

Implementasi Algoritma Backtracking Dengan Optimasi Menggunakan Teknik Hidden Single Pada Penyelesaian Permainan Sudoku

Valdo Septiansen Widjaja
Program Studi Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi
Universitas Multimedia Nusantara
Gading Serpong, Tangerang, Indonesia
valdo_widjaja@yahoo.com

Dodick Z. Sudirman
Program Studi Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi
Universitas Multimedia Nusantara
Gading Serpong, Tangerang, Indonesia
dodick@umn.ac.id

Abstrak—Perkembangan teknologi informasi telah merambah dalam berbagai bidang, salah satunya bidang permainan seperti *puzzle* atau teka-teki. Sudoku adalah sebuah permainan teka-teki logika yang cukup menarik untuk dimainkan. Hingga saat ini permainan Sudoku telah populer di kalangan masyarakat. Berbagai jenis variasi *puzzle* dan tingkat kesulitan yang terdapat dalam Sudoku membuat para ilmuwan berusaha untuk melakukan penelitian terhadap permainan ini. Penggunaan algoritma *backtracking* dalam penyelesaian *puzzle* Sudoku merupakan salah satu penelitian yang telah dilakukan sebelumnya. Hanya saja, algoritma ini masih membutuhkan waktu yang cukup lama dalam melakukan komputasi penyelesaian *puzzle* Sudoku. Dalam penelitian ini, dibangun sebuah aplikasi untuk mengoptimalkan algoritma *backtracking* dalam menyelesaikan *puzzle* Sudoku menggunakan sebuah teknik optimasi yang disebut teknik *hidden single*. Aplikasi ini telah berhasil mengimplementasikan teknik *hidden single* ke dalam algoritma *backtracking* yang digunakan. Hasil uji coba penelitian menunjukkan bahwa dengan menggunakan optimasi teknik *hidden single*, algoritma *backtracking* mampu mengoptimalkan waktu dan kinerja komputasi pada penyelesaian *puzzle* Sudoku.

Kata kunci—Sudoku, permainan, logika, *puzzle*, teka-teki, algoritma, optimasi, *backtracking*, *hidden single*.

I. PENDAHULUAN

Perkembangan teknologi informasi terjadi dalam berbagai bidang, salah satunya dalam bidang permainan[6]. Selain sebagai hiburan, permainan juga dapat menjadi suatu hal yang menantang ataupun untuk mengasah kemampuan otak pemain, seperti permainan teka-teki yang saat ini sedang populer di kalangan masyarakat[8].

Sudoku adalah sebuah permainan teka-teki angka yang berbasis logika[9]. Pada umumnya, sebuah *puzzle* Sudoku terdiri dari 81 kotak yang disusun menjadi 9 baris, 9 kolom, dan 9 subbagian. Tujuan utama dari permainan ini adalah mengisi seluruh kotak tersebut dengan angka 1 sampai 9[1]. Permainan Sudoku mengharuskan pemainnya untuk berpikir secara logis sesuai dengan aturan-aturan permainan yang ada. Namun demikian, para ilmuwan komputer dan matematikawan berusaha untuk meneliti permainan ini karena tingkat kerumitan dan banyaknya variasi *puzzle* Sudoku yang masing-

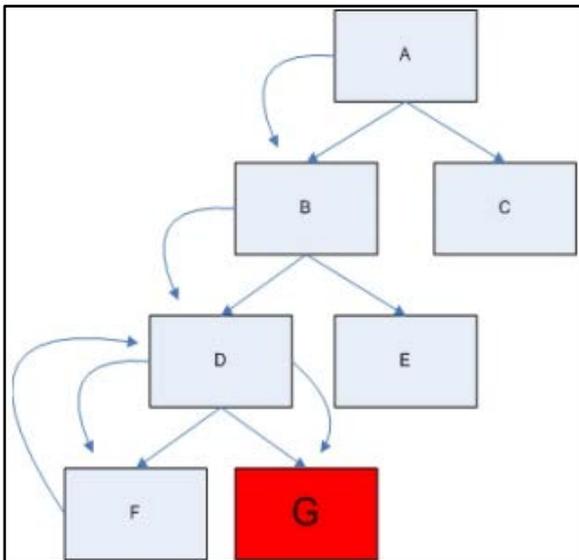
masing memiliki suatu solusi unik[4]. Salah satu teknik sederhana dan paling umum yang sering digunakan untuk menyelesaikan permainan Sudoku adalah teknik *hidden single*[1]. Meskipun begitu, pada beberapa variasi *puzzle* Sudoku tertentu tidak dapat diselesaikan hanya dengan menggunakan teknik *hidden single* saja.

Algoritma *backtracking* merupakan perbaikan dari algoritma *brute-force* dan algoritma *exhaustive-search*, di mana algoritma ini membangun pohon ruang status (*state-space tree*) untuk menemukan solusi[3]. Prinsip dari algoritma *backtracking* adalah jika terjadi kesalahan dalam pencarian solusi pada sebuah *node*, maka akan dilakukan *backtrack* ke *node* sebelumnya. Meski begitu, algoritma ini menjadi tidak efisien jika kandidat solusi yang ditemukan sama dengan kandidat solusi yang ditemukan pada algoritma *exhaustive-search*.

Berdasarkan pemaparan tersebut di atas, maka dilakukan penelitian untuk melakukan implementasi teknik *hidden single* pada algoritma *backtracking* dengan tujuan melakukan optimasi pada algoritma tersebut sehingga dapat membantu dalam menyelesaikan *puzzle* Sudoku dengan lebih mudah.

II. ALGORITMA BACKTRACKING

Algoritma *backtrack* pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950[10]. Dalam perkembangan algoritma ini, beberapa ahli seperti Rwalker, Golomb, dan Baumert menyajikan uraian umum tentang *backtrack* dan penerapannya dalam berbagai persoalan dan aplikasi. Algoritma *backtracking* adalah sebuah algoritma yang berbasis *depth-first search* (DFS) dalam pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Algoritma ini membangun solusi parsial (*partial solution*) dari sebuah kandidat solusi dan mengevaluasi solusi parsial tersebut pada suatu waktu. Jika solusi parsial yang dibangun tidak memenuhi syarat, maka kandidat solusi tersebut tidak akan dibangun lebih lanjut dan dilakukan *backtrack* ke kandidat solusi lain yang memenuhi syarat. Algoritma *backtracking* dilakukan secara berulang-ulang hingga menemukan sebuah solusi yang sesuai dengan syarat.



Gambar 1. Contoh Pohon Ruang Status Algoritma Backtracking[10]

Langkah-langkah pencarian solusi pada algoritma *backtracking* adalah sebagai berikut[10].

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Simpul yang telah dilahirkan dinamakan simpul hidup dan simpul hidup yang diperluas dinamakan simpul-E (*Expand node*).
2. Jika lintasan yang diperoleh dari perluasan simpul-E tidak mengarah ke solusi, maka simpul itu akan menjadi simpul mati yang tidak dapat diperluas lagi.
3. Jika posisi terakhir ada di simpul mati, maka pencarian dilakukan dengan membangkitkan simpul anak yang lainnya dan jika tidak ada simpul anak maka dilakukan *backtracking* ke simpul induk.
4. Pencarian dihentikan jika kita telah menemukan solusi atau tidak ada simpul hidup yang dapat ditemukan.

```

procedure RunutBalik(input k: integer)
{ Mencari semua solusi dengan metode
backtracking.
Masukan: k, yaitu indeks komponen vektor
solusi x[k].
Keluaran: solusi x = (x[1], x[2], ...,
x[n]). }
Algoritma:
for tiap x[k] yang belum dicoba
sedemikian sehingga (x[k] ← T(k)) and
B(x[1], x[2], ..., x[k]) = true do
if (x[1], x[2], ..., x[k]) adalah
lintasan dari akar ke daun then
CetakSolusi(x)
endif
RunutBalikR(k + 1)
Endforeach

```

Gambar 2. Contoh Pseudocode Algoritma Backtracking[7]

III. SUDOKU DAN TEKNIK HIDDEN SINGLE

Sebuah permainan teka-teki Sudoku terdiri dari 9x9 kotak, yang dibagi menjadi 9 subbagian berukuran 3x3 kotak[2]. Beberapa kotak telah diisi dengan angka-angka tertentu (antara 1 sampai 9) sebagai petunjuk awal permainan. Tujuan dari permainan Sudoku adalah untuk meletakkan sebuah angka (antara 1 sampai 9) ke dalam sebuah kotak kosong dengan aturan masing-masing baris, kolom, dan subbagian hanya boleh menampilkan angka tersebut sekali.

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 3. Contoh Sudoku Valid Berukuran 9x9 kotak[11]

Secara umum, permainan Sudoku dapat diselesaikan secara manual dengan kombinasi beberapa teknik dasar, antara lain sebagai berikut[5].

1) Pemindaian (*scanning*)

Berupa proses memindai baris atau kolom untuk mengidentifikasi baris mana yang terdapat angka-angka tertentu. Proses ini kemudian diulang pada setiap kolom (atau baris) secara sistematis. Kemudian menentukan nilai dari suatu kotak dengan membuang nilai-nilai yang tidak mungkin.

2) Penandaan (*marking*)

Penandaan berupa analisis logika, dengan menandai kandidat angka yang dapat dimasukkan ke dalam sebuah kotak.

3) Analisis (*analyzing*)

Analisis berupa eliminasi kandidat, di mana kemajuan dicapai dengan mengeliminasi kandidat angka secara berturut-turut hingga sebuah kotak hanya mempunyai sebuah kandidat.

Teknik *hidden single* adalah salah satu teknik dalam menyelesaikan permainan Sudoku[11]. Teknik *hidden single* bertujuan untuk mencari kandidat tunggal dari sebuah kotak, tetapi kandidat tunggal tersebut agak tersembunyi. Teknik ini dilakukan dengan mencari kandidat yang unik pada suatu baris, kolom, atau subbagian. Sebuah kotak yang memiliki *hidden single* dapat memiliki lebih dari satu kandidat,

tetapi salah satu kandidat tersebut adalah unik pada suatu baris, kolom, atau subbagian.

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 4. Contoh Teknik Hidden Single Pada Baris[11]

Gambar 4 menunjukkan salah satu kondisi *hidden single* pada baris. Kotak warna kuning, yaitu pada baris ke-1, kolom ke-5, memiliki *hidden single*, yaitu angka 2. Pada subbagian ke-1, sudah terdapat angka 2 di baris ke-2 sehingga kotak kosong pada baris ke-1, kolom ke-1 tidak mungkin diisi dengan angka 2. Kemudian pada kolom ke-8, sudah terdapat angka 2 di baris ke-5 sehingga pada baris ke-1, kolom ke-8 tidak mungkin diisi dengan angka 2. Maka pada baris ke-1, yang bisa diisi dengan angka 2 hanya pada kolom ke-5.

IV. IMPLEMENTASI DAN UJI COBA

A. Implementasi Algoritma Backtracking Hidden Single

Implementasi algoritma *backtracking hidden single* terdiri dari beberapa proses, antara lain sebagai berikut.

- 1) *Find and save any empty cell into array*
Proses ini melakukan pencarian dan memasukkan setiap kotak kosong yang ada pada *puzzle* ke dalam sebuah *array*.
- 2) *Find and save any candidate number in the empty cell*
Proses ini melakukan pengecekan dan menyimpan kandidat angka yang mungkin pada sebuah kotak.
- 3) *Check if is there any candidate*
Proses ini melakukan pengecekan apakah kotak tersebut memiliki kandidat angka. Jika tidak ada kandidat angka pada kotak tersebut, maka dilakukan proses *backtrack*.
- 4) *Check if is there any hidden single number*
Proses ini melakukan pengecekan apakah kotak tersebut memiliki *hidden single number*. Jika terdapat *hidden single number*, maka dilakukan proses *backtrack*.
- 5) *Check if all candidates have been tried*
Proses ini melakukan pengecekan terhadap seluruh kandidat angka yang telah dicoba sebelumnya. Jika

seluruh kandidat angka telah dicoba sebelumnya, maka dilakukan proses *backtrack*.

6) Backtrack

Proses ini melakukan *backtrack* untuk kembali ke kotak (*node*) yang telah diproses sebelumnya. Kemudian dilakukan pengecekan kembali pada kotak tersebut.

7) Fill the empty cell with one of the candidate number

Proses ini melakukan pengisian pada kotak kosong dengan salah satu kandidat angka yang sesuai dengan aturan permainan Sudoku.

B. Hasil Uji Coba

Pengujian dilakukan dengan membandingkan algoritma *backtracking* biasa dengan algoritma *backtracking hidden single* dalam menyelesaikan *puzzle* Sudoku. Pengujian dilakukan terhadap beberapa tingkat kesulitan yang masing-masing berjumlah 100 *puzzle*. Berikut akan dijelaskan hasil pengujian salah satu tingkat kesulitan.

TABLE I. HASIL PENGUJIAN PENYELESAIAN SUDOKU TINGKAT KESULITAN HARD

Eksp. Ke-	Jumlah kotak kosong (45 - 49)	Waktu		Jumlah Backtrack		Optimasi
		Backtrack	Backtrack H. Single	Backtrack	Backtrack H. Single	
1	45	941	729	24	9	8
2	45	7173	2355	229	46	48
3	45	5308	2883	224	82	44
4	45	5260	2447	129	72	26
5	45	821	567	19	0	8
6	45	632	536	10	2	4
7	45	559	511	5	0	3
8	45	4217	5337	164	46	30
9	45	1102	709	31	7	9
10	45	8120	3415	362	107	46
11	45	6682	1806	282	31	36
12	45	6131	4499	266	167	38
13	45	2471	2472	49	38	10
14	45	441	441	1	0	1
15	45	13110	5281	538	165	53
16	45	4220	3251	178	102	38
17	45	6321	2895	281	67	59
18	45	737	759	0	0	0
19	45	593	469	7	0	3
20	45	1955	948	70	18	10
21	45	1073	788	28	6	11
22	46	3528	2423	146	66	35
23	46	9791	2453	440	78	24
24	46	5486	1564	236	38	21
25	46	715	552	8	1	4
26	46	1418	1145	47	29	8
27	46	460	486	1	0	1
28	46	3733	2334	157	69	31
29	46	983	742	24	9	5

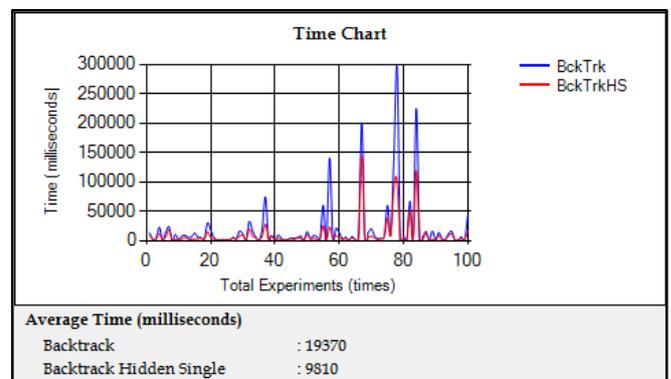
TABLE I. HASIL PENGUJIAN PENYELESAIAN SUDOKU TINGKAT KESULITAN HARD (LANJUTAN)

Eksp. Ke-	Jumlah kotak kosong (45 - 49)	Waktu		Jumlah Backtrack		Optimasi
		Backtrack	Backtrack H. Single	Backtrack	Backtrack H. Single	
30	46	20813	8391	960	209	180
31	46	9283	4943	401	169	58
32	46	1269	525	38	0	3
33	46	1792	1348	58	20	25
34	46	818	631	13	3	5
35	46	702	547	10	0	4
36	47	12795	8379	514	286	118
37	47	2237	1563	84	34	24
38	47	13700	6010	571	199	83
39	47	774	690	14	6	6
40	47	1940	1080	71	21	10
41	47	8828	5539	369	206	49
42	47	4786	3685	210	133	35
43	47	2540	2151	101	70	18
44	47	513	506	3	1	2
45	47	18288	7114	849	185	149
46	47	9195	3162	274	77	57
47	47	4040	3172	170	103	33
48	47	3561	1778	146	36	32
49	47	3384	2378	74	47	14
50	47	59611	25151	2703	838	397
51	47	13334	4005	603	104	76
52	47	7545	5935	332	210	68
53	47	951	912	23	23	0
54	47	4460	2980	190	86	42
55	47	15983	1837	710	33	33
56	48	23445	17577	1065	638	209
57	48	5087	2444	210	60	38
58	48	12774	2105	572	57	25
59	48	17699	5371	824	158	83
60	48	1288	623	35	1	8
61	48	16127	7610	656	267	94
62	48	32184	18855	1383	670	236
63	48	6233	4604	269	157	52
64	48	73803	27744	3462	862	265
65	48	3808	1452	158	28	22
66	48	2005	1076	70	7	21
67	48	14461	9554	657	350	101
68	48	2637	2226	98	76	13
69	48	6472	4459	277	146	45
70	48	577	608	4	0	3
71	48	7848	4136	265	129	54
72	48	2762	949	108	12	14
73	48	59469	38091	2326	1148	499
74	48	140302	90179	5076	2658	933
75	48	2259	1325	83	16	11
76	48	6648	1468	81	20	21
77	48	13296	8312	499	253	108
78	48	42440	16337	1904	592	158
79	49	22500	11363	1051	409	147
80	49	29696	14490	1402	512	189
81	49	5216	4054	215	140	38
82	49	3193	2127	124	55	29
83	49	11054	9087	492	350	81
84	49	1157	897	10	0	2
85	49	16524	7703	761	302	55

TABLE I. HASIL PENGUJIAN PENYELESAIAN SUDOKU TINGKAT KESULITAN HARD (LANJUTAN)

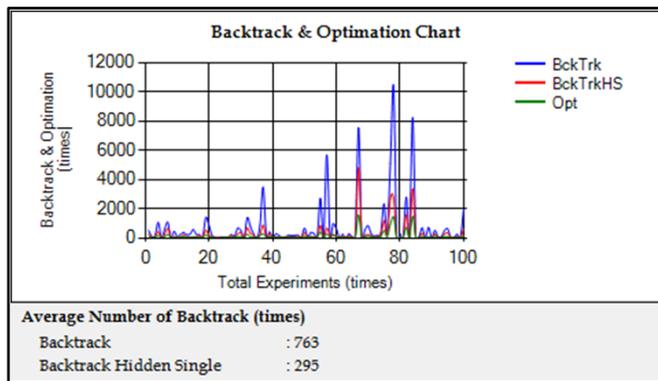
Eksp. Ke-	Jumlah kotak kosong (45 - 49)	Waktu		Jumlah Backtrack		Optimasi
		Backtrack	Backtrack H. Single	Backtrack	Backtrack H. Single	
86	49	8792	7966	397	297	82
87	49	7449	3404	180	134	17
88	49	140159	22795	5673	663	264
89	49	5445	3381	234	103	45
90	49	2162	1818	82	55	14
91	49	200309	143237	7544	4800	1571
92	49	19405	7327	824	209	122
93	49	9270	8253	416	215	63
94	49	294944	102372	10405	2751	1418
95	49	66455	50819	2784	1575	693
96	49	224351	118881	8223	3355	1462
97	49	15112	12227	672	295	98
98	49	11348	7441	492	262	88
99	49	15841	11606	625	334	83
100	49	5952	3508	249	92	37

Pada TABLE I di atas menunjukkan hasil pengujian penyelesaian Sudoku terhadap 100 *puzzle* dengan jumlah awal kotak kosong secara acak antara 45 sampai 49 buah. Parameter yang diuji pada penelitian ini adalah waktu penyelesaian (diukur dalam *milliseconds*) dan jumlah *backtrack* yang dilakukan. Terlihat pada tabel tersebut bahwa 95% soal diselesaikan lebih cepat dengan menggunakan algoritma *backtracking hidden single*, sedangkan jumlah *backtrack* yang dilakukan oleh *backtracking hidden single* lebih sedikit daripada jumlah *backtrack* yang dilakukan oleh *backtrack* biasa.



Gambar 5. Chart Time Hasil Pengujian Penyelesaian 100 Puzzle Sudoku Tingkat Kesulitan Hard

Pada Gambar 5 di atas menunjukkan hasil statistik waktu penyelesaian dari hasil pengujian yang telah dilakukan, berdasarkan TABLE I. Garis biru menggambarkan waktu penyelesaian *backtracking* biasa dan garis merah menggambarkan waktu penyelesaian *backtracking hidden single*. Selain itu terdapat juga rata-rata waktu penyelesaian dari masing-masing algoritma *backtracking*. Terlihat pada gambar tersebut bahwa rata-rata waktu penyelesaian *backtracking* biasa, dengan waktu 19370 *milliseconds*, lebih lama daripada rata-rata waktu penyelesaian *backtracking hidden single*, dengan waktu 9810 *milliseconds*.



Gambar 6. Backtrack & Optimization Chart Hasil Pengujian Penyelesaian 100 Puzzle Sudoku Tingkat Kesulitan Hard

Pada Gambar 6 di atas menunjukkan hasil statistik jumlah *backtrack* dari hasil pengujian yang telah dilakukan, berdasarkan TABLE I. Garis biru menggambarkan jumlah *backtrack* yang dilakukan algoritma *backtracking* biasa dan garis merah menggambarkan jumlah *backtrack* yang dilakukan algoritma *backtracking hidden single*. Selain itu terdapat juga rata-rata jumlah *backtrack* dari masing-masing algoritma *backtracking*. Terlihat pada gambar tersebut bahwa rata-rata jumlah *backtrack* yang dilakukan algoritma *backtracking* biasa, dengan 763 kali, lebih banyak daripada rata-rata jumlah *backtrack* yang dilakukan algoritma *backtracking hidden single*, dengan 295 kali.

Hasil pengujian penyelesaian 100 *puzzle* Sudoku tingkat kesulitan *hard* di atas menunjukkan bahwa algoritma *backtracking hidden single* lebih efisien daripada algoritma *backtracking* biasa, dilihat dari segi waktu penyelesaian dan jumlah *backtrack* yang dilakukan.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Kesimpulan yang diperoleh dari penelitian antara lain sebagai berikut.

1. Optimasi pada algoritma *backtracking* dengan menggunakan teknik *hidden single* untuk menyelesaikan permainan Sudoku telah berhasil diimplementasikan dan diuji.
2. Optimasi pada algoritma *backtracking* dengan menggunakan teknik *hidden single* mampu mengoptimalkan waktu penyelesaian dan meminimalkan jumlah penggunaan *backtrack* pada algoritma *backtracking* biasa dalam menyelesaikan permainan Sudoku.
3. Tingkat kesulitan sebuah *puzzle* Sudoku mempengaruhi hasil uji coba kedua algoritma yang telah diuji. Semakin rumit tingkat kesulitan *puzzle*, maka perbedaan waktu penyelesaian dan jumlah *backtrack* yang digunakan semakin besar.

B. Saran

Saran untuk penelitian dan pengembangan aplikasi selanjutnya adalah sebagai berikut.

1. Adanya keterbatasan pada saat aplikasi berjalan, yaitu terkadang aplikasi menjadi *not responding* pada saat aplikasi melakukan uji coba. Diharapkan pada penelitian selanjutnya, penulisan kode menggunakan *threading* sehingga aplikasi dapat berjalan dengan lebih baik.
2. Aplikasi dapat dikembangkan dengan menambahkan optimasi pada algoritma *backtracking* dengan menggunakan teknik-teknik penyelesaian Sudoku lainnya, seperti *hidden pair* atau *hidden triple*.

DAFTAR PUSTAKA

- [1] D. Adler & Associates. 2012. "Sudoku Solver Manual Version 1.6.0". Dalam <http://www.dadler.net/>.
- [2] Lee, Wei-Meng. 2006. *Programming Sudoku*. New York: Springer-Verlag New York, Inc.
- [3] Levitin, Anany & Maria Levitin. 2011. *Algorithmic Puzzles*. New York: Oxford University Press, Inc.
- [4] Majumder, Abhisek dkk. 2010. "The Game of Sudoku-Advanced Backtrack Approach". Dalam *International Journal of Computer Science and Network Security*, VOL.10 No.8, August 2010.
- [5] Mujaddid, Sibghatullah. 2009. "Penerapan Algoritma Runut-Balik (Backtracking) Dalam Penyelesaian Permainan Sudoku". Dalam <http://informatika.stei.itb.ac.id/~rinaldi.munir/Smik/>.
- [6] Noviyanto, Fiftin. 2008. "Membangun Sistem Pembelajaran Pengenalan Bentuk Untuk Anak Berbasis Multimedia Dan Game Interaktif". Dalam <http://www.journal.uad.ac.id/>.
- [7] Sulun, Hafni Syaeful & Rinaldi Munir. 2010. "Pembangkit Teka-Teki Silang Dengan Algoritma Backtracking Serta Aplikasi Permainannya Yang Berbasis Web". Dalam *Jurnal Informatika*, Vol.4 No.2, Juli 2010.
- [8] Sutanto, Hendy. 2009. "Combination of BFS and Brute Force Algorithm Implementation in Futoshiki Puzzle Game". Dalam <http://informatika.stei.itb.ac.id/~rinaldi.munir/Smik/>.
- [9] Syam, Umi Fadilah Wardati. 2007. "Perumusan Sudoku Dengan Solusi Unik Menggunakan Algoritma Pencabangan Dan Pembatasan (Branch And Bound)". Dalam <http://informatika.stei.itb.ac.id/~rinaldi.munir/Smik/>.
- [10] Teneng, Joko Purwadi, & Erick Kurniawan. 2010. "Penerapan Algoritma Backtracking Pada Permainan Math Maze". Dalam <http://ti.ukdw.ac.id/>.
- [11] Wihandika, Randy Cahya, Nur Rosyid Mubtada'i & Rizky Yuniar H. 2011. "Penyelesaian Puzzle Sudoku Menggunakan Algoritma Genetika". Dalam <http://www.eepis-its.edu/>.