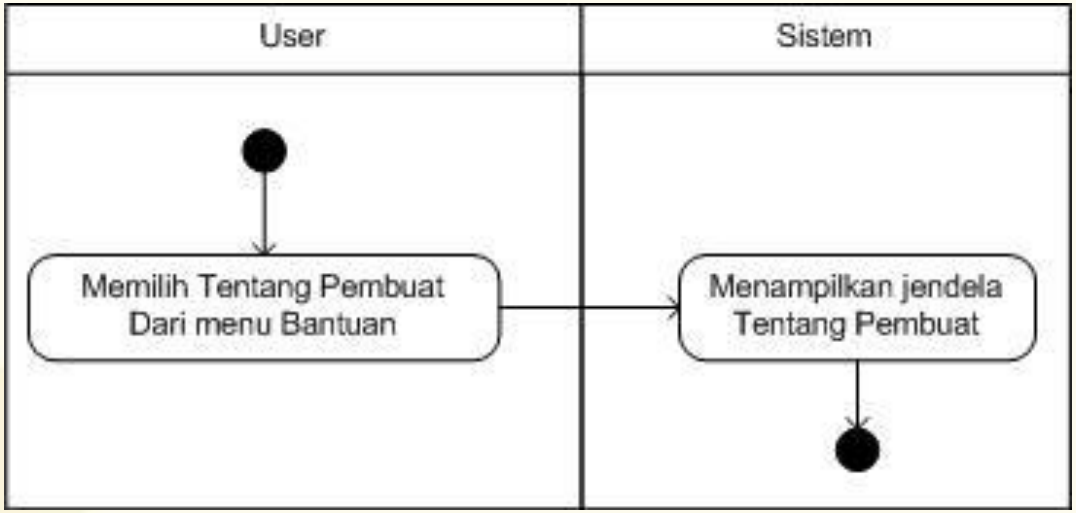
Kebutuhan Fungsional

- Activity Diagram Melihat Petunjuk



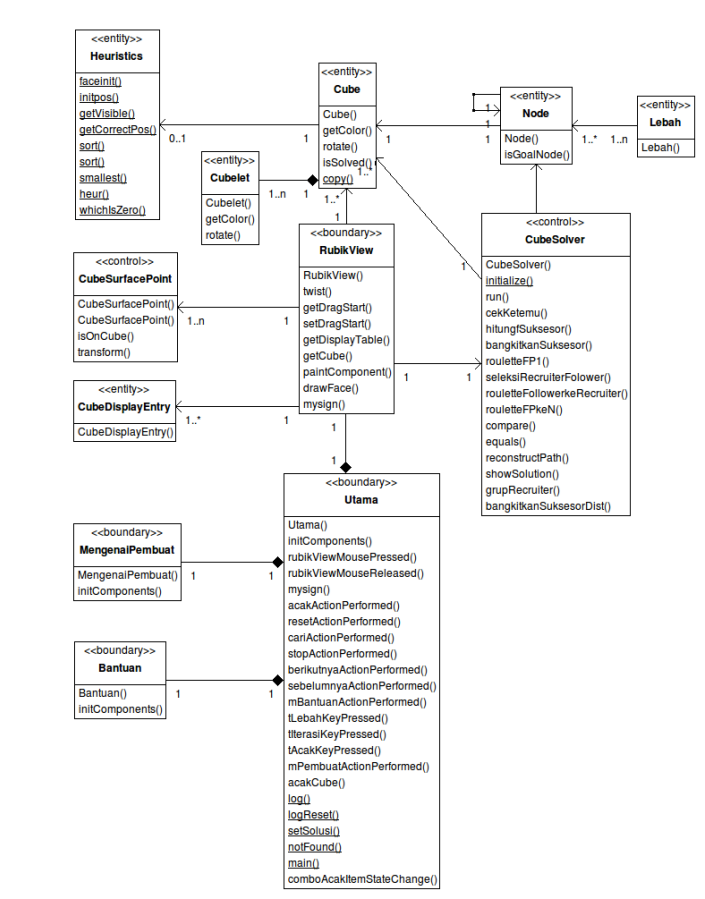
Kebutuhan Fungsional

- Activity Diagram Melihat Informasi Pembuat



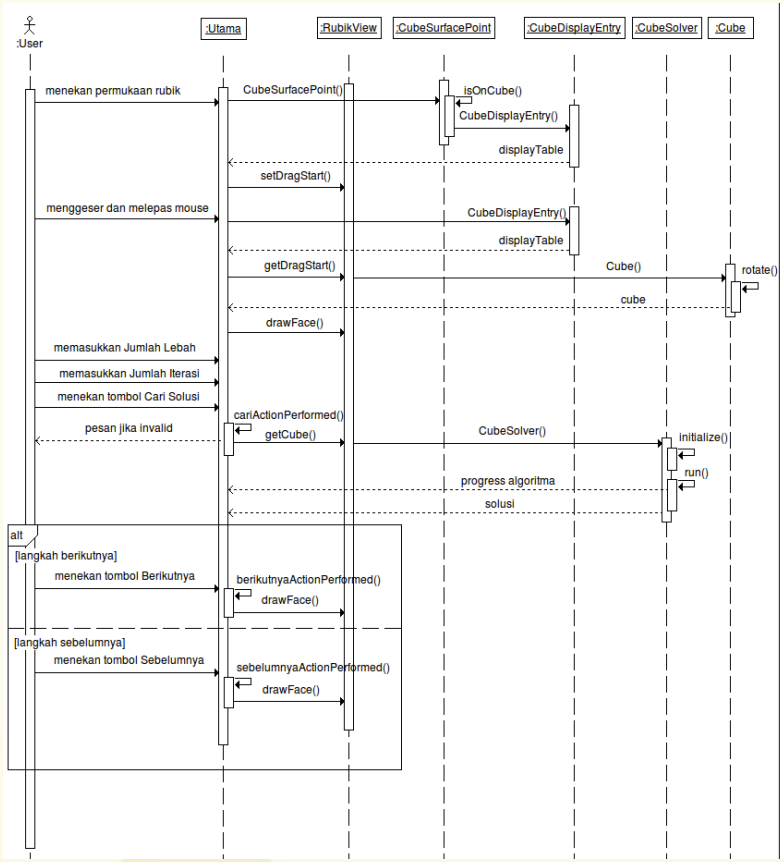
Kebutuhan Fungsional

- Class Diagram



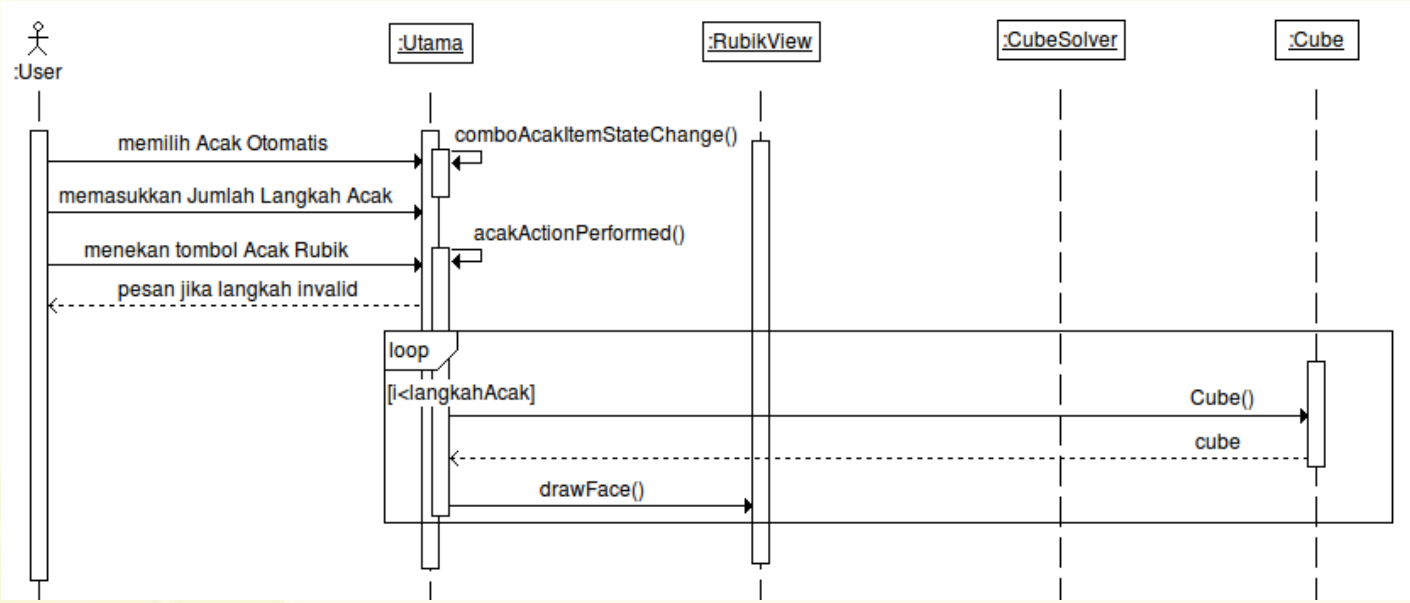
Kebutuhan Fungsional

- Sequence Diagram Mencari Solusi Rubik's Cube



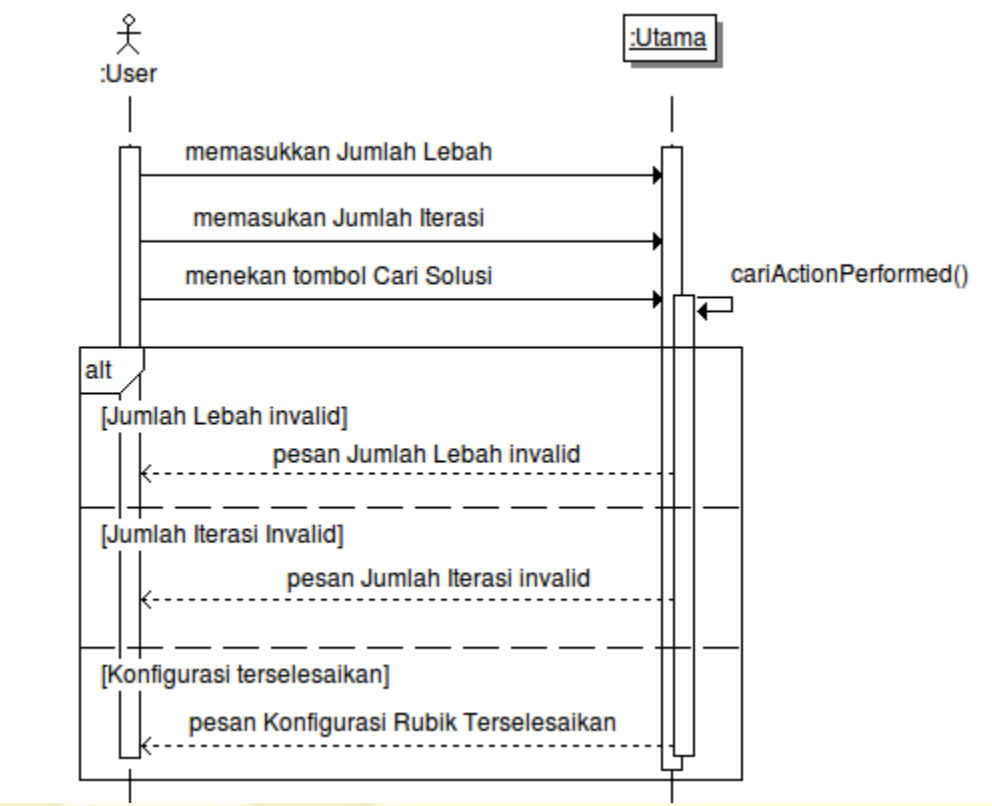
Kebutuhan Fungsional

- Sequence Diagram Memilih Acak Otomatis



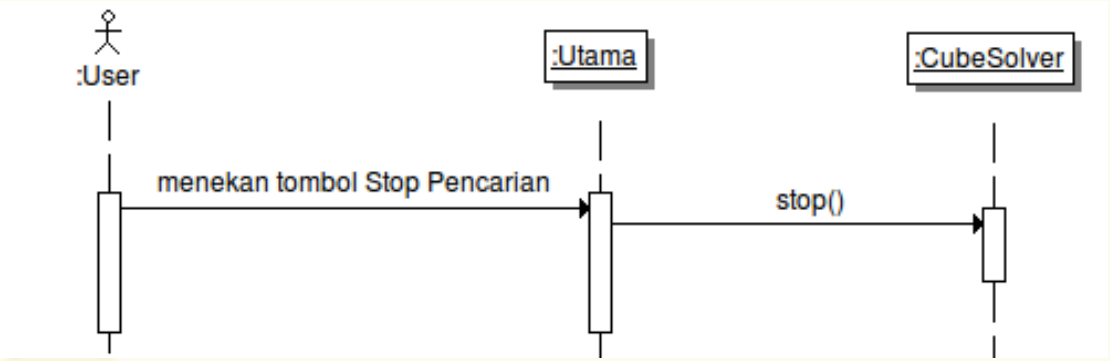
Kebutuhan Fungsional

- Sequence Diagram Parameter Invalid



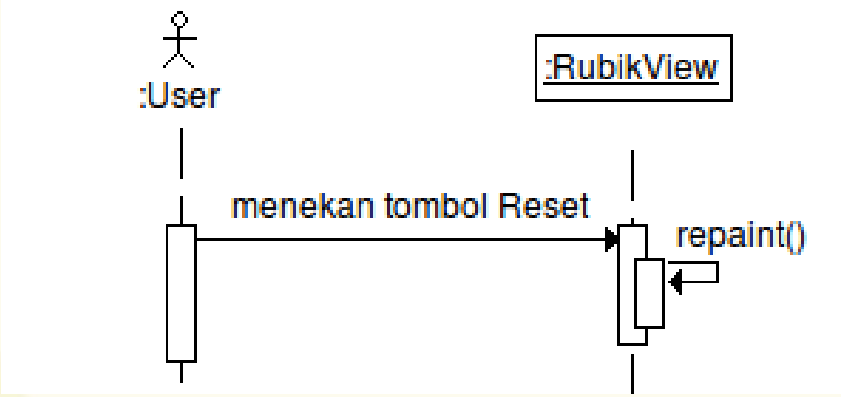
Kebutuhan Fungsional

- Sequence Diagram Menghentikan Pencarian



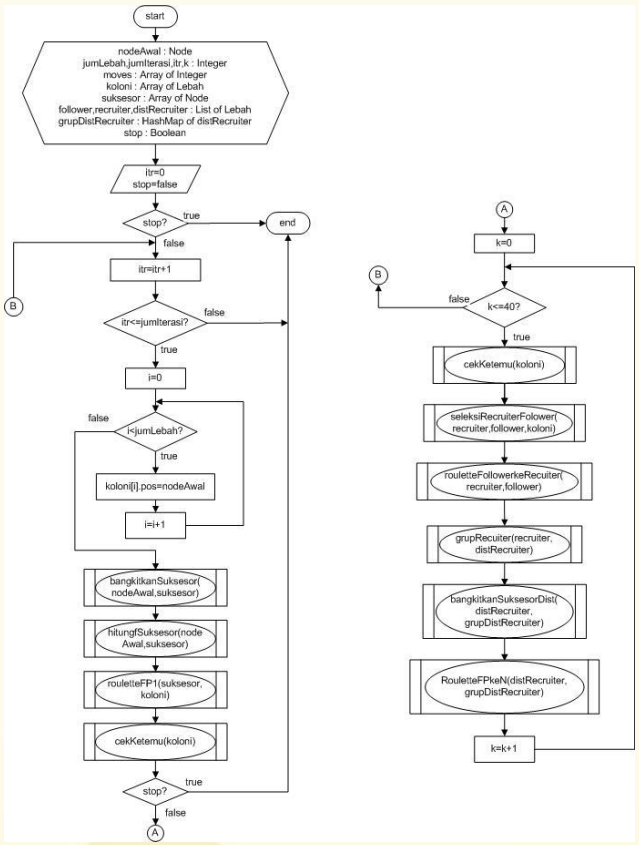
Kebutuhan Fungsional

- Sequence Diagram Mereset Konfigurasi



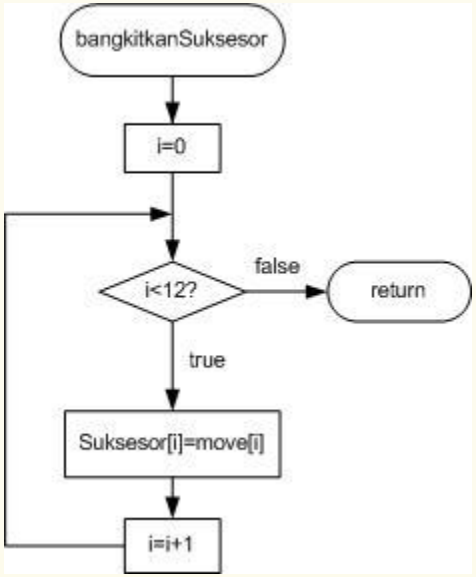
Perancangan Metode

- Flowchart Algoritma BCO untuk Rubik's Cube



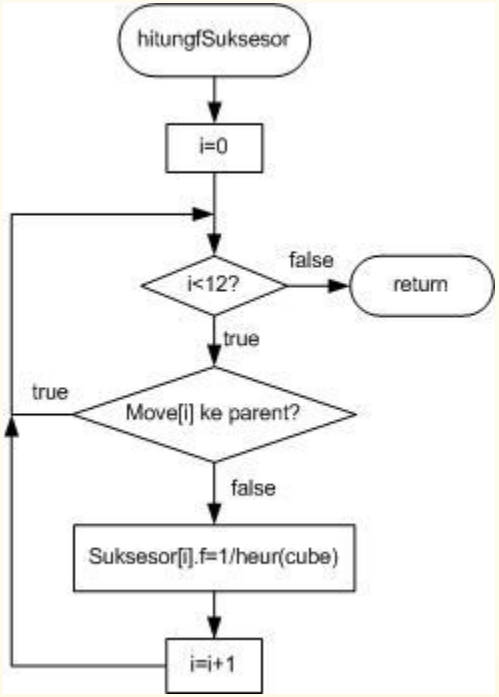
Perancangan Metode

- Flowchart Subrutin bangkitkanSuksesor



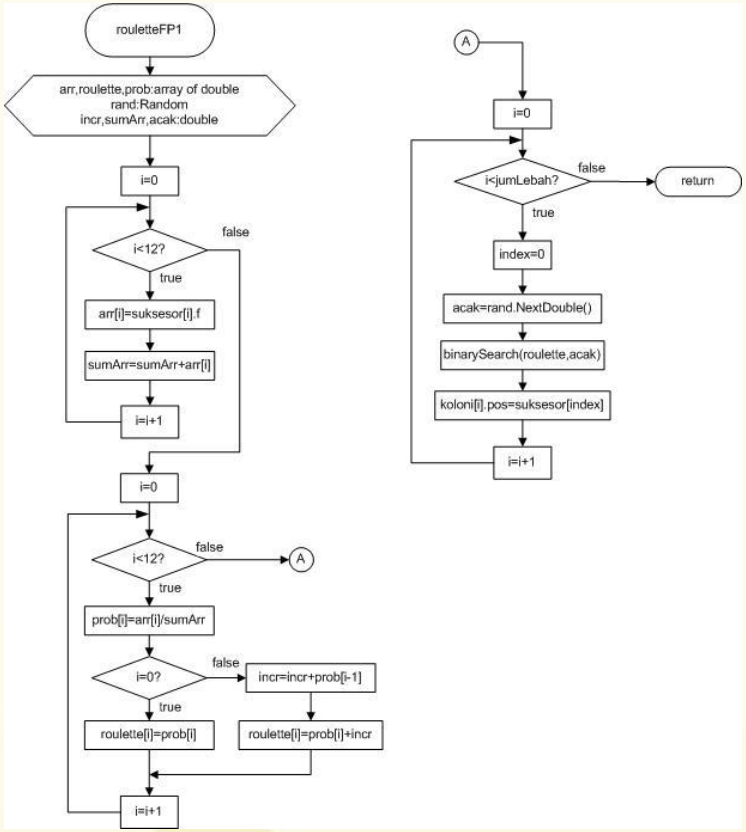
Perancangan Metode

- Flowchart Subrutin hitungfSuksesor



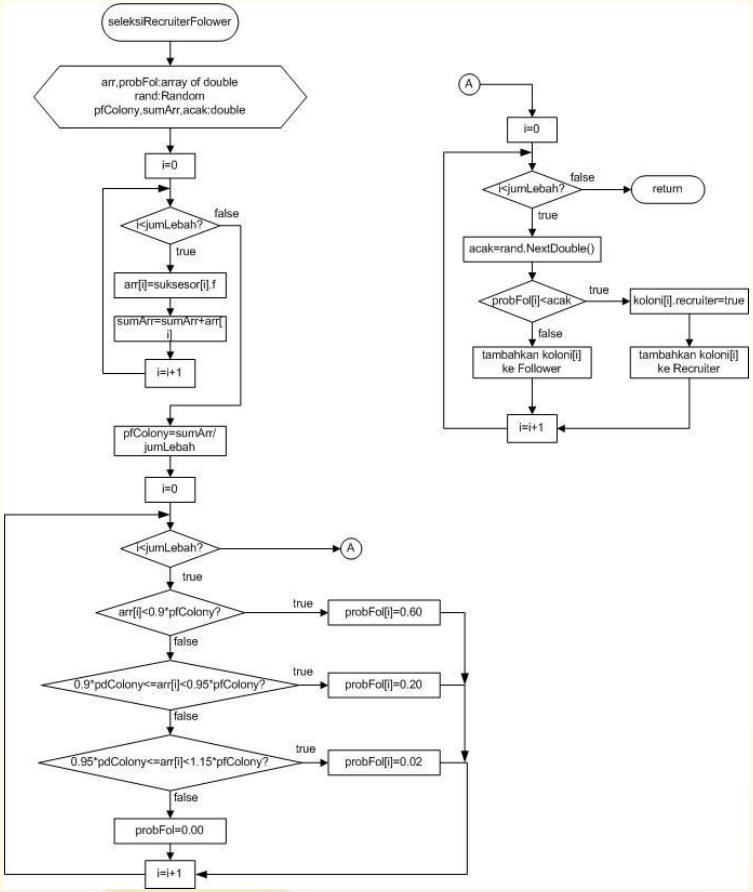
Perancangan Metode

- Flowchart Subrutin rouletteFP1



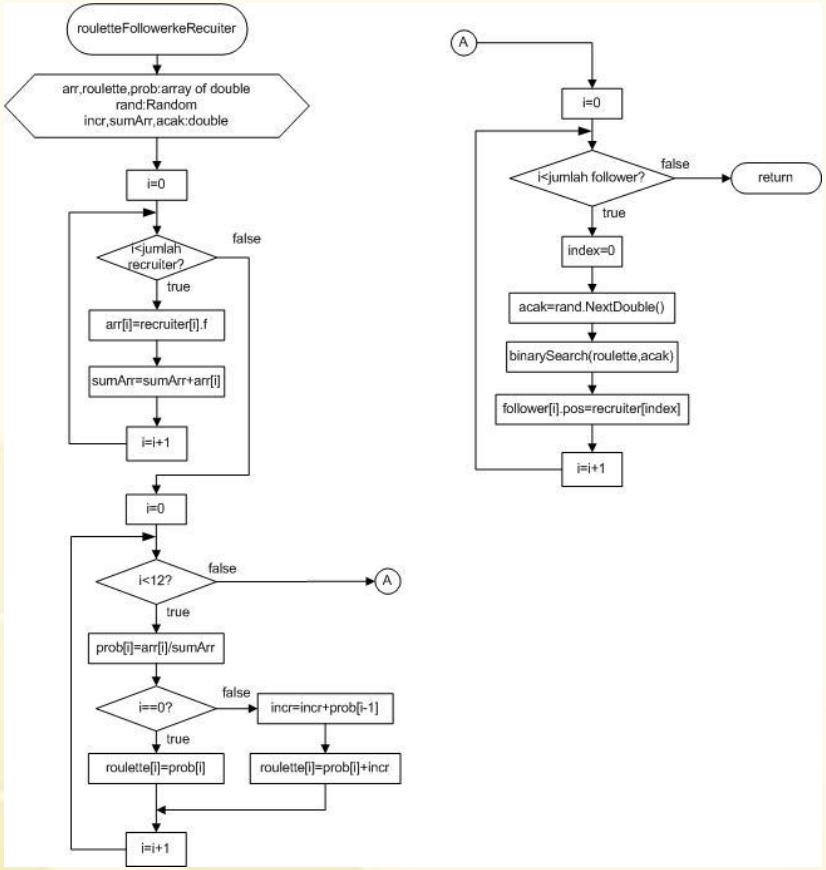
Perancangan Metode

- Flowchart Subrutin seleksiRecruiterFollower



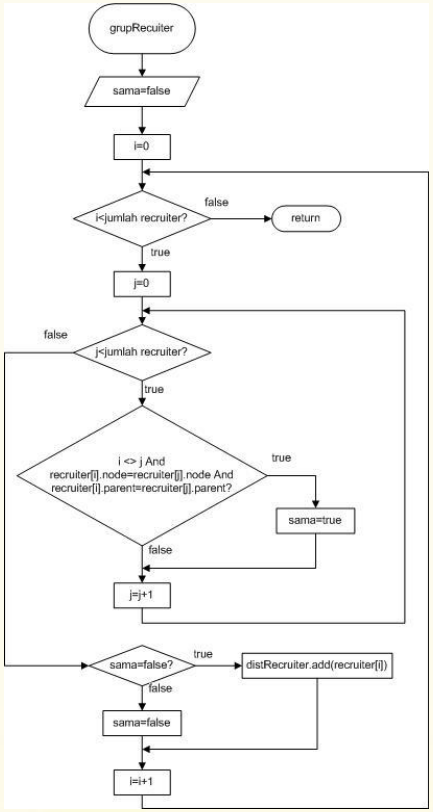
Perancangan Metode

- Flowchart Subrutin rouletteFollowerkeRecruiter



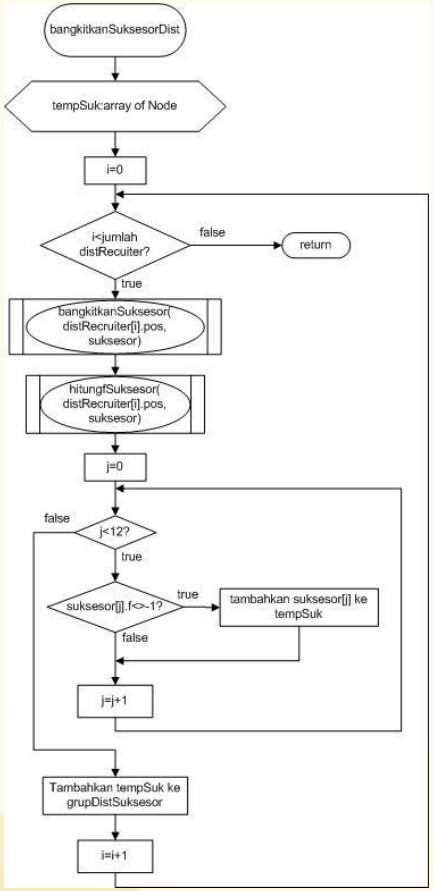
Perancangan Metode

- Flowchart Subrutin grupRecruiter



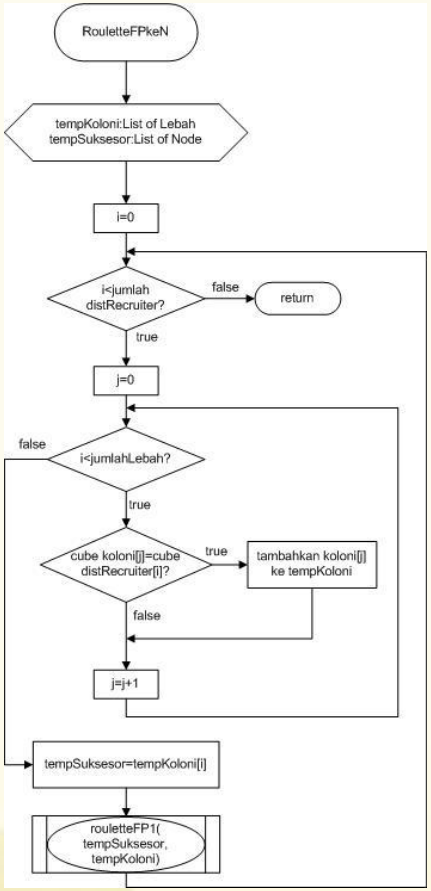
Perancangan Metode

- Flowchart Subrutin bangkitkanSuksesorDist



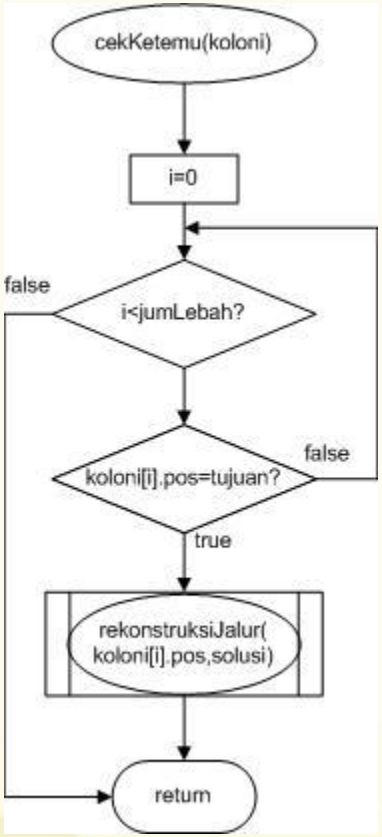
Perancangan Metode

- Flowchart Subrutin rouletteFPkeN



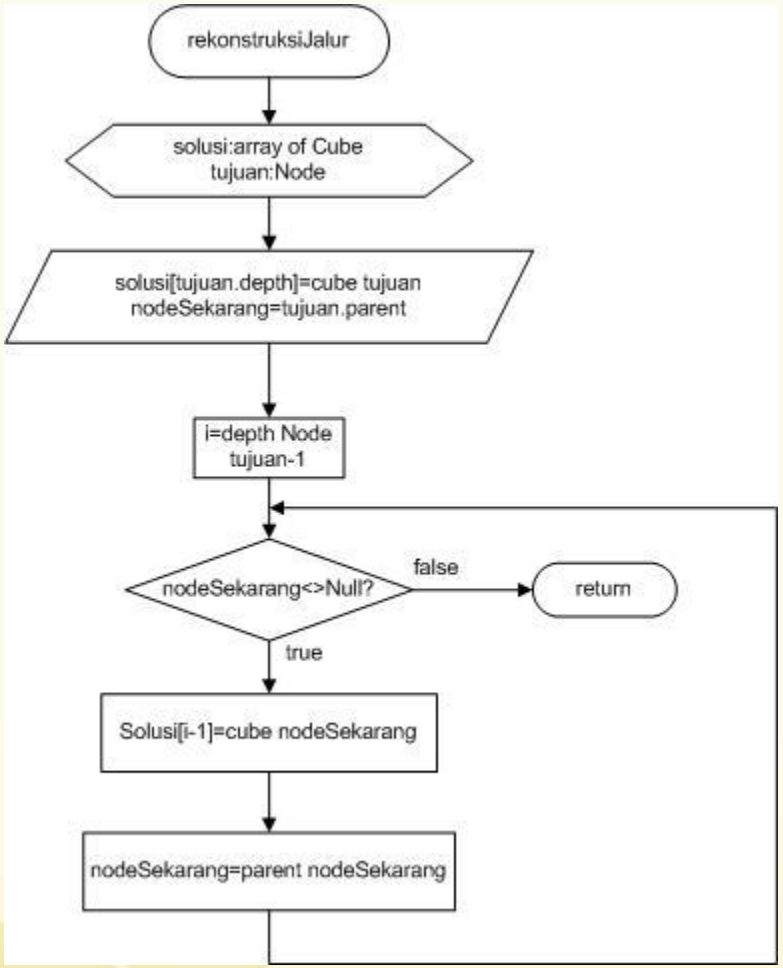
Perancangan Metode

- Flowchart Subrutin cekKetemu



Perancangan Metode

- Flowchart Subrutin rekonstruksiJalur



Perancangan Antarmuka

- Rubik's Cube Java Applet – Michael Schubart
- Antarmuka Form Utama
- Antarmuka Form Petunjuk
- Antarmuka Form Tentang Pembuat

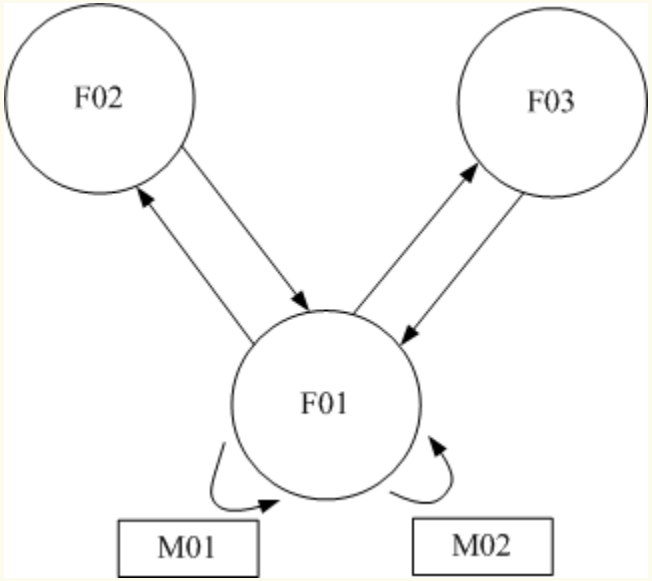


Perancangan Antarmuka

- Antarmuka Pesan Masukan Tidak Valid
- Antarmuka Pesan Konfigurasi Terselesaikan



Jaringan Semantik



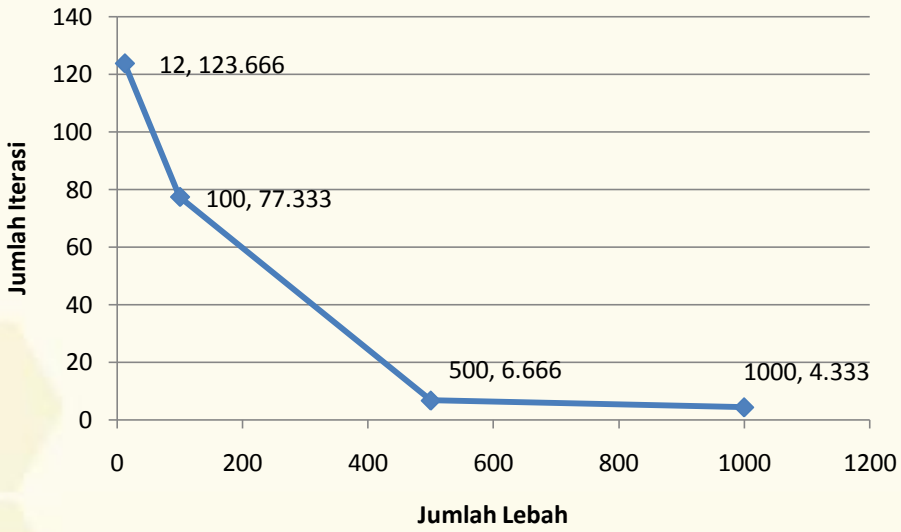
Pengujian Parameter Aplikasi

- Konfigurasi Rubik's Cube yang Diuji

No	Jumlah Langkah Acak	Langkah Acak Pembangkit
1	5	F F F L R
2	5	FB'DUL
3	6	U' F L' L' D L
4	7	B' U R' F' B R B
5	8	F F U L R' B' D R
6	10	F' B B' F' F' D D L U' D

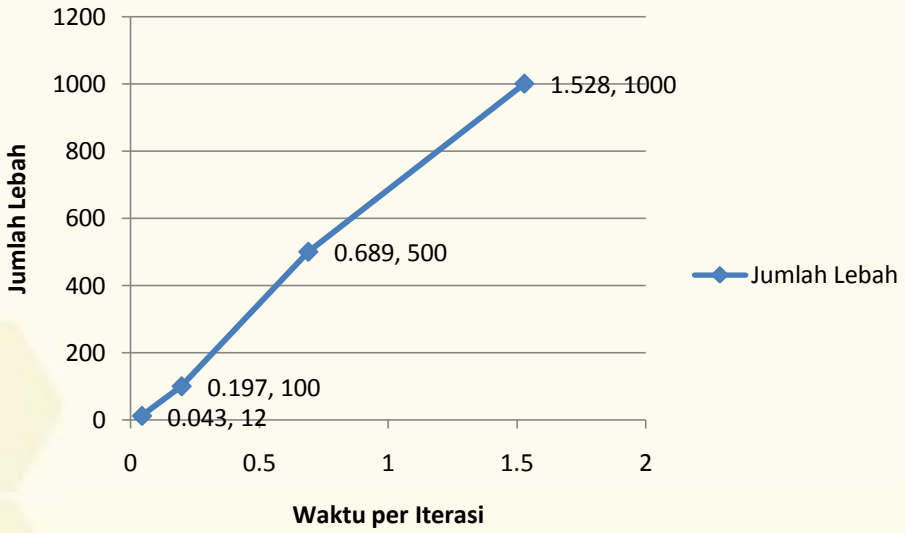
Pengujian Parameter Aplikasi

- Perbandingan Jumlah Lebah dengan Jumlah Iterasi



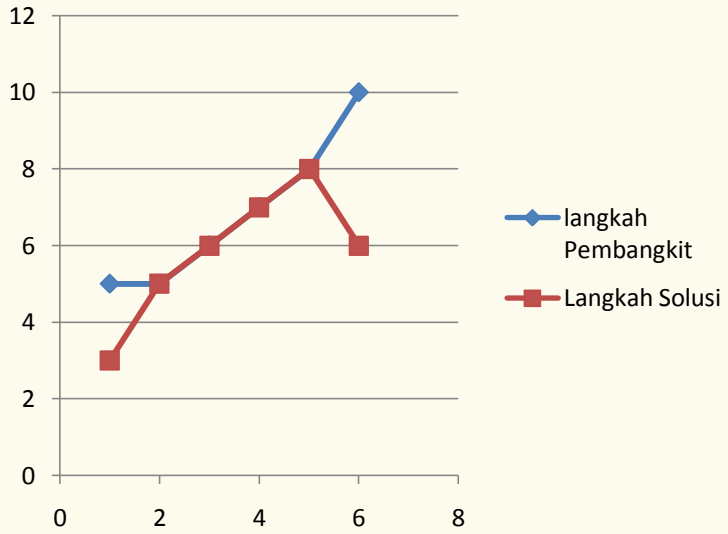
Pengujian Parameter Aplikasi

- Perbandingan Jumlah Lebah dengan Waktu Per Iterasi



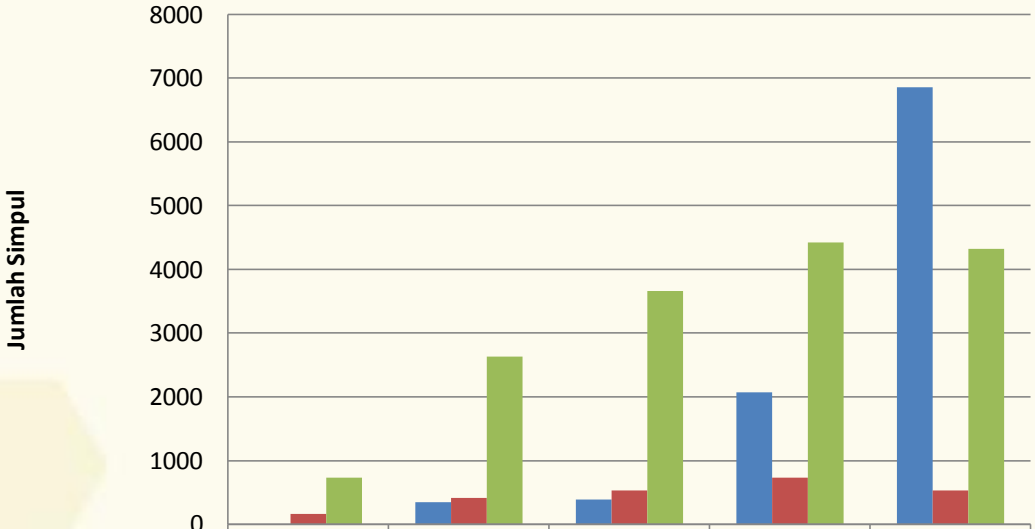
Pengujian Parameter Aplikasi

- Perbandingan Jumlah Langkah pembangkit dengan Jumlah Langkah solusi



Perbandingan dengan Algoritma lain

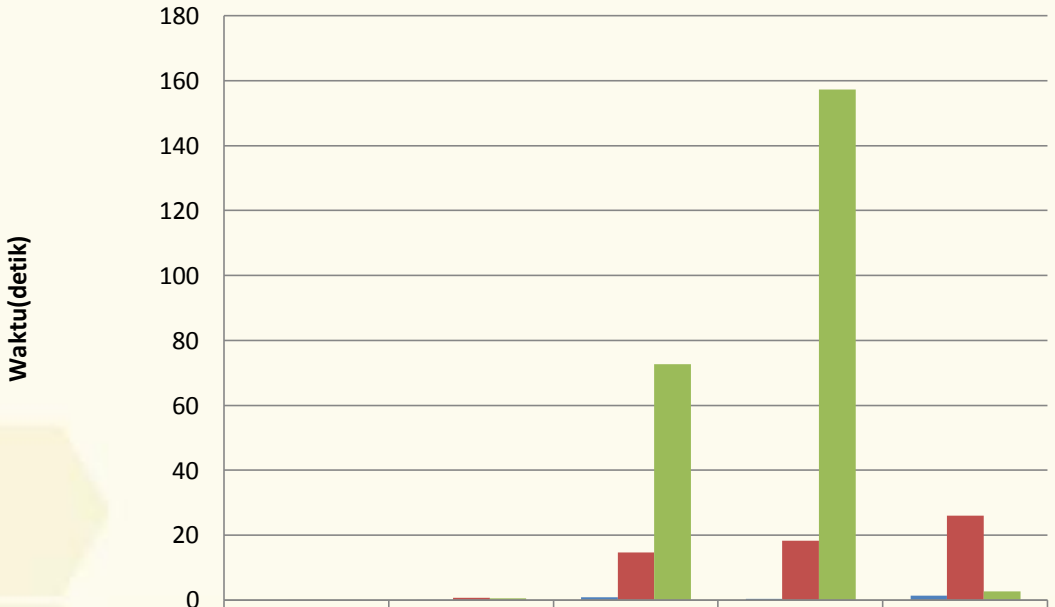
- Perbandingan Jumlah Simpul Yang ditelusuri



	1	2	3	4	5
IDA*	6	351	389	2076	6865
BCO (12 Lebah)	155	408	529	727	529
BCO (100 Lebah)	727	2630	3664	4423	4324

Perbandingan dengan Algoritma lain

- Perbandingan Waktu Pencarian



	1	2	3	4	5
IDA*	0	0.094	0.94	0.39	1.404
BCO(12 Lebah)	0.075	0.808	14.67	18.376	26.021
BCO(100 Lebah)	0.019	0.515	72.681	157.214	2.589

Kesimpulan

- Algoritma BCO yang diterapkan untuk mencari langkah terpendek pada puzzle Rubik's Cube berhasi menemukan langkah solusi terpendek
- Dari segi waktu, performansi Algoritma BCO lebih lama bila dibandingkan dengan algoritma lain, karena terdapat kemungkinan terjebak dalam *local optimum*. Akan tetapi dari segi ruang algoritma BCO lebih efisien bila dibandingkan dengan algoritma lain, karena jumlah simpul yang ditelusuri tergantung pada jumlah lebah.

Saran

- Mengganti fungsi heuristik untuk memperkirakan jarak dari suatu konfigurasi acak ke konfigurasi terselesaikan dengan fungsi selain 3D Manhattan Distance.
- Mencoba metode seleksi yang lain pada tahap *backward pass*.
- Menambahkan fitur untuk memasukkan konfigurasi Rubik's Cube yang sebenarnya diluar simulasi untuk membantu user lain selain peneliti untuk menyelesaikannya.

Terima Kasih

