# PERBANDINGAN KINERJA ALGORITMA GOLOMB CODE DAN TABOO CODE PADA KOMPRESI FILE MP4

Megawati[1], Sinar Sinurat[2], Kurnia Ulfa[3]
[1,2,3]Universitas Budi Darma, Medan
[1]megawati8135@gmail.com, [2]sinurat.sin@gmail.com,
[3]kurniaulfa82@gmail.com

## ABSTRAK

*Kompresi file adalah salah satu teknik penting dalam pengolahan data untuk mengurangi ukuran file tanpa mengorbankan kualitas data. Algoritma kompresi yang efektif tidak hanya mengurangi ukuran file tetapi juga mempertahankan integritas data. Pada penelitian ini, kami membandingkan kinerja dua algoritma kompresi, yaitu Golomb Code dan Taboo Code, dalam konteks kompresi file MP4. Golomb Code adalah algoritma kompresi yang efisien untuk data dengan distribusi geometrik, sering digunakan dalam kompresi lossless. Taboo Code adalah algoritma kompresi yang lebih baru, dirancang untuk mengatasi beberapa keterbatasan dari metode kompresi tradisional dengan menggunakan teknik pengkodean yang lebih adaptif dan fleksibel. Penelitian ini mengevaluasi kinerja kedua algoritma berdasarkan beberapa metrik, yaitu rasio kompresi, kecepatan kompresi, dan kualitas file hasil kompresi. Percobaan dilakukan pada serangkaian file MP4 dengan berbagai ukuran dan konten untuk mendapatkan hasil yang komprehensif dan generalisasi yang lebih baik. Hasil penelitian menunjukkan bahwa masing-masing algoritma memiliki keunggulan dan kelemahan tersendiri. Golomb Code menunjukkan performa yang lebih baik pada file dengan pola data yang lebih teratur dan distribusi data yang sesuai dengan asumsi algoritma. Sementara itu, Taboo Code menunjukkan keunggulan dalam fleksibilitas dan adaptabilitasnya pada file dengan pola data yang lebih kompleks dan tidak teratur.*

*Kata Kunci: kompresi, golomb code, taboo code, MP4, kualitas file*

## *ABSTRACT*

*File compression is a crucial technique in data processing for reducing file size without sacrificing data quality. Effective compression algorithms not only reduce file size but also maintain data integrity. This study compares the performance of two compression algorithms, Golomb Code and Taboo Code, in the context of MP4 file compression. Golomb Code is an efficient compression algorithm for data with geometric distribution, commonly used in lossless compression. Taboo Code is a newer compression algorithm designed to address some limitations of traditional compression methods by using more adaptive and flexible encoding techniques. The study evaluates the performance of both algorithms based on several metrics, including compression ratio, compression speed, and quality of the compressed file. Experiments were conducted on a range of MP4 files with varying sizes and content to achieve comprehensive results and better generalization. The findings indicate that each algorithm has its own strengths and weaknesses. Golomb Code performs better with files that have more regular data patterns and data distributions that align with the algorithm's assumptions. In*

*contrast, Taboo Code excels in flexibility and adaptability with files that have more complex and irregular data patterns.*

*Keywords: compression, golomb code, taboo code, mp4, file quality*

## A. Pendahuluan

Kompresi data adalah teknik untuk mengubah data, seperti kumpulan karakter, menjadi format kode dengan tujuan menghemat ruang penyimpanan dan waktu transmisi data (Imani et al., 2021);(Darnita, 2019). Proses ini sangat penting karena ukuran data yang semakin besar memerlukan media penyimpanan yang lebih luas. Dalam memilih algoritma kompresi, beberapa faktor krusial diperhatikan, termasuk kebutuhan sumber daya seperti memori dan kecepatan PC, serta kecepatan kompresi dan ukuran file hasil kompresi (Andriyani, 2024). Metode kompresi dikembangkan untuk mengurangi kebutuhan penyimpanan dan mempermudah pengelolaan data, salah satunya adalah kompresi data teks yang sering digunakan untuk mengurangi ukuran file (Iqbal, 2019). File digital dengan format MP4 biasanya memiliki ukuran yang sangat besar, sehingga membutuhkan media penyimpanan yang cukup luas (Setiawan, 2023). Kompresi data menjadi sangat penting dalam konteks ini untuk mengurangi kebutuhan penyimpanan, mempercepat proses pengiriman data, dan mengurangi kebutuhan bandwidth. Salah satu format kompresi video yang umum digunakan adalah MPEG-4, yang berfungsi mengompresi video dengan mengurangi ukuran setiap frame dan mengompresi sekumpulan frame secara bergantian (Setiawan, 2023). Ini memungkinkan pengelolaan video yang lebih efisien tanpa mengorbankan kualitas secara signifikan.

Terdapat berbagai algoritma kompresi yang tersedia, seperti *Golomb Code dan Taboo Code*, masing-masing dengan kelebihan dan kekurangan tersendiri. *Algoritma Golomb Code*, misalnya, sering digunakan dalam kompresi data yang memiliki distribusi probabilitas yang tidak merata, sedangkan *Taboo Code* lebih fokus pada pengurangan redundansi (Gulo, 2017). Setelah melakukan kompresi dengan algoritma-algoritma ini, penting untuk

membandingkannya untuk menentukan mana yang lebih efisien dalam hal rasio kompresi dan kecepatan (Bant et al., 2018). Perbandingan ini dapat memberikan wawasan tentang algoritma mana yang lebih unggul dalam aplikasi tertentu. Penelitian mengenai perbandingan antara *Golomb Code dan Taboo Code* dalam konteks kompresi file audio masih sangat terbatas. Oleh karena itu, penulis tertarik untuk mengeksplorasi perbedaan performa kedua algoritma ini dalam kompresi file audio untuk menentukan mana yang lebih baik dan lebih cepat. Penelitian ini bertujuan untuk memberikan panduan yang lebih jelas bagi pemilihan algoritma kompresi yang tepat dalam aplikasi nyata, serta meningkatkan efisiensi proses kompresi data secara keseluruhan.

Penelitian Agung Maulana Nst pada tahun 2019 mengenai kompresi file audio mengungkapkan bahwa *algoritma Half Byte dan Deflate efektif* dalam mengurangi ukuran file, sehingga mengurangi kebutuhan kapasitas penyimpanan di memori atau RAM. Algoritma *Half Byte* berhasil mengurangi ukuran file dengan baik, sedangkan algoritma *Deflate* juga memberikan hasil yang memuaskan dalam hal efisiensi kompresi. Selain itu, kedua algoritma tersebut mampu menjaga tingkat keamanan file yang dikompresi tanpa mengalami kerusakan (NST, 2019). Berdasarkan penelitian yang dilakukan oleh Nuryasin pada tahun 2014 tentang perbandingan kompresi file data menggunakan algoritma Huffman, Half Byte, dan Run Length, ditemukan bahwa algoritma Huffman adalah yang paling efisien dalam hal pemampatan file dibandingkan dengan *algoritma Run Length dan Half Byte*. Hasil perbandingan yang disajikan dalam bentuk tabel menunjukkan keunggulan algoritma Huffman dalam mengurangi ukuran file secara lebih efektif (Nuryasin, 2014). Berdasarkan penelitian Eka Kristian Gulo pada tahun 2017 tentang perancangan aplikasi kompresi audio menggunakan algoritma Golomb, dapat disimpulkan bahwa algoritma *Golomb* efektif untuk kompresi audio digital. *Algoritma* ini bekerja dengan menyederhanakan kode penyusun file audio dan mengoptimalkan pengurangan ukuran file dengan menghitung frekuensi kemunculan setiap *byte* dalam file audio. Proses ini

membantu dalam mengurangi ukuran file secara signifikan [5]. Berdasarkan penelitian Linni Hairani Butar-Butar pada tahun 2020 mengenai penerapan *algoritma Golomb Coding* pada kompresi *Short Message Service* (SMS), disimpulkan bahwa *Golomb Coding* efektif dalam mengompresi SMS. Algoritma ini bekerja dengan mengganti setiap bit dan memperkecil ukuran setiap karakter, yang mengurangi jumlah karakter dalam pesan. Dengan semakin banyaknya teks yang dikirim, biaya pengiriman SMS dapat meningkat, sehingga kompresi ini penting untuk menghemat biaya pengiriman (Hairani & Butar, 2021). Berdasarkan penelitian yang dilakukan oleh Saiful Simanjuntak mengemukakan bahwa algoritma *taboo code* yang dikembangkan oleh Steven Pigeon, menggunakan pendekatan terhadap panjang variabel. Prinsipnya adalah memilih bilangan positif n dan membalikkan pola n untuk menandakan akhir kode tersebut (Simanjuntak, 2022).

Dari beberapa penjabaran diatas, penulis telah mengumpulkan beberapa jurnal dari beberapa sumber yang berkaitan dengan permasalahan yang sedang penulis bahas. Oleh sebab itu penulias melakukan penelitian untuk memecahkan masalah yang telah disebutkan diatas dengan mengangkat judul "***Perbandingan Kinerja Algoritma Golomb Code Dan Taboo Code Pada Kompresi File MP4 ".***

## B. Metode Penelitian

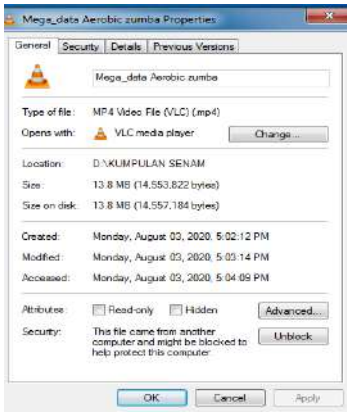
Dalam metodologi penelitian, kerangka kerja mencakup beberapa tahapan sistematis untuk mempermudah penelitian. Tahapan tersebut meliputi: 1) Identifikasi Masalah untuk menguraikan isu terkait ukuran dan transfer file MP4; 2) Studi Pustaka untuk memahami objek penelitian melalui referensi; 3) Analisa Kompresi dengan algoritma *Golomb Code dan Taboo Code* untuk mengurangi ukuran file MP4; 4) Perbandingan Algoritma untuk menilai efektivitas kedua algoritma; 5) Perancangan Sistem untuk membuat aplikasi perbandingan menggunakan *Microsoft Visual Studio* 2010; 6) Implementasi untuk membangun dan memastikan perangkat lunak berfungsi dengan baik; dan 7) Pengujian untuk mengevaluasi hasil akhir dari penelitian.

**Sampel Data**

Dengan melakukan kompresi *file MP4*, *file* yang berukuran besar akan dikompresi menjadi ukuran yang lebih kecil dan akan mengurangi alokasi penyimpanan. Sebagai penerapan algoritma *file audio* yang akan dikompresi adalah *file* dengan format *MP4.*

Tabel 1 *File Audio* Yang Akan Dikompresi

| Nama File | Ukuran | Type |
|---|---|---|
| Mega_data Aerobic zumba | 13,8 MB | *MP4* |




Gambar 1 Gambar Detail *File MP4*

## C. Hasil Penelitian dan Pembahasan

Untuk memperkecil ukuran *file* mp4 menggunakan algoritma *golomb code*, di bagi menjadi 2 bagian antara lain :

1. Proses Kompresi

Tahap pembacaan isi *file* nilai heksadecimal yang diambil dimasukkan kedalam tabel untuk dilakukan pembacaan frekuensi. Tahap ini akan dilakukan secara manual dengan mengurutkan nilai berdasarkan kemunculan terbanyak dari tiap niali yang sama dan frekuensi tersebut adan berada di urutan pertama dari susunan.

Langkah awal dalam kompresi algoritma *Golomb Code* yaitu menentukan nilai frekuensi dengan mengurutkan dari frekuensi terbesar ke frekuensi terkecil.Dapat dilihat pada tabel 2 berikut.

Tabel 2 Frekuensi sebelum dikompresi

| n | Nilai *Hexadecimal* | Biner | Frek | Bit | Frek x Bit |
|---|---|---|---|---|---|
| 1 | 45 | 01000101 | 3 | 8 | 24 |
| 2 | 49 | 01001001 | 3 | 8 | 24 |
| 3 | 4E | 01001110 | 2 | 8 | 16 |
| 4 | 41 | 01000001 | 2 | 8 | 16 |
| 5 | 20 | 00100000 | 2 | 8 | 16 |
| 6 | 57 | 01010111 | 2 | 8 | 16 |
| 7 | 55 | 01010101 | 2 | 8 | 16 |
| 8 | 50 | 01010000 | 1 | 8 | 8 |
| 9 | 4C | 01001100 | 1 | 8 | 8 |

| | | | | | |
|---|---|---|---|---|---|
| 10 | 54 | 01010100 | 1 | 8 | 8 |
| 11 | 44 | 01000100 | 1 | 8 | 8 |
| 12 | 53 | 01010011 | 1 | 8 | 8 |
| 13 | 4D | 01001101 | 1 | 8 | 8 |
| 14 | 52 | 01010010 | 1 | 8 | 8 |
| Total Bit | | | | | 184 |

Setelah mengetahui ukuran sebelum dikompresi, selanjutnya akan dilakukan proses kompresi menggunakan algoritma *Golomb Code*. Satu nilai karakter dari bilangan *hexadecimal* tersebut masing-masing bernilai 8 bit bilangan biner. Sehingga 23 bilangan *hexadecimal* mempunyai nilai biner sebanyak 184 bit. Untuk mengubah satuan menjadi *byte*, maka jumlah keseluruhan bit akan dibagi 8 yang menghasilkan 184/8 = 23.

**Proses Dekompresi**

Selanjutnya proses dekompresi hal yang dilakukan adalah menganalisa keseluruhan bit hasil dari kompresi sebelumnya. Adapun bit keseluruhan hasil kompresi dapat dilihat pada tabel berikut.

Tahapan berikutnya dalam melakukan dekompresi menggunakan algoritma *Golomb Code* adalah mengubah karakter kedalam nilai desimal dan nilai biner dari hasil kompresi sebelumnya yang telah diurutkan. Berikut adalah tabel urutan karakter berdasarkan nilai desimal dan biner:

Tabel 3 Nilai Desimal dan Nilai Biner Hasil Kompresi

| No | Karakter | Nilai Desimal | Nilai Biner |
|---|---|---|---|
| 1 | X | 88 | 01011000 |
| 2 | DLE ( *Data Line Escape*) | 16 | 00010000 |
| 3 | 0 | 48 | 00110000 |
| 4 | , | 44 | 00101100 |
| 5 | Ë | 137 | 10001001 |
| 6 | Æ | 146 | 10010010 |
| 7 | SUB ( *Substitute* ) | 26 | 00011010 |
| 8 | ACK ( *Acknowledge* ) | 6 | 00000110 |
| 9 | ■ | 220 | 11011100 |
| 10 | LF ( *Line Feed* ) | 10 | 00001010 |
| 11 | DC 2 ( *Device Control* 2 ) | 18 | 00010010 |
| 12 | N | 110 | 01101110 |
| 13 | T | 84 | 01010100 |
| 14 | Ò | 149 | 10010101 |
| 15 | SCH ( *Start of Heading* ) | 1 | 00000001 |
| 16 | BS ( *Backstage* ) | 8 | 00001000 |

**Penerapan Algoritma Taboo Code**

Proses yang pertama dilakukan untuk memperkecil ukuran *file MP4* dengan menggunakan algoritma *taboo Code*, yaitu :

1. Proses Kompresi

   Pada tahap ini, akan dilakukan proses kompresi dengan algoritma *Taboo Code* dengan langkah pertama yang dilakukan adalah menyusun nilai-nilai *hexadecimal* yang diperoleh sebelumnya berdasarkan frekuensi kemunculannya,dimulai dari karakter yang jumlah terbanyak hingga yang jumlahnya sedikit.

Tabel 4 Nilai bilangan *Hexadecimal* pada sampel *file* teks

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | 45 | 4E | 45 | 4C | 49 | 54 | 49 | 41 | 4E | 20 | 44 |
| 45 | 53 | 4D | 49 | 20 | 57 | 41 | 52 | 55 | 57 | 55 | |

Tabel tersebut bertujuan untuk menampilkan nilai-nilai *hexadecimal* yang dapat dihasilkan secara manual untuk teks yang akan dihitung. Dengan urutan nilai *hexadecimal* disusun berdasarkan frekuensinya, di mana nilai frekuensi terbanyak ditempatkan di urutan teratas.

Tabel 5 Frekuensi nilai *hexadecimal* belum terkompresi

| N | Nilai *Hexadecimal* | ASCII ( Biner ) | Frek | Bit | Frek x Bit |
|---|---|---|---|---|---|
| 1 | 45 | 01000101 | 3 | 8 | 24 |
| 2 | 49 | 01001001 | 3 | 8 | 24 |
| 3 | 4E | 01001110 | 2 | 8 | 16 |
| 4 | 41 | 01000001 | 2 | 8 | 16 |
| 5 | 20 | 00100000 | 2 | 8 | 16 |
| 6 | 57 | 01010111 | 2 | 8 | 16 |
| 7 | 55 | 01010101 | 2 | 8 | 16 |
| 8 | 50 | 01010000 | 1 | 8 | 8 |
| 9 | 4C | 01001100 | 1 | 8 | 8 |
| 10 | 54 | 01010100 | 1 | 8 | 8 |
| 11 | 44 | 01000100 | 1 | 8 | 8 |
| 12 | 53 | 01010011 | 1 | 8 | 8 |
| 13 | 4D | 01001101 | 1 | 8 | 8 |

| | | | | | |
|---|---|---|---|---|---|
| 14 | 52 | 01010010 | 1 | 8 | 8 |
| Total Bit | | | | | 184 |

Setelah diurutkan berdasarkan frekuensi terbanyak, bilangan biner yang akandikompresi pada setiap karakter diganti dengan kode kebenaran dari algoritma *Taboo Code* yang diurutkan berdasarkan frekuensi kemunculan. Yakni jumlah bit yang akan dikompresi merupakan jumlah bit pada *codeword*,dimana bit sebelumnya berdasarkan jumlah bit dari bilangan biner.

Tabel 6 Frekuensi *file* teks sudah terkompresi

| n | Nilai *Hexadecimal* | *Codeword* | Bit | Frek | Bit x Frek |
|---|---|---|---|---|---|
| 1 | 45 | 0100 | 4 | 3 | 12 |
| 2 | 49 | 1000 | 4 | 3 | 12 |
| 3 | 4E | 1100 | 4 | 2 | 8 |
| 4 | 41 | 010100 | 6 | 2 | 12 |
| 5 | 20 | 011000 | 6 | 2 | 12 |
| 6 | 57 | 011100 | 6 | 2 | 12 |
| 7 | 55 | 100100 | 6 | 2 | 12 |
| 8 | 50 | 101000 | 6 | 1 | 6 |
| 9 | 4C | 101100 | 6 | 1 | 6 |
| 10 | 54 | 110100 | 6 | 1 | 6 |
| 11 | 44 | 111000 | 6 | 1 | 6 |
| 12 | 53 | 111100 | 6 | 1 | 6 |
| 13 | 4D | 01010100 | 8 | 1 | 8 |
| 14 | 52 | 01011000 | 8 | 1 | 8 |
| Total Bit | | | | | 126 |

Setelah kode berhasil di *encode* berdasarkan kode kebenaran algoritma *Taboo Code*, langkah selanjutnya adalah mengatur ulang urutan *string* bit yang telah dihasilkan dari proses kompresi sesuai dengan posisi awal nilai *hexadecimal* pada *string* sebelumnya.

Tabel 7 *String* bit sesudah kompresi

| | | | | |
|---|---|---|---|---|
| **50** | **45** | **4E** | **45** | **4C** |

| | | | | |
|---|---|---|---|---|
| 101000 | 0100 | 1100 | 0100 | 101100 |
| **49** | **54** | **49** | **41** | **4E** |
| 1000 | 110100 | 1000 | 010100 | 1100 |
| **20** | **44** | **45** | **53** | **4D** |
| 011000 | 111000 | 0100 | 111100 | 01010100 |
| **49** | **57** | **41** | **52** | **55** |
| 1000 | 011100 | 010100 | 01011000 | 100100 |
| **57** | **55** | **20** | | |
| 011100 | 100100 | 011000 | | |

Setelah mengatur ulang *string* bit, kemudian menyusun kembali *string* bit yang sudah dikompresidapat dilihat pada tabel sebagai berikut.

*Tabel 8. String* bit yang telah dikompresi

| | | | | | |
|---|---|---|---|---|---|
| 101000 | 0100 | 1100 | 0100 | 101100 | 1000 |
| 110100 | 1000 | 010100 | 1100 | 011000 | 111000 |
| 0100 | 111100 | 01010100 | 1000 | 011100 | 010100 |
| 01011000 | 100100 | 011100 | 100100 | 011000 | |

Dari perhitungan sebelumnya, diketahui hasil penjumlahan bit dan frekuensi adalah 126 bit.Dalam konteks komputer, setiap karakter diwakili oleh kode ASCII (*American Standard Code for Information Interchange*)*,* yang terdiri dari 8 bit biner. Jika data memiliki jumlah bit kurang dari 8 bit, biasanya komputer tidak dapat membacanya secara langsung sebelum dilakukan proses enkripsi. Selama proses enkripsi, bit data akan diatur berurutan setiap 8 bit untuk membentuk karakter yang dapat dibaca oleh komputer.

**Perancangan dan Pemodelan Sistem**

Perancangan adalah gambaran dari pembuatan aplikasi pengamanan pesan teks akan membantu pengguna lebih mudah dalam menggunakan algoritma tersebut untuk mengamankan data atau pun informasi penting. Tujuan dari perancangan antarmuka yaitu membuat tampilan sistem yang sederhana dan mudah digunakan

(*userfriendly*) sehingga *user* dapat lebih mudah dalam menggunakan sistem.

**Implementasi Sistem**

Implementasi adalah fase di mana analisis diputuskan, dan data diolah dalam bentuk perangkat lunak sistem (kode sumber). Tujuannya adalah untuk menilai apakah sistem berfungsi sesuai kebutuhan dan beroperasi dengan baik, atau apakah perlu perbaikan. Proses implementasi melibatkan beberapa langkah, termasuk persetujuan aplikasi, instalasi, pengujian data, dan penggunaan sistem yang diperbaiki atau baru. Untuk menjalankan sistem yang dibuat, perangkat lunak dan perangkat keras yang mendukung diperlukan. Implementasi merupakan langkah selanjutnya yang digunakan untuk mengoperasikan sistem yang dirancang. Sistem perancangan merupakan suatu kesatuan pengolahan yang terdiri dari prosedur dan pelaksanaan data.

**Perangkat Keras *(Hardware)***

Spesifikasi perangkat keras *(hardware)* yang digunakan untuk implementasi pembuatan dan menajalankan program adalah sebagai berikut :

1. *Processor* laptop yang digunakan pada penelitian ini adalah menggunakan minimal AMD A9 @ 3,00 GHz.
2. Memory yang digunakan minimal 2 GB
3. *Hard Disk* yang digunakan sebagai media penyimpanan minimal 50 GB

**Perangkat Lunak *(Software)***

Perangkat lunak *(software)* yang digunakan didalam perancangan aplikasi ini adalah sebagai berikut:

1. Sistem *windows 7 Ultimate*
   Sistem *Windows* 7 *Ultimate*" merupakan sebuah versi dari sistem operasi Windows yang dikenal sebagai *Windows* 7 *Ultimate*. *Windows* 7 *Ultimate* adalah edisi tertinggi dari Windows 7 yang diluncurkan oleh Microsoft. Ini adalah sistem operasi komputer yang memiliki fitur-fitur tambahan dan lebih banyak opsi dibandingkan dengan edisi Windows 7 lainnya, seperti *Windows* 7 *Home Premium* atau *Windows* 7 Professional.
2. *Microsoft Visual Studio 2010*
   Microsoft visual studio Versi ini juga memiliki fitur pemecahan masalah yang memungkinkan pengembang untuk melacak dan memperbaiki kesalahan dalam

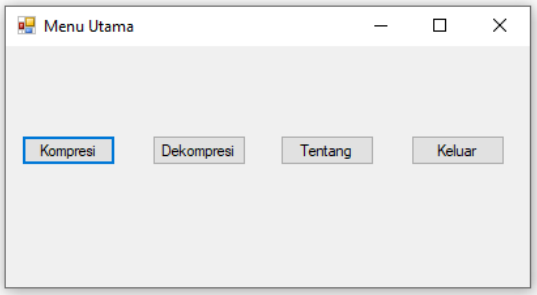
kode mereka dengan lebih efisien. Selain itu, Visual Studio 2010 mendukung pengembangan aplikasi berbasis database dan integrasi dengan teknologi seperti SQL Server.

**Pengujian Sistem**

Berikut ini merupakan tampilan hasil dari pengujian perbandingan Algoritma *golomb* dan Algoritma *taboo code* yang dapat dilihat sebagai berikut:

1. Tampilan Halaman Menu Utama

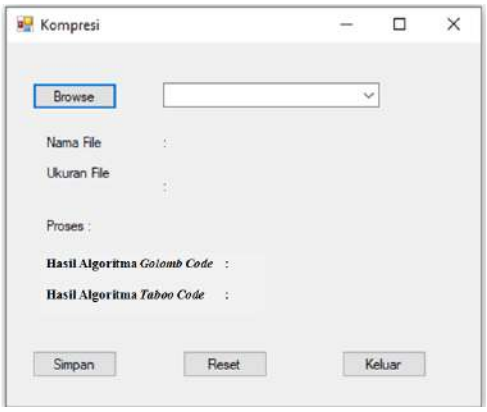
   Tampilan menu utama adalah program yang menyajikan berbagai pilihan menu, di mana pengguna harus memilih menu yang ingin diakses.




Gambar 2.Tampilan Halaman Menu Utama

2. Tampilan Halaman Kompresi

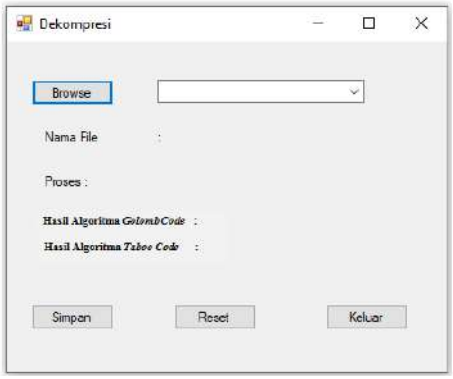
   Pada tampilan kompresi, pengguna dapat menjalankan proses kompresi dengan menginput atau memilih *file* yang ingin dikompresi. Tampilan halaman menu kompresi dapat dilihat pada gambar berikut.




Gambar 3. Tampilan Halaman Kompresi

3. Tampilan *Form* Dekompresi

   Tampilan dekompresi digunakan untuk mengembalikan *file* yang sudah dikompresi, di mana pengguna harus terlebih dahulu mencari *file* yang telah dikompresi. Tampilan halaman menu dekompresi dapat dilihat pada gambar berikut.




Gambar 4. Tampilan Halaman Dekompresi

4. Tampilan Halaman Tentang

   Tampilan "Tentang" akan menampilkan informasi mengenai pembuat program. Tampilan

"Tentang" ini dapat dilihat pada gambar berikut.


Gambar 5 Tampilan Halaman Tentang

## D. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan perbandingan algoritma untuk mengkompresi file MP4, dapat disimpulkan bahwa: 1) *Algoritma Golomb Code* lebih efisien dibandingkan *Taboo Code* dalam kompresi file MP4 karena *Golomb Code* lebih cocok untuk data dengan distribusi panjang kode yang tidak merata, yang sering ditemukan dalam file MP4; 2) Kedua algoritma dapat memulihkan data dengan kualitas yang sama setelah dekompresi tanpa kehilangan informasi, tetapi *Golomb Code* umumnya menghasilkan ukuran file yang lebih kecil setelah kompresi tanpa mengurangi kualitas data; 3) *Algoritma Golomb Code* juga lebih mudah diimplementasikan karena memiliki aturan dan struktur yang lebih sederhana dibandingkan dengan *Taboo Code.*

## DAFTAR PUSTAKA

Andriyani. (2024). Data Sebagai Fondasi Kecerdasan Buatan. In *TOHAR MEDIA*.

Bant, M., Data, L., Compression, I. I., Encoding, B., Encoding, R. B., Reduction, M. B., & Climbing, H. (2018). Terim- Doküman Matrisleri için Sıralamaya Dayalı Bir Kayıpsız Sıkıştırma Şeması Reordering Based Lossless Compression Scheme for Term-Document Matrices. *TÜRKİYE BİLİŞİM VAKFI BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ DERGİSİ*, *5*(1), 30–40.

Darnita, Y. (2019). Kompresi Data Teks Dengan Menggunakan Algoritma Sequitur. *Jurnal SISTEMASI*, *8*(1), 104–113.

Gulo, E. K. (2017). Perancangan Aplikasi Kompresi Audio dengan Menerapkan Algoritma Golomb. *Pelita Informatika Budi Darma*, *16*(1), 436–439.

Hairani, L., & Butar, B. (2021). Penerapan Algoritma Golomb Coding Pada Aplikasi Kompresi Short Message Service ( SMS ). *Journal of Informatics Management and Information Technology*, *1*(4), 159–165.

Imani, M. L., Muhima, R. R., Agustini, S., Informatika, T., Teknologi, I., & Tama, A. (2021). Penerapan Metode Huffman dalam Kompresi Data. *Seminar Nasional Sains Dan Teknologi Terapan IX*, *1*(1), 457–462.

Iqbal, D. (2019). IMPLEMENTASI ALGORITMA LEVENSTEIN UNTUK KOMPRESI FILE VIDEO. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *3*(1), 266–273.

NST, A. M. (2019). Analisa Perbandingan Algoritma Half Byte Dan Algoritma Deflate Pada File Dengan Metode Independent Sample T-Test. *Informasi Dan Teknologi Ilmiah (INTI)*, *6*(2), 164–170.

Nuryasin. (2014). PERBANDINGAN KOMPRESI FILE DATA DENGAN ALGORITMA HUFFMAN , HALF BYTE DAN RUN LENGTH. *Studi Informatika: Jurnal Sistem Informasi*, *7*(2), 1–6.

Setiawan. (2023). PENDIDIKAN MULTIMEDIA: Konsep dan Aplikasi pada era revolusi industri 4.0 menuju society 5.0. In *PT. Sonpedia Publishing Indonesia*.

Simanjuntak, S. (2022). Implementasi Metode Taboo Code Untuk Kompresi File Video. *Explorer*, *2*(1), 32–38. https://doi.org/10.47065/explorer.v2i1.156