JURNAL TEKNIK INFORMATIKA DAN TEKNOLOGI INFORMASI
Halaman Jurnal: https://ejurnal.politeknikpratama.ac.id/index.php/JUTITI
Halaman UTAMA: https://ejurnal.politeknikpratama.ac.id/index.php

PERBANDINGAN KEEFEKTIFAN ALGORITMA BACKTRACKING DAN SOFT COMPUTING DALAM MEMECAHKAN PERMAINAN PAPAN NONOGRAM

[1] Muhammad Ali Zafar Sidiq, [3] Aldi Supriyadi, [3] Asti Herliana
Universitas Adhirajasa Reswara Sanjaya, muhammadalizafarsidiq@gmail.com
Universitas Adhirajasa Reswara Sanjaya, aldisupriyadi11@gmail.com
Universitas Adhirajasa Reswara Sanjaya, asti@ars.ac.id

Abstract : *Solving logic puzzles using specific algorithms presents an intriguing challenge where the efficiency of the approaches is crucial. One such puzzle involves solving nonograms, where the task is to fill in board fields according to the conditions specified for each row and column. The availability of various methods allows for comparing their efficiency and effectiveness. This study aimed to evaluate the effectiveness of two algorithms from different categories. We selected a modified Depth-First Search (DFS) method and a soft computing method based on permutations generation to solve a set of chosen nonograms. The research was conducted using four different board sizes, and the results indicated that the effectiveness of the methods largely depends on the complexity of the nonogram. The algorithm employing permutations consistently produced stable results, while the DFS method did not always guarantee a complete solution.*

Keywords**:** *Soft computing; Nonogram; Depth-First Search*

Abstrak : Menggunakan algoritma-algoritma tertentu untuk memecahkan teka-teki logika menawarkan tantangan yang menarik di mana efisiensi pendekatan menjadi krusial. Salah satu teka-teki tersebut melibatkan pemecahan nonogram, di mana tugasnya adalah mengisi bidang-bidang papan sesuai dengan kondisi yang ditentukan untuk setiap baris dan kolom. Ketersediaan berbagai metode memungkinkan untuk membandingkan efisiensi dan efektivitasnya. Penelitian ini bertujuan untuk mengevaluasi efektivitas dua algoritma dari kategori yang berbeda. Kami memilih metode Depth-First Search (DFS) yang dimodifikasi dan metode soft computing berdasarkan generasi permutasi untuk memecahkan serangkaian nonogram yang dipilih. Penelitian ini dilakukan menggunakan empat ukuran papan yang berbeda, dan hasilnya menunjukkan bahwa efektivitas metode-metode tersebut sangat bergantung pada kompleksitas nonogram. Algoritma yang menggunakan permutasi secara konsisten menghasilkan hasil yang stabil, sedangkan metode DFS tidak selalu menjamin solusi yang lengkap.

Kata kunci: Soft computing; Nonogram; Depth-First Search;

1. Pendahuluan

Selama bertahun-tahun, Kecerdasan Buatan (AI) telah semakin populer di kalangan pengikut dan peneliti yang ingin menerapkan teknologi ini secara praktis [11, 20]. Prinsip dan metodenya terus berkembang pesat, sehingga penggunaan AI dapat diperluas lebih luas [22, 23]. Sejak asumsi dasar Kecerdasan Buatan diperkenalkan, metode yang digunakan terus ditingkatkan dan dioptimalkan untuk meningkatkan operasi dan efisiensi secara berturut-turut [10, 13, 25, 27].

Kecerdasan Buatan saat ini telah diterapkan dalam berbagai bidang, dengan tugas utamanya termasuk klasifikasi dan prediksi [1, 17, 18]. Selain itu, Kecerdasan Buatan juga digunakan dalam teknik soft computing [8, 28]. Soft computing adalah pendekatan yang mengikuti prinsip-proses pemikiran manusia untuk memecahkan masalah dan mencapai solusi [12, 15, 38]. Pendekatan ini mencakup metode probabilistik, logika fuzzy, jaringan saraf, dan algoritma evolusioner [12, 15, 38].

Algoritma soft computing digunakan dalam berbagai masalah dengan efisiensi untuk mencapai hasil yang diharapkan [7, 28] berdasarkan kekhususannya dalam operasi tersebut. Salah satu contoh penggunaan *soft computing* adalah dalam memecahkan nonogram. Secara sederhana, nonogram adalah teka-teki logika di mana kita harus mengisi bagian-bagian kosong [4, 37]. Nonogram adalah papan dua dimensi yang perlu kita isi dengan benar sehingga tampilan akhirnya sesuai dengan batasan-batasan yang diberikan. Pada papan tersebut, terdapat kolom dan baris yang memiliki deret angka sebagai petunjuk untuk menentukan jarak antara blok-blok yang terisi dan kosong, serta urutan blok yang serupa [3, 35].

Namun, terdapat algoritma-algoritma lain yang dapat digunakan untuk menyelesaikan nonogram, seperti algoritma A-star atau Depth-First Search (DFS) [14, 32, 36]. Meskipun solusi-solusi ini cenderung kurang efisien, namun masih populer karena implementasinya yang sederhana dan kemampuannya dalam memberikan hasil [6, 24, 26]. Dalam menyelesaikan nonogram, mempertimbangkan penggunaan beberapa solusi memberikan kesempatan untuk menguji efektivitas berbagai algoritma ini dan membandingkan kualitas solusi yang dihasilkan [16, 33, 34].

Selain itu, aspek penting lainnya dalam mempelajari perilaku metode-metode ini adalah bagaimana keadaan awal dihasilkan. Hal ini sering kali menjadi faktor kunci yang menentukan tingkat efisiensi suatu solusi atau apakah solusi dapat ditemukan atau tidak [5, 21, 29].

Karena adanya kesenjangan dalam bidang ini dan potensi untuk membawa wawasan baru dalam memecahkan masalah nonogram, kami memutuskan untuk melakukan upaya guna menguji kinerja dua algoritma yang dipilih dan efektivitasnya dalam berbagai kondisi.

Dalam artikel ini, kami menggunakan dua algoritma yang mewakili dua kelompok metode untuk menyelesaikan nonogram yang non-linier, dan kami menganalisis kinerja algoritma-algoritma tersebut dalam menghasilkan solusi, jumlah iterasi, dan waktu yang diperlukan untuk mencapai keadaan akhir.

Selanjutnya, kami menyelidiki bagaimana pembentukan keadaan awal mempengaruhi hasilnya. Penyelesaian nonogram cukup menantang karena kendala yang ada dan aturan yang jelas untuk mengisi papan.

Penelitian ini dilakukan dengan tujuan memperkenalkan perspektif baru mengenai kemungkinan penggunaan metode yang sudah ada dan modifikasi mereka untuk mendapatkan solusi yang memuaskan untuk teka-teki nonogram.

Penggunaan dua algoritma yang berbeda bertujuan untuk menguji kualitas kinerja mereka dalam berbagai kondisi. Hal ini memungkinkan penelitian yang komprehensif mengenai pengaruh tingkat kesulitan nonogram terhadap waktu yang diperlukan.

2. Metode Penelitian

Untuk memberikan pemahaman umum tentang asumsi utama nonogram dan menjelaskan prinsip-prinsip algoritma yang digunakan, kami memutuskan untuk menguraikan informasi dasar berikut ini agar mempermudah interpretasi kegiatan dalam penelitian ini. Penjelasan tentang asumsi dibagi menjadi dua bagian. Bagian pertama akan membahas dan memberikan contoh-contoh nonogram serta cara-cara untuk mendefinisikan kendala dan mencari solusi. Di sisi lain, bagian kedua akan mencakup presentasi algoritma yang dipilih dan mengilustrasikan prinsip operasinya.

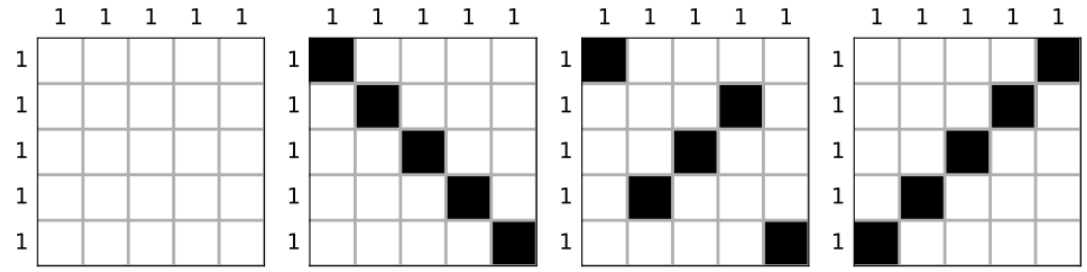
2.1. Nonogram

Nonogram merupakan teka-teki logika yang didasarkan pada konsep mengisi kotak-kotak pada papan berdasarkan batasan yang telah ditentukan [9, 39]. Baik pada baris maupun kolom, terdapat batasan numerik yang terletak di atas atau di sebelahnya. Tujuan dari nonogram adalah mengisi papan dengan cara yang tepat untuk mendapatkan solusi yang benar.

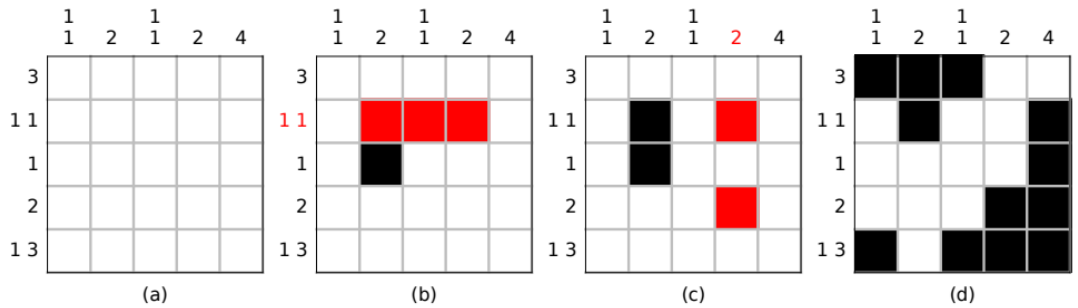
Solusi yang benar adalah solusi yang memenuhi semua batasan tersebut. Sebagai contoh, jika terdapat baris dengan batasan 2, 3, 4, maka solusi yang benar akan memiliki tiga kelompok dengan dua, tiga, dan empat kotak yang diisi dengan warna yang sesuai, dan minimal satu kotak kosong di antara kelompok-kelompok yang berdekatan.

Nonogram dapat memiliki ukuran yang bervariasi, baik berbentuk persegi maupun persegi panjang. Selain itu, nonogram dan batasan-batasannya dapat didefinisikan dengan cara yang tidak jelas mengenai cara pengisiannya [19, 31]. Hal ini menyebabkan kemungkinan terdapat beberapa hasil yang berbeda, sehingga meskipun memiliki batasan yang sama, nonogram yang dihasilkan dapat terlihat berbeda.

Gambar 1 mengilustrasikan contoh papan nonogram dengan berbagai kemungkinan solusi. Di sisi lain, Gambar 2 menampilkan jenis nonogram yang berbeda. Kendala-kendala ini menunjukkan bahwa setiap papan nonogram memiliki aturan yang unik dalam pengisiannya, sehingga mengisi papan dengan cara yang berbeda dapat menghasilkan solusi yang bertentangan dan salah. Dua kasus disajikan di mana bidang yang diisi dengan buruk ditandai dengan warna merah untuk menyoroti kesalahan (b, c). Pada kasus (b), semua kolom valid, tetapi baris kedua diisi dengan kesalahan. Batasan menunjukkan bahwa harus ada dua grup dengan satu bidang terisi pada setiap kelompok, tetapi dalam kasus ini terdapat tiga piksel yang terisi. Kasus (c) menunjukkan nonogram dengan baris yang diisi dengan benar, tetapi terdapat kesalahan pada kolom keempat. Kolom ini seharusnya memiliki satu grup dengan dua bidang terisi, tetapi dalam kasus ini terdapat dua grup dengan masing-masing satu bidang yang terisi.



Gambar 1. Contoh nonogram 5x5 dengan beberapa solusi yang mungkin



Gambar 2. Contoh nonogram 5x5 dengan solusi yang benar dan solusi yang salah

Dalam studi ini, fokus kami adalah pada jenis masalah nonogram dengan ukuran yang berbeda yang memiliki definisi yang unik. Artinya, kami tidak mempertimbangkan kasus di mana ada lebih dari satu solusi yang mungkin untuk sebuah papan nonogram.

2.2. Algoritma

Untuk menjelaskan prinsip operasi dari algoritma-algoritma yang digunakan, kami memutuskan untuk menampilkan pseudo-code dari dua metode terpilih yang digunakan untuk menyelesaikan nonogram tertentu. Algoritma-algoritma ini telah dikembangkan dengan baik

untuk memastikan solusi yang konsisten dan tidak bertentangan dengan cara pengisian untuk nonogram berukuran universal dan mengikuti kendala yang diberikan.

2.2.1.Algoritma Depth-First Search

Berikut ini adalah pseudo-code untuk algoritma Depth-First Search yang dimodifikasi untuk menemukan solusi dari papan nonogram [2, 30]. Fungsi yang tercantum pada daftar 1 bekerja secara rekursif untuk mencari solusi.

Pada awalnya, pemanggilan fungsi ini melintasi papan bersama dengan kolom-kolomnya. Ketika indeks mencapai jumlah kolom, indeks baris ditingkatkan, dan proses ini diulangi hingga seluruh papan telah diperiksa.

Fungsi verifikasi dalam pseudo-code bertujuan untuk memeriksa apakah baris dan kolom yang diberikan sesuai dengan batasan numerik yang ditentukan. Jika kondisi tersebut terpenuhi dan pemanggilan rekursif ke fungsi findSolution mengembalikan nilai True, maka kolom yang sedang dianalisis akan diisi. Sebaliknya, jika kondisi tersebut tidak terpenuhi, nilai bidang yang sedang dianalisis diubah kembali menjadi 0, yang menunjukkan bahwa bidang tersebut tidak diisi. Proses ini diperiksa kembali menggunakan fungsi verify dan findSolution. Jika tidak ada solusi yang ditemukan, nilai False dikembalikan, menandakan bahwa solusi yang valid tidak dapat ditemukan.

Daftar 1. Pseudocode untuk algoritma Depth-First Search yang dimodifikasi

```
def findSolution(i, j):
    if i == height:
        return True

    if j + 1 == width:
        nextI = i + 1
    else:
        nextI = i
    nextJ = (j + 1) % width

    board[i][j] = 1
    if verify(i, j) and findSolution(nextI, nextJ):
        return True

    board[i][j] = 0
    if verify(i, j) and findSolution(nextI, nextJ):
        return True

return False
```

2.2.2.Algoritma Soft Computing

Dalam daftar kedua, disajikan kode semu untuk algoritma soft computing yang dipilih. Algoritma ini bekerja dengan menghasilkan permutasi dari berbagai kondisi yang mungkin terjadi. Kemudian, keadaan yang dihasilkan dibandingkan dengan batasan yang telah ditentukan untuk baris atau kolom tertentu. Tujuannya adalah untuk memeriksa apakah keadaan yang sedang dianalisis memenuhi kondisi yang telah ditentukan. Fungsi "solve_row" digunakan untuk memperoleh kemungkinan solusi baik untuk baris maupun kolom. Pendekatan ini membantu menyederhanakan proses pencarian solusi untuk bidang nonogram yang sedang dianalisis. Fungsi "get_permutations" bertanggung jawab dalam menghasilkan solusi yang memenuhi batasan yang telah ditentukan. Jika solusi yang dibuat memenuhi kondisi akhir, maka keadaan saat ini akan disimpan sebagai solusi yang valid. Selanjutnya, berbagai keadaan yang telah disimpan dibandingkan satu sama lain untuk mendapatkan solusi nonogram yang telah selesai sebagian.

Di sisi lain, fungsi "solve" digunakan untuk mendapatkan solusi untuk baris dan kolom secara keseluruhan. Fungsi ini menggunakan fungsi "get_permutations" yang telah disebutkan sebelumnya untuk mendapatkan sebagian solusi. Kemudian, proses pencarian solusi dilanjutkan dengan mencoba semua permutasi yang mungkin. Setelah semua pergerakan yang mungkin dilakukan telah dicoba, proses pencarian solusi akan berakhir dan papan nonogram yang telah berhasil dipecahkan akan disajikan. Namun, metode ini memiliki kelemahan yaitu kemungkinan tidak adanya solusi nonogram yang dapat dipecahkan secara keseluruhan.

Daftar 2. Algoritma soft computing berdasarkan pembangkitan permutasi

```
def solve_row(counts, row):
    permutations = get_permutations(counts, len(row))
    valid_permutations = []
    for permutation in permutations:
        valid = True
        for x in range(len(row)):
            if row[x] != -1 and row[x] != permutation[x]:
                valid = False
        if valid:
            valid_permutations.append(permutation)

    new_row = copy(valid_permutations[0])
    for permutation in valid_permutations:
        for x in range(len(row)):
            if new_row[x] != permutation[x]:
                new_row[x] = -1

    return new_row

def solve():
    changed = True
    while changed:
        changed = False
        for x in range(width):
            col = grid[:, x]
            col = self.solve_row(col_counts[x], col)
            for y in range(height):
                if col[y] != -1 and grid[y, x] != col[y]:
                    changed = True
                grid[y, x] = col[y]

        # Sama seperti pengulangan (loop) untuk baris
        ...
    return grid
```

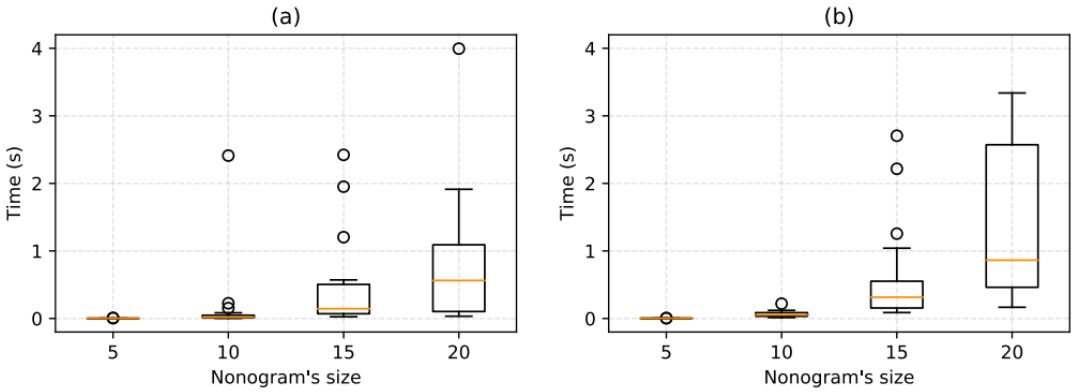
	Matriks:

3.

4. Hasil dan Pembahasan

Algoritma yang disajikan di atas digunakan untuk memecahkan nonogram pada papan yang telah dipilih. Algoritma ini telah diuji untuk memastikan bahwa mereka dapat menghasilkan solusi yang lengkap, serta waktu yang dibutuhkan untuk mencapai solusi tersebut diukur. Dalam pengujian ini, empat ukuran nonogram (5, 10, 15, 20) dipilih, dengan jumlah baris dan kolom yang sama. Setiap ukuran nonogram diuji sebanyak 20 kali, dan papan yang digunakan memiliki tingkat kesulitan yang berbeda-beda. Penting untuk dicatat bahwa setiap nonogram telah didefinisikan secara jelas, sehingga hanya ada satu solusi yang mungkin.

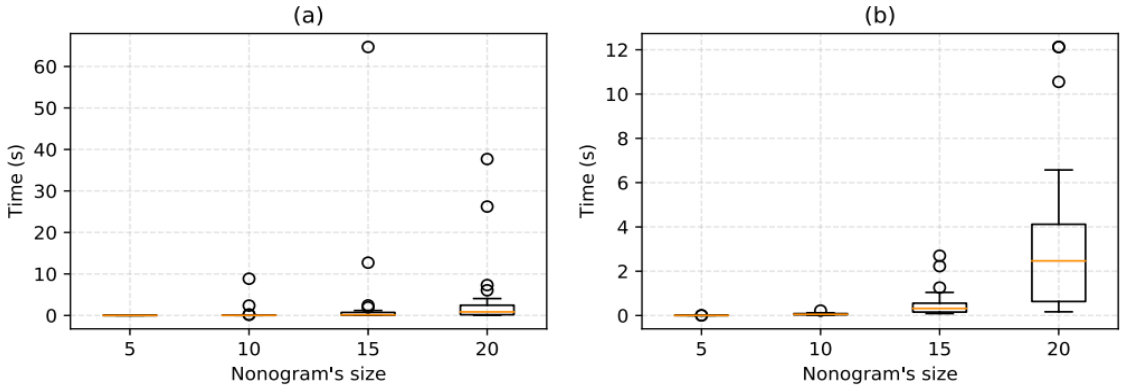
Gambar 3. Waktu penyelesaian dengan penghapusan data outlier. a) DFS yang dimodifikasi b)



Algoritma Soft Computing yang dimodifikasi.

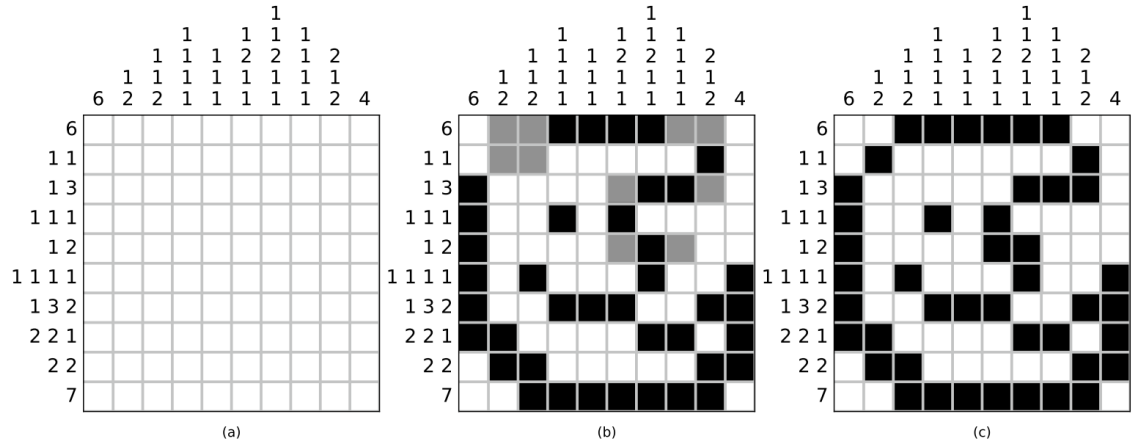
Waktu yang dihasilkan dari studi ditampilkan dalam Gambar 3 dan 4. Pada Gambar pertama, visualisasi diberikan tanpa mempertimbangkan data yang berada jauh dari rata-rata. Dari gambar tersebut, dapat dilihat bahwa hasilnya berfluktuasi antara rentang [0, 4] detik, dengan rata-rata hasil berada dalam rentang [0, 1] detik untuk semua ukuran nonogram yang berhasil dipecahkan. Penting untuk dicatat bahwa seiring dengan meningkatnya ukuran papan, rentang interkuartil juga meningkat, dan pengamatan data yang berada jauh dari rata-rata lebih sering terjadi. Selain itu, terlihat bahwa pada papan yang lebih kecil dan lebih sederhana, solusi yang diperoleh dengan menggunakan algoritma DFS yang dimodifikasi memberikan hasil yang lebih baik. Namun, saat menganalisis papan yang lebih sulit, algoritma berbasis permutasi menunjukkan perilaku yang lebih seimbang.

Di sisi lain, pada visualisasi kedua yang mencakup grafik dengan data yang berada jauh dari rata-rata, terlihat bahwa untuk ukuran nonogram yang lebih besar, algoritma DFS mencatat waktu yang lebih lama dibandingkan dengan algoritma kedua. Hal ini menunjukkan bahwa algoritma berbasis permutasi memiliki tingkat kestabilan yang lebih baik dan memberikan hasil yang serupa dalam hal efisiensi untuk semua ukuran dan nonogram yang berhasil dipecahkan.



Gambar 4. Waktu penyelesaian dengan adanya data outlier. a) DFS yang dimodifikasi b) Algoritma Soft Computing yang dimodifikasi.

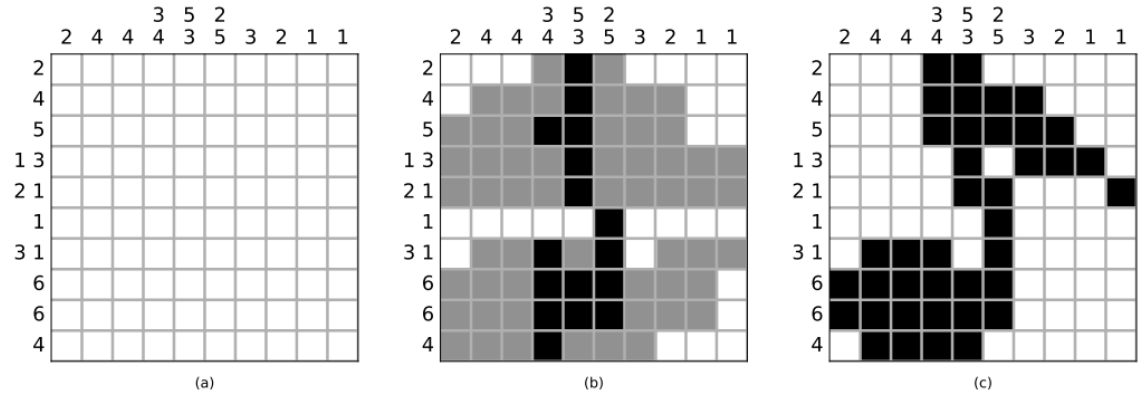
Di sisi lain, Gambar 5 menampilkan contoh papan nonogram dengan ukuran 10x10. Visualisasi ini mencakup papan kosong dengan batasan yang telah ditentukan mengenai cara mengisinya (a), nonogram yang telah diselesaikan sebagian (b), dan nonogram yang telah diselesaikan sepenuhnya (c). Nonogram yang telah berhasil dipecahkan ditunjukkan dalam bagian (c). Sekitar 88% dari total solusi diisi dengan benar menggunakan algoritma berbasis permutasi. Area yang berwarna abu-abu menunjukkan area yang masih belum diketahui, sedangkan area yang berwarna hitam menunjukkan area yang telah terisi dengan benar sesuai dengan batasan yang telah ditentukan.



Gambar 5. Nonogram yang membutuhkan beberapa tebakan: (a) nonogram kosong, (b) sebagian terpecahkan oleh algoritma kedua (pixel abu-abu tetap tidak diketahui), (c) terselesaikan oleh algoritma DFS. Sumber: webpbn.com #16270.

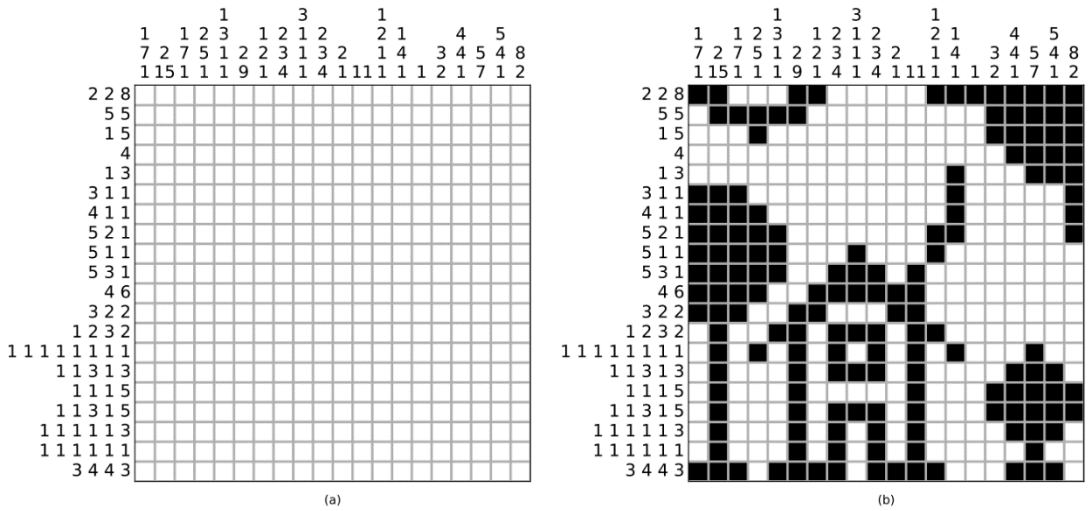
Di sisi lain, Gambar 6 menampilkan visualisasi solusi yang diperoleh menggunakan algoritma dari kelompok metode soft computing. Gambar ini menunjukkan tiga hal: (a) sebuah papan kosong dengan kendala, (b) sebuah nonogram yang sebagian terselesaikan, dan (c) sebuah nonogram yang berhasil terselesaikan secara benar. Perlu diperhatikan bahwa pada contoh ini, terdapat lebih banyak bidang yang kondisi akhirnya tidak dapat ditentukan. Hal ini menunjukkan bahwa algoritma berbasis permutasi tidak selalu menghasilkan nonogram yang dapat diselesaikan, berbeda dengan algoritma DFS yang telah dimodifikasi.

Gambar 6. Nonogram yang membutuhkan beberapa tebakan: (a) nonogram kosong, (b)



sebagian terpecahkan oleh algoritma kedua (pixel abu-abu tetap tidak diketahui), (c) terselesaikan oleh algoritma DFS. Sumber: webpbn.com #4573.

Terdapat juga beberapa nonogram yang dianggap sulit oleh kedua algoritma tersebut. Sebagai contoh, pada Gambar 7 terlihat bahwa nonogram ini dapat diselesaikan dengan kedua



algoritma, namun membutuhkan lebih banyak waktu dibandingkan algoritma yang lebih sederhana. Algoritma DFS menyelesaikan nonogram ini dalam waktu kira-kira 36 detik, sementara algoritma berbasis permutasi menyelesaikannya dalam waktu kira-kira 12 detik. Dalam nonogram ini, terlihat bahwa terdapat kelompok-kelompok kecil bidang yang terisi, yang menyebabkan banyak kemungkinan permutasi dan juga meningkatkan jumlah tebakan yang diperlukan dalam algoritma DFS.

Gambar 7. Nonogram yang memakan waktu lama untuk diselesaikan oleh kedua algoritma. Sumber: webpbn.com #11.

Tabel 1 menyajikan ringkasan waktu yang dihasilkan dari pengukuran saat algoritma memecahkan nonogram berukuran 20×20. Terdapat enam papan yang berbeda yang dianalisis, dan waktu ditampilkan untuk kedua algoritma yang digunakan. Dari hasil yang diperoleh, dapat disimpulkan bahwa dalam banyak kasus, metode kedua menghasilkan solusi dengan lebih cepat. Namun, terdapat juga kasus di mana metode pertama jauh lebih efisien dan efektif, seperti pada nonogram nomor 240 atau 360. Hal ini mungkin disebabkan oleh kompleksitas dalam menyelesaikan nonogram dengan benar, terutama karena batasan numerik pada papan. Selain itu, perbedaan waktu yang signifikan terlihat pada nonogram nomor 6, di mana perbedaan waktunya mencapai hampir 25 detik, atau sekitar 13 kali lebih lama dibandingkan algoritma kedua. Penelitian ini menunjukkan bahwa untuk nonogram yang lebih besar, kualitas dan kecepatan solusi sangat dipengaruhi oleh tingkat kompleksitas nonogram tersebut. Dalam beberapa kasus, algoritma DFS bekerja lebih efisien, sementara dalam kasus lain, algoritma berbasis pembangkitan permutasi memberikan hasil yang lebih baik.

tabel 1. Waktu pemecahan untuk beberapa nonogram dengan ukuran 20.

Sumber	Algoritma DFS [s]	Algoritma Permutasi [s]
webpbn.com #6	26.2064	1.6269
webpbn.com #240	0.8569	6.5333
webpbn.com #387	5.9944	3.3328
webpbn.com #360	0.4997	12.1421
webpbn.com #1352	4.0307	0.9397
webpbn.com #1304	7.2368	4.9681

5. Kesimpulan

Algoritma yang dikembangkan oleh para ilmuwan telah diterapkan dalam berbagai masalah yang ada di berbagai bidang. Selain itu, upaya terus dilakukan untuk meningkatkan kinerja metode yang ada agar lebih efisien. Penggunaan dan peningkatan algoritma melalui pendekatan Soft Computing menjadi sangat populer. Sebagian besar usaha dilakukan untuk membuat metode ini dapat diterapkan pada kelompok masalah yang lebih luas daripada sebelumnya.

Dalam artikel ini, digunakan dua metode yang mewakili kelompok yang berbeda untuk membandingkan efektivitas mereka dalam menyelesaikan nonogram. Metode pertama adalah algoritma DFS, sedangkan metode kedua adalah algoritma yang berdasarkan pembuatan permutasi dari keadaan-keadaan yang berurutan. Kedua metode ini dimodifikasi untuk digunakan dalam memecahkan nonogram, di mana bidang harus diisi dengan benar sesuai dengan persyaratan numerik di sekitar papan, sehingga membentuk gambar yang tepat.

Dalam penelitian ini, nonogram dengan ukuran yang berbeda diuji untuk menyelidiki waktu yang dibutuhkan dalam mendapatkan solusi. Hasil penelitian menunjukkan bahwa algoritma yang berdasarkan pembuatan permutasi lebih stabil, artinya waktu yang diperlukan relatif konsisten terlepas dari tingkat kesulitan nonogram yang diuji. Namun, pada metode DFS, terdapat beberapa kasus di mana waktu yang diperlukan jauh lebih lama dan variabel dibandingkan dengan hasil rata-rata dari metode tersebut. Meskipun demikian, perlu dicatat bahwa algoritma DFS tetap dapat menjamin solusi yang benar, tidak peduli seberapa sulitnya nonogram tersebut, berbeda dengan metode yang lain.

Untuk masa depan, penting untuk mempertimbangkan ekspansi metode yang digunakan guna menghasilkan hasil yang lebih komprehensif dan memungkinkan perbandingan yang lebih baik. Selain itu, perlu dilakukan upaya untuk meningkatkan algoritma yang digunakan agar lebih efisien, mengurangi waktu yang dibutuhkan untuk mendapatkan solusi, serta meningkatkan efektivitasnya. Selain itu, masalah yang diberikan dapat diperluas dengan memilih kriteria yang tepat untuk menggambarkan kinerja algoritma dan metode. Metode pembuatan keputusan berbasis multi-kriteria juga dapat digunakan untuk mengevaluasi kualitas algoritma yang digunakan.

Bibliography

[1] Agrawal, A., Gans, J., Goldfarb, A., 2018. Prediction machines: the simple economics of artificial intelligence. Harvard Business Press.

[2] Asano, T., Izumi, T., Kiyomi, M., Konagaya, M., Ono, H., Otachi, Y., Schweitzer, P., Tarui, J., Uehara, R., 2014. Depth-first search using o(n) bits, in: International Symposium on Algorithms and Computation, Springer. pp. 553–564.

[3] Batenburg, K.J., Henstra, S., Kosters, W.A., Palenstijn, W.J., 2009. Constructing simple nonograms of varying difficulty. Pure Mathematics and Applications (Pu. MA) 20, 1–15.

[4] Batenburg, K.J., Kosters, W.A., 2009. Solving nonograms by combining relaxations. Pattern Recognition 42, 1672–1683.

[5] Berend, D., Pomeranz, D., Rabani, R., Raziel, B., 2014. Nonograms: Combinatorial questions and algorithms. Discrete Applied Mathematics 169, 30–42.

[6] Bonet, B., Geffner, H., 2006. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to mdps., in: ICAPS, pp. 142–151.

[7] Chandrasekaran, M., Muralidhar, M., Krishna, C.M., Dixit, U., 2010. Application of soft computing techniques in machining performance prediction and optimization: a literature review. The International Journal of Advanced Manufacturing Technology 46, 445–464.

[8] Chaturvedi, D.K., 2008. Soft computing. Studies in Computational intelligence 103.

[9] Chen, Y.C., Lin, S.S., 2019. A fast nonogram solver that won the taai 2017 and icga 2018

tournaments. ICGA Journal 41, 2–14.

[10] Conant, E.F., Toledano, A.Y., Periaswamy, S., Fotin, S.V., Go, J., Boatsman, J.E., Hoffmeister, J.W., 2019. Improving accuracy and efficiency with concurrent use of artificial intelligence for digital breast tomosynthesis. Radiology: Artificial Intelligence 1, e180096.

[11] Dick, S., 2019. Artificial intelligence .

[12] Du, K.L., Swamy, M.N., 2006. Neural networks in a softcomputing framework. Springer Science & Business Media.

[13] Fethi, M.D., Pasiouras, F., 2010. Assessing bank efficiency and performance with operational research and artificial intelligence techniques: A survey. European journal of operational research 204, 189–198.

[14] Firmansyah, E.R., Masruroh, S.U., Fahrianto, F., 2016. Comparative analysis of a* and basic theta* algorithm in android-based pathfinding games, in: 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M), IEEE. pp. 275–280.

[15] Herrera, F., Lozano, M., Molina, D., et al., 2010. Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. Last accessed: July .

[16] Khan, K.A., 2020. Solving nonograms using integer programming without coloring. IEEE Transactions on Games .

[17] Konar, A., 2018. Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain. CRC press.

[18] Kotsiantis, S.B., Zaharakis, I., Pintelas, P., 2007. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering 160, 3–24.

[19] Malhotra, R.K., Indrayan, A., 2010. A simple nomogram for sample size for estimating sensitivity and specificity of medical tests. Indian journal of ophthalmology 58, 519.

[20] McCarthy, J., 1998. What is artificial intelligence? .

[21] Mingote, L., Azevedo, F., 2009. Colored nonograms: an integer linear programming approach, in: Portuguese Conference on Artificial Intelligence, Springer. pp. 213–224. [22] Mitchell, R., Michalski, J., Carbonell, T., 2013. An artificial intelligence approach. Springer.

[23] Nilsson, N.J., 2014. Principles of artificial intelligence. Morgan Kaufmann.

[24] Rahim, R., Abdullah, D., Simarmata, J., Pranolo, A., Ahmar, A.S., Hidayat, R., Napitupulu, D., Nurdiyanto, H., Febriadi, B., Zamzami, Z., 2018. Block architecture problem with depth first search solution and its application, in: Journal of Physics: Conference Series, IOP Publishing. p. 012006.

[25] ur Rehman, A., Sałabun, W., Faizi, S., Hussain, M., W ˛atróbski, J., 2021. On graph structures in fuzzy environment using optimization parameter. IEEE Access 9, 75699–75711.

[26] Rhodes, C., Blewitt, W., Sharp, C., Ushaw, G., Morgan, G., 2014. Smart routing: A novel application of collaborative path-finding to smart parking systems, in: 2014 IEEE 16th Conference on Business Informatics, IEEE. pp. 119–126.

[27] Russell, S., Norvig, P., 2002. Artificial intelligence: a modern approach .

[28] Senthilkumaran, N., Rajesh, R., 2009. Image segmentation-a survey of soft computing approaches, in: 2009 International Conference on Advances in Recent Technologies in Communication and Computing, IEEE. pp. 844–846.

[29] Sohn, Y.S., Oh, K., Kim, B.S., 2007. A recognition method of the printed alphabet by using nonogram puzzle, in: Proceedings of The 8th International Symposium on Advanced Intelligent

Systems, pp. 232–236.

[30] Stojmenovic, I., Russell, M., Vukojevic, B., 2000. Depth first search and location based localized routing and qos routing in wireless networks, in: Proceedings 2000 International Conference on Parallel Processing, IEEE. pp. 173–180.

[31] Tan, X., Ma, Z., Yan, L., Ye, W., Liu, Z., Liang, C., 2019. Radiomics nomogram outperforms size criteria in discriminating lymph node metastasis in resectable esophageal squamous cell carcinoma. European radiology 29, 392–400.

[32] Tsai, J.T., 2012. Solving japanese nonograms by taguchi-based genetic algorithm. Applied Intelligence 37, 405–419.

[33] Tsai, J.T., Chou, P.Y., Fang, J.C., 2011. Learning intelligent genetic algorithms using japanese nonograms. IEEE Transactions on Education 55, 164–168.

[34] Wang, W.L., Tang, M.H., 2014. Simulated annealing approach to solve nonogram puzzles with multiple solutions. Procedia Computer Science 36, 541–548.

[35] Wiggers, W., van Bergen, W., 2004. A comparison of a genetic algorithm and a depth first search algorithm applied to japanese nonograms, in: Twente student conference on IT, Citeseer.

[36] Wu, I.C., Sun, D.J., Chen, L.P., Chen, K.Y., Kuo, C.H., Kang, H.H., Lin, H.H., 2013. An efficient approach to solving nonograms. IEEE Transactions on Computational Intelligence and AI in Games 5, 251–264.

[37] Yu, C.H., Lee, H.L., Chen, L.H., 2011. An efficient algorithm for solving nonograms. Applied Intelligence 35, 18–31.

[38] Zadeh, L.A., 1996. Soft computing and fuzzy logic, in: Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi a Zadeh. World Scientific, pp. 796–804.

[39] Zavistanavicius, R., 2013. Nonogram solving algorithms analysis and implementation for augmented reality system