PENYELESAIAN PUZZLE SUDOKU MENGGUNAKAN ALGORITMA GENETIKA

Randy Cahya Wihandika[1], Nur Rosyid Mubtada'i, S.Kom[2], Rizky Yuniar H, S.Kom, M.T[2]
Mahasiswa Jurusan Teknik Informatika, Dosen Jurusan Teknik Informatika
Jurusan Teknik Informatika
Politeknik Eletronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya 60111, Indonesia
Tel: +62 (31) 594 7280; Fax: +62 (31) 594 6114
E-mail: pens@eepis-its.edu or rendicahya@yahoo.com
Homepage: http://www.eepis-its.edu/

Abstrak

Sudoku adalah sebuah permainan puzzle pada papan berukuran 9x9 yang menggunakan angka 1 sampai 9. Target dari permainan ini adalah melengkapi papan dengan mengisi kotak-kotak yang kosong dengan sebuah angka sehingga dalam satu baris, kolom, dan region (kotak 3x3) tidak ada angka yang berulang. Karena itu, dalam sebuah puzzle Sudoku, hanya terdapat 1 solusi yang valid.

Penyelesaian permainan ini secara manual akan memakan waktu beberapa lama, sehingga mulai dilakukan penelitian untuk menyelesaikan puzzle Sudoku menggunakan beberapa algoritma yang dalam prosesnya diperlukan banyak iterasi.

Dengan Algoritma Genetika, suatu solusi dari puzzle Sudoku direpresentasikan dalam sebuah kromosom atau individu dengan struktur tertentu. Kromosom yang terkumpul dalam populasi mengalami berbagai proses, mulai dari seleksi, pindah silang, mutasi, hingga pergantian generasi. Kromosom yang lolos diharapkan adalah yang terbaik, yang merupakan solusi puzzle.

***Kata Kunci**: Puzzle, Sudoku, Algoritma Genetika*

Abstract

Sudoku is a puzzle game in a 9x9 board which uses the digits 1 through 9. The target of the game is to complete the board by filling in the empty cells with a number so that in one row, column, and region (3x3 cells) no numbers are repeated. Therefore, in a Sudoku puzzle, there is only one valid solution.

Completion of this game manually takes a while, until some researches try to solve the puzzle using multiple algorithms which process requires many iterations.

With Genetic Algorithm, a solution of the Sudoku puzzle is represented in a chromosome or an individual with a particular structure. Chromosomes in a population will pass through several processes: selection, crossover, and mutation. Chromosome which passes is expected the best, which is a solution to the puzzle.

***Keyword**: Puzzle, Sudoku, Genetic Algorithm*

1 PENDAHULUAN

1.1 Latar Belakang

Teknologi komputer terus mengalami kemajuan yang pesat. Perkembangan ini mendorong berkembangnya teknologi Kecerdasan Buatan (Artificial Intelligence, AI). Kecerdasan Buatan merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Teknologi tersebut telah diaplikasikan dalam banyak bidang di kehidupan nyata untuk memecahkan beberapa masalah, mulai dari yang sederhana hingga yang sangat rumit. Bidangnya pun beragam, mulai dari bidang kesehatan, bidang industri, bidang transportasi, hingga game.

Kecerdasan buatan dalam game adalah salah satu bidang penelitian yang menarik bahkan hingga saat ini. Teknologi ini telah berhasil diimplementasikan dalam berbagai jenis game. Salah satu tanda keberhasilannya adalah kalahnya Garry Kasparov, juara catur dunia, dari komputer Deep Blue.

Ada beberapa algoritma pada kecerdasan buatan, seperti Jaringan Saraf Tiruan (Neural Network, NN), Min-Max, dan Algoritma

Genetika (Genetic Algorithm, GA). Namun tidak semua algoritma-algoritma tersebut cocok untuk diaplikasikan untuk memecahkan suatu masalah.

Pada penelitian ini, akan dibangun sebuah sistem untuk penyelesaian game puzzle Sudoku menggunakan Algoritma Genetika. Sistem ini ditujukan untuk dapat menyelesaikan permainan-permainan Sudoku dengan semua tingkat kesulitan.

1.2 Tujuan

Tujuan dari penelitian ini adalah mengimplementasikan Algoritma Genetika dalam penyelesaian puzzle Sudoku hingga mendapat satu solusi puzzle yang valid.

1.3 Permasalahan

Berdasarkan uraian di atas, maka permasalahan yang timbul dalam pengerjaan penelitian ini antara lain:

1. Bagaimana membuat pembangkit yang dapat membuat puzzle Sudoku yang valid.
2. Bagaimana mengimplementasikan Algoritma Genetika secara efisien untuk menyelesaikan puzzle.
3. Bagaimana merancang kromosom yang dapat merepresentasikan suatu solusi puzzle.
4. Bagaimana membuat *fitness function* yang dapat mengukur kualitas suatu solusi puzzle.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini antara lain:

1. Puzzle sudoku yang digunakan adalah ukuran standard 9 x 9 kotak.

2 PERANCANGAN SISTEM

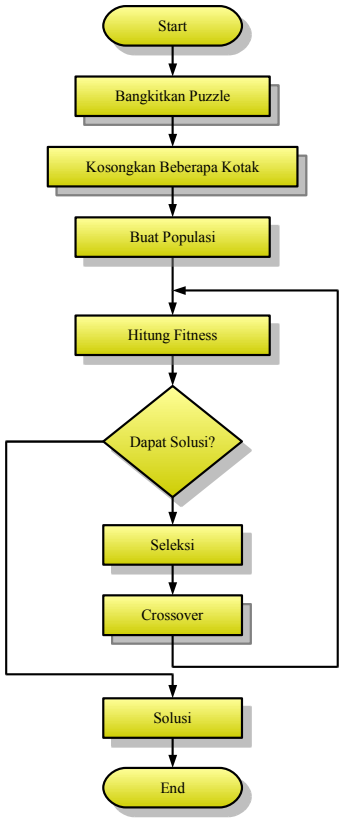
Pada bab ini akan dijelaskan mengenai perencanaan dan pembuatan perangkat lunak Penyelesaian Puzzle Sudoku menggunakan Algoritma Genetika, serta langkah-langkah alur program. Bahasa pemrograman yang digunakan pada perangkat lunak ini adalah Java.

2.1 Blog Diagram Sistem

Ada dua proses yang dilakukan oleh perangkat lunak ini, yaitu pembangkitan (generate) puzzle Sudoku dan penyelesaiannya menggunakan Algoritma Genetika. Proses yang pertama adalah proses membuat puzzle Sudoku yang valid, yang artinya memiliki solusi. Pengguna permainan kemudian dapat melakukan permainan untuk menyelesaikan puzzle tersebut.

Proses kedua adalah penyelesaian puzzle tersebut menggunakan Algoritma Genetika. Sebelum memasuki proses tersebut, terlebih dahulu dicari kandidat-kandidat solusi untuk masing-masing kotak puzzle. Hal ini bertujuan untuk mengeliminasi kandidat-kandidat solusi yang tidak mungkin. Contoh, jika suatu kotak telah terisi dengan angka 9, maka angka 9 tersebut dieliminasi dari kandidat pada kotak-kotak di baris, kolom, dan region yang sama. Dengan langkah tersebut, diharapkan Algoritma Genetika akan memiliki konvergensi yang baik. Solusi dari suatu puzzle diapatkan jika suatu kromosom memiliki nilai *fitness* yang sempurna, karena kromosom dengan nilai *fitness* di bawah nilai *fitness* sempurna berarti bukan solusi dari puzzle Sudoku.

Pada Algoritma Genetika, satu kromosom adalah satu kandidat solusi puzzle. Karena nilai yang mungkin pada solusi adalah antara 1 sampai dengan 9, maka representasi kromosom yang digunakan adalah deretan bilangan bulat (integer) dengan panjang kromosom sama dengan jumlah kotak yang kosong pada puzzle. Nilai *fitness* diambil dari panjang kromosom dikurangi dengan jumlah gen error. Gen error adalah gen yang berisi angka yang berulang pada baris, kolom, atau region. Metode seleksi yang digunakan adalah *Roulette-Wheel*.



Gambar 2.1 Blok Diagram Sistem

2.2 Pembangkitan Puzzle

Pembangkitan puzzle adalah proses pembuatan puzzle Sudoku yang valid. Puzzle Sudoku direpresentasikan dengan array dua

dimensi. Puzzle ini harus memenuhi aturan dalam permainan Sudoku, yaitu tidak boleh ada angka yang berulang pada baris, kolom, dan region. Puzzle ini kemudian dikosongkan beberapa kotaknya yang jumlahnya tergantung pada tingkat kesulitan permainan.

Tekniknya, kandidat-kandidat pada setiap cell direpresentasikan dengan sebuah List. Pada kondisi awal, list tersebut berisi angka 1 hingga 9. Karena papan berukuran 9x9, penyimpanan cell-cell tersebut dilakukan menggunakan array dua dimensi dengan ukuran 9x9. Kemudian looping dilakukan untuk semua cell tersebut, dari kiri ke kanan, atas ke bawah. Untuk masing-masing cell, diambil angka secara random dari kandidat-kandidat yang tersedia untuk cell tersebut. Kemudian angka yang terpilih tersebut dihapuskan dari kandidat pada cell-cell pada baris, kolom, dan region yang sama. Dengan teknik ini, satu puzzle Sudoku yang valid dapat dibangkitkan dalam waktu yang cukup cepat (< 1 detik).

Pseudocode:

1. Lakukan langkah 2 hingga 9 selama tidak terjadi error.
2. Buat array of `List` berukuran 9x9.
3. Isi setiap list dengan angka 1 hingga 9.
4. Lakukan langkah 5 hingga 9 untuk semua cell.
5. Ambil satu angka dari kandidat secara acak.
6. Hapus angka yang terpilih dari kandidat pada baris yang sama.
7. Hapus angka yang terpilih dari kandidat pada kolom yang sama.
8. Hapus angka yang terpilih dari kandidat pada region yang sama.
9. Jika kandidat pada cell kosong, ulangi dari langkah 2 (terjadi error).
10. Kosongkan secara random sejumlah cell sesuai dengan tingkat kesulitan.

3	4	5	2	8	7	6	1	9
6	2	1	5	9	4	8	3	7
7	8	9	3	1	6	2	5	4
8	6	7	1	3	5	4	9	2
2	1	4	9	6	8	5	7	3
9	5	3	4	7	2	1	8	6
4	9	2	7	5	1	3	6	8
1	7	8	6	2	3	9	4	5
5	3	6	8	4	9	7	2	1

Gambar 2.2 Puzzle Sudoku valid

2.3 Representasi Kromosom

Kromosom adalah representasi solusi dari suatu masalah pada Algoritma Genetika. Pada Sudoku, kromosom adalah representasi suatu solusi dari puzzle yang berisi angka-angka yang akan diisikan pada kotak-kotak yang kosong sehingga panjang kromosom adalah sama dengan jumlah kotak yang kosong pada puzzle. Nilai untuk masing-masing gen pada kromosom diambil secara acak dari kandidat-kandidat pada setiap cell. Kandidat nilai untuk setiap kromosom adalah angka 1 hingga 9. Pada pemrograman, kromosom direpresentasikan dengan *array of integer* (`int[]`).

4	7	8	9	4	3	6	1	7	4	6	5	2	5	6	9	9	7	4	7

Gambar 2.3 Contoh bentuk kromosom

2.4 Eliminasi Kandidat

Eliminasi kandidat dilakukan untuk menghilangkan kandidat-kandidat solusi yang tidak mungkin. Kandidat yang tidak mungkin adalah kandidat yang memiliki angka yang berulang pada baris, kolom, atau region yang sama.

Pseudocode:

1. Lakukan langkah 2 hingga 8 untuk semua cell kosong.
2. Buat `List`, isi dengan angka 1 hingga 9.
3. Lakukan langkah 3 untuk cell-cell berisi pada baris yang sama.
4. Hapus angka yang ditemui dari list.
5. Lakukan langkah 6 untuk cell-cell berisi pada kolom yang sama.
6. Hapus angka yang ditemui dari list.
7. Lakukan langkah 8 untuk cell-cell berisi pada region yang sama.
8. Hapus angka yang ditemui dari list.

2.5 Naked Single

Naked Single adalah salah satu teknik dalam penyelesaian puzzle Sudoku secara manual. Tujuan dari teknik ini adalah mencari cell-cell kosong yang hanya memiliki satu kandidat (kandidat tunggal) yang artinya cell tersebut hanya mungkin diisikan dengan angka tersebut.

Teknik ini biasanya dilakukan dengan menuliskan kandidat-kandidat setiap cell secara manual. Teknik ini merupakan langkah awal dalam penyelesaian karena teknik ini adalah teknik yang paling sederhana. Pada perangkat lunak ini, pencarian dilakukan dengan memanfaatkan hasil dari eliminasi kandidat. Cell-cell dengan kandidat tunggal akan diisi langsung dengan nilai kandidat tunggal tersebut.

Pseudocode:

1. Lakukan langkah 2 hingga untuk

semua cell kosong.
2. Hitung jumlah kandidat.
3. Jika jumlah kandidat sama dengan 1, isi cell dengan kandidat tersebut
4. Jika tidak, maka cell tersebut tidak dalam kondisi Naked Single.

8	**2**				**1**			
	7	**5**			2			
3	1	**9**		5	4	6	2	
		7	4		8		6	
	4		6		7		5	2
			2	9	5		4	
	6			2	3		8	6
	8		1			2		
			5		6			9

Gambar 2.4 Naked Single

2.6 Hidden Single

Hidden Single adalah teknik yang hampir sama dengan teknik Naked Single yaitu pencarian kandidat tunggal. Namun pada teknik ini, kandidat tunggal tersebut agak tersembunyi. Teknik ini dilakukan dengan mencari kandidat yang unik pada baris, kolom, atau region. Cell dengan hidden single memiliki lebih dari satu kandidat, namun salah satu kandidat tersebut adalah unik pada baris, kolom, atau region.

2.6.1 Row Hidden Single

Row Hidden Single adalah Hidden Single yang ditemukan pada baris. Indikasinya adalah ditemukannya kandidat yang unik pada suatu cell dalam satu baris.

Contoh:

x	5	1	8	**2**	7	9	**x**	3
		2						5
8				9				
	9		2					
3			7	6			**2**	4
	7		5			1	9	
				5	3	8	1	
1			9	7			5	2
5	4	9				6	3	7

Gambar 2.5 Row Hidden Single

Pada contoh di atas, ditemukan Row Hidden Single pada baris pertama. Di region pertama, sudah ada angka 2 di baris kedua sehingga cell kosong pada baris tidak mungkin diisi dengan angka 2. Kemudian pada baris pertama, kolom ke-8, sudah ada angka 2 pada baris ke-5 sehingga pada baris pertama, kolom ke-8 tidak bisa diisi dengan angka 2. Maka pada baris pertama, yang bisa diisi dengan angka 2 hanya pada kolom ke 5.

Pseudocode:

1. Lakukan langkah 2 hingga 5 untuk setiap baris.
2. Lakukan langkah 3 untuk setiap cell yang kosong pada baris tersebut.
3. Catat semua kandidat pada cell yang ditelusuri.
4. Cek apakah ada angka unik.
5. Jika ada, isikan angka unik tersebut ke cell yang bersangkutan.

2.6.2 Column Hidden Single

Column Hidden Single adalah Hidden Single yang ditemukan pada baris. Indikasinya adalah ditemukannya kandidat yang unik pada suatu cell dalam satu baris.

Contoh:

7	4	9		6			3	
2	**x**						**1**	5
5	8							
	x	2						
1	**x**							9
	x	4			1			
	2				6			4
4	1				3	2	5	6
9	6							1

Gambar 2.6 Column Hidden Single

Pada contoh di atas, ditemukan Column Hidden Single pada region ke-2, kolom ke-2 dengan nilai 1. Pada region 1, kolom ke-2, tidak ada cell kosong. Pada region ke-7, kolom ke-2, hanya ada 1 cell kosong, yaitu pada baris ke-8, sehingga angka 1 pada kolom ke-2 hanya dapat diisikan pada cell tersebut.

Pseudocode:

1. Lakukan langkah 2 hingga 5 untuk setiap kolom.
2. Lakukan langkah 3 untuk setiap cell yang kosong pada kolom tersebut.
3. Catat semua angka yang ditelusuri.
4. Cek apakah ada angka unik.
5. Jika ada, isikan angka unik tersebut ke cell yang bersangkutan.

2.6.3 Region Hidden Single

Column Hidden Single adalah Hidden Single yang ditemukan pada baris. Indikasinya adalah ditemukannya kandidat yang unik pada suatu cell dalam satu baris.

Contoh:

5	x	2	1	8	6	**3**		9
3	7	x		5		8		
x	x	x	**3**		7			2
	3					9		5
9	1							7
		8			5	1	3	
		7			1		5	
8		1			9		2	
4		**3**	5		8	7	9	

Gambar 2.7 Region Hidden Single

Pada contoh di atas, walaupun cell berwarna kuning memiliki kandidat 1, 3, dan 6, namun kandidat 3 pada region 1 hanya ditemukan pada cell tersebut. Artinya, cell-cell kosong yang lain pada region tersebut tidak ada yang memiliki kandidat 3. Dan karena pada satu region harus ada minimal satu angka 3, maka cell tersebut dapat langsung diisi dengan angka 3.

Pseudocode:

1. Lakukan langkah 2 hingga 5 untuk setiap region.
2. Lakukan langkah 3 untuk setiap cell yang kosong pada region tersebut.
3. Catat semua angka yang ditelusuri.
4. Cek apakah ada angka unik.
5. Jika ada, isikan angka unik tersebut ke cell yang bersangkutan.

2.7 Fungsi Fitness

Untuk menghitung nilai *fitness* masing-masing kromosom, terlebih dahulu dihitung jumlah angka yang berulang pada baris, kolom, dan region pada puzzle setelah angka-angka pada kromosom diisikan pada kotak-kotak yang kosong pada puzzle. Fungsi *fitness* yang digunakan cukup sederhana, yaitu adalah pengurangan panjang kromosom dengan jumlah angka yang berulang sehingga kromosom yang merupakan solusi puzzle adalah kromosom yang mempunyai nilai *fitness* sebesar panjang kromosom.

Contoh:

Untuk puzzle di bawah ini:

3	4			8	7		1	9
6		1		9		8	3	7
7	8		3	1	6		5	4
8	6	7			5		9	2
2		4	9		8	5		3
9	5			7	2	1	8	6
4		2	7		1		6	8
1		8	6	2	3	9	4	5
5	3	6	8	4	9	7	2	1

Jika diisikan kromosom:

5	2	2	2	2	4	9	2	1	3	4	1	6	7	3	4	7	5	3	7

Maka akan didapatkan:

3	4	**5**	2	8	7	2	1	9
6	2	1	2	9	**4**	8	3	7
7	8	**9**	3	1	6	2	5	4
8	6	7	**1**	**3**	5	**4**	9	2
2	**1**	4	9	**6**	8	5	**7**	3
9	5	**3**	**4**	7	2	1	8	6
4	**7**	2	7	**5**	1	**3**	6	8
1	**1**	8	6	2	3	9	4	5
5	3	6	8	4	9	7	2	1

Gambar 2.8 Kotak-kotak yang error pada puzzle

Angka berwarna merah adalah cell-cell yang error. Maka nilai *fitness* untuk kromosom tersebut adalah:

fitness = panjang kromosom – jumlah error
= 20 – 7
= 14

Solusi dari suatu puzzle Sudoku adalah kromosom dengan nilai *fitness* sempurna, yaitu sebanyak jumlah kotak yang kosong.

2.8 Seleksi

Metode seleksi yang digunakan adalah metode *Roulette-Wheel* yang membuat kromosom-kromosom dengan nilai *fitness* tinggi memiliki kemungkinan terpilih yang

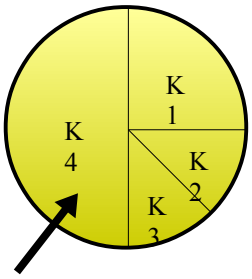
tinggi pula.

Probabilitas keterpilihan setiap kromosom ditentukan dengan cara membagi nilai *fitness* kromosom tersebut dengan total nilai *fitness*. Untuk itu, yang terlebih dahulu harus dihitung adalah total nilai *fitness*. Kemudian dilakukan pembagian masing-masing nilai *fitness* dengan total *fitness* tersebut. Nilai tersebut adalah probabilitas setiap kromosom yang berkisar antara nol dan satu.

Kemudian pada proses pemutaran roda *roulette*, teknisnya juga sederhana. Suatu bilangan acak dibangkitkan, dengan jangkauan nilai antara nol dan satu. Nilai acak tersebut kemudian dibandingkan dengan nilai probabilitas masing-masing kromosom. Pembandingan dilakukan dari kromosom yang terkecil. Kromosom yang memiliki nilai *fitness* yang lebih besar atau sama dengan nilai acak akan menjadi individu yang terpilih.

Tabel 2.1 Contoh probabilitas seleksi untuk setiap kromosom

Kromosom	Fitness	Probabilitas	Nilai Probabilitas
C1	2	1/4	0.25
C2	1	1/8	0.375
C3	1	1/8	0.5
C4	4	1/2	1
Total	**8**	**1**	**1**



Gambar 2.9 Ilustrasi metode seleksi Roulette-Wheel

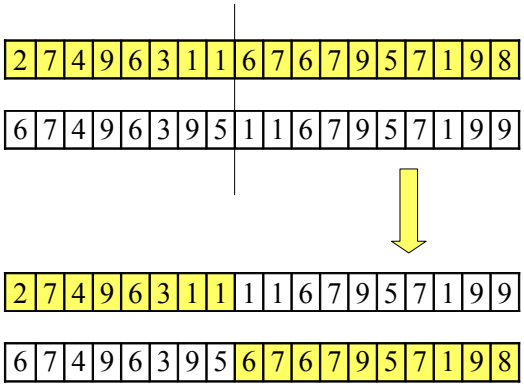
2.9 Elitisme

Elitisme dilakukan untuk mempertahankan individu terbaik karena ada kemungkinan bahwa individu terbaik itu tidak terpilih pada proses seleksi. Jika itu terjadi, maka bisa saja Algoritma Genetika kehilangan konvergensinya.

Implementasi mekanisme elitisme pada pemrograman cukup sederhana: pilih dua individu dengan nilai *fitness* tertinggi.

2.10 Pindah Silang

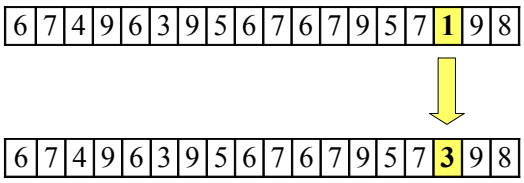
Metode pindah silang yang digunakan adalah pindah silang dengan satu titik. Pertukaran gen kemudian dilakukan pada kedua kromosom mulai titik gen tertentu sampai akhir kromosom.



Gambar 2.10 Crossover single point

2.11 Mutasi

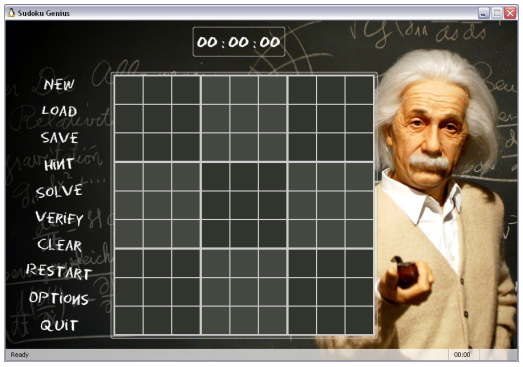
Mutasi dilakukan satu titik, yaitu dengan mengganti nilai pada gen tertentu dengan nilai yang lain yang mungkin diisikan pada kotak yang bersesuaian.



Gambar 2.11 Mutasi single point

3 PEMBUATAN PROGRAM

Program dibuat menggunakan bahasa pemrograman Java Standard Edition. Tampilan perangkat lunak adalah seperti di bawah ini. Tema tampilan dibuat custom agar menarik pengguna perangkat lunak.



Gambar 3.1 Tampilan perangkat lunak

4 UJI COBA DAN ANALISA

Setelah perencanaan dan pembuatan sistem, maka langkah selanjutnya yaitu melakukan pengujian terhadap sistem yang telah dibuat. Pengujian dilakukan untuk mengetahui apakah sistem yang dibangun dapat dijalankan sesuai dengan yang diinginkan. Pada bab ini, pengujian difokuskan pada implementasi Algoritma Genetika untuk menyelesaikan suatu puzzle.

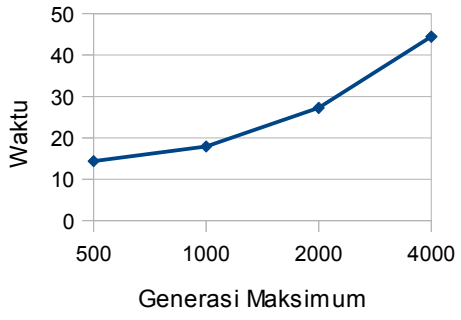
Puzzle yang digunakan untuk percobaan adalah puzzle dengan tingkat kesulitan Very Hard (55 kotak kosong). Puzzle ini disimpan ke dalam file dan di-load pada setiap percobaan. Percobaan yang dilakukan adalah dengan mengubah-ubah konfigurasi Algoritma Genetika, yaitu jumlah individu, generasi maksimum, *crossover rate*, jumlah titik crossover, dan *mutation rate*.

Untuk setiap konfigurasi, dilakukan 10 percobaan dan diambil nilai rata-ratanya. Hal ini dilakukan karena Algoritma Genetika memberikan hasil yang berbeda-beda untuk konfigurasi yang sama.

4.1 PERCOBAAN JUMLAH GENERASI MAKSIMUM

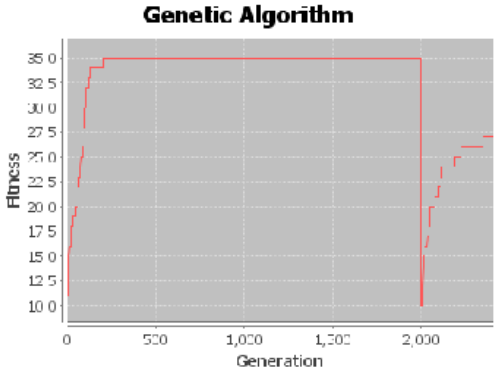
Percobaan ini dilakukan untuk mencari batas maksimum generasi yang optimal. Pada proses evolusi, jika jumlah generasi telah mencapai batas maksimum, maka populasi akan diinisialisasi kembali dan proses evolusi diulang dari awal. Hal ini bertujuan untuk menghindari proses evolusi yang stagnan, yang artinya tidak ada peningkatan nilai *fitness* pada populasi.

Grafik di bawah ini adalah grafik nilai rata-rata dari percobaan tersebut.



Gambar 4.1 Grafik pengaruh jumlah generasi maksimum terhadap waktu evolusi

Grafik di atas menunjukkan pengaruh jumlah generasi maksimum pada waktu evolusi. Penambahan jumlah generasi maksimum dapat menurunkan performa Algoritma Genetika. Hal itu membawa kepada analisa yang lain, seperti ditunjukkan oleh grafik berikut.



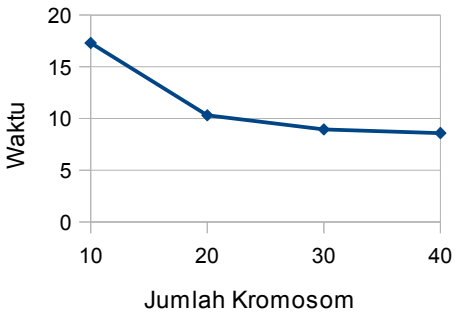
Gambar 4.2 Stagnasi pada proses evolusi

Grafik di atas menunjukkan peningkatan nilai *fitness* pada proses evolusi suatu populasi. Grafik tersebut juga menggambarkan nilai *fitness* yang tidak meningkat hingga generasi ke-2000. Ada kemungkinan nilai *fitness* tersebut akan meningkat pada generasi-generasi selanjutnya, namun akan lebih baik jika dilakukan inisialisasi populasi dari awal. Hal ini disebabkan karena inisialisasi populasi baru dapat membawa kepada solusi dengan lebih cepat. Dan penambahan jumlah generasi maksimum berarti memperpanjang stagnasi dan akan membuang waktu.

4.2 PERCOBAAN JUMLAH KROMOSOM

Percobaan jumlah kromosom dilakukan dengan menambah jumlah kromosom untuk dilihat pengaruh pada performa Algoritma Genetika.

Hasil percobaan digambarkan pada grafik berikut.



Gambar 4.3 Grafik pengaruh jumlah kromosom terhadap waktu evolusi

Grafik di atas menunjukkan pengaruh jumlah kromosom dalam populasi terhadap waktu

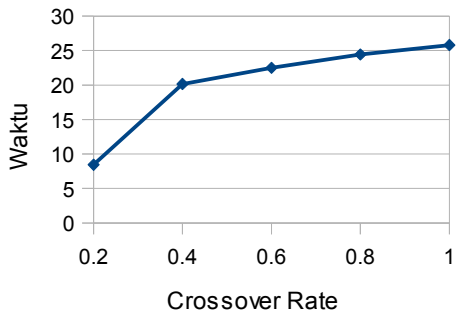
evolusi. Penambahan jumlah kromosom akan menurunkan waktu yang dibutuhkan oleh Algoritma Genetika untuk mencapai solusi. Hal ini karena jumlah kromosom yang semakin banyak akan memperkaya populasi akan variasi yang membawa kepada solusi.

Namun begitu, grafik menunjukkan penurunan waktu yang tidak signifikan dari jumlah kromosom 20 hingga 40. Ini disebabkan karena semakin banyak kromosom pada populasi berarti semakin banyak operasi-operasi seleksi, crossover, dan mutasi yang harus dilakukan, dan itu menambah waktu evolusi.

4.3 PERCOBAAN CROSSOVER RATE

Percobaan *crossover rate* dilakukan untuk melihat pengaruh probabilitas crossover pada waktu evolusi Algoritma Genetika.

Hasil percobaan digambarkan pada grafik berikut.



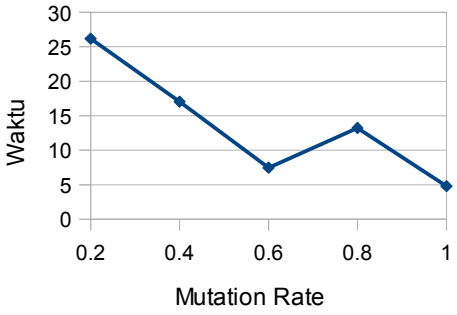
Gambar 4.4 Pengaruh *crossover rate* terhadap waktu evolusi

Grafik di atas menunjukkan pengaruh *crossover rate* atau probabilitas crossover terhadap waktu evolusi. Pada grafik tersebut ada penurunan performa Algoritma Genetika yang sebanding dengan penambahan *crossover rate*.

4.4 PERCOBAAN MUTATION RATE

Percobaan mutation rate dilakukan untuk mengetahui pengaruhnya terhadap waktu evolusi.

Hasil percobaan digambarkan pada grafik berikut.W



Gambar 4.5 Grafik pengaruh *mutation rate* terhadap waktu evolusi

Grafik di atas menunjukkan pengaruh *mutation rate* atau probabilitas mutasi terhadap waktu evolusi. Pada grafik tersebut terjadi penurunan performa Algoritma Genetika yang sebanding dengan penambahan *mutation rate*. Hal ini disebabkan karena mutasi dibutuhkan terus-menerus untuk mencapai solusi. Menaikkan *mutation rate* berarti memperbanyak proses mutasi yang terjadi sehingga solusi akan lebih cepat tercapai.

5 SIMPULAN DAN SARAN

Berdasarkan hasil pengujian dan analisa yang telah dibahas pada bab sebelumnya, maka dapat dibuat beberapa kesimpulan sebagai berikut.

1. Dalam proses evolusi menuju nilai *fitness* yang sempurna, seringkali terjadi stagnasi yang membuang waktu dan menyebabkan waktu evolusi menjadi panjang. Pengulangan proses evolusi dari awal (inisialisasi populasi) dapat memperbaiki waktu evolusi.
2. Inisialisasi populasi di awal proses evolusi sangat menentukan konvergensi proses evolusi.
3. Penambahan jumlah kromosom dalam populasi dapat meningkatkan performa algoritma namun tidak signifikan karena membutuhkan komputasi yang lebih tinggi.
4. Peningkatan *crossover rate* dan *mutation rate* dapat meningkatkan konvergensi.

Untuk pengembangan di masa mendatang, saran yang dapat kami berikan adalah:

1. Menggunakan bahasa native seperti C/C++ akan dapat meningkatkan performa.

DAFTAR PUSTAKA

Suyanto. 2008. Soft Computing: Membangun Mesin Ber-IQ Tinggi. Bandung: Informatika.

ANZAC, The. 2009. Sudoku Algorithm: Generates a Valid Sudoku in 0.018 seconds. Website: http://www.codeproject.com/KB/game/SudokuGen.aspx?msg=2928863

Algorithmics of sudoku. 2007. Website: http://en.wikipedia.org/wiki/Algorithmics_of_sudoku

An introduction to genetic algorithms. Website: http://www.realintelligence.net/tutorials.php?category=optim&tutName=geneticalgos

Setia Y, Narapratama D. 2006. Analisis Beberapa Algoritma Untuk Menyelesaikan Puzzle Sudoku. Teknik Informatika ITB.