Penyelesaian Permainan Sudoku Menggunakan Algoritma *Backtracking* Berbasis *Artificial Intelligence*

Alya Aulia Hanafi[1], Naseh Hibban[2], Fawwaz Muhammad Zulfikar[3], Faisal Dharma Adhinata[4,*]

[1,2,3,4] Program Studi Rekayasa Perangkat Lunak, Institut Teknologi Telkom Purwokerto, Indonesia
Email: [1]19104004@ittelkom-pwt.ac.id, [2]19104003@ittelkom-pwt.ac.id, [3]19104058@ittelkom-pwt.ac.id, [4,*]faisal@ittelkom-pwt.ac.id
[*)] faisal@ittelkom-pwt.ac.id

Abstrak−Kebutuhan teknologi untuk mempermudah dan menyelesaikan suatu permasalahan menjadi efisien sangat diperlukan di era modern ini. Salah satu bidang ilmu yang sering digunakan untuk memecahkan masalah dengan meniru kebiasaan manusia adalah Artificial Intelligence (AI). Permainan Sudoku merupakan permainan dengan jenis teka teki logika. Sudoku termasuk pada permasalahan NP – complete, sehingga sulit atau bahkan tidak bisa untuk diselesaikan dengan waktu yang sama. Dari permasalahan tersebut, dibutuhkan sebuah algoritma AI untuk menyelesaikan Permainan Sudoku. Salah satu algoritmanya adalah *Backtracking* (runut-balik). Hasil percobaan menunjukkan semakin banyak jumlah proses *backtracking* berbanding lurus dengan semakin banyaknya waktu yang dibutuhkan untuk menyelesaikan permainan Puzzel Sudoku. Kemudian, tingkat akurasi pada program sudoku solving dengan algortima *backtracking* mencapai 100% dengan kotak kosong berjumlah 43.

Kata Kunci: Artificial Intelligence, Sudoku, NP – complete, Runut-balik, Puzzel

Abstract− *The need for technology to simplify and solve a problem to be efficient is indispensable in this modern era. One of the fields of science that is often used to solve problems by imitating human habits is Artificial Intelligence (AI). Sudoku game is a game with a type of logic puzzle. Sudoku has included in the NP-complete problem, so it is difficult or even impossible to solve it at the same time. From these problems, an AI algorithm is needed to solve the Sudoku Game. One of the algorithms is Backtracking. The experimental results show that the more the number of backtracking processes is directly proportional to the more time it takes to complete the Puzzle Sudoku game. Then, the level of accuracy in the sudoku solving program with the backtracking algorithm reaches 100% with 43 empty boxes.*

Keywords: Artificial Intelligence, Sudoku, NP – complete, *Backtracking*, Puzzle

1. PENDAHULUAN

Pada era modern kini, banyak perkembangan yang terjadi di dunia teknologi. Salah satunya adalah Arificial Intelegence atau Kecerdasan Buatan.. Kecerdasan Buatan atau Artificial Intelligence (AI) merupakan simulasi dari kecerdasan yang ada pada manusia yang diimplementasikan di dalam hardware dan deprogram untuk berfikir. Dengan kata lain AI adalah system computer yang dapat melakukan pekerjaan yang umunya memerlukan kecerdasan dan tenaga manusia untuk menyelesaikanya [1].

Peran Kecerdasan Buatan atau AI pada era modern ini sangat dibutuhkan untuk mempermudah dan menyelesaikan suatu permasalahan dengan lebih efisien dan cepat. Kecerdasan Buatan sudah banyak dipakai dalam aplikasi sebagai fitur yang bertujuan untuk mengatasi berbagai masalah sesuai dengan sasaran aplikasi. Salah satu contoh permasalahanya yang dapat ditangani oleh Kecerdasan Buatan adalah penyelesaian Permainan Sudoku secara cepat dan tepat.

Permainan Sudoku merupakan permainan dengan jenis teka teki logika. Permainan ini sudah terkenal di kalangan usia muda hingga orang tua [2]. Permainan Sudoku banyak diminati karena melatih otak dalam hal berlogika Misi dari permainan ini adalah dengan mengisikan angka-angka dari 1 sampai 9 ke dalam jarring – jarring 9x9 yang terdiri dari 9 kotak 3x3 tanpa ada angka yang berulang pada satu baris, kolom atu kotak. Sudoku mendorong para pemain berpikir menggunakan logika secara teliti. Dalam mengisi angka diperlukan keakuratan yang tinggi karena bila sembarangan memasukan angka , maka permainan tidak dapat diselesaikan.

Permainan Sudoku memiliki prinsip keunikan dan menciptakan banyak kombinasi angka yang membuat permainan Sudoku menghasilkan banyak kemungkinan. Dibutuhkan kesabaran dan ketajaman akurasi dalam mengisi angka pada permainan Sudoku ini. Sudoku termasuk pada permasalahan NP – complete, sehingga sulit atau bahkan tidak bisa untuk diselesaikan dengan waktu yang sama.

Dari masalah diatas, dibutuhkan sebuah AI yang berisi algoritma yang tepat untuk menyelesaikan Permainan Sudoku. Salah satu algoritmanya adalah *Backtracking* (runut-balik). Algoritma ini akan merunut kemungkinan angka yang hanya mengarah pada solusi atau penyelesaian. Jadi penyelesaian atau solusi dapat ditemukan dengan runut penelusuran yang sedikit dan efisien karena tidak memeriksa semua kemungkinan angka

penyelesaian yang ada. Dengan diterapkanya AI ini, Permainan Sudoku akan diselesaikan dengan lebih cepat dan efisien.

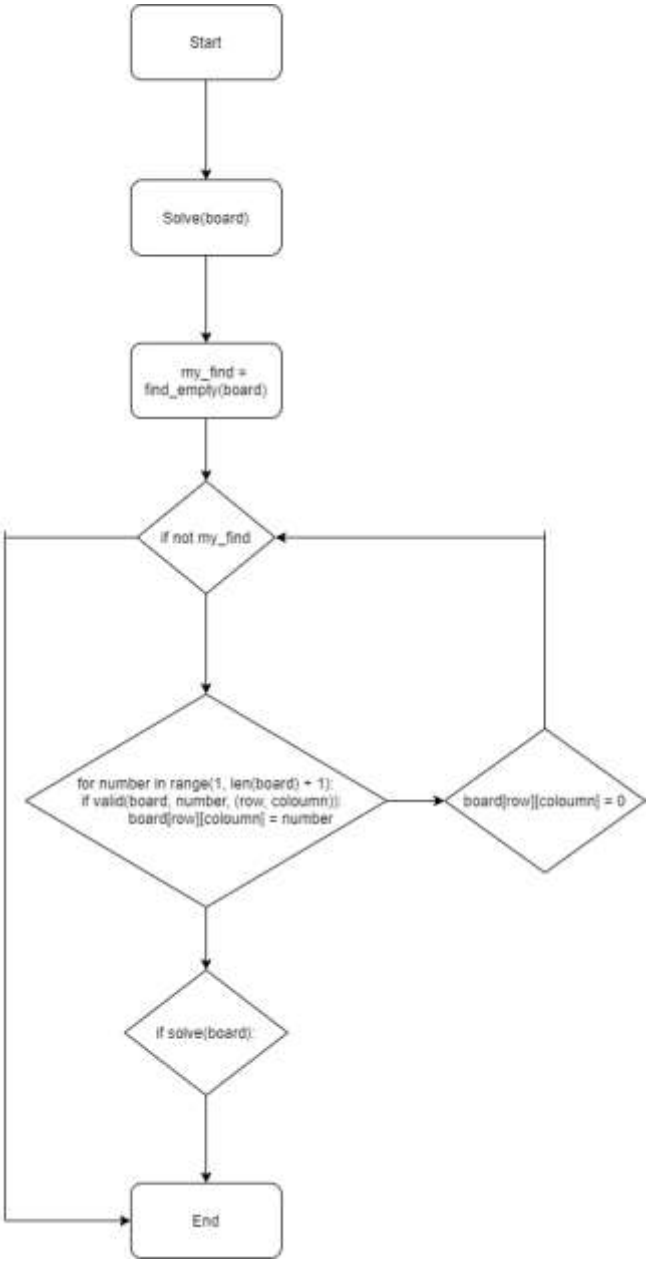
2. METODE PENELITIAN

2.1 Kajian Literatur

Sudoku merupakan permainan puzzle yang memiliki konsep Latin Square. Sudoku memiliki bagian-bagian yang diberi angka, jumlah kotak pada sudoku terdiri 3X3 yang masing-masing kotak memiliki Sembilan kotak yang lebih kecil [3].

Backtracking (Runut-balik) adalah algoritma pencarian solusi berbasis pada DFS yang dapat mengatasi sebuah permasalahan [4]. Algoritma *Backtracking* banyak diterapkan untuk program games, salah satunya game sudoku. Pada *backtracking*, hanya ada pilihan yang mengarah ke solusi eksplorasi, pilihan yang tidak mengarah ke solusi tidak dipertimbangjan lagi.

2.2 Metode



Gambar 1. Flowchart Algoritma solving permainan sudoku

Berikut merupakan penjabaran dari alur Flowchart pada Gambar 1.

1) Pertama proses masuk ke fungsi Solve dengan parameternya board
2) Tahap berikutnya pada pendeklarasian my_find dengan fungsi find empty (board) terdapat percabangan, jika kondisi not my find maka program akan berakhir atau bisa dibilang sudoku terselesaikan. Fungsi find_empty digunakan untuk mengecek apakah coloum dan row masih ada yang kosong atau tidak.
3) Tahap berikutnya, else dari percabangan diatas adalah jika row dan coloum = my_find maka penelusuran akan dilakukan dengan algoritma

   ```
   for number in range(1, len(board) + 1):
       if valid(board, number, (row, coloumn)):
           board[row][coloumn] = number
   ```

   pada algoritma ini terdapat fungsi valid yang berguna untuk mengecek kebenaran dari kemungkinan angka solusi yang diberikan oleh fungsi solve.
4) Tahap selanjutnya, terdapat percabangan jika angka yang dihasilkan dari algoritma diatas solve maka return true yang artinya angka tersebut merupakan solusi. Jika tidak maka return false dan Kembali ke pencarian angka solusi.

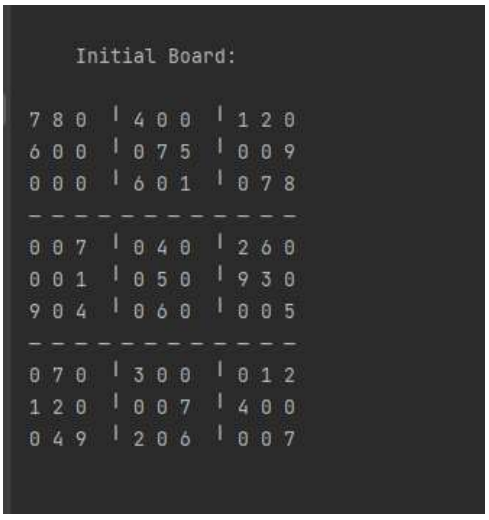
Metode yang digunakan pada penelitian ini adalah Backtraking. *Backtracking* adalah salah satu Algoritma yang diperlukan untuk mengatasi sebuah masalah yang bersangkutan dengan pemenuhan terbatas, misalnya permainan puzzle pada umumnya [5]. Pada penyelesaian suatu masalah Algoritma *Backtracking* memiliki beberapa cara yang dapat menciptakan beberapa berbagai peluang solusi. Saat proses penyelesaian masalah Algoritma *backtracking* melakukannya dengan bertahap dari peluang satu ke peluang lainnya

Pada penyelesaian permainan sudoku, Algoritma Backtraking akan mencari solusi angka yang tepat untuk mengisi kotak kotak kosong yang ada sudoku. Algoritma Backtraking akan melakukan penelusuran hingga menemukan solusi, jika tidak menemukan pada sebuah node kandidat, maka Algoritma akan kembali dan melakukan penelusuran pada node kandidat kemungkinan yang lain.

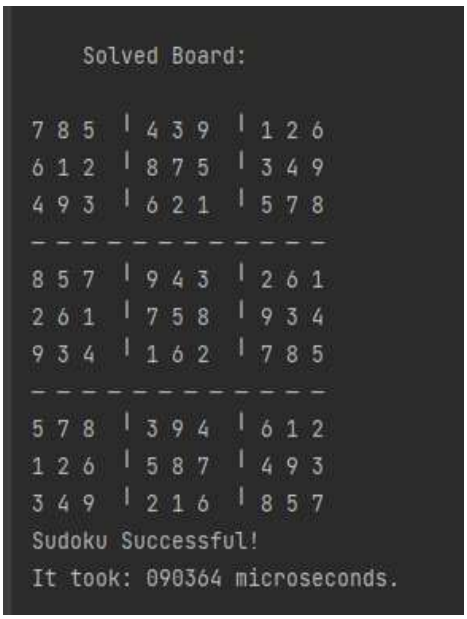
3. HASIL DAN PEMBAHASAN

3.1 Hasil Running Program Sudoku Solving

Pengujian ini menggunkan aplikasi pycharm dan menggunkan bahasa pemrograman python. Pada Gambar 2 menunjukan initial board atau Puzzel sudoku yang belum dikerjakan yang merupakan hasil run dari kode program sudoku solving



Gambar 2. Puzzel Sudoku



Gambar 3. Puzzel sudoku yang terselesaikan

Pada Gambar 3 menunjukan puzzle sudoku yang sudah terselesaikan menggunakan algoritma *backtracking* yang merupakan hasil kode program sudoku solving. Puzzel Sudoku pada program ini memiliki ukuran 3 x 3 kotak yang didalmnya terdapat angka dengan ukuran 3 x 3.

3.2 Hasil Percobaan

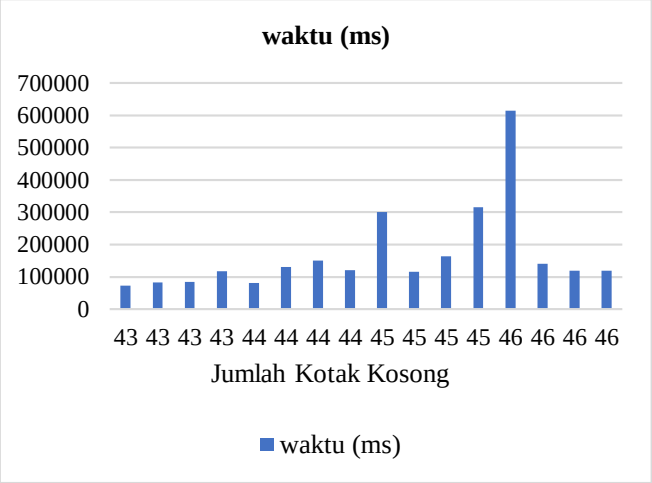
Pada penelitian ini pengujian program sudoku dilakukan pada 16 puzzel sudoku berbeda yang memiliki angka random dan tingkat kesulitan sulit. Pengujian ini dilakukan pada perangkat PC yang memiliki spesifikasi AMD Rayzen 5, RAM 8GB DDR4 Memory dan HDD 1000 GB. Pengujiann ini juga dilakukan pada software yang Bernama Pycharm dan menggunakan Bahasa Pemrograman Pyhton Berikut hasil pengujian beserta penjelasanya.

Tabel 1. Hasil Pengujian Sudoku

Percobaan ke	jumlah kotak kosong (43-46)	waktu (ms)	Jumlah proses *backtracking*
1	43	072160	828
2	43	081928	936
3	43	085013	828
4	43	118202	1269
5	44	081048	855
6	44	130791	1305
7	44	150576	1386
8	44	120871	1143
9	45	300986	2835
10	45	116479	1179
11	45	164175	1485
12	45	315750	2889
13	46	613958	5535
14	46	141243	1251
15	46	119281	1053
16	46	118934	1161

Tabel 1 menunjukan hasil dari pengujian program sudoku pada 16 puzzel sudoku berbeda yang memiliki tingkat kesulitas sulit. Jumlah kotak kosong pada setiap pengujian diatur mulai dari 43 – 46 kotak kosong. Pada pengujian ini terdapat dua parameter yang diuji, yaitu waktu penyelesaian puzzle dalam satuan microsecond (ms) dan jumlah proses *backtracking*. Melihat hasil pengujian, dari 16 percobaan yang dilakukan dapat dcermati bahwa semakin banyaknya kotak kosong pada puzzle sudoku tidak semua berbanding lurus dengan bertambahnya waktu dan

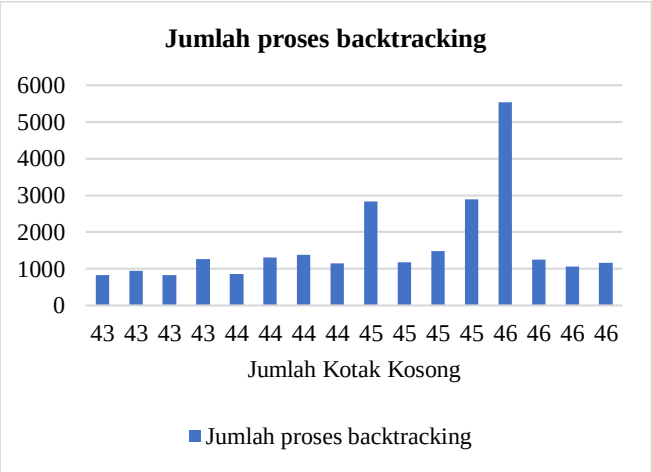
jumlah *backtracking*. Berdasarkan hasil dari pengujian, kami membuat sebuah grafik sesuai dengan parameter yang di ujikan. Berikut merupakan grafik dan penjelasanya.



Gambar 4. Grafik pengujian pada parameter waktu (ms)

Perhitungan Rata – rata waktu penyelesaian Puzzel Sudoku
X = Waktu(ms)
S = Jumlah Percobaan
Y = Rata – Rata waktu penyelesaian
Y $= \frac{\Sigma X}{S} = \frac{2623395}{16}$ = 163962 ms
Y = 163926 ms

Gambar 4 merupakan grafik yang menunjukan statistic waktu yang dibutuhkan oleh program untuk menyelesaikan Puzzle Sudoku dari pengujian yang dilakukan pada penelitian ini. Parameter Waktu pada pengujian ini menggunakan satuan microsecond (ms). Hal tersebut dilakukan untuk melihat secara detil perbedaan waktu antar pengujian dengan kotak kosong dan angka yang berbeda. Dari grambar grafik pengujian ke – 9 dan pengujian ke – 13 memiliki waktu penyelesaian yang menonjol dari pada yang lain. Hal ini dapat terjadi disebabkan oleh tingkat kesulitan yang lebih tinggi dari pada yang lain. Semakin sulit puzzle sudoku yang di uji coba, maka akan semakin banyak proses *backtracking* yang dilakukan dan hal itu berbanding lurus dengan semakin banyaknya waktu yang diperlukan. Jika dirata – rata, Algoritma *Backtracking* pada program sudoku solving ini menghabiskan waktu 163962 ms (microseconds) untuk menyelesaikan puzzle sudoku. Hal tersebut menunjukan bahwa penggunaan Algoritma Backtraking pada penyelesaian Puzzel Sudoku dapat mengefisienkan waktu pengerjaan dibandingkan dengan mengerjakan Puzze Sudoku secara manual.



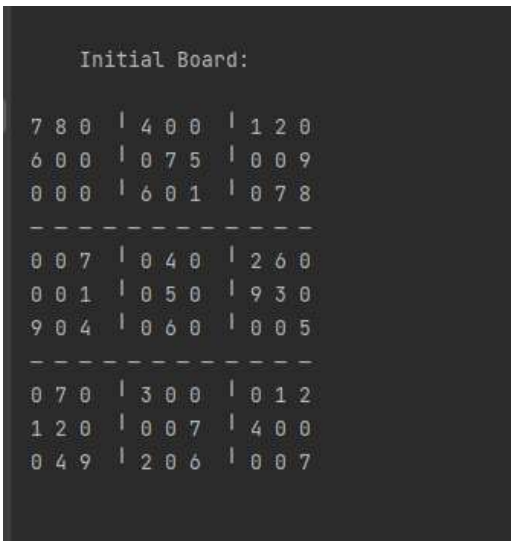
Gambar 5. Grafik pengujian parameter Jumlah proses *backtracking*

Perhitungan Rata – rata jumlah proses *backtracking*
X = Jumlah Proses *Backtracking*
S = Jumlah Pengujian
Y = Rata – rata Jumlah Proses Backtraking
Y $= \frac{\sum X}{S} = \frac{25.938}{16}$ = 1621
Y = 1621

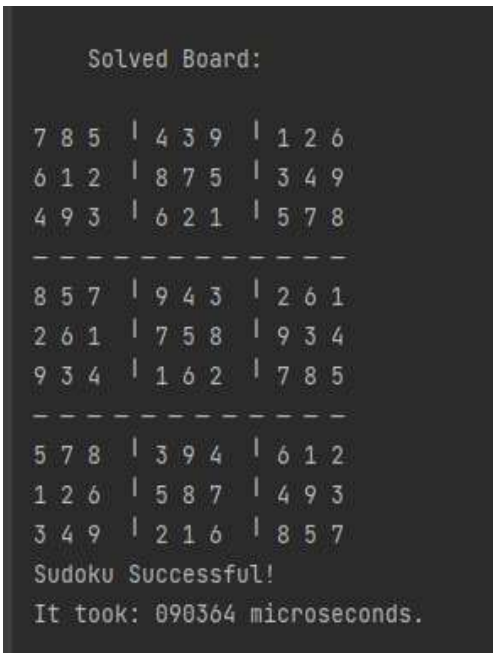
Gambar 5 merupakan grafik yang menunjukan statistic jumlah proses *backtracking* pada penyelesaian 16 Puzzel Sudoku. Berdasarkan gambar grafik, jumlah proses *backtracking* yang paling tinggi berada pada perngujian ke – 13. Namun jika dilihat secara keseluruhan, jumlah proses *backtracking* mengalami perubahan yang tidak terlalu signifikan pada saat kotak kosong terus bertembah dari 43 – 46. Hanya 3 pengujian yang memiliki jumlah proses *backtracking* paling menonjol dari yang lain dalam artian 3 tertinggi diantara yang lain. Jumlah yang banyak tersebut bisa disebabkan oleh tingkat kesulitas Puzzel Sudoku yang berbeda dari yang lain. Rata – rata jumlah *backtracking* yang digunakan untuk menyelesaikan Puzzel Sudoku pada pengujian ini adalah 1621. Jika dilihat bersamaan dengan grafik waktu , maka dapat disimpulkan bahwa semakin banyak proses algoritma *backtracking* maka akan semakin banyak waktu yang diperlukan. Tetapi jika dilihat bersamaan dengan jumlah kotak kosong, maka dapat disimpulkan bahwa semakin banyak kotak kosong yang ada pada Puzzel Sudoku tidak berbanding lurus dengan semakin banyaknya jumlah proses *backtracking* yang diperlukan untuk menyelesaikan Puzzel Sudoku.

Dari pembahasan dan hasil uji coba algoritma *backtracking* pada 16 Puzzel Sudoku berbeda menunjukkan bahwa algoritma *backtracking* dapat menjadi solusi untuk menyelesaikan permainan Puzzel Sudoku dengan waktu dan proses yang efisien dilihat dari perbandingan antara pengerjaan manual dengan pengerjaan menggunakan algoritma *backtracking*.

3.2 Akurasi algoritma *backtracking* terhadap penyelesaian Puzzel Sudoku



Gambar 6. Puzzel Sudoku dengan ukuran 9 x 9



Gambar 7. Puzzel Sudoku yang telah diselesaikan dengan algoritma *backtracking*

Pada Gambar 7 menunjukan bahwa semua kotak kosong yang tertera pada Gambar 6 telah terisi semua. Jika dilihat secara aturan permainan Puzzel Sudoku yang tidak memperbolehkan adanya angka yang sama pada satu baris vertical dan satu baris horizontal, hasil pengujian ini sudah sesuai dengan aturan yang berlaku. Selain itu angka angka yang terisi pada setiap kotak kosong pada Puzzel Sudoku terisi dengan benar tanpa ada yang salah satupun. Berdasarkan Analisa dan penemuan tersebut dapat disimpulkan bahwa pada pengujian Puzzel Sudoku ini penyelesaian puzzle menggunakan program dengan algoritma *backtracking* memiliki tingkat ketepatan 100 %. Hal itu dilihat dari jumlah benarnya angka yang dimasukan dengan jumlah kotak kosong yang ada pada puzzle sudoku diatas. Pada pengujian akurasi, kami menggunakan Puzzel Sudoku dengan kotak kosong berjumlah 43.

4. KESIMPULAN

Berdasarkan penelitian dan pengujian yang telah kami lakukan, kami dapat mendapatkan bebrapa kesimpulan sebagai berikut :

1. Banyaknya kotak kosong pada Puzzel Sudoku tidak terlalu mempengaruhi banyaknya jumlah proses *backtracking*.
2. Semakin banyak jumlah proses *backtracking* berbanding lurus dengan semakin banyaknya waktu yang dibutuhkan untuk menyelesaikan permainan Puzzel Sudoku.
3. Pada beberapa Puzzel Sudoku yang diuji, tingkat kesulitan mempengaruhi jumlah proses *backtracking* dan juga waktu penyelesaian.
4. Tingkat akurasi pada program sudoku solving dengan algortima *backtracking* mencapai 100% dengan kotak kosong berjumlah 43.
5. Penyelesaian Puzzel Sudoku dengan program sudoku solving dengan algoritma *backtracking* berhasil dijalankan dan berhasil menyelesaikan Puzzel Sudoku.

Berdasarkan penelitian yang telah dilakukan terdapat beberapa kekurangan. Untuk iu saran bagi penelitian selanjutnya adalah sebagai berikut ini.

1. Terdapat keterbatasan pada proses perhitungan jumlah proses *backtracking*. Kami berharap pada penelitian selanjutnya dapat menambah kode yang dapat menghitung jumlah proses *backtracking* secara otomatis.
2. Program Sudoku solving masih dapat dikembangkan dengan menggunakan algoritma lain yang dapat mempercepat dan mengurangi jumlah proses bcaktracking.

REFERENCES

[1] D. A. Gultom, “Penerapan Kecerdasan Buatan Dalam Menyelesaikan Permainan Pergeseran Angka Pada Bintang David Dengan Metode Pencarian Breadth-First Dan Pencarian Heuristic Menggunakan Bahasa

Pemrograman Visual," vol. 2, no. 2, pp. 174–181, 2017.

[2] J. Hadinata, "Problem Solving Sudoku Menggunakan Algoritma Genetika," *Sisfotenika*, no. Vol 1, No 1 (2011): SISFOTENIKA, pp. 48–58, 2011, [Online]. Available: http://sisfotenika.stmikpontianak.ac.id/index.php/ST/article/view6.

[3] K. Literatur, "BLOK : Kumpulan tiga boks , vertical maupun horizontal . 81 sel , atau 9 boks , atau 9 deret , atau dapat dilakukan dari mana saja , tergantung struktur soal . SEL : Kotak terkecil yang seharusnya berisi angka . BOKS : Kotak lebih besar yang mengandung Se," no. 1, pp. 207–215, 2013.

[4] M. H. Rifqo and Y. Apridiansyah, "Implementasi Algoritma *Backtracking* Dalam Sistem Informasi Perpustakaan Untuk Pencarian Judul Buku (Studi Kasus Unit Pelayanan Terpadu Perpustakaan Universitas Muhammadiyah Bengkulu)," *Pseudocode*, vol. 4, no. 1, pp. 90–96, 2017, doi: 10.33369/pseudocode.4.1.90-96.

[5] P. N. Salsabila, "Penerapan Algoritma *Backtracking* dalam Menyelesaikan Sudoku with 4 Given Digits," no. 13518094, 2020.