Analisis Performa *Min-max Fitness Mutation* dalam Algoritma Genetika pada Permainan Puzzle Sudoku

Putra Noba Nurima[1], Poltak Sihombing[2], Erna Budhiarti Nababan[3]

[1]*Magister Teknik Informatika Fasilkom-TI USU*
[2,3]*Fakultas Ilmu Komputer dan Teknologi Informasi USU*
putra.noba.nurima@students.usu.ac.id

Abstrak--Penerapan operator mutasi pada algoritma genetika apabila tidak di koordinasikan dengan baik dapat membuat menurunnya nilai fitness pada suatu individu. Untuk mengatasi masalah tersebut dilakukan pengaturann penggunaan operator mutasi sehingga diharapkan dapat membuat proses pencarian solusi dengan algoritma genetika menjadi lebih terarah. Pada penelitian ini, proses pencarian solusi menggunakan algoritma genetika selanjutnya dilakukan dengan terfokus pada beberapa individu yaitu individu dengan nilai fitness minimal dan maksimal. Pendekatan metode mutasi ini yang digunakan adalah mutasi min-max, dimana individu dengan nilai fitness minimal dimutasi agar menurun sehingga kecil kemungkinannya bertahan, sebaliknya individu dengan nilai fitness maksimal dimutasi agar kemungkinan bertahannya meningkat sehingga dapat bertahan pada proses selanjutnya. Pada pengujian yang dilakukan, solusi dapat ditemukan dengan cepat atau dengan jumlah generasi yang sedikit, namun pada pembatasan jumlah generasi tidak selalu ditemukan solusi yang valid. Selanjutnya pengujian dengan jumlah individu yang kecil menunjukkan kinerja yang lebih baik dibandingkan dengan jumlah individu yang lebih besar.

I. PENDAHULUAN

Algoritma genetika digunakan untuk menyelesaikan permasalahan *searching* dan optimasi yang mempunyai kompleksitas tinggi yang banyak terjadi dalam *dynamic programming.* Operator utama yang terdapat dalam algoritma genetika meliputi *selection, crossover* dan *mutation.*

Pada proses penerapan operator mutasi, dalam prosesnya dapat menghasilkan individu dengan fitness yang lebih buruk dari sebelumnya apabila dilakukan secara acak serta tidak terkoordinasi dengan baik. Pada beberapa aplikasi penerapan algoritma genetika, terkadang operator mutation tidak digunakan untuk menjaga agar hasil yang sudah mendekati solusi optimal tidak dirusak oleh penggunaan operator mutasi [14]. Seperti pada penelitian [13], penggunaan operator mutasi hanya digunakan jika diperlukan saja yaitu untuk menghindari generasi yang tidak dibutuhkan.

Pada penelitian [2], ditawarkan penggunaan *controlled crossover* dan *guided mutation* untuk membuat proses menjadi lebih efisien dengan membatasi menurunnya nilai fitness. Pendekatan ini hanya diterapkan pada kromosom yang menggunakan nilai gen berbasis biner {0 dan 1} dan untuk memastikan bahwa penggunaan operator mutasi akan selalu cenderung menaik yaitu gen 0 menjadi 1.

Pada penelitian [9], mereka menggunakan pemanggilan fungsi fitness untuk mengurutkan individu atau kromosom secara *descending* yaitu dari *fitness* tertinggi hingga terendah. Proses mutasi gen selanjutnya hanya diterapkan pada kromosom dengan rangking 1 sampai 5. Hasil yang didapatkan menunjukkan peningkatan kinerja algoritma genetika namun masih ditemukan kerentanan dari proses persilangan yang mengakibatkan menurunnya nilai fitness pada individu yang sudah baik.

Berkaitan dengan penelitian di atas, untuk menjaga nilai fitness menjadi lebih baik dan dapat meminimalisasi mutasi kearah yang lebih buruk dengan ini kiranya dapat digunakan *min-max fitness mutation*. Penerapan min-max ini adalah dengan mengaplikasikan dahulu perhitungan jumlah fitness sebelum dilakukan proses mutasi.

Kemudian dipilih individu dengan nilai *fitness* yang tertinggi dan terendah saja untuk kemudian dilakukan mutasi. Hal ini dilakukan dengan tujuan untuk mendapatkan kemungkinan nilai *fitness* yang lebih tinggi pada max individu dan nilai *fitness* yang lebih kecil pada min individu.

II. MUTASI

Mutasi berperan untuk menggantikan informasi bit yang hilang akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada insialisasi populasi [16].

Percobaan mutasi merupakan suatu percobaan efektif yang cukup memadai dalam teknik pencarian solusi, namun proses komputasi penerapan mutasi membutuhkan sumberdaya yang tinggi apabila digunakan untuk mengeksekusi suatu percobaan terhadap individu yang banyak dalam populasi [10].

Berdasarkan bagian yang termutasi, proses mutasi dapat dibedakan atas tiga bagian [16]:

1) Mutasi pada tingkat kromosom;
2) Mutasi pada tingkat gen;
3) Mutasi pada tingkat bit.

Berdasarkan mutasinya dalam gen, terbagi menjadi 3, yaitu [8]:

1) Swap mutation;
2) 3 *Swap Mutation;*
3) *Insertion Mutation.*

III. MIN-MAX FITNESS MUTATION

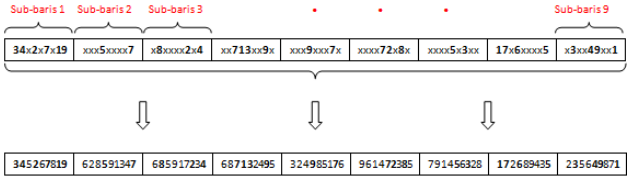
Min-max fitness mutation merupakan penerapan dari operator mutasi yang digunakan dalam algoritma genetika. Sebelum dilakukan mutasi pada individu dalam populasi, terlebih dahulu dilakukan perhitungan *fitness* masing-masing individu untuk kemudian didapatkan sebuah individu yang memiliki jumlah *fitness* tertinggi dan sebuah individu dengan *fitness* terendah. Penerapan proses mutasi kemudian hanya dilakukan pada dua buah individu tersebut.

Tujuan yang ingin didapatkan pada proses mutasi ini adalah untuk menambah nilai *fitness* pada individu *max* dan mengurangi jumlah *fitness* pada individu *min*. Hal ini berguna untuk mempertegas nilai *fitness* individu tersebut. Individu *max* akan memiliki probabilitas tertinggi untuk bertahan pada proses *selection*, demikian pula pada individu *min* akan memiliki probabilitas terkecil sehingga terseleksi dan hilang dari rantai evolusi.

IV. INISIALISASI POPULASI AWAL

Untuk menyelesaikan persoalan sudoku dilakukan inisialisasi populasi awal yang berisi individu-individu yang merepresentasikan solusi yang mungkin di dapatkan. Pada populasi awal terdiri dari individu yang dibentuk dari 9 sub-baris yang merepresentasikan tiap-tiap baris dari matriks Sudoku. Masing-masing sub-baris berisi 9 gen yang merepresentasikan angka-angka dalam matriks Sudoku. Setiap individu selanjutnya dicreate berdasarkan petunjuk atau soal yang diberikan dan kotak kosong yang diisi angka secara random namun dengan memperhatikan bahwa dalam satu sub-baris tidak terdapat angka yang sama.

Pada gambar 1 diilustrasikan bagaimana kromosom 1 terbentuk, dimulai dari kromosom yang hanya berisi gen-gen petunjuk (yang ditulis tebal) dan gen-gen kosong (yang dituliskan dengan symbol x) untuk selanjutnya diisi dengan nilai random.



Gambar 1. Proses inisialisasi populasi awal

Tahap pembentukan kromosom selanjutnya akan mengikuti cara yang sama pada seluruh pembentukan kromosom namun karena gen kosong diisi dengan nilai acak maka kromosom yang dihasilkan akan berbeda.

V. SELEKSI

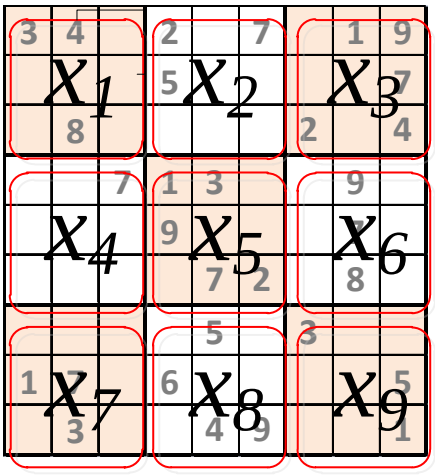
Sebelum dilakukan tahap seleksi, terlebih dahulu individu yang telah ada harus dihitung dahulu nilai *fitness*nya. Metode seleksi yang digunakan pada penelitian ini adalah *rollette whell selection* atau roda rolet.

a. Evaluasi nilai *fitness*

Nilai *fitness* digunakan untuk menentukan baik atau tidaknya suatu solusi dan dijadikan acuan untuk mencapai hasil yang optimal.

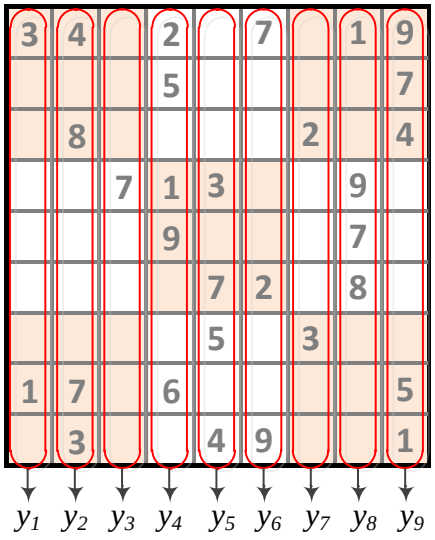
Nilai *fitness* selanjutnya dihitung dengan 2 (dua) kali perhitungan yaitu perhitungan nilai *x* dan nilai *y*. Pada perhitungan nilai variabel *x*, diambil dari sub-blok matriks, dimana berisi gen-gen yang terdiri dari bilangan {1..9}

dan masing-masing bilangan tersebut tidak terdapat angka yang sama. Gambar 2 menjelaskan bagaimana pembagian sub-blok matriks Sudoku untuk perhitungan nilai *x*.



Gambar 2. Aturan *encoding* nilai *x*

Selanjutnya perhitungan variabel *y* diambil dari gen yang memiliki indeks awal (*i*) yang berbeda atau jika dilihat pada matriks adalah perhitungan gen secara vertikal. Untuk lebih jelasnya, aturan perhitungan nilai *y* dapat dilihat pada gambar 3 berikut.



Gambar 3. Aturan *encoding* nilai *y*

Nilai fitness pada persoalan sudoku dapat dihitung menggunakan rumus, sebagai berikut:

$$f(x) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i \qquad (2.1)$$

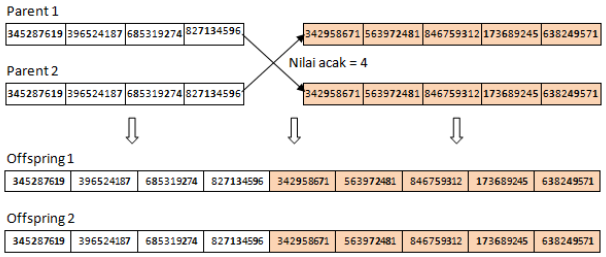
Dimana:

$f(x)$ = nilai *fitness* setiap kromosom
n = jumlah gen
x_i = nilai *variabel* gen *x*
y_i = nilai *variabel* gen *y*

VI. PERSILANGAN

Titik persilangan pada penelitian ini hanya akan dilakukan antara masing-masing sub-baris, untuk memastikan proses persilangan dapat menjaga angka nilai soal tidak berubah selama optimasi.

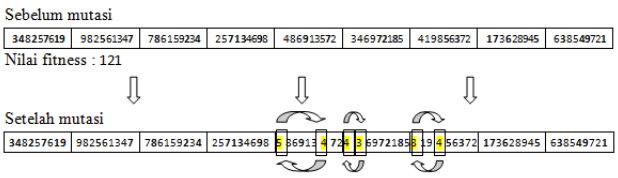
Persilangan dilakukan pada kromosom yang berdekatan, misalnya kromosom 1 dipersilangkan dengan kromosom 2, kromosom 3 dipersilangkan dengan kromosom 4, dan seterusnya.



Gambar 4. Persilangan 1 titik pada sub-baris kromosom

Pada penelitian ini proses mutasi menggunakan perhitungan *fitness* untuk menentukan kromosom yang akan terpilih untuk dilakukan proses mutasi dimana kromosom yang dimutasi hanyalah pada kromosom yang memiliki *fitness* yang tertinggi dan terendah.

Metode mutasi yang digunakan adalah mutasi gen dan mutasi *swap*. Mutasi gen dilakukan untuk mengganti gen lain yang sama dalam 1 kromosom, gen tersebut akan diganti dengan kemungkinan gen yang didapatkan secara acak, sedangkan mutasi *swap* dilakukan untuk mengganti gen yang sudah dimutasi untuk menjaga agar dalam suatu sub-baris tidak ada gen yang sama. Pada mutasi untuk kromosom dengan *fitness* tertinggi, mutasi dilakukan untuk memperbaiki nilai *fitness* supaya menjadi lebih tinggi.

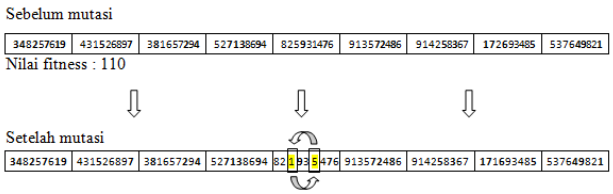


Gambar 5. Proses penerapan mutasi max

Untuk lebih jelasnya dapat dilihat pada proses mutasi dalam bentuk matriks pada gambar 6.

Gambar 6. Proses penerapan mutasi max dalam bentuk matriks

Selanjutnya mutasi pada kromosom dengan nilai fitness terendah akan dilakukan pengambilan posisi gen secara acak untuk kemudian dilakukan mutasi gen secara acak pula dengan cara mempertukarkan gen pada sub-baris yang sama. Gambar 7 menunjukkan bagaimana mutasi min dilakukan.



Gambar 7. Proses penerapan mutasi min

VII. PENGUJIAN

Pengujian dilakukan pada data matriks Sudoku 9x9 dengan menggunakan beberapa parameter pada percobaan dengan nilai parameter yang berbeda-beda, yaitu: jumlah kromosom, batas generasi, probabilitas persilangan *(Pc)* dan probabilitas mutasi *(Pm)*. Hasil pengujian dapat dinyatakan *solved* apabila nilai *fitness* yang didapatkan adalah 162.

Tidak ada aturan pasti dalam menentukan parameter dalam algoritma genetika, maka untuk jumlah individu akan dipilih jumlah yang sedikit dengan batas generasi sebanyak 500 untuk mempersingkat komputasi, selanjutnya untuk *Pc* dan *Pm* digunakan parameter yang disarankan oleh [5]. Adapun nilai parameter yang digunakan untuk algoritma genetika adalah sebagai berikut:

1. Jumlah individu : 6, 12 dan 18;
2. Batas generasi : 500;
3. Probabilitas persilangan (*Pc*) : 0,5;
4. Probabilitas mutasi (*Pm*) : 0,1.

Selanjutnya pengujian akan dilakukan dengan program algoritma genetika untuk menyelesaikan *puzzle* sudoku. Misalkan pada pengujian didapatkan solusi yang dapat menyelesaikan *puzzle* dengan nilai *fitness* sebesar 162 dan dengan jumlah generasi sebanyak 186. Hal ini menunjukkan bahwa pada generasi ke-186, sudah terdapat solusi berupa individu dengan nilai *fitness* optimum sehingga proses dapat dihentikan. Penghentian proses dapat dilakukan untuk menghemat proses komputasi karena solusi akan tetap stabil meskipun proses tetap dilanjutkan.

Pada pengujian pertama dilakukan dengan 6 individu yang diproses menggunakan algoritma genetika, rangkuman hasil yang didapatkan dimuat pada tabel I.

TABEL I. HASIL PENGUJIAN DENGAN 6 INDIVIDU

No. Pengujian	Jumlah Generasi			Persentase Solusi optimum yang Tercapai
	Terbaik	Terburuk	Rata-rata	
1-10	34	403	146	90%
11-20	10	446	215	90%
21-30	34	420	171	100%
31-40	40	282	148	90%
41-50	19	218	105	90%
51-60	152	358	239	80%
61-70	17	500	193	90%
71-80	21	393	154	90%
81-90	55	406	202	90%
91-100	36	393	206	90%
	10	**500**	**178**	**90%**

Pada tabel I dapat dilihat rangkuman hasil untuk mengetahui kinerja pada setiap kelompok pengujian. Sebagai contoh, pada nomor pengujian 1-10 menunjukkan bahwa dari 10 kali pengujian terdapat 9 kali pengujian yang berhasil menemukan solusi optimum.

Pada keseluruhan hasil, solusi optimum dapat ditemukan dengan generasi terbaik sebanyak 10 generasi. Nilai rata-rata keseluruhan dari jumlah generasi yang didapatkan sebanyak 178 generasi dengan tingkat persentase pencapaian solusi sebesar 90%.

Selanjutnya pada pengujian kedua dilakukan dengan 12 individu, rangkuman hasil yang didapatkan dimuat pada tabel II.

TABEL II. HASIL PENGUJIAN DENGAN 12 INDIVIDU

No. Pengujian	Jumlah Generasi			Persentase Solusi optimum yang Tercapai
	Terbaik	Terburuk	Rata-rata	
1-10	36	365	216	70%
11-20	67	401	229	80%
21-30	45	139	87	30%
31-40	12	408	176	70%

41-50	13	313	153	80%
51-60	33	453	171	100%
61-70	18	277	95	80%
71-80	37	189	110	60%
81-90	17	470	155	80%
91-100	11	489	266	100%
	11	**489**	**166**	**75%**

Pada keseluruhan hasil pada tabel II, solusi optimum dapat ditemukan dengan generasi minimal sebanyak 11 generasi. Nilai rata-rata keseluruhan dari jumlah generasi yang didapatkan sebanyak 166 generasi dengan tingkat persentase pencapaian solusi sebesar 75%.

Selanjutnya pada pengujian ketiga dilakukan dengan 18 individu, rangkuman hasil yang didapatkan dimuat pada tabel III.

TABEL III. HASIL PENGUJIAN DENGAN 18 INDIVIDU

No. Pengujian	Jumlah Generasi			Persentase Solusi optimum yang Tercapai
	Terbaik	Terburuk	Rata-rata	
1-10	17	464	174	80%
11-20	28	456	225	80%
21-30	24	300	93	90%
31-40	11	351	153	40%
41-50	78	164	111	50%
51-60	13	372	154	70%
61-70	16	465	164	70%
71-80	23	348	161	60%
81-90	13	331	112	60%
91-100	22	408	213	80%
	11	**465**	**156**	**68%**

Pada keseluruhan hasil pada tabel III, solusi optimum dapat ditemukan dengan generasi minimal sebanyak 11 generasi. Nilai rata-rata keseluruhan dari jumlah generasi yang didapatkan sebanyak 156 generasi dengan tingkat persentase pencapaian solusi sebesar 68%.

VIII. PEMBAHASAN

Setelah dilakukan serangkaian pengujian, didapatkan hasil bahwa penggunaan mutasi min-max dapat menemukan hasil yang optimum dengan generasi yang sedikit. Meskipun hasil yang didapatkan tidak selalu merupakan solusi yang dapat menyelesaikan permainan (*solved*), namun hasil optimum bisa ditemukan dengan generasi yang sedikit.

Tabel IV merangkum hasil pengujian yang mendapatkan nilai optimum yang dapat menyelesaikan *puzzle* dari seluruh hasil pengujian yang telah dilakukan sebelumnya.

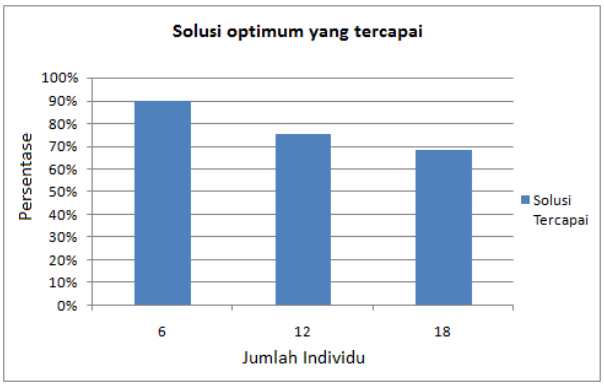
TABEL IV. HASIL PENGUJIAN BERDASARKAN JUMLAH INDIVIDU

INDIVIDU	Jumlah Rata-rata Generasi yang dibutuhkan			Persentase Solusi optimum yang Tercapai
	Terbaik	Terburuk	Rata-rata	
6	10	500	178	90%
12	11	489	166	75%
18	11	465	156	68%

Berdasarkan jumlah individu dalam suatu populasi, hasil terbaik pada pengujian ini didapatkan dari jumlah individu yang kecil yaitu sebanyak 6. Sedangkan pada jumlah individu sebanyak 12 dan 18, hasil yang diperoleh kurang maksimal. Hal ini dikarenakan mutasi hanya dilakukan pada 2 individu yaitu individu dengan *fitness* terbesar dan individu dengan *fitness* terkecil, sehingga pada individu dengan jumlah banyak akan memperbesar kemungkinan nilai fitness pada kromosom menjadi menurun.

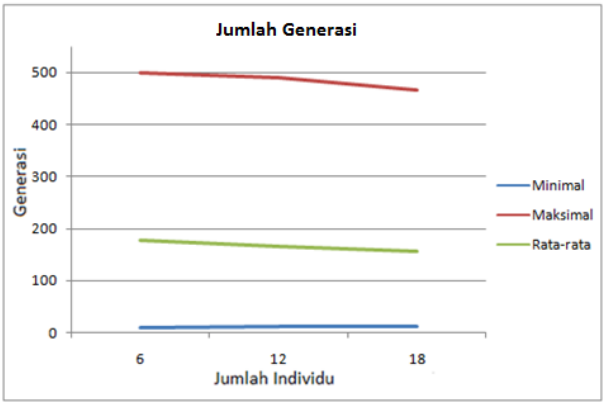
Solusi optimum yang diperoleh bisa didapatkan sebelum batas generasi tercapai namun apabila batas generasi sudah tercapai namun apabila tidak ditemukan solusi yang optimum maka solusi yang diambil adalah solusi terbaik yang mampu dihasilkan.

Pada gambar 7 dapat dilihat grafik rangkuman hasil pengujian berdasarkan solusi optimum yang berhasil dicapai. Pada grafik tersebut terlihat bahwa semakin besar jumlah individu dalam suatu populasi maka akan semakin menurun persentase solusi optimum yang tercapai.



Gambar 8. hasil pengujian berdasarkan solusi optimum tercapai

Pada gambar 9 dapat dilihat grafik hasil pengujian berdasarkan jumlah generasi yang dibutuhkan yaitu kurva yang terlihat menunjukkan bahwa semakin besar jumlah individu maka jumlah generasi yang dibutuhkan akan semakin kecil.



Gambar 9. Hasil pengujian berdasarkan jumlah generasi

IX. KESIMPULAN

Penggunaan algoritma genetika dengan mutasi pada individu min-max fitness mampu menemukan solusi dengan generasi yang sedikit sehingga dapat mempersingkat komputasi yang dibutuhkan.

Penerapan mutasi pada min-max fitness dengan populasi yang lebih kecil menunjukkan kinerja yang lebih baik dibandingkan dengan populasi yang lebih besar. Hal ini dapat dilihat dari hasil pengujian yang menunjukkan bahwa nilai fitness optimum yang memenuhi kriteria untuk menyelesaikan puzzle lebih sering didapatkan pada populasi dengan sedikit individu.

X. SARAN

Untuk penelitian selanjutnya penggunaan mutasi pada min-max mutasi dapat diujicobakan pada studi kasus yang lain yang memiliki banyak solusi dan dapat juga dengan penambahan metode lain yang sesuai.

Untuk penerapan mutasi pada min-max fitness pada individu yang lebih banyak, selanjutnya dapat dilakukan pengkajian probabilitasnya seperti jumlah individu min-max dapat disesuaikan dengan jumlah individu yang ada pada populasi.

REFERENSI

[1] Chen, Y., Yue, C., Mabu, S. & Hirasawa, K. 2009. A genetic relation algorithm with guided mutation for the large-scale portfolio optimization. *ICROS-SICE International Joint Conference* **2**: 2579-2584.

[2] Chowdhury, B. 2014. A genetic approach with controlled crossover and guided mutation for biological sequence alignment. *IEEE Fourth International Conference of Emerging Applications of Information Technology* **14**: 307-312.

[3] Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Publishing Company, Inc: Massachusetts.

[4] Hadinata, J. 2011. Problem solving sudoku menggunakan algoritma genetika. *Jurnal Sisfotenika* **1** (1): 48-58.

[5] Hopgood, A.A. 2001. *Intelligence Systems for Engineers dan Scientists*. 2nd Edition. CRC Press: Boca Raton.

[6] Kusumadewi, S. 2003. *Artificial Intelligence: Teknik & aplikasi*. Graha Ilmu: Yogyakarta.

[7] Liu, L. & Zhang, Y. 2010. Genetic algorithm design of the min -max weighted distance problem. *International Conference on Computer Application and System Modeling (ICCASM 2010)* **6**: 620-623.

[8] Mantere, T. & Koljonen, J. 2007. Solving, rating and generating sudoku puzzles with GA. *IEEE Congress on Evalutionary Computation (CEC 2007)* **7**: 1382-1389.

[9] Metaxiotis, K. & Liagkouras, K. 2013. A fitness guided mutation operator for improved performance of MOEAs. *IEEE* **13**: 751-754.

[10] Mirshokraie, S., Mesbah, A. & Pattabiraman, K. 2015. Guided mutation testing for javascript web applications. *IEEE Transactions on Software Engineering* **41**(5): 429-444.

[11] Mitchell, M. 1999. *An Introduction to Genetic Algorithms*, The MIT Press: London.

[12] Negnevitsky, M. 2005. *Artificial Intelligence: A guide to intelligent systems*. Pearson Education.

[13] Pehlivanoglu, Y.V. 2013. A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks. *IEEE Transactions on Evolutionary Computation* **17** (3): 436-452.

[14] Saragih, R.I.E. 2015. Fuzzy logic controller (FLC) dalam penentuan mutation rate algoritma genetika. Tesis. Universitas Sumatera Utara.

[15] Song, Y.H. 1996. Improved genetic algorithms with fuzzy logic controlled crossover and mutation. *UKACC International Conference on CONTROL* **427**: 140-144.

[16] Suyanto. 2007. *Algoritma Genetika dalam Matlab*. Penerbit Andi: Yogyakarta.

[17] Thiang., Kurniawan, R. & Ferdinando, H. 2001. Implementasi algoritma genetika pada mikrokontroler MCS51 untuk mencari rute terpendek. *Prosiding Seminar of Intelligent Technology and Its Applications (SITIA)*.