Evaluasi Algoritma Runut Balik dan *Simulated Annealing* pada Permainan Sudoku

Dyah Sulistyowati Rahayu[#1], Arie Suryapratama[#2], Azka Zulham Amongsaufa[#3], Bheli Isya Kurniawan Koloay[#4]

[#]*Jurusan Teknik Informatika, Universitas Pancasila*
Jalan Raya Srengseng Sawah, Jagakarsa, Jakarta Selatan

[1]dyah.s.rahayu@univpancasila.ac.id

[2]ariesuryap11@gmail.com

[3]azkazulham.amongsaufa@gmail.com

[4]bheliisya17@gmail.com

Abstract — **Sudoku is one of the popular puzzle games in many countries. How to solve this game is being the Non Polynomial Completeness problem. Because of that, a lot of research has been done to find the most effective algorithm completing Sudoku. This study presents the evaluation of algorithm in completing the Sudoku 2 x 2 and 3 x 3. The most common algorithm is brute force and backtracking. There is also simulated annealing algorithm. The evaluation is done by comparing the step of each algorithm to solve the Sudoku and also their running time. The result of this evaluation shows the differences of each algorithm characteristic in solving this Sudoku problem. By comparing the result of this study, the most effective algorithm can be chosen. Based on the test, concluded that the backtracking algorithm is more effective than the simulated annealing algorithm which the value reach 50%.**

Keywords— **backtracking, evaluation, simulated annealing, Sudoku.**

I. PENDAHULUAN

Sudoku merupakan permainan teka teki angka yang sangat popular. Media masa (Koran, majalah dan buku) banyak yang mencantumkan teka teki Sudoku di dalam konten hiburan atau kuisnya. Bahkan ada pula kompetisi yang dilaksanakan untuk menyelesaikan permainan Sudoku dengan cepat.

Terdapat banyak algoritma yang digunakan untuk menyelesaikan permainan Sudoku. Salah satu algoritma sederhana yaitu algoritma pencil and paper [1]. Algoritma ini menunjukkan cara menyelesaikan Sudoku dengan berbekal pensil dan kertas seperti pada umumnya orang menyelesaikan permainan ini. Algoritma Crook's pencil and paper bekerja berdasarkan teorema okupansi dan himpunan.

Pada bidang komputer, tentu yang paling sederhana adalah algoritma brute force dimana algoritma akan mencari penyelesaian satu persatu yang mungkin dari seluruh kemungkinan yang dimiliki [2]. Algoritma ini tentunya adalah algoritma yang kurang optimal dikarenakan harus mengulang kembali semua langkah dari awal jika tidak ditemukan solusi pada kotak tersebut.

Algoritma yang umum digunakan untuk menyelesaikan permainan teka teki salah satunya adalah algoritma runut balik (backtracking). Algoritma ini tidak akan mencari lagi langkah-langkah yang sudah pasti tidak mungkin menemukan solusi. Jika tidak ditemukan solusi pada suatu waktu, maka algoritma hanya akan mengulang satu langkah sebelumnya dan menggantinya dengan angka yang lain [3]. Algoritma runut balik cukup banyak digunakan di dalam Sudoku solver di berbagai situs karena sederhana namun cukup efektif.

Terdapat pula algoritma berbasis aturan (*rule based*), *Boltzman machine, simulated annealing*, dan algoritma genetika. Algoritma *Boltzman machine* ternyata kurang efektif dibanding algoritma runut balik [4]. Begitupun dengan algoritma genetika. Meskipun algoritma genetika sangat baik meyelesaikan kasus berjenis non *polynomial* (misalnya *travelling salesman problem*) namun algoritma genetika tidak mampu menyelesaikan Sudoku dengan optimal. Algoritma genetika memiliki tingkat konvergensi yang rendah dan sering terjebak pada nilai lokal minimum [5].

Ada pula pendekatan *heuristic* yang dilakukan melalui transformasi Sudoku menjadi graf. Permasalahan Sudoku diubah menjadi permasalahan *Hamiltonian cycle* dimana keduanya merupakan permasalahan *non polynomial*. Graf pada *Hamiltonian cycle* untuk Sudoku ini bisa berupa graf langsung dan graf tak langsung [6]

Simulated annealing pada penyelesaian permainan Sudoku bisa diterapkan dengan menganggap bahwa Sudoku adalah sebuah permasalahan optimalisasi dimana jumlah pelanggaran atas batasan yang diberikan menjadi minimal [7]. Dengan pendekatan *meta heuristic* tersebut, permainan Sudoku dapat diselesaikan dengan benar dan waktu yang relatif cepat [8][9].

Selain algoritma-algoritma yang telah disebutkan diatas, masih sangat banyak algoritma lain yang ditemukan untuk menyelesaikan permainan Sudoku ini. Penelitian ini akan fokus pada tiga buah algoritma yaitu *bruteforce*, runut balik, dan *simulated annealing*. Selain karakteristik masing-masing algoritma, penelitian ini akan melakukan implementasi dan menganalisa waktu komputasi masing-masing.

II. LANDASAN TEORI

Dalam sebuah perancangan algoritma, pada umumnya akan ditampilkan diagram alir serta *pseudocode* ataupun kode program dari algoritma tersebut. Analisa memori, kompleksitas serta waktu komputasi suatu algoritma ada pada tahap selanjutnya.

A. Algoritma Brute Force

Algoritma *brute force* merupakan algoritma yang akan menelusuri seluruh kemungkinan solusi dari suatu permasalahan. Pada Sudoku berukuran 3 x 3 area, atau 9 x 9 kotak maka akan ada 9 x 9 baris perulangan.

Potongan kode program dari algoritma *brute force* disajikan pada Gambar 1 dimana *n* merupakan ukuran banyaknya range pada Sudoku.

```
final int n = 3;
final int[][] field = new int[n*n][n*n];
for (int i = 0; i < n*n; i++)
    for (int j = 0; j < n*n; j++)
        field[i][j] = (i*n + i/n + j) % (n*n) + 1;
```

Gambar 1. Potongan kode program Sudoku dengan *Brute force* [10]

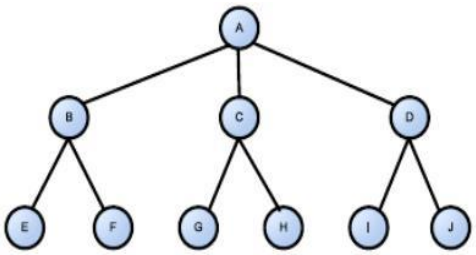
Kompleksitas algoritma *brute force* ini adalah O(n*n) dimana *n* adalah jumlah baris Sudoku. Pada Sudoku yang umum, ukuran Sudoku adalah 9 baris dan 9 kolom. Meskipun algoritma *brute force* terlihat sederhana, namun waktu eksekusi yang diperlukan akan sangat besar mengingat ada banyaknya kemungkinan angka untuk mengisi kotak kosong di dalam teka teki Sudoku. Untuk Sudoku dengan 51 kotak kosong maka akan ada 9 kemungkinan angka di setiap kotaknya sehingga jumlah iterasi yang dilakukan sebanyak 9^{51} yaitu 4.63×10^{48}.

B. Algoritma Runut Balik (Backtracking)

Algoritma runut balik merupakan pengembangan dari algoritma *brute force*. Jika tidak ditemukan solusi *s* pada langkah ke-*n*, maka algoritma akan mengubah solusi pada tahap ke *n*-1 dengan nilai *s*+1. Metode runut baliknya ini lah yang berbeda dengan *brute force*.

Algoritma runut balik merupakan bentuk tipikal dari algoritma rekursif dan berbasis pada *Depth First Search* (DFS) dalam mencari solusi yang tepat. Algoritma runut balik tidak akan memeriksa semua kemungkinan solusi yang ada dengan menerapkan *pruning* (pemotongan) terhadap garis solusi yang sudah pasti tidak mungkin. Karena kesederhanaan konsepnya maka algoritma ini banyak diterapkan pada bidang kecerdasan buatan dan berbagai bentuk permainan.

Misalkan pada pohon solusi yang ditampilkan pada Gambar 2. Jika solusi yang dicari adalah solusi E maka jalur yang harus ditempuh adalah A – B – E. Algoritma DFS yang diterapkan oleh algoritma runut balik ini akan mencari solusi di jalur kedalaman. Jika solusi yang akan dicari adalah F maka setelah melalui jalur A – B – E, program akan bergerak memeriksa solusi F.



Gambar 2. Contoh Pohon Solusi

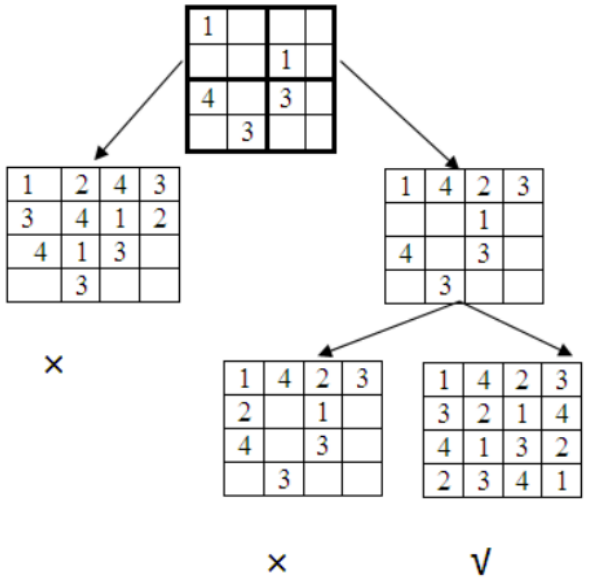
Properti umum pada algoritma runut balik adalah:

- Solusi dinyatakan dalam bentuk vektor solusi *S* dengan indeks *n*
  $S = (s_1, s_2, s_3, \ldots, s_n)$
- Fungsi pembangkit nilai *sn* dinyatakan dengan *T(n)*
- Fungsi pembatas atau disebut juga fungsi kriteria dinyatakan dengan $B(s_1, s_2, s_3, \ldots s_n)$

Fungsi pembatas menentukan apakah vektor solusi mengarah ke solusi yang benar. *B* akan bernilai *true* jika vektor solusi mengarah ke solusi yang benar sehingga pembangkit nilai dilanjutkan. Namun jika *B* bernilai *false* maka vektor solusi tersebut dibuang dan tidak dipertimbangkan lagi dalam ruang pencarian.

Algoritma runut balik akan mengisi kotak kosong pertama dengan angka "1". Jika angka "1" tersebut tidak sesuai dengan batasan Sudoku yang telah didefinisikan maka akan diganti dengan "2", "3", dan seterusnya sampai angka "9". Ketika sebuah angka yang diisikan dalam kotak sudah sesuai dengan batasan Sudoku, maka algoritma akan bergerak ke kotak kosong kedua dan mengulang proses pengisian mulai dari angka "1" sampai "9". Namun, jika angka "1" sampai "9" tidak sesuai maka algoritma akan melakukan runut balik ke langkah sebelumnya dan mengganti angka di kotak yang telah terisi tersebut [11]. Gambar 3 menampilkan contoh proses runut balik pada Sudoku dengan ukuran 4 x 4.

Pada langkah pertama, pengisian kotak di indeks (1,2) dengan angka 1 tidak memenuhi batasan karena pada baris tersebut sudah memiliki angka 1. Maka nilai angkanya ditingkatkan menjadi 2. Karena tidak ada batasan yang dilanggar maka pada indeks (1,2) diisikan angka 2. Pada indeks ke (1,3) angka 1 dan 2 tidak bisa diisikan karena telah terdapat pada baris 1 dan angka 3 telah terdapat pada kolom 3 sehingga angka 4 diisikan pada indeks tersebut.



Gambar 3. Proses runut balik pada Sudoku 4 x 4

Proses tersebut terus dilanjutkan sampai dengan indeks ke (3,3). Pada pengisian indeks ke (3,4) tidak ada angka yang memenuhi batasan Sudoku sehingga program melakukan runut balik ke indeks (3,2). Pada proses pengisian indeks (3,2), angka yang diisikan dinaikkan dari 1 menjadi 2, 3, dan 4 namun tidak ada yang sesuai sehingga program melakukan runut balik ke indeks (2,4). Proses runut balik akan terus dilakukan jika angka yang diisikan tidak ada yang memenuhi batasan. Pada contoh ini runut balik berhenti pada indeks (1,2) dengan mengubah angka 2 menjadi 4. Proses pengisian kemudian dilanjutkan sampai selesai.

Potongan kode program untuk algoritma runut balik ditampilkan pada Gambar 4. Kuncinya adalah proses pengecekan apakah angka yang diisikan valid atau tidak. Jika angka yang diisikan valid maka algoritma melanjutkan langkahnya dengan mengisi kolom selanjutnya. Namun jika isian angkanya tidak valid maka nilai angka di langkah sebelumnya akan ditinjau ulang. Jika pada langkah ke *n-1* tidak ditemukan isian yang valid, maka algoritma akan melangkah lagi ke langkah *n-2*, begitu seterusnya.

Pada dasarnya, algoritma runut balik merupakan algoritma *brute force* yang dijadikan efisien sehingga sudah pasti kompleksitasnya lebih rendah dibanding algoritma *brute force*. Untuk Sudoku dengan 51 kotak kosong maka akan ada kurang lebih 37660 iterasi [3]. Jumlah iterasi tersebut jauh lebih kecil dibanding iterasi pada algoritma *brute force* yang mencapai 4.63×10^{48} .

```
function backtrack(position){
   if (isEndOfGrid == true){ //Kotak kosong terisi. Solusi
ditemukan. Keluar
      return true;
   }

   foreach (x from 1 ... 9){
      grid[position] = x;
      if (gridIsValid == true){// Periksa apakah ada yang tidak
valid
         if (backtrack(nextPosition) == true){// lanjut ke kotak
kosong berikutnya
            return true; // Kotak kosong terisi. Solusi ditemukan.
Keluar
         }
      }
   }

   grid[position] = NULL;
 // mengosongkan isi kotak
   return false;
//Solusi tidak ditemukan. Runut balik.
}
```

Gambar 4. *Pseudocode* algoritma runut balik [3]

C. Algoritma Simulated Annealing

Algoritma *simulated annealing* merupakan algoritma yang prosesnya mengikuti proses *annealing* dalam bidang kimia, yaitu proses pengerasan kristal. Ketika suhu udara naik mencapai titik leleh maka material padat akan meleleh atau mencair. Sebaliknya, ketika suhu udara diturunkan maka lelehan material padat tersebut akan perlahan-lahan mengeras. Untuk meningkatkan tingkat kualitas kristal yang dihasilkan, maka dalam proses penurunan suhu tersebut ada kalanya suhu dinaikkan sedikit kemudian diturunkan kembali.

Algoritma *simulated annealing* dapat pula dijelaskan sebagai algoritma *meta heuristik* yang diterapkan pada kasus minimalisasi. Algoritma *simulated annealing* akan mengevaluasi solusi awal dan berusaha memperbaiki solusi awal tersebut secara iteratif. Solusi tujuan yang ditemukan adalah nilai fungsi tujuan yang paling minimum.

Solusi pada iterasi ke-(k+1) adalah satu diantara alternatif berikut ini:

- $s_{k+1} = s^*$ jika $f^* < f_k$
- $s_{k+1} = s^*$ jika $f^* \geqslant f_k$ dan $r = 1$
- $s_{k+1} = s_k$

dengan s_{k+1} adalah solusi pada iterasi ke-($k+1$), s^* adalah solusi yang dicari pada iterasi ke-($k+1$), f^* adalah nilai fungsi yang dihasilkan dari iterasi ke-($k+1$), f_k adalah nilai fungsi pada iterasi sebelumnya, dan r adalah bilangan acak Bernoulli dengan peluang $p = f_k / (k \times f^*)$. Meskipun besar nilai peluang p tidak seperti yang disebutkan pada versi asli bilangan Bernoulli, namun esensi perhitungannya sama

yaitu nilai peluang *p* akan semakin kecil untuk iterasi *k* yang semakin besar.

Hal yang paling penting dari algoritma *simulated annealing* ini adalah cara penentuan solusi s_{k+1}. *Simulated annealing* tidak bekerja secara *greedy* dimana akan selalu memilih solusi yang memiliki nilai fungsi lebih kecil. *Simulated annealing* mempertimbangkan pula nilai peluang pada setiap iterasinya.

Nilai peluang ini akan menjaga algoritma *simulated annealing* tidak berhenti pada jebakan nilai lokal minimum layaknya pada proses *annealing* kristal. Algoritma *simulated annealing* memungkinkan mengambil nilai fungsi yang lebih buruk, dalam kata lain tidak lebih minimum dibanding nilai fungsi sebelumnya sehingga bisa menemukan lembah minimum lokal di berbagai titik. Dengan memiliki ruang pencarian yang luas ini maka *simulated annealing* dapat memberikan solusi yang baik dan memuaskan.

Simulated annealing dapat langsung diterapkan pada permasalahan optimasi apapun dengan mendefinisikan operator ketetanggaan dan fungsi evaluasi yang sesuai. Dimulai dari mendefinisikan sebuah kandidat solusi, maka eksplorasi pada semua kemungkinan solusi dilakukan dengan melakukan iterasi dengan operator ketetanggaan tersebut.

Sebuah kandidat solusi akan diterima jika:

- *s'* (tetangga) memiliki fungsi biaya yang lebih baik dari *s* (kandidat solusi)
- probabilitasnya diterima dengan persamaan $exp(-\delta/t)$ dimana δ adalah nilai yang berubah pada fungsi biaya dan *t* (temperatur) adalah parameter control.

Simulated annealing hanya akan menganggap pengisian angka tersebut benar jika telah menemui salah satu syarat diterimanya solusi tersebut.

Dijelaskan pula secara singkat bahwa algoritma *simulated annealing* adalah algoritma yang langkahnya dilakukan berdasarkan nilai probabilitas terbesar dari keberhasilan suatu kemungkinan. Pendekatan probabilitas ini digunakan untuk menemukan perkiraan nilai global optimum dalam area pencarian yang luas (biasanya area bersifat diskrit). Fokus algoritma *simulated annealing* ini adalah untuk memperkirakan nilai global optimum dengan waktu yang singkat, dan bukan mendapatkan nilai pasti yang hanya bersifat lokal minimum [12].

Pseudocode untuk algoritma *simulated annealing* disajikan pada Gambar 5.

```
Let s = s0
For k = 0 through kmax (exclusive):
    T ← temperature(k / kmax)
    Pick a random neighbour, snew ← neighbour(s)
    If P(E(s), E(snew), T) ≥ random(0, 1),
    move to the new state:
        s ← snew
Output: the final state s
```

Gambar 5. *Pseudocode* algoritma *Simulated Annealing* [11]

TABEL I
KARAKTERISTIK, KELEBIHAN, DAN KELEMAHAN ALGORITMA UNTUK MENYELESAIKAN SUDOKU

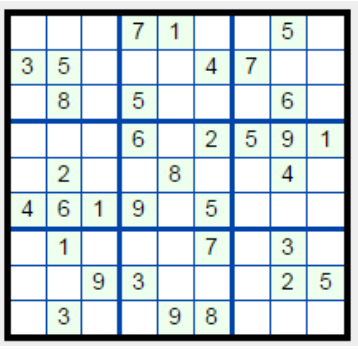
Algoritma	Karakteristik	Kelebihan	Kelemahan
Brute force	• Mencari semua kemungkinan • Langkah terurut mulai dari kotak pertama di baris pertama, kolom pertama • Jika gagal, mengulang dari langkah pertama	• Pasti menemukan solusi • Kode sederhana • Waktu eksekusi tidak tergantung tingkat kesulitan soal Sudoku	• Waktu yang relative lama
Runut Balik	• Mengeliminasi ketidakmungkinan • Langkah terurut mulai dari kotak pertama di baris pertama, kolom pertama • Jika gagal, mengulang satu langkah sebelumnya	• Pasti menemukan solusi • Waktu relative lebih cepat dari brute force	• Mungkin masih ada pendekatan lain yang lebih cepat
Simulated annealing	• Berdasarkan nilai probabilitas setiap langkah • Langkah pertama mengisi kotakkosong yang dipilih secara acak	• Pasti menemukan solusi • Waktu relative cepat	• Pemilihan secara random pada langkah pertama bisa sangat mempengaruhi efektivitas langkah selanjutnya

III. HASIL DAN PEMBAHASAN

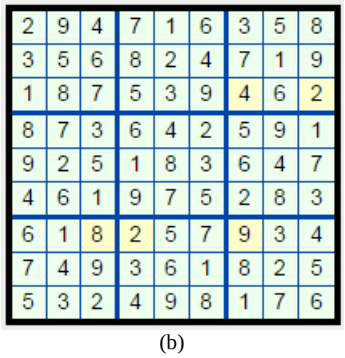
Karakteristik, kelebihan dan kelemahan algoritma *brute force*, runut balik, serta *simulated annealing* dijelaskan pada Tabel I. Metode runut balik merupakan pengembangan dari metode *brute force* dimana metode tersebut pasti akan menemukan solusi dari Sudoku selama permasalahan Sudokunya benar.

Algoritma *simulated annealing* menggunakan bentuk pendekatan yang berbeda dibanding *brute force* maupun runut balik. Algoritma *simulated annealing* ini menetapkan langkah pertama secara acak sehingga performanya tergantung pula pada hasil langkah pertama tersebut. Algoritma ini menerapkan perhitungan probabilitas untuk menentukan langkah terbaiknya di dalam menemukan solusi Sudoku.

Untuk contoh Sudoku berukuran 3 x 3 area seperti pada gambar 6 dan gambar 7, terdapat 30 kotak yang sudah terisi dan 51 kotak kosong. Setiap kotak kosong terdapat 9 kemungkinan angka yang bisa diisikan. *Brute force* akan melakukan iterasi sebanyak 9^{51} yaitu 4.63 x 10^{48} dimana nilai tersebut sangat besar. Sedangkan dengan algoritma runut balik, Sudoku tersebut dapat diselesaikan dengan 37660 iterasi.

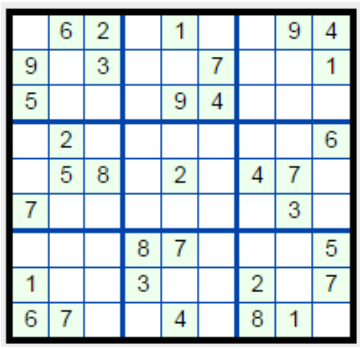


(a)



(b)

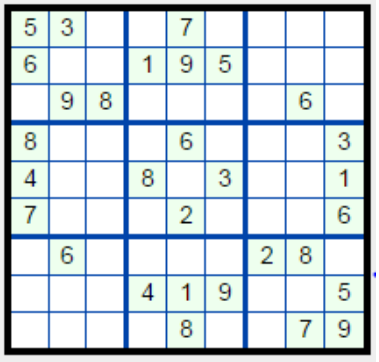
Gambar 6. (a) Sudoku dengan 48 kotak kosong dan (b) solusi yang dihasilkan



(a)

(b)

Gambar 7. (a) Sudoku lain dengan 48 kotak kosong dan (b) solusi yang dihasilkan



(a)

(b)

Gambar 8. (a) Sudoku dengan 51 kotak kosong dan (b) solusi yang dihasilkan

Perbedaan jumlah iterasi antar kedua metode ini sangat besar sehingga sudah pasti perbedaan waktu eksekusinya pun besar. Oleh karena hal itu, pengujian yang dilakukan untuk mengetahui waktu eksekusi algoritma hanya membandingkan antara algoritma runut balik dan *simulated annealing*.

Lima buah pengujian pertama menggunakan Sudoku dengan 48 angka yang harus diisikan. Contoh Sudoku yang digunakan pada pengujian ini dengan 48 angka kosong ada pada Gambar 6 (a) dan Gambar 7 (a). Lima buah pengujian selanjutnya yaitu mengisi 51 angka pada kotak yang kosong seperti pada Gambar 8 (a). Jumlah kotak kosong yang berbeda untuk menguji pengaruh jumlah solusi yang dicari dengan standar deviasi tiap pengujian.

Gambar 6(b), 7(b), dan 8(b) menampilkan hasil dari algoritma runut balik dan simulated annealing. Solusi yang dihasilkan oleh kedua algoritma telah benar dan kedua algoritma dapat menemukan solusi yang benar tersebut.

Tabel II menampilkan waktu eksekusi dari 10 kali percobaan yang masing-masing menggunakan algoritma runut balik dan *simulated annealing*. Algoritma runut balik dan *simulated annealing* dibandingkan karena kedua algoritma tersebut dinyatakan sebagai algoritma yang efektif pada jenis permainan Sudoku. Algoritma *brute force* tidak digunakan sebagai pembanding karena sudah pasti waktu kerjanya akan lebih lama dibanding dengan algoritma runut balik.

Tabel II juga mencantumkan nilai rata – rata dan standar deviasi dari percobaan yang dilakukan. Nilai rata-rata memudahkan dalam melakukan perbandingan antara kedua algoritma sedangkan nilai standar deviasi memudahkan melakukan perbandingan antara percobaan yang berbeda dalam kasus yang sama. Waktu eksekusi ditampilkan dalam hitungan detik.

TABEL II.
WAKTU EKSEKUSI ALGORITMA RUNUT BALIK DAN SIMULATED ANNEALING

No	Jumlah Kotak Kosong	Waktu Penyelesaian (Detik)	
		Backtracking	Simulated Annealing
1	48	0.0120010	0.0160009
2	48	0.0099991	0.019002
3	48	0.0100091	0.0170009
4	48	0.0110008	0.018001
5	48	0.0109999	0.0170001
Rata-rata		**0.010802**	**0.017401**
Standar deviasi		**0.000747**	**0.00102**
6	51	0.0140008	0.0180010
7	51	0.0120010	0.0190010
8	51	0.0119998	0.0280011
9	51	0.0130018	0.0255999
10	51	0.0158701	0.0332000
Rata-rata		**0.013375**	**0.024761**
Standar deviasi		**0.001452**	**0.00568**

Rata-rata waktu eksekusi untuk Sudoku 48 kotak kosong adalah 0.0108 detik untuk algoritma runut balik dan 0.0174 detik untuk algoritma *simulated annealing*. Sedangkan untuk Sudoku 51 kotak kosong, rata-rata waktu eksekusi untuk algoritma runut balik adalah 0.0133 detik dan untuk algoritma *simulated annealing* adalah 0.0247 detik. Dari data tersebut dapat diketahui bahwa algoritma runut balik lebih cepat menemukan solusi dibanding algoritma *simulated annealing*. Waktu eksekusi algoritma runut balik lebih cepat 0.007 detik untuk Sudoku 48 kotak kosong dan lebih cepat 0.011 detik untuk Sudoku 51 kotak kosong dibandingkan algoritma *simulated annealing*. Dari data tersebut dapat disimpulkan bahwa perbedaan waktu antar kedua algoritma cukup signifikan mencapai 50% dengan perhitungan waktu eksekusi algoritma runut balik dibagi waktu eksekusi algoritma *simulated annealing* kali 100%. Untuk Sudoku 48 kotak kosong performanya adalah (0.010802/0.017401) x 100% yaitu 62%. Sedangkan untuk Sudoku 51 kotak kosong performanya adalah (0.013375/0.024761) x 100% yaitu 54%.

Perlu dilihat pula bahwa peningkatan waktu eksekusi algoritma runut balik dengan tingkat isian yang berbeda pun lebih kecil dibanding peningkatan waktu eksekusi pada algoritma *simulated annealing*. Hal tersebut menunjukkan bahwa algoritma runut balik lebih stabil dibanding algoritma *simulated annealing*.

Standar deviasi rata-rata dari 10 percobaan tersebut sebesar 0.001 detik untuk algoritma runut balik dan 0.003 detik untuk algoritma *simulated annealing*.Standar deviasi pada kedua algoritma sangat kecil menunjukkan bahwa hasilnya konvergen dan langkah acak pada algoritma *simulated annealing* tidak berpengaruh besar terhadap waktu eksekusi penyelesaian Sudoku.

Peningkatan waktu eksekusi antara jumlah kosong yang berbeda tidak terlalu signifikan yaitu sebesar 0.003 detik untuk algoritma runut balik dan 0.007 detik untuk algoritma *simulated annealing*. Untuk itu akan dilakukan pengujian lebih lanjut dengan tingkat kesulitan yang berbeda. Pengujian ini diperlukan untuk memastikan pengaruh adanya tingkat kesulitan yang berbeda terhadap waktu eksekusi algoritma. Jika pada algoritma *brute force*, perbedaan tingkat kesulitan Sudoku ini tidak akan mempengaruhi waktu eksekusi programnya. Oleh karena itu, penelitian ini perlu membuktikan bahwa hal tersebut tidak berlaku pada algoritma runut balik dan *simulated annealing*.

Pengujian selanjutnya dilakukan berdasarkan empat buah Sudoku hasil dari sebuah generator dengan tingkat kesulitan yang berbeda [13]. Ada lima buah tingkatan yaitu sangat mudah (*extremely easy*), mudah (*easy*), sedang (*medium*), sulit (*difficult*), dan sangat sulit (*evil*). Namun pada penelitian ini digunakan 4 tingkatan yang berbeda mulai dari sangat mudah (*extremely easy*) sampai sulit (*difficult*). Gambar 9-12 menampilkan salah satu Sudoku dari tiap

tingkatan yang digunakan sebagai bahan pengujian dengan angka yang tercetak besar adalah solusi yang diberikan dan angka yang tercetak kecil adalah solusi yang harus dicari oleh kedua algoritma.

Pada generator tersebut telah ditetapkan batasan tingkat kesulitan yang salah satunya ditentukan berdasarkan jumlah kotak kosong yang harus diisi angka. Untuk tingkat sangat mudah, kotak kosongnya kurang dari 32. Tingkat mudah kotak kosong berjumlah 32 sampai 45. Tingkat sedang kotak kosongnya berjumlah 46 sampai 52. Sedangkan tingkat sulit kotak kosongnya berjumlah 53 sampai 55.

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

Gambar 9. Sudoku dengan tingkat kesulitan: mudah (extremely easy)

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

Gambar 10. Sudoku dengan tingkat kesulitan: mudah (easy)

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

Gambar 11. Sudoku dengan tingkat kesulitan: sedang (medium)

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

Gambar 12. Sudoku dengan tingkat kesulitan: sulit (difficult)

Tabel III menampilkan waktu penyelesaian Sudoku dengan 4 tingkat kesulitan yang berbeda oleh kedua algoritma. Masing-masing tingkat kesulitan diujicoba dengan 5 buah soal Sudoku yang berbeda supaya dapat diambil rata-rata waktu eksekusinya. Nilai yang akan dijadikan dasar analisa adalah nilai rata-rata waktu eksekusi dalam satuan detik.

Pada algoritma runut balik, tingkat kesulitan sangat mudah mampu diselesaikan pada 0.004 detik. Pada tingkat kesulitan mudah dan sedang memerlukan waktu 0.019 detik dan 0.013 detik. Sedangkan untuk tingkat kesulitan sulit memerlukan waktu 0.032 detik. Dari hasil tersebut terlihat bahwa peningkatan waktu antar tingkat kesulitan berbeda cukup signifikan. Tingkat sangat mudah ke mudah perbedaan waktunya mencapai 4 kali lipat. Dari tingkat kesulitan sedang ke sulit, peningkatan waktunya hampir mencapai 3 kali lipat. Namun untuk tingkat kesulitan mudah ke sedang tidak ada perbedaan yang signifikan.

TABEL III
WAKTU EKSEKUSI ALGORITMA RUNUT BALIK DAN SIMULATED ANNEALING PADA SUDOKU DENGAN TINGKAT KESULITAN YANG BERBEDA

No	Tingkat Kesulitan	Jumlah kotak kosong	Waktu Penyelesaian (Detik)	
			Back tracking	Simulated Annealing
1	Extremely Easy	< 32	0.004	0.003
			0.003	0.002
			0.004	0.003
			0.005	0.003
			0.004	0.002
	Rata – rata		**0.004**	**0.0046**
2	Easy	32 – 45	0.020	0.023
			0.023	0.020
			0.019	0.021
			0.017	0.024
			0.018	0.022
	Rata – rata		**0.0194**	**0.022**
3	Medium	46 – 52	0.021	0.024
			0.012	0.016
			0.013	0.014
			0.009	0.012
			0.010	0.010
	Rata – rata		**0.013**	**0.015**
4	Difficult	53 – 55	0.026	0.033
			0.041	0.030
			0.024	0.028
			0.035	0.039
			0.037	0.029
	Rata – rata		**0.0326**	**0.0318**

Pada algoritma *simulated annealing* berlaku hal yang hampir sama. Peningkatan waktu antara tingkat kesulitan sangat mudah ke mudah mencapai 4 kali lipat, sedangkan dari tingkat kesulitan sedang ke sulit lebih dari 2 kali lipat.

Dari hasil tersebut terlihat bahwa pemeringkatan yang dibuat sudah cukup mewakili tingkat kesulitan yang sebenarnya yang dibuktikan dengan meningkatnya waktu eksekusi yang diperlukan oleh masing-masing algoritma.

Algoritma runut balik maupun algoritma *simulated annealing* memiliki rata-rata waktu eksekusi yang hampir sama untuk setiap jenis peringkat kesulitan Sudoku. Perbedaan waktu eksekusi untuk tingkat kesulitan sangat mudah (*extremely easy*) adalah 0.0006 detik, untuk tingkat kesulitan mudah (*easy*) sebesar 0.008 detik, untuk tingkat kesulitan sedang (*medium*) sebesar 0.002 detik, dan untuk tingkat kesulitan sulit (*difficult*) 0.008 detik. Nilai perbedaan terbesarnya mencapai 42% dari waktu eksekusi aslinya.

Gambar 13 (a) menampilkan soal untuk peringkat soal sangat mudah (*extremely easy*) dengan 30 kotak kosong yang harus diisi dan terbukti dapat diselesaikan oleh kedua algoritma dengan baik yang ditampilkan pada Gambar 13 (b). Hasil tersebut sama dengan Gambar 9 yang merupakan kunci dari generator Sudoku.

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

(a)

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

(b)

Gambar 13. (a) Sudoku dengan peringkat kesulitan: sangat mudah (*extremely easy*) dan (b) solusi yang dihasilkan

Gambar 14 (a) menampilkan soal untuk peringkat soal mudah (*easy*) dengan 41 kotak kosong yang harus diisi dan terbukti dapat diselesaikan oleh kedua algoritma dengan baik yang ditampilkan pada Gambar 14 (b). Hasil tersebut sama dengan Gambar 10 yang merupakan kunci dari generator Sudoku.

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

(a)

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

(b)

Gambar 14. (a) Sudoku dengan peringkat kesulitan: mudah (*easy*) dan (b) solusi yang dihasilkan

Gambar 15 (a) menampilkan soal untuk peringkat soal sedang (*medium*) dan terbukti pula dapat diselesaikan oleh kedua algoritma dengan baik yang ditampilkan pada Gambar 15 (b). Hasil tersebut sama dengan Gambar 11 yang membuktikan bahwa solusi yang dihasilkan merupakan solusi yang benar.

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

(a)

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

(b)

Gambar 15. (a) Sudoku dengan peringkat kesulitan sedang dan (b) solusi yang dihasilkan

Gambar 16 (a) menampilkan soal untuk peringkat soal sulit (*difficult*) dan terbukti pula dapat diselesaikan oleh kedua algoritma dengan baik yang ditampilkan pada Gambar 16 (b). Hasil tersebut sama dengan Gambar 12 yang membuktikan bahwa solusi yang dihasilkan merupakan solusi yang benar.

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

(a)

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

(b)

Gambar 16. (a) Sudoku dengan peringkat kesulitan sulit dan (b) solusi yang dihasilkan

IV. KESIMPULAN

Algoritma *brute force* merupakan algoritma primitif yang mencari solusi dari semua kemungkinan secara berurutan. Meskipun hasilnya dipastikan kebenarannya namun waktu eksekusi yang diperlukan akan lama. Oleh karena itu, dikembangkanlah algoritma runut balik yang membuat perulangan pada algoritma *brute force* lebih efektif.

Algoritma runut balik terbukti lebih efektif dibanding algoritma *simulated annealing* dalam menyelesaikan permainan Sudoku dimana nilai efektifitasnya mencapai 50%. Kedua algoritma memiliki karakteristik yang hampir sama dalam menyelesaikan permainan Sudoku dengan berbagai tingkat kesulitan yang terbukti dengan jumlah

peningkatan waktu yang hampir sama antar tingkat kesulitan yang berbeda pada kedua algoritma.

Meskipun algoritma *simulated annealing* menggunakan pendekatan langkah pertama secara acak dan langkah selanjutnya berdasarkan bobot probabilitas namun algoritma ini mampu menemukan solusi dengan benar dan konvergen yang dibuktikan dengan nilai standar deviasi yang kecil.

Karena efektifitas algoritma runut balik tersebut lah maka algoritma ini sering dipakai dalam menyelesaikan permainan teka teki, salah satunya Sudoku.

DAFTAR PUSTAKA

[1] Crook, J. F., A Pencil and paper algorithm for solving Sudoku puzzles. Notices of the AMS. Volume 56 Number 4 pp.460-468. April 2009

[2] Kumari, T., Preety, Y., Lavina. Study of Brute Force and heuristic Approach to Solve Sudoku. International Journal of Emerging Trends and Technology in Computer Science. Vol. 4 Issue 6(2), September-October 2015

[3] Lee, Y. Y., Solving Sudoku by Backtracking. Code My Road Programming Projects and Articles. 1 May 2014 (diakses 16 februari 2017) dari https://codemyroad.wordpress.com/2014/05/01/solving-sudoku-by-backtracking/

[4] Berggren, P., Nellson, D., A Study of Sudoku solving algorithm. Bachelor Thesis at CSC. Stockholm. 2011.

[5] Weiss, M. J. Genetic Algorithm and Sudoku. Proceeding of the 42nd Midwest Instruction and Computing Symposium (MICS 2009). 17-18 April 2009. South Dakota School of Mines and Technology.

[6] Haythorpe, M., Reducing the generalized Sudoku probem to the Hamiltonian cycle problem, AKCE International Journal of Graph and Combinatorics, Elsevier, Vol. 13, pp. 272-282 2016.

[7] Sartono, B., Penggunaan algoritma simulated annealing untuk menyelesaikan teka-teki binary dan Sudoku, Forum Statistika dan Komputasi, Vol. 16 No.2 pp. 1-6, Oktober 2011.

[8] Lewis, R., Metaheuristics can solve Sudoku puzzle, Journals of Heuristics, Vol. 13 No. 4 pp. 387-401, 2007.

[9] Dehkordi, Z., Zamanifar, K., Dasjerdi, A., Aghaee, N., Sudoku using parallel simulated annealing, ICSI 2010 Part II LNCS 6146 pp. 461-467. 2010

[10] no name, Sudoku solving algorithm, Wikipedia: The Free Encyclopedia, https://en.wikipedia.org/wiki/Sudoku_solving_algorithms diakses tanggal 16 Februari 2017.

[11] Ekne, S., Gylleus, K., Analysis and comparison of solving algorithms for Sudoku: Focusing on effiency.Degree project in computer science. 2015. Diakses dari http://www.diva-portal.se/smash/get/diva2:811020/FULLTEXT01.pdf pada tanggal 16 februari 2017.

[12] No nome. Simulated Annealing. Wikipedia, the free encyclopedia. Diubah terakhir 7 Desember 2016, diakses tanggal 17 februari 2017

[13] Team of Xiang-Sun Zhang Research Group, Sudoku Puzzles Generating: from easy to evil, Operation Research and Bioinformatics Group at Academy of mathematics and Systems Science, diakses dari http://zhangroup.aporc.org/images/files/Paper_3485.pdf pada tanggal 28 Februari 2017.