Optimasi Metode *Line-solving* dengan Teknik Heuristik Pencarian *Pseudo-Exhaustive* untuk Menyelesaikan Permainan Nonogram

Michael Leon Putra Widhi - 13521108[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521108@std.stei.itb.ac.id

***Abstrak*— Permainan Nonogram adalah sebuah permainan berbasis logika yang mengasah pikiran. Belum cukup banyak algoritma yang diusulkan untuk dapat menyelesaikan permainan ini secara komputasional. Penyelesaian sebuah nonogram dengan cara lempang menggunakan algoritma *brute force* memerlukan kompleksitas waktu yang sangat tinggi sehingga algoritma ini tidak efisien. Permainan ini sendiri sudah digolongkan dalam permasalahan NP-*complete* sehingga akan sulit untuk merumuskan sebuah algoritma yang berhasil menyelesaikan permasalahan ini dalam kompleksitas polinomial. Teknik Heuristik sering digunakan untuk melakukan akselerasi pencarian solusi tanpa harus memperoleh seluruh kemungkinan secara mendalam. Pada makalah ini akan dibahas sebuah algoritma berbasis *line-solving* yang penulis usulkan untuk menyelesaikan permasalahan ini dalam basis kasus-per-kasus. Algoritma diimplementasikan secara rangkap sebagai gabungan dari beberapa algoritma dengan pendekatan heuristik untuk melakukan optimasi mendekati kompleksitas polinomial. Pada makalah ini juga akan dibahas mengenai ide pengembangan lebih lanjut solusi dengan menggunakan proses propagasi dan program dinamis yang jika diimplementasikan dengan baik dapat mencapai kompleksitas rata-rata mendekati polinomial.**

***Kata Kunci*— *Line-solving*, Pseudo-*exhaustive*, Heuristik, Optimasi**

I. PENDAHULUAN

Permainan berbasis logika sudah menjadi hal yang umum di kalangan masyarakat. Berbagai permainan seperti catur, kakuro, dan sudoku membawa daya tarik tersendiri bagi para peminatnya. Ide dasar dari setiap permainan ini cukup sederhana, hal yang kemudian membuatnya menjadi menarik adalah proses merumuskan sebuah strategi untuk memenangkan permainan-permainan tersebut. Banyak orang yang telah berhasil merumuskan sebuah strategi penyelesaian dengan melakukan komputasi, baik secara heuristik maupun matematis sebagai bagian dari pembuatan kecerdasan buatan atau pembelajaran mesin. Akan tetapi, karena beberapa permainan dengan ide sederhana memerlukan pemikiran yang lebih kompleks, tak jarang pada implementasinya, beberapa algoritma yang diusulkan memiliki kompleksitas ruang dan waktu yang sangat besar.

Permainan nonogram merupakan salah satu permainan logika yang saat ini masih cukup jarang diketahui, tidak setenar permainan logika lain seperti sudoku dan catur. Belum banyak solusi komputasi algoritma yang diusulkan untuk menyelesaikan permainan ini. Beberapa solusi yang telah diusulkan sebelumnya menyatakan bahwa permainan ini masuk dalam persoalan NP-*complete* [5] yang artinya diperlukan algoritma dengan kompleksitas yang tinggi untuk menyelesaikan permasalahan ini. Pengujian algoritma dengan ide lempang seperti algoritma *brute force* telah membuktikan bahwa setidaknya diperlukan kompleksitas waktu sebesar $O(2^{MN})$ untuk melakukan pengujian seluruh kemungkinan pengisian setiap petak pada nonogram biner (keputusan pengisian yang timbul hanya diisi atau dicoret) dengan M dan N menyatakan jumlah baris dan kolom secara berurutan. Tentu saja ini merupakan sebuah nilai yang sangat besar, terutama untuk pemrosesan dengan jumlah baris dan kolom yang banyak.

Heuristik berasal dari bahasa Yunani yaitu *eureka* yang berarti "menemukan". Teknik ini dapat digunakan untuk mempercepat pencarian solusi yang memiliki banyak jumlah kemungkinan. Secara umum, Teknik ini dapat mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasi seluruh kemungkinan secara penuh. Pada makalah ini, akan dibahas mengenai bagaimana cara melakukan optimasi terhadap algoritma *brute force* yang diimplementasikan dengan *exhaustive search* dan dikombinasikan dengan algoritma runut-balik (*backtracking*) sehingga menjadi pseudo-*exhaustive* dengan serangkaian teknik heuristik. Penulis akan merumuskan sebuah strategi algoritma rangkap (terdiri atas beberapa algoritma) berbasis pemecahan garis (*line-solving*) kasus-per-kasus yang dapat menyelesaikan permainan nonogram dengan kompleksitas mendekati kompleksitas waktu polinomial. Penulis juga akan menuliskan beberapa alternatif pengembangan solusi yang membuat algoritma usulan menjadi lebih mangkus dan sangkil lagi pada beberapa instansiasi kasus dengan pendekatan proses propagasi dan program dinamis (*dynamic programming*).

II. TEORI DASAR

A. Metode Line-Solving

Metode Pemecah baris atau yang bisa disebut sebagai *Line-solving Method* adalah sebuah algoritma berisi kumpulan prosedur yang menyelesaikan permasalahan setiap baris secara bergantian [2]. Pemecah baris secara umum adalah alat dasar untuk memecahkan permasalahan yang dinyatakan dalam dimensi $m \times n$.

Pada metode ini, langkah pertama pemecahan masalah yang dilakukan adalah mencari informasi yang dapat diperoleh dengan mempertimbangkan setiap baris dan kolom secara terpisah. Hal ini, secara tidak langsung, membuat perlunya melakukan penelusuran terhadap semua baris dan kolom dari ruang masalah dan menandai semua masukan yang nilainya dapat disimpulkan hanya dari petunjuk pada baris atau kolom tersebut. Meskipun seringkali solusi yang perlu dianalisis berjumlah eksponensial, beberapa permasalahan dapat diselesaikan dalam kompleksitas waktu polinomial untuk permasalahan dengan *n* buah baris [1].

B. Teknik Heuristik

Heuristik adalah seni dan ilmu menemukan. Kata heuristik diturunkan dari Bahasa Yunani yaitu "*eureka*" yang berarti menemukan (*to find* atau *to discover*) [7]. Heuristik mengacu pada teknik memecahkan persoalan berbasis pengalaman, dari proses pembelajaran, dan penemuan solusi meskipun tidak dijamin optimal.

Teknik ini umum digunakan untuk menyelesaikan permasalahan yang tidak memerlukan pembuktian besar secara matematis karena secara umum diimplementasikan menggunakan terkaan, intuisi, dan akal sehat, tetapi sangat berguna pada banyak kasus. Teknik heuristik yang baik dapat digunakan untuk mengeliminasi jumlah kemungkinan secara masif tanpa harus mengeksplorasi seluruh kemungkinan solusi secara penuh.

C. Algoritma Brute Force dan Exhaustive Search

Algoritma *Brute Force* adalah sebuah pendekatan yang mudah untuk memecahkan suatu masalah, biasanya didasarkan langsung pada pernyataan masalah dan definisi konsep yang dilibatkan atau lempang (*straight-forward*). Algoritma *Brute Force* memecahkan masalah dengan sangat sederhana, langsung, dan dengan cara yang cukup jelas [6].

Kelebihan dari algoritma ini adalah dapat diterapkan untuk memecahkan hampir sebagian besar masalah dan menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, dan menentukan elemen minimum atau maksimum dalam sebuah senarai. Algoritma ini juga sederhana dan mudah dimengerti. Adapun kekurangan dari algoritma ini adalah jarang menjadi algoritma dengan solusi yang mangkus dan umumnya lamban dalam melakukan pemrosesan data masukan dalam jumlah besar.

Teknik *Exhaustive Search* adalah adalah teknik pencarian solusi dengan menerapkan *brute force* untuk persoalan-persoalan kombinatorik seperti permutasi, kombinasi, dan operasi himpunan. Berikut adalah beberapa langkah implementasi *exhaustive search* [6].

1. Enumerasi setiap kemungkinan solusi dengan cara yang sistematis.
2. Evaluasi setiap kemungkinan solusi satu per satu, simpan solusi terbaik yang ditemukan sampai sejauh ini.
3. Bila pencarian berakhir, umumkan solusi terbaik.

Meskipun *exhaustive search* secara teoritis menghasilkan solusi, tetapi waktu atau sumber daya yang dibutuhkan dalam pencarian solusinya sangat besar. Hal yang menjadi poin utama dari makalah ini adalah tidak diperlukannya penelusuran ruang solusi secara penuh untuk memperoleh solusi yang tetap optimal. Pendekatan ini disebut sebagai pencarian pseudo-*exhaustive* (*pseudo-exhaustive search*).

D. Algoritma Runut-Balik (Backtracking)

Secara umum, runut-balik (*Backtracking*) dapat dipandang sebagai sebuah fase di dalam algoritma traversal DFS atau sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis, baik untuk persoalan optimasi maupun non-optimasi [8].

Algoritma runut-balik merupakan perbaikan dari *exhaustive search*. Jika pada *exhaustive search*, semua kemungkinan solusi dieksplorasi dan dievaluasi satu per satu, maka pada algoritma runut-balik, hanya pilihan yang mengarah pada solusi yang akan dieksplorasi, pilihan lainnya tidak akan dipertimbangkan lagi dan akan dipangkas dengan menggunakan fungsi pembatas (*bounding function*).

E. Proses Propagasi

Propagasi secara umum dapat diartikan sebagai penyebaran atau perambatan dari suatu hal atau informasi. Dalam konteks tertentu, propagasi dapat merujuk pada penyebaran gelombang atau sinyal dalam media seperti udara, air, atau kabel [9].

Dalam konteks algoritma, propagasi dapat merujuk pada penyebaran informasi atau data melalui jaringan atau struktur data. Propagasi sering digunakan dalam algoritma pembelajaran mesin atau jaringan saraf tiruan untuk mentransfer informasi antar neuron atau layer pada jaringan saraf [10]. Propagasi juga dapat merujuk pada cara penyebaran informasi pada algoritma optimasi, seperti pada algoritma genetika atau algoritma evolusi. Pada algoritma ini, propagasi digunakan untuk mengoptimalkan nilai parameter dalam populasi dan mencari solusi terbaik dengan cara mempertukarkan informasi antar individu pada setiap iterasi algoritma.

Beberapa jenis propagasi yang akan dibahas pada makalah ini adalah *forward checking*, *arc consistency*, dan *constraint propagation*. *Forward checking* adalah teknik propagasi yang paling umum digunakan pada algoritma runut-balik. Dengan *forward checking*, setiap kali sebuah nilai diisi, nilai yang mungkin pada suatu kondisi terkait akan diperiksa dan nilai-nilai yang tidak memenuhi batasan akan dihapus. Langkah ini mengurangi jumlah kemungkinan solusi yang perlu diperiksa pada setiap level runut-balik dan mempercepat waktu eksekusi. *Arc consistency* dan *constraint propagation* juga merupakan teknik propagasi yang dapat digunakan pada algoritma runut-balik. *Arc consistency* memastikan bahwa setiap nilai pada sebuah kondisi memenuhi semua batasan yang terkait dengan kondisi tersebut, sedangkan *constraint propagation* menyebarkan informasi batasan dari sebuah titik ke titik lain untuk mengurangi jumlah kemungkinan solusi yang perlu diperiksa [11].

F. Program Dinamis (Dynamic Programming)

Program dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan sedemikian sehingga solusi persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan [12]. Pada

program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas.

Prinsip Optimalitas menyatakan jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Prinsip optimalitas berarti hasil optimal dari tahap k dapat digunakan ketika proses akan dikerjakan dari tahap k ke tahap $k+1$, tanpa harus kembali ke tahap awal. Karakteristik dari persoalan yang dapat diselesaikan dengan algoritma ini adalah persoalan yang dapat dibagi menjadi beberapa tahap, pada setiap tahap hanya diambil satu keputusan dan masing-masing tahap terdiri dari sejumlah status yang berhubungan dengan tahap tersebut [12].

G. Permainan Nonogram

Permainan nonogram adalah sebuah teka-teki logika gambar yang dinyatakan dalam sebuah multi-larik berukuran $M \times N$. Permainan ini diciptakan oleh Non Ishida pada tahun 1987 dan pada tahun 1990, James Dalgety mempopulerkan permainan ini dengan sebutan nonogram yang diambil dari nama sang pencipta permainan [5].

Ide dari permainan ini cukup sederhana, setiap elemen pada multi-larik tersebut harus ditentukan apakah akan diisi (diwarnai) atau tidak (dicoret). Pada bagian kiri dari setiap baris dan bagian atas dari setiap kolom terdapat sekumpulan angka yang dinyatakan dalam sebuah larik sebagai petunjuk pengisian jumlah elemen matriks pada baris atau kolom tertentu. Larik tersebut dapat pula kosong. Salah satu aturan penting yang dinyatakan dalam permainan ini adalah tidak boleh ada sekumpulan blok hasil pemetaan yang diletakkan persis bertetangga. Misalkan terdapat sebuah baris yang memiliki petunjuk larik $[2, 1, 5]$, maka akan terdapat sebuah blok dengan dua petak berwarna, diikuti sebuah blok dengan satu petak berwarna dan sebuah blok dengan lima petak berwarna yang harus diseparasi oleh minimal satu buah petak. Petak separator ini kemudian dapat diberi tanda silang (dicoret) agar tidak ditempati pada iterasi mendatang.

Berikut adalah sebuah contoh nonogram kosong berukuran 6×6 dan penyelesaiannya.

	1	5	2	5	2 1	2
2 1						
1 3						
1 2						
3						
4						
1						

	1	5	2	5	2 1	2
2 1			x	x	x	
1 3	x		x			
1 2	x		x			x
3	x				x	x
4	x					x
1	x	x	x		x	x

Tabel 2.1. *Kondisi awal nonogram yang belum terpecahkan (kiri) dan setelah dipecahkan (kanan).*
Sumber : Dokumen pribadi.

Tampak pada bagian kanan bahwa pengisian petak pada suatu baris atau kolom harus sesuai dengan aturan penyusunan pada larik angka di bagian kiri dan atas dari kolom dan baris yang berkoresponden. Metode ini kemudian disebut sebagai pemrosesan garis (*line-solving*). Hal yang membuat permainan ini kemudian jadi menarik adalah terdapat sebuah gambar atau ilustrasi tersembunyi yang timbul setelah sebuah nonogram berhasil dipecahkan. Pada contoh ini, ilustrasi yang muncul adalah sebuah burung.

Secara umum solusi pengisian petak dalam sebuah nonogram adalah solusi tunggal unik. Akan tetapi, terdapat beberapa instansiasi khusus yang memiliki komponen larik penyusun yang sama, tetapi dengan kombinasi yang berbeda. Misalnya nonogram berukuran 3×3 berikut.

	1	1	1
1		x	x
1	x		x
1	x	x	

	1	1	1
1	x	x	
1	x		x
1		x	x

Tabel 2.2. *Contoh instansiasi sebuah nonogram dengan elemen larik penyusun yang sama, tetapi memiliki solusi berbeda.*
Sumber : Dokumen pribadi.

Meskipun mungkin, instansiasi diatas cukup jarang dijumpai pada kasus nonogram dengan jumlah elemen larik penyusun lebih dari satu.

Selain aturan pengisian petak, terdapat pula variasi lain dari nonogram, yaitu pengisian petak berwarna, membuat gambar ilustrasi yang dihasilkan menjadi lebih menarik lagi. Berikut adalah contoh nonogram berwarna berukuran 5×6.

	1	2	3	2	2
	1	1	1	1	1
			1		
1	x	x		x	x
3	x				x
5					
1	x	x		x	x
5					

Tabel 2.3. *Contoh sebuah nonogram berwarna yang membentuk gambar pohon.*
Sumber : Dokumen pribadi.

Tampak pada Tabel 2.3. sebuah nonogram berwarna yang menggambarkan pohon. Akan tetapi, sebagai batasan implementasi dari makalah ini, hanya akan ada aturan pewarnaan biner pada setiap petak, artinya setiap petak hanya akan mungkin untuk diisi (diberi warna lebih gelap) atau tidak diisi (perlu dicoret).

III. ANALISIS PERMASALAHAN

A. Pengantar Penyelesaian

Pada bagian ini, akan dijabarkan beberapa definisi, istilah, dan notasi yang digunakan pada makalah ini, baik sebagai bagian dari eksplanasi, maupun pada bagian penjelasan algoritma yang diusulkan. Permainan nonogram digambarkan dalam sebuah multi-larik $M \times N$ yang terdiri atas M buah baris dan N buah kolom penyusun. Adapun notasi r_i dan c_j berturut-turut menyatakan blok penyusun baris ke-i dan kolom ke-j. Notasi t_{ij} digunakan untuk menyatakan secara spesifik elemen pada petak baris ke-i dan kolom ke-j pada nonogram dengan ketentuan $t_{ij} = 0$ berarti petak belum diproses, $t_{ij} = -1$ berarti petak tidak akan diproses (petak dicoret), dan $t_{ij} = 1$ menandakan petak telah terisi.

Dalam setiap baris dan kolom terdapat kumpulan angka yang menandakan jumlah petak yang harus diisi. Bagian ini disebut sebagai larik angka penyusun. Dalam sebuah larik angka penyusun mungkin terdapat lebih dari satu angka yang

proses pemetaannya harus memenuhi ketentuan kecocokan mengikuti metode *line-solving*. Suatu baris r_i atau kolom c_j dikatakan memenuhi aturan pengisian yang cocok jika seluruh komponen petak pada baris atau kolom tersebut berhasil ditentukan (akan diisi atau tidak diisi/dicoret), tanpa melanggar batasan umum permainan nonogram, diantaranya.

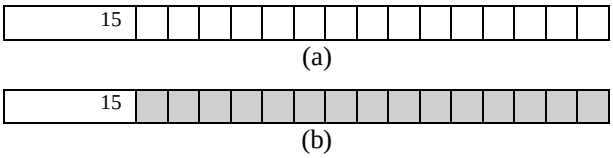
1. Proses pemetaan larik angka penyusun pada pengisian petak baris atau kolom tertentu memiliki jeda petak (akan dicoret) minimal satu buah petak.
2. Jika masih terdapat komponen petak yang masih belum dapat ditentukan, maka setidaknya terdapat sejumlah petak kosong bertetangga yang cukup untuk diisi sejumlah angka pada larik angka penyusun yang belum terpetakan.

Batasan diatas memungkinkan terciptanya solusi majemuk bagi baris atau kolom tertentu. Akan tetapi, perlu dicatat bahwa pada akhirnya, hanya terdapat satu buah solusi penyelesaian yang valid. Hal ini disebabkan untuk menentukan pengisian petak t_{ij} pada sebuah nonogram diperlukan terpenuhinya aturan pengisian yang cocok pada baris dan kolom yang berhubungan dengan baris atau kolom terkait.

B. Heuristik Dasar Metode Line-solving

Secara umum ada dua hal utama yang dapat dilakukan untuk menyelesaikan sebuah nonogram, yaitu pengisian petak dan pencoretan petak. Terdapat beberapa kasus khusus yang dapat dijadikan sebagai heuristik dasar pengisian dan pencoretan petak. Berikut adalah penjelasannya.

1. Pengisian Penuh (*complete fill*)
   Misalkan terdapat sebuah baris atau sebuah kolom dari nonogram yang larik angka penyusunnya hanya terdiri atas satu buah elemen yang memiliki nilai sama dengan baris atau kolom tersebut. Satu-satunya cara pengisian yang dapat dilakukan adalah dengan mengisi seluruh petak pada baris atau kolom tersebut. Berikut adalah ilustrasinya.



Tabel 3.1. *(a) Contoh ilustrasi pemecahan baris dengan pengisian penuh dan (b) hasil pengisiannya.*
Sumber : Dokumen pribadi.

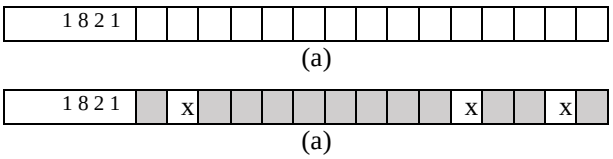
2. Pengisian Penuh dengan Beberapa Petak Tercoret (*complete fill with definite blanks*)
   Kasus ini mirip dengan kasus sebelumnya. Misalkan terdapat sebuah baris M atau kolom N dengan elemen penyusun larik angka n_i sebanyak k elemen yang memenuhi kondisi

$$\sum_{i=1}^{k} n_i + (k-1) = M$$

   untuk baris, atau

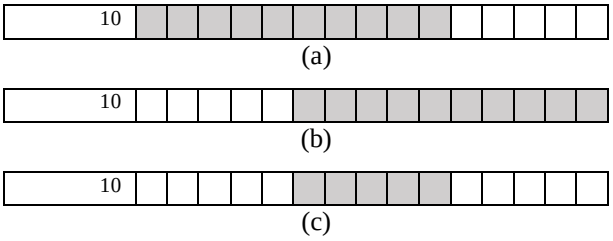
$$\sum_{i=1}^{k} n_i + (k-1) = N$$

   untuk kolom, maka akan terdapat $(k-1)$ petak yang tidak akan diisi (tercoret) berdasarkan elemen-elemen pada larik tersebut untuk memenuhi aturan pengisian yang cocok. Berikut adalah ilustrasi contoh kasusnya.



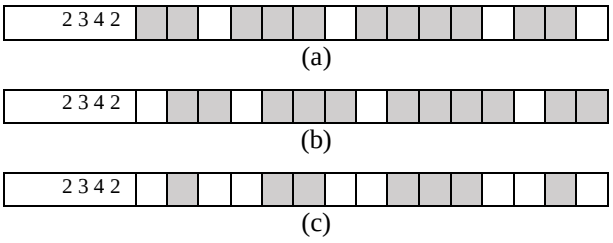
Tabel 3.2. *(a) Contoh ilustrasi pemecahan baris pengisian penuh dengan beberapa petak tercoret dan (b) hasil pengisiannya.*
Sumber : Dokumen pribadi.

3. Pengisian Parsial (*partial fill*)
   Misalkan terdapat sebuah baris atau kolom yang larik angka penyusunnya hanya terdiri atas satu elemen, tetapi memiliki nilai yang lebih besar dari separuh panjang baris atau kolom terkait. Maka pasti terdapat beberapa petak di tengah yang harus diisi. Kumpulan petak ini kemudian disebut sebagai superposisi petak ekstrem [3]. Berikut adalah ilustrasinya.



Tabel 3.3. *(a) Hasil pengisian petak pada tepi paling kiri dan (b) pada tepi paling kanan menghasilkan (c) pengisian petak pasti (obvious path) pada bagian tengah sebagai hasil superposisi keduanya.*
Sumber : Dokumen pribadi.

Bagian ini juga berlaku untuk jumlah blok elemen angka penyusun larik yang berjumlah lebih dari satu. Berikut adalah contohnya.



Tabel 3.4. *(a) Hasil pengisian petak pada tepi paling kiri dan (b) pada tepi paling kanan menghasilkan (c) pengisian petak pasti (obvious path) pada bagian tengah sebagai hasil superposisi keduanya.*
Sumber : Dokumen pribadi.

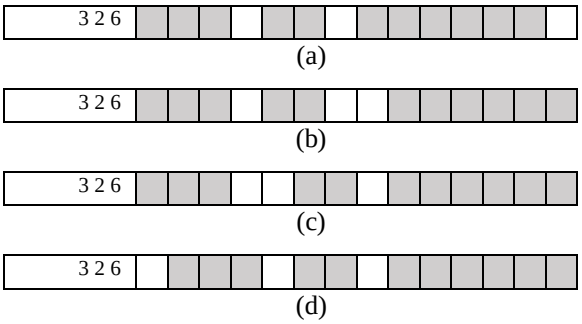
Tampak pada Tabel 3.4. (c) terdapat lebih banyak petak pasti (*obvious path*) yang harus diisi.

C. Heuristik dalam Optimasi Solusi Kombinatorial

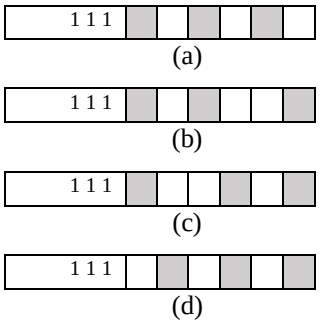
Dengan memanfaatkan berbagai karakteristik yang telah dijelaskan sebelumnya, dapat dilakukan pencarian seluruh kemungkinan pengisian dengan solusi kombinatorial yang lebih elegan.

Langkah awal yang perlu dilakukan adalah dengan menentukan baris atau kolom pada nonogram yang memenuhi kondisi pengisian penuh (*complete fill*) atau pengisian penuh dengan beberapa petak tercoret (*complete fill with definite blanks*). Langkah ini memberikan sebuah gambaran mengenai kotak yang perlu diisi terlebih dahulu. Pada akhir langkah ini hanya akan tersisa baris dan kolom yang memerlukan pengisian parsial (*partial fill*). Langkah yang dilakukan selanjutnya adalah melakukan instansiasi sebuah matriks untuk menampung seluruh kemungkinan solusi yang memenuhi aturan pengisian yang cocok. Proses pencarian semua kemungkinan dilakukan secara *exhaustive search* dengan pendekatan kombinatorik.

Misalkan terdapat sebuah instansiasi larik angka penyusun [3, 2, 6] pada sebuah baris dengan 14 petak untuk diisi dan larik [1, 1, 1] pada sebuah baris dengan 6 petak untuk diisi. Berikut adalah ilustrasi implementasi pengisian parsial pada kedua instansiasi tersebut.



Tabel 3.5. *Aturan pengisian yang cocok untuk instansiasi larik angka penyusun [3, 2, 6] pada baris dengan 14 petak.*
Sumber : Dokumen pribadi.



Tabel 3.6. *Aturan pengisian yang cocok untuk instansiasi larik angka penyusun [1, 1, 1] pada baris dengan 6 petak.*
Sumber : Dokumen pribadi.

Hal yang membuat instansiasi ini menarik untuk dijadikan contoh adalah karena diperoleh jumlah kemungkinan kombinasi yang sama untuk instansiasi kasus yang berbeda dari segi larik angka penyusun dan jumlah petak baris. Dengan pemikiran sederhana, pasti dapat disimpulkan terdapat hubungan dari kedua instansiasi tersebut. Hubungan ini dapat dinyatakan dalam kombinatorial.

Dengan mengabaikan petak separator, akan terdapat n buah blok dengan n menyatakan jumlah angka yang membentuk larik angka penyusun, sehingga perlu dilakukan pemilihan diantara $m+n$ buah blok untuk mendapatkan jumlah kemungkinan pengisian petak yang cocok. Secara matematis, hubungan ini dapat dinyatakan sebagai C_n^{m+n} dengan m menyatakan jumlah petak kosong yang bukan merupakan petak separator. Jika prosedur ini diaplikasikan pada instansiasi yang diberikan, maka akan diperoleh $C_3^{3+1} = C_3^4 = 4$ kemungkinan solusi, sesuai dengan yang diilustrasikan pada Tabel 3.5 dan 3.6. Hal ini akan mengurangi jumlah enumerasi kemungkinan yang tidak valid karena akan langsung terpangkas sejak awal proses pencarian.

Setelah ditemukan semua kemungkinan pengisian yang cocok untuk semua baris dan semua kolom pada nonogram tersebut, dilakukan pengurutan terhadap setiap baris dan kolom berdasarkan jumlah kemungkinan yang paling sedikit dan jumlah petak terbanyak yang belum terisi. Kemudian, dilakukan pengisian petak yang merupakan superposisi dari semua kemungkinan solusi baris atau kolom terkait, sesuai dengan heuristik pengisian parsial yang dijelaskan sebelumnya. Setelah dilakukan pengisian, dilakukan pembaharuan jumlah petak yang belum terisi dan pemrosesan dilanjutkan pada baris atau kolom berikutnya. Pemrosesan dilakukan hingga seluruh baris dan kolom selesai diproses dan memenuhi kondisi pengisian yang cocok. Berikut adalah bentuk *pseudocode* dari implementasinya.

Prosedur penyelesaian nonogram solusi kombinatorial

```
Terapkan complete fill
Terapkan complete fill with definite blanks
Enumerasi semua kemungkinan penyusunan baris dan kolom
while ada petak yang belum diproses do
    Pilih baris atau kolom dengan kemungkinan paling sedikit
    Terapkan partial fill dengan hasil superposisi seluruh kemungkinan
    if petak tersebut terisi untuk setiap kemungkinan then
        Isi petak tersebut
    else if petak tersebut tercoret untuk setiap kemungkinan then
        Coret petak tersebut
    else
        Tinggalkan petak tersebut tidak terisi
    endif
    Lakukan pembaharuan jumlah petak yang belum terisi pada baris
    dan kolom yang berkoresponden
    Lakukan pembaharuan kemungkinan yang dapat dipenuhi baris dan
    kolom yang berkoresponden
endwhile
```

Tampak pada *pseudocode* bahwa terdapat langkah untuk memperbaharui kemungkinan yang dapat dipenuhi oleh baris dan kolom yang berkoresponden. Prosedur ini dilakukan dengan mengkombinasikan algoritma diatas dengan algoritma runut-balik (*backtracking*) untuk menghasilkan solusi yang lebih optimal. Kemungkinan yang tidak memenuhi kondisi pengisian yang cocok akan dipangkas sebelum dilakukan ekspansi ke simpul (dalam hal ini baris atau kolom) yang lain.

Permainan nonogram merupakan permainan yang cukup kompleks. Misalkan terdapat sebuah nonogram dengan M buah baris dan N buah kolom yang belum terselesaikan. Dengan menggunakan implementasi algoritma *brute force* murni diperlukan setidaknya $\binom{M}{r}\binom{N}{c}$ cara untuk dapat memperoleh satu buah solusi yang valid dengan r adalah jumlah rata-rata elemen penyusun larik pada baris tertentu dan c adalah jumlah rata-rata elemen penyusun larik pada kolom tertentu. Jumlah ini membawa kompleksitas algoritma yang diperlukan untuk menyelesaikan sebuah nonogram adalah $O(2^{MN})$ [4]. Sangat tidak mangkus untuk pemrosesan dengan

jumlah baris dan kolom yang masif.

Akan tetapi, dengan implementasi sesuai dengan skema diatas akan diperoleh solusi dengan waktu pemecahan yang lebih baik. Karena dalam setiap baris hanya akan terdapat N buah kombinasi solusi yang memenuhi aturan pengisian yang cocok, maka kompleksitas algoritma dapat berkurang menjadi $O(M^N)$. Solusi masih memiliki kompleksitas eksponensial, tetapi solusi ini memperbaiki kompleksitas algoritma *brute force* yang sangat besar. Perlu diingat bahwa $2^{MN} > M^N$ karena $2^M > M$ pasti benar untuk $M > 0$.

IV. IMPLEMENTASI

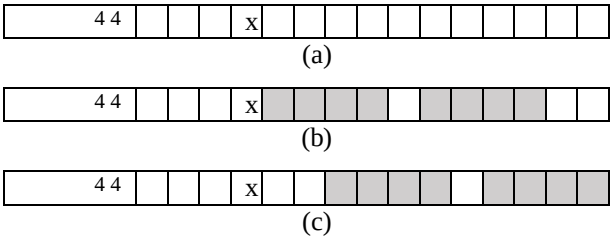
Pada bagian ini akan dibahas mengenai bagaimana cara melakukan optimasi teknik heuristik untuk melakukan optimasi solusi kombinatorial yang telah dibahas sebelumnya. Strategi dari teknik heuristik yang telah dibahas pada bagian III umumnya menggunakan teknik perbandingan dari solusi penyusunan paling kiri (*leftmost solution*) dan solusi penyusunan paling kanan (*rightmost solution*) dengan penyusunan paling kanan melakukan penekanan pengisian beberapa petak pertama pada elemen penyusun larik, dan penyusunan paling kiri melakukan penekanan pengisian beberapa petak terakhir pada elemen penyusun larik. Proses penyusunan solusi ini yang akan dioptimasi.

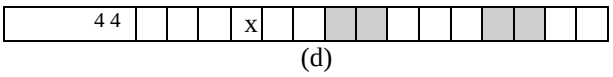
Solusi dibagi menjadi beberapa prosedur logis yang pada makalah ini diimplementasikan secara prosedural. Adapun solusi yang dibangun merupakan implementasi konkrit pseudo-*exhaustive search* dengan pendekatan heuristik yang teroptimalisasi.

A. Prosedur Pertama – Pertimbangan Tepi

Setelah melakukan enumerasi terhadap semua kemungkinan, dibantu dengan proses runut-balik pada akhir setiap pemrosesan baris atau kolom, terdapat beberapa langkah yang sejatinya tidak perlu ikut disimpan karena tidak mungkin.

Misalkan terdapat sebuah larik angka penyusun [4, 4] pada baris dengan 15 petak dan petak keempat dari kiri tidak boleh diisi (lihat Tabel 3.7 (a)). Proses pengisian parsial solusi paling kanan tidak perlu dilakukan dari petak paling kanan, cukup mulai dari petak yang berada di sebelah kanan petak coret tersebut. Hal ini bisa dilakukan karena tidak mungkin blok tersebut diisi sejumlah elemen paling awal dari larik angka penyusun (dalam kasus ini tidak mungkin mengisi empat petak pada saat hanya tersisa tiga petak kosong pada petak penyusunan paling awal). Solusi diselesaikan dengan mengisi petak yang memenuhi kondisi superposisi setiap komponen n_k. Berikut adalah ilustrasinya.

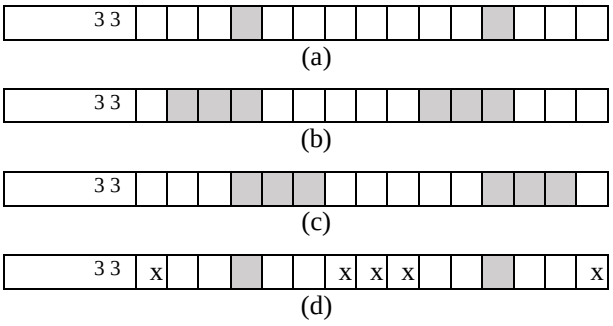




Tabel 3.7. *(a) Kondisi awal baris yang akan diproses, (b) pemrosesan solusi paling kiri, (c) pemrosesan solusi paling kanan, dan (d) pengisian petak pasti hasil superposisi keduanya.*
Sumber : Dokumen pribadi.

B. Prosedur Kedua – Pencoretan Kemungkinan Non-Solusi

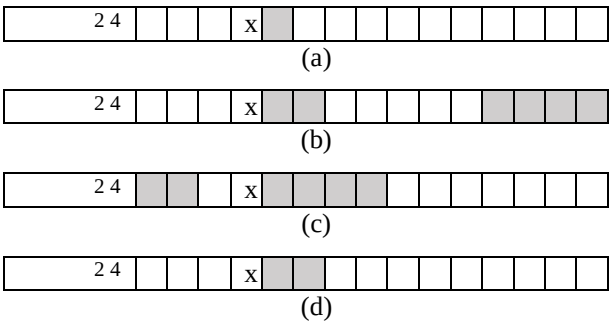
Prosedur ini akan menyelesaikan masalah petak kosong antara kondisi petak berurutan yang merupakan solusi blok paling kanan dan blok paling kiri. Hal ini akan mengurangi jumlah kemungkinan solusi dengan mencoret petak yang sudah tidak mungkin dikunjungi jika batasan wilayah dari masing-masing angka pada larik angka penyusun tidak akan mungkin tercapai. Prosedur diakhiri dengan mencoret petak yang berada diluar batas penyusunan ini. Berikut adalah ilustrasinya.



Tabel 3.8. *(a) Kondisi awal baris yang akan diproses, (b) pemrosesan solusi kiri blok terisi, (c) pemrosesan solusi kanan blok terisi, dan (d) pencoretan petak yang tidak akan mungkin diisi.*
Sumber : Dokumen pribadi.

C. Prosedur Ketiga – Pengisian Lintas Blok

Prosedur ini berfokus pada penyelesaian petak pada posisi diantara petak yang sudah tercoret sebelumnya. Setelah menemukan petak yang memenuhi kondisi tersebut, lakukan proses pencarian kemungkinan pengisian di sekitar petak yang sebelumnya telah tercoret, hindari setiap kemungkinan pengisian petak pada melewati bagian yang dicoret, memilih jumlah kemungkinan tersedikit dengan pengisian jumlah petak terbesar, dan mengisi petak dengan komponen yang sesuai (diisi atau dicoret). Berikut adalah ilustrasinya.

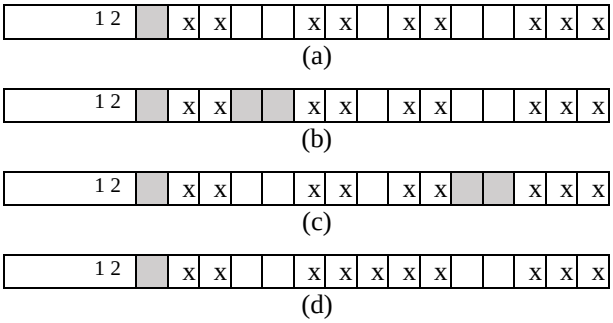


Tabel 3.9. *(a) Kondisi awal baris yang akan diproses, (b) pemrosesan solusi paling kanan, (c) pemrosesan solusi paling kiri, dan (d) pengisian petak pasti hasil superposisi.*
Sumber : Dokumen pribadi.

Hal unik dari prosedur ini adalah dimungkinkan terjadinya pengisian lintas blok (sesuai dengan judul prosedur). Dapat dilihat pada Tabel 3.9. (d) bahwa blok yang diisi merupakan irisan dari blok pertama solusi pertama (Tabel 3.9. (b)) dan blok kedua solusi kedua (Tabel 3.9. (c)). Akan tetapi, kondisi ini tetap benar adanya karena irisan keduanya merupakan salah satu petak pasti (*obvious path*) minimum yang harus diisi untuk membentuk konfigurasi blok sesuai elemen penyusun larik.

D. Prosedur Keempat – Restriksi Alternatif

Prosedur ini melakukan penyelesaian terhadap kasus sebuah baris atau kolom yang memiliki jumlah petak kosong lebih kecil dari angka pada larik angka penyusun yang belum terpetakan. Kondisi ini membuat perlunya dilakukan pemeriksaan baris atau kolom secara paralel untuk menambah restriksi pada sebuah blok (mencoret blok yang berada pada posisi yang kurang relevan). Berikut adalah contoh instansiasi kasusnya.

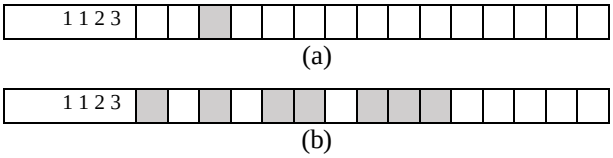


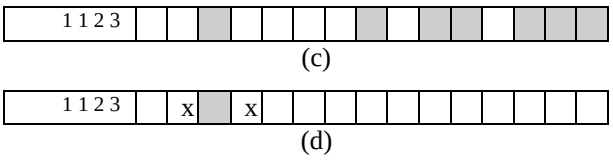
Tabel 3.10. *(a) Kondisi awal baris yang akan diproses, (b) pemrosesan solusi paling kiri, (c) pemrosesan solusi paling kanan, dan (d) pencoretan petak yang kurang relevan.*
Sumber : Dokumen pribadi.

Jika dicermati lebih jauh, petak kedelapan dari kiri berada diantara dua petak pasti yang tercoret (lihat Tabel 3.10 (a)). Ukuran dari petak kosong ini lebih kecil dari nilai larik angka penyusun yang belum terpetakan (dalam hal ini, $n = 2$). Keputusan memilih petak ini untuk diisi menjadi tidak relevan karena pilihan pengisian blok solusi paling kanan dan paling kiri tidak menyinggung petak tersebut. Akibatnya, petak kedelapan tersebut dapat ditetapkan sebagai sebuah petak pasti yang tidak akan diisi (dicoret).

E. Prosedur Kelima – Pembatasan Ekspansi

Prosedur ini merupakan langkah pamungkas yang menyelesaikan permasalahan minor yang mungkin dimiliki sebuah instansiasi pasca pemrosesan prosedur-prosedur sebelumnya. Secara umum, prosedur ini akan menangani pembatasan ekspansi dari masing-masing petak yang telah terisi sebelumnya. Berikut adalah contoh instansiasinya.





Tabel 3.11. *(a) Kondisi awal baris yang akan diproses, (b) pemrosesan solusi paling kiri, (c) pemrosesan solusi paling kanan, dan (d) pengisian petak pasti yang tidak mungkin terisi.*
Sumber : Dokumen pribadi.

Pada instansiasi awal (Tabel 3.11. (a)) terdapat banyak sekali petak yang belum terisi dan belum diketahui kondisinya apakah akan diisi atau tidak. Dengan melakukan percobaan pencarian solusi paling kanan dan paling kiri dari instansiasi tersebut, diperoleh kenyataan bahwa sejatinya terdapat beberapa petak yang tidak mungkin diisi untuk memenuhi aturan pengisian yang cocok. Petak ini kemudian dinyatakan sebagai petak pasti yang dicoret sehingga tidak akan mungkin diisi, dalam kasus ini adalah petak kedua dan keempat dari kiri (lihat Tabel 3.11. (d)).

F. Prosedur Penuh

Setelah berbagai prosedur penyelesaian masalah didefinisikan, semua prosedur tersebut diintegrasikan menjadi sebuah prosedur penuh yang dapat melakukan optimasi teknik heuristik selama pemrosesan penyelesaian berlangsung. Dengan mengaplikasikan seluruh prosedur, maka dapat dirumuskan *pseudocode* dari algoritma penyelesaian nonogram yang dioptimasi sebagai berikut.

Prosedur penyelesaian nonogram yang dioptimasi

```
Terapkan complete fill
Terapkan complete fill with definite blanks
Enumerasi semua kemungkinan penyusunan baris dan kolom
while ada petak yang belum diproses do
    Pilih baris atau kolom dengan kemungkinan paling sedikit
    if ini merupakan iterasi pertama then
        Terapkan partial fill dengan hasil superposisi seluruh
        kemungkinan
        if petak tersebut terisi untuk setiap kemungkinan then
            Isi petak tersebut
        else if petak tersebut tercoret untuk setiap kemungkinan then
            Coret petak tersebut
        else
            Tinggalkan petak tersebut tidak terisi
        endif
    else
        Terapkan partial fill dengan hasil superposisi seluruh
        kemungkinan
        for setiap kemungkinan yang akan diproses do
            Terapkan complete fill with definite blanks (jika mungkin)
            Terapkan prosedur pertama
            Terapkan prosedur kedua
            Terapkan prosedur ketiga
            Terapkan prosedur keempat
            Terapkan prosedur kelima
        endfor
    endif
    Lakukan pembaharuan jumlah petak yang belum terisi pada baris
    dan kolom yang berkoresponden
    Lakukan pembaharuan kemungkinan yang dapat dipenuhi baris dan
    kolom yang berkoresponden
endwhile
```

G. Propagasi dan Implementasi Program Dinamis

Proses propagasi, seperti yang telah dijelaskan sebelumnya, dapat digunakan untuk melakukan optimasi terhadap proses pemangkasan jumlah solusi yang perlu ditelusuri. Implementasi proses propagasi dalam menyelesaikan nonogram dapat diwujudkan dengan proses menyebarluaskan informasi atau kondisi yang telah diketahui pada sebuah blok tertentu terhadap setiap petak pada baris atau kolom yang sama. Berikut adalah skema pemeriksaan yang dilakukan dengan propagasi.

1. Analisis larik angka penyusun pada setiap baris dan kolom untuk mengetahui petak yang harus diisi dan yang perlu dicoret.
2. Terapkan propagasi pada baris atau kolom terkait. Misalnya, jika pada sebuah baris telah diketahui bahwa petak pertama dan ketiga harus diisi, maka petak di antara keduanya harus dicoret. Proses propagasi dapat dikombinasikan dengan pembangkitan logika heuristik yang dibahas sebelumnya.
3. Setelah propagasi dilakukan, lakukan pemeriksaan kembali petak yang masih kosong dan terapkan propagasi bagi setiap simpul jelajah (dalam hal ini, setiap kemungkinan) yang mungkin. Proses ini diulang sampai seluruh baris dan kolom pada nonogram memenuhi aturan pengisian yang cocok.

Prosedur yang diterapkan pada bagian ini secara umum mirip dengan prosedur yang diterapkan pada algoritma runut-balik yang telah diimplementasikan sebelumnya. Akan tetapi, solusi ini tidak memerlukan penelusuran secara mendalam dengan percobaan semi-*exhaustive search* seperti yang diterapkan pada algoritma runut-balik. Akibatnya proses ini akan secara umum akan lebih mangkus.

Implementasi lebih jauh dari program dapat diterapkan dengan menggunakan program dinamis (*dynamic programming*) untuk melakukan pemilihan logika heuristik yang lebih spesifik bagi suatu instansiasi. Misalnya pada suatu instansiasi lebih cocok untuk menerapkan prosedur ketiga terlebih dahulu daripada prosedur pertama, atau lebih cocok menerapkan prosedur keempat terlebih dahulu daripada prosedur ketiga. Akan tetapi, perlu dicatat bahwa ada kemungkinan program dinamis tidak menghasilkan solusi yang optimal, terutama pada kasus nonogram yang kompleks (jumlah elemen larik angka penyusun yang banyak dan bernilai kecil), sehingga metode ini lebih baik tidak diimplementasikan secara independen, melainkan dikombinasikan dengan proses propagasi. Tahap yang ditekankan dalam proses pembentukan solusi dengan program dinamis adalah proses pemilihan skema heuristik sehingga diperoleh pendekatan solusi yang lebih mangkus.

H. Analisis Kompleksitas Algoritma

Membuat sebuah algoritma yang ditargetkan lebih mangkus dan sangkil akan terasa kurang lengkap tanpa adanya analisis kompleksitas algoritma. Penyelesaian dari nonogram sendiri sudah digolongkan sebagai masalah NP-*complete* [5] yang artinya kompleksitas eksponensial sudah tergolong cukup bagus pada permasalahan ini.

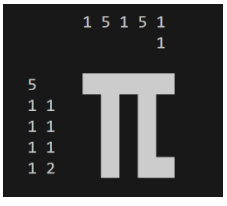
Telah dibahas pada bagian akhir bagian III bahwa untuk sebuah nonogram dengan M buah baris dan N buah kolom yang belum terselesaikan, setiap baris hanya akan terdapat N buah kombinasi solusi yang memenuhi aturan pengisian yang cocok. Kondisi ini membawa implementasi algoritma dengan solusi kombinatorial tersebut secara umum dapat didelesaikan kompleksitas algoritma $O(M^N)$, jauh lebih mangkus daripada kasus terburuk pencarian solusi algoritma *brute force* murni dengan kompleksitas algoritma $O(2^{MN})$.

Setelah dilakukan berbagai skema optimasi dengan menggunakan beberapa pendekatan heuristik yang dijelaskan secara panjang lebar pada bagian ini, penyelesaian sebuah nonogram secara umum akan lebih mangkus karena tidak diperlukan proses pencarian kemungkinan solusi yang menyeluruh untuk kasus yang sejatinya bisa diselesaikan lebih cepat. Dengan melakukan implementasi teknik heuristik, akan tercapai kompleksitas algoritma $O(M^{(N-n)})$ untuk kasus terburuk dengan n menyatakan jumlah kemungkinan yang bisa dipangkas akibat penerapan heuristik dan $O(MN)$ untuk kasus terbaik dimana hanya diperlukan sejumlah iterasi untuk menyelesaikan nonogram supaya memenuhi aturan pengisian yang cocok. Kembali, solusi pada kasus terburuk masih merupakan solusi eksponensial. Akan tetapi, solusi ini memperbaiki kompleksitas algoritma sebelumnya meskipun dengan nilai yang relatif kecil. Perlu diingat bahwa $M^N > M^{(N-n)}$ karena $N > N - n$ pasti benar untuk $n > 0$ dan MN jauh lebih kecil daripada M^N.

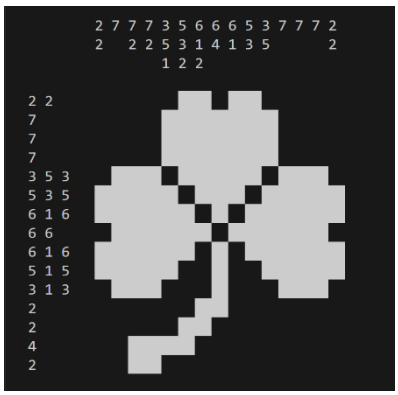
Lebih lanjut, penerapan proses propagasi dalam menentukan simpul jelajah selanjutnya memangkas jumlah solusi yang perlu ditelusuri secara dramatis. Beberapa teknik propagasi seperti *forward checking* dan *constraint propagation* dapat menekan kompleksitas algoritma pada kasus rata-rata penyelesaian nonogram hingga $O(MN)$, tergantung pada kompleksitas nonogram dan efektivitas propagasi dalam mengurangi kemungkinan yang perlu diperiksa. Program dinamis yang dikombinasikan dengan proses propagasi juga akan menghasilkan kompleksitas algoritma pada kasus rata-rata yang sama, tetapi dengan rentang distribusi kompleksitas algoritma kasus yang lebih leptokurtosis.

V. UJI COBA

Proses uji coba dilakukan dengan menggunakan berbagai kasus uji dengan ukuran yang bervariasi untuk setiap sampelnya. Sebagai catatan pengujian dilakukan pada program yang dibangun oleh penulis selama proses eksplorasi solusi dan penyusunan algoritma berlangsung. Berikut adalah beberapa kasus uji dan hasil yang diperoleh.



Gambar 5.1. *Hasil pemrosesan nonogram bergambar phi berukuran* 5×5.
Sumber : Dokumen pribadi.



Gambar 5.2. *Hasil pemrosesan nonogram bergambar daun clover berukuran* 15×15.
Sumber : Dokumen pribadi.

Gambar 5.3. *Hasil pemrosesan nonogram bergambar penguin dan pohon cemara berukuran* 20×15.
Sumber : Dokumen pribadi.

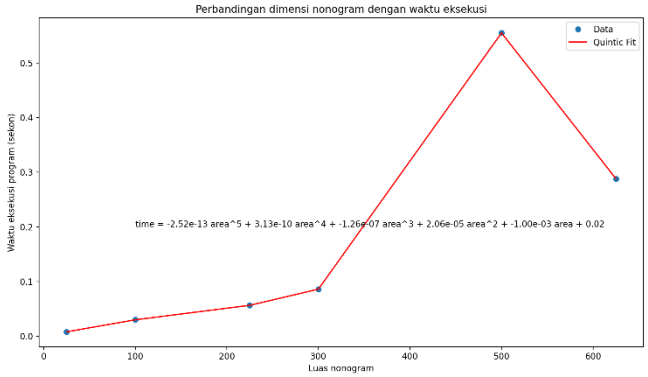
Gambar 5.4. *Hasil pemrosesan nonogram bergambar sebuah kode QR berukuran* 25×25. *Silahkan pindai untuk memperoleh informasi yang dimiliki kode QR tersebut.*
Sumber : Dokumen pribadi.

Hasil eksplorasi dan pengujian berupa implementasi program berhasil menyelesaikan berbagai nonogram mulai dari ukuran yang kecil (5×5) hingga ukuran yang relatif lebih besar (25×25). Pengujian dengan nonogram berukuran cukup besar untuk bentuk non-persegi (20×15) juga terbukti cukup efektif untuk diselesaikan dengan algoritma penyelesaian yang sebelumnya dibangun.

Sebagai informasi penjelas, berikut adalah hasil percobaan untuk pemrosesan ukuran nonogram yang bervariasi.

Panjang	Lebar	Luas	Bentuk	Waktu eksekusi
5	5	25	Phi	0.00800 s.
10	10	100	Ikan	0.02999 s.
15	15	225	Daun *clover*	0.05654 s.
20	15	300	Penguin & pohon	0.08604 s.
20	25	500	Kelinci	0.55500 s.
25	25	625	Kode QR	0.28799 s.

Tabel 5.1. *Tabel hasil pemrosesan berbagai ukuran nonogram.*
Sumber : Dokumen pribadi.



Gambar 5.5. *Hasil pemrosesan grafik variasi percobaan ukuran nonogram yang diujikan.*
Sumber : Dokumen pribadi.

Setelah dilakukan berbagai percobaan, dibuat sebuah grafik regresi polinomial untuk menyatakan hubungan antara luas daerah penyelesaian nonogram dengan waktu eksekusi yang diperlukan untuk menyelesaikan nonogram terkait. Dengan regresi quintik (polinomial berderajat 5) diperoleh persamaan

$$t = -2.52 \times 10^{-13}\, A^5 + 3.13 \times 10^{-10}\, A^4 - 1.26 \times 10^{-7}\, A^3 + 2.06 \times 10^{-5}\, A^2 - 1.00 \times 10^{-3}\, A + 0.02$$

dengan t menyatakan total waktu eksekusi dalam sekon dan A menyatakan area atau luas dari nonogram yang diproses.

Melalui hubungan ini, tampak jelas bahwa korelasi antara luas nonogram dengan waktu eksekusinya masih berhubungan eksponensial (terbukti dengan diperlukannya hampiran persamaan regresi polinomial derajat tinggi, setidaknya sejumlah ukuran data). Akan tetapi, nilai antar data tidak mengalami perubahan yang sangat signifikan. Tentu saja terdapat berbagai faktor lain yang mempengaruhi proses eksekusi solusi seperti intensitas pengisian baris atau kolom tertentu (contoh pada gambar kelinci dengan dimensi 20×25), maupun banyaknya petak yang harus dicoret. Namun, hampiran nilai dalam bentuk persamaan polinomial yang landai (koefisien polinomial derajat 5 berada pada orde 10^{-13}) membuktikan bahwa solusi algoritma yang diusulkan telah cukup efektif mengurangi kompleksitas waktu pada berbagai instansiasi persoalan nonogram.

Penulis berhasil melakukan implementasi fisik algoritma yang telah disusun dan mengarsipkannya dalam sebuah tautan yang dapat diakses secara publik. Uji coba dan pengembangan lebih lanjut dapat dilakukan dengan mengakses laman *github* penulis berikut :

https://github.com/mikeleo03/Nonogram-Solver

VI. KESIMPULAN

Permainan nonogram merupakan permainan berbasis logika yang masih cukup jarang dikenal banyak orang. Permainan ini memiliki dasar permainan yang cukup sederhana, tetapi memerlukan proses pencarian sedikit rumit untuk memperoleh solusi dalam waktu yang optimal. Sebelumnya telah banyak dibahas mengenai solusi penyelesaian metode *line-solving* yang lebih mangkus untuk menyelesaikan permainan nonogram dengan kompleksitas yang masih eksponensial, tetapi jauh lebih landai. Selain itu, telah dilakukan pula serangkaian proses uji coba yang berhasil menunjukkan bahwa penyelesaian permainan nonogram sejatinya dapat dihampiri dengan kompleksitas polinomial orde tinggi, memberikan harapan bagi salah sebuah permasalahan yang tergolong NP-*complete* untuk dapat diselesaikan dengan kompleksitas algoritma yang lebih baik. Adapun tautan kode sumber (*source code*) terlampir dan penjelasan mengenai teknik propagasi dan program dinamis yang juga dijelaskan sebelumnya harapannya dapat mendukung proses pengembangan lebih lanjut sehingga dapat tercipta secara fisik sebuah penyelesaian nonogram secara komputasional yang lebih sangkil kedepannya.

VII. UCAPAN TERIMA KASIH

Puji Syukur hanya kepada Tuhan Yang Maha Esa karena hanya atas berkat dan limpahan rahmatNya, penulis dapat menyelesaikan makalah ini dengan baik. Terima kasih juga penulis sampaikan kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc. sebagai dosen pengampu dalam mata kuliah IF2211 Strategi Algoritma kelas K02 atas ilmu yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan makalah ini. Tidak lupa terima kasih juga penulis sampaikan kepada orang tua yang senantiasa memberikan dukungan dan motivasi kepada penulis.

VIII. LAMPIRAN

Tautan video penjelasan : https://youtu.be/K1HWkHLoVuc

REFERENSI

[1] https://www.sciencedirect.com/science/article/pii/S0166218X14000080 diakses pada 1 April 2023
[2] https://www.scirp.org/journal/paperinformation.aspx?paperid=104406 diakses pada 1 April 2023
[3] https://www.nonograms.org/methods diakses pada 1 April 2023
[4] https://homepages.cwi.nl/~kbatenbu/papers/bako_pr_2009.pdf diakses pada 1 April 2023
[5] http://en.wikipedia.org/wiki/Nonogram diakses pada 1 April 2023
[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf diakses pada 1 April 2023
[7] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf diakses pada 1 April 2023
[8] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf diakses pada 1 April 2023
[9] https://kbbi.kemdikbud.go.id/entri/propagasi diakses pada 1 April 2023
[10] https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd diakses pada 1 April 2023
[11] https://home.csulb.edu/~tebert/teaching/lectures/551/ac/ac.pdf diakses pada 1 April 2023
[12] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf diakses pada 1 April 2023

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Mei 2023

Michael Leon Putra Widhi
13521108