KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)
Volume 6, Nomor 1, November 2022
DOI: 10.30865/komik.v6i1.5728
ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Analisa Perbandingan Algoritma Taboo Codes Dan Algoritma Yamamoto's Recursive Code Untuk Kompresi File Teks Menggunakan Metode Exponential

Arianti Rhamadani

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: [1]arianti612@gmail.com

Abstrak-Di era digital saat ini penggunaan teknologi kompresi data sangat penting disebabkan pemanfaatan data dalam bentuk elektronik lebih cepat untuk melakukan pertukaran informasi. Ukuran *file* yang besar membutuhkan banyak ruang penyimpanan. *File* teks berekstensi rich text format (rtf) memiliki ukuran yang besar dibanding dengan jenis *file* teks yang berekstensi lain. Dikarenakan ukuran *file* berekstensi .rtf terlalu besar mengakibatkan proses pengiriman data memerlukan waktu yang cukup lama. Salah satu teknik yang diperlukan untuk masalah ini adalah melakukan kompresi. Kompresi adalah proses pengubahan ukuran data ke ukuran yang lebih kecil. Algoritma yang digunakan pada penelitian ini adalah algoritma *Taboo Codes* dan *Yamamoto's Recursive Code* yang memiliki kelebihan dan kekurangan masing-masing. Setelah kedua algoritma dikompres, kemudian dilakukan perbandingan dari kedua algoritma tersebut menggunakan metode Exponential. Metode perbandingan exponential merupakan salah satu metode yang digunakan untuk menentukan urutan prioritas alternatif keputusan dengan beberapa kriteria. Hasil akhir dari proses perbandingan dengan metode exponential didapatkan hasil 7,203 yang merupakan algoritma *Yamamoto's Recursive Code*. Sehingga algoritma tersebut menjadi algoritma tercepat dan efektif dalam melakukan proses kompresi *file* teks, hal ini dikarenakan semakin kecil total nilai yang diperoleh maka semakin sedikit jumlah usaha yang dilakukan oleh algoritma tersebut dalam melakukan kompresi.

Kata Kunci: Kompresi; Rtf; Taboo Codes; *Yamamoto's Recursive Code*; Exponential

Abstract-In the current digital era, the use of data compression technology is very important because the use of data in electronic form is faster to exchange information. Large *file* sizes require a lot of storage space. Rich text format (rtf) text *file*s have a large size compared to other types of text *file*s with extensions. Because the *file* size with the extension .rtf is too large, the data transmission process takes a long time. One of the techniques required for this problem is to perform compression. Compression is the process of resizing data to a smaller size. The algorithms used in this study are the *Taboo Codes* and *Yamamoto's Recursive Code* algorithms, each of which has advantages and disadvantages. After the two algorithms are compressed, then a comparison of the two algorithms is carried out using the Exponential method. The exponential comparison method is one of the methods used to determine the priority order of decision alternatives with several criteria. The final result of the comparison process with the exponential method is 7,203, which is *Yamamoto's Recursive Code* algorithm. So that the algorithm becomes the fastest and most effective algorithm in compressing text *file*s, this is because the smaller the total value obtained, the less effort is made by the algorithm in compression.

Keywords: Compression; Rtf; Taboo Codes; *Yamamoto's Recursive Code;* Exponential

1. PENDAHULUAN

Di era digital saat ini penggunaan teknologi kompresi data sangat penting disebabkan pemanfaatan data dalam bentuk elektronik lebih cepat untuk melakukan pertukaran informasi. Peningkatan dalam penggunaan data telah menimbulkan masalah penyimpanan dan secara tidak langsung meningkatkan kebutuhan pada penyimpanan data. Ukuran *file* yang lebih besar membutuhkan lebih banyak ruang penyimpanan.

Dibanding dengan jenis format *file* teks yang lain, rich text format (rtf) memiliki ukuran yang sangat besar. Terkadang ada risiko *file* teks tidak dapat ditampung di media penyimpanan dan dikarenakan ukuran *file* yang berekstensi rich text format (rtf) yang terlalu besar mengakibatkan proses pengiriman data memerlukan waktu yang cukup lama. Solusi yang diperlukan untuk masalah ini adalah melakukan proses kompresi. Melakukan kompresi data pada saat ini sangat menguntungkan ketika memiliki data yang sangat besar dan data di dalamnya mengandung banyak karakter berulang sehingga dalam melakukan penyimpanan akan lebih efisien. Faktor utama yang terjadi karena pada proses kompresi dapat dipengaruhi oleh algoritma kompresi yang digunakan untuk mengurangi jumlah bit dari data asli. Algoritma yang digunakan pada penelitian ini antara lain algoritma *Taboo Codes* dan *Yamamoto's Recursive Code*.

Algoritma *Taboo Codes* dan *Yamamoto's Recursive Code* adalah jenis lossless compression di mana data yang terkompresi dapat dikembalikan ke data aslinya serta membutuhkan lebih sedikit ruang penyimpanan daripada *file* teks yang tidak terkompresi. Namun, algoritma *Taboo Codes* dan *Yamamoto's Recursive Code* memiliki kelebihan dan kekurangannya masing-masing. Setelah kedua algoritma dikompres, kemudian dilakukan perbandingan dari kedua algoritma tersebut menggunakan metode Exponential. Metode perbandingan exponential merupakan salah satu metode yang digunakan untuk menentukan urutan prioritas alternatif keputusan dengan beberapa kriteria. Dengan perhitungan exponential, perbedaan nilai antar kriteria dapat dibedakan berdasarkan kemampuan seseorang dalam menilai[1]. Dengan membandingkan algoritma *Taboo Codes* dan *Yamamoto's Recursive Code*, akan menghasilkan algoritma kompresi yang paling efisien sehingga bisa menentukan algoritma yang akan digunakan ketika akan mengompresi sebuah *file* teks.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Berdasarkan penelitian Muhammad Asnawi Latif dkk pada tahun 2018 dapat disimpulkan dari hasil yang diperoleh dengan menggunakan metode Exponential untuk melakukan perbandingan Algoritma Rice Codes dengan Algoritma Goldbach Codes untuk mengompresi *file* teks tidak jauh berbeda. Dengan menerapkan metode perbandingan Exponential dengan 3 kriteria, hasil kompresi yang terbaik adalah algoritma Rice Codes dengan hasil ration of compression 2,07% dan space saving 52%[2].

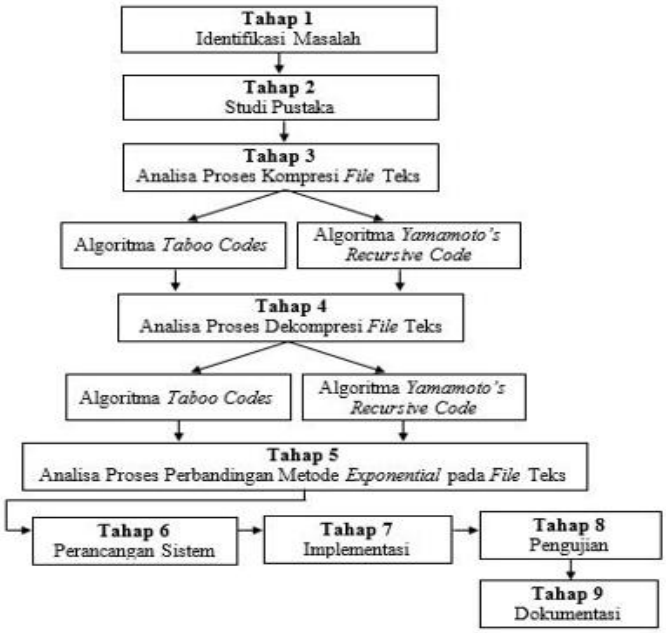
Berdasarkan penelitian Nur Aftikasyah pada tahun 2021 dapat disimpulkan bahwa Algoritma *Yamamoto's Recursive Code* telah melalui proses implementasi dan pembuktian bahwa *file* dokumen yang berukuran besar bisa dikompresi menjadi ukuran yang lebih kecil[3].

Berdasarkan penelitian Saiful Simanjuntak pada tahun 2021 dapat disimpulkan dengan menggunakan Algoritma *Taboo Codes* untuk mengompresi *file* video bahwa hasil kompresi yang digunakan dengan parameter analisis kompresi rasio dan penghematan ruang dengan rasio kompresi 58,33% dan hasil space saving 0,1467% hasil penghematan ruang menunjukkan tingkat hasil kompresi yang sangat baik dalam sebuah *file* video, dan sudah baik untuk diterapkan[4].

Mengetahui tentang kinerja kompresi *file* teks dalam beberapa penelitian sebelumnya, yang mengarah pada kemampuan untuk meningkatkan transfer data, mengurangi pemanfaatan ruang penyimpanan dengan mengurangi byte data, dan mempercepat penyimpanan dan transfer data gambar. Oleh karena itu dengan adanya penelitian terdahulu sangat membantu dalam mempelajari pengkompresian *file* teks.

2. METODOLOGI PENELITIAN

2.1 Kerangka Penelitian



Gambar 1. Kerangka Penelitian

Berdasarkan kerangka penelitian di atas, masing-masing tahapan dijelaskan sebagai berikut:

a. Tahap Identifikasi Masalah
   Langkah ini adalah cara penulis untuk memprediksi, mengevaluasi, dan menjelaskan masalah kapasitas penyimpanan secara keseluruhan dan apa yang menyebabkan *file* teks .rtf mentransfer cukup lambat sehingga tidak lagi muat di ruang penyimpanan.
b. Tahap Studi Pustaka
   Pada langkah ini melakukan pemahaman terhadap pokok bahasan dilakukan dengan membaca berbagai bahan referensi seperti buku, artikel, jurnal dan bahan bacaan lainnya.
c. Tahap Analisa Proses Kompresi Pada *File* Teks
   Langkah analisa ini dilakukan untuk mengetahui proses kompresi *file* teks menggunakan algoritma *Taboo Codes* dan Algoritma *Yamamoto's Recursive Code*, yang bertujuan untuk memperkecil ukuran data.
d. Tahap Analisa Proses Dekompresi Pada *File* Teks
   Proses mengembalikan *file* teks terkompresi ke *file* asli sebelum dikompresi menggunakan algoritma *Taboo Codes* dan Algoritma *Yamamoto's Recursive Code*.
e. Tahap Analisa Proses Perbandingan Pada *File* Teks

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Langkah analisa ini dilakukan untuk mengetahui proses perbandingan *file* teks menggunakan metode perbandingan exponential yang bertujuan untuk mengetahui algoritma mana yang lebih efektif dan efisien yang akan digunakan untuk mengompresi *file* teks.

f. Tahap Implementasi
Langkah yang dilakukan dalam proses pengembangan/pengkodean perangkat lunak, yang tujuannya untuk menentukan apakah suatu sistem dapat berfungsi dengan baik sesuai kebutuhan atau perlukah perbaikan sistem.

g. Tahap Pengujian
Langkah ini dilakukan untuk mengecek hasil kompresi *file* teks dalam format .rtf.

h. Tahap Dokumentasi
Tahap dokumentasi merupakan tahap akhir dari suatu penelitian yang ditulis dalam bentuk laporan.

2.2 Kompresi

Kompresi adalah proses pengubahan ukuran data ke ukuran yang lebih kecil dari data asli tanpa kehilangan maksud dan tujuan dari data asli. Kompresi *file* teks dimaksudkan untuk memperkecil ukuran *file* teks, sehingga memudahkan dalam mentransfer data dan menyimpan *file* teks. Kompresi *file* teks dilakukan dengan mengompresi isi data teks[4][5].

Ada beberapa parameter yang umum digunakan untuk mengetahui kinerja metode kompresi yang bisa digunakan untuk menentukan kualitas data yang akan dikompres adalah sebagai berikut[5]:

a. *Ratio of Compression* (RC)
Ratio of Compression (RC) yaitu proses perbandingan antara ukuran bit data sebelum dikompres dan ukuran bit data yang telah dikompres. Formula yang digunakan dapat dituliskan sebagai berikut:

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Sesudah\ Dikompresi} \quad (1)$$

b. *Compression Ratio* (CR)
Compression Ratio (CR) yaitu proses menghitung persentase dari perbandingan antara data yang sudah dikompres dengan data yang belum dikompres. Formula yang digunakan dapat dituliskan sebagai berikut:

$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\% \quad (2)$$

c. *Redudancy (Rd)*
Redudancy yaitu proses menghitung perbedaan nilai persentase antara ukuran data sebelum kompres dan ukuran data setelah dikompres. Formula yang digunakan dapat dituliskan sebagai berikut:

$$Rd = \frac{file\ sebelum\ dikompresi - file\ setelah\ dikompresi}{Ukuran\ file\ sebelum\ dikompresi}\ x\ 100\% \quad (3)$$

d. *Space Saving* (Ss)
Space Saving (Ss) yaitu proses menghituung perbedaan nilai persentase antara data yang belum terkompres dan yang sudah terkompres.

$$S_s = 100\% - C_R \quad (4)$$

2.3 *File* Teks

File teks adalah *file* yang berisi data dalam bentuk teks[6][7]. Data yang diperoleh berasal dari dokumen pengolah kata, bilangan angka yang digunakan untuk melakukan perhitungan, serta menginput nama dan alamat dalam database merupakan contoh input data teks yang terdiri dari karakter, angka dan huruf. Fungsi *file* teks yaitu untuk menyimpan data atau informasi yang dapat digunakan dan dikelola oleh *user*. Jika semakin banyak *file* teks yang akan disimpan, maka semakin banyak pula ruang yang akan digunakan. Maka daripada itu diperlukan suatu metode untuk mengecilkan ukuran *file* teks supaya lebih sedikit menggunakan ruang penyimpanan yang biasa disebut dengan kompresi[8].

Jenis kompresi yang digunakan untuk *file* teks pada penelitian ini adalah *lossless compression*, yaitu *file* yang sudah dikompres bisa dikembalikan lagi keukuran awal. *File* teks yang dikompresi adalah *file* teks yang memiliki ekstensi .rtf.

2.4 Algoritma *Taboo Codes*

Algoritma *Taboo Codes* mengambil pendekatan terhadap panjangnya variabel. Algoritma *Taboo Codes* adalah ide orisinal oleh Steven Pigeon. Prinsip dari algoritma *Taboo Codes* adalah memilih bilangan positif n dan membalikkan pola n untuk mewakili akhir kode. Ada dua tipe dalam Algoritma *Taboo Codes* diantaranya[4]:

a. Jenis pertama dari algoritma *Taboo Codes* adalah berbasis blok dan panjangnya adalah kelipatan n. Algoritma integer *Taboo Codes* berbasis blok adalah urutan blok n-bit, nilai n dipilih oleh pengguna, dan blok terakhir

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

memiliki pola tab yang tidak terjadi di blok lain. Nilai blok n-bit adalah 2. Jika satu dicadangkan, setiap blok cipher yang tersisa dapat memiliki salah satu blok ini dengan pola biner 2n-1.

b. Jenis kedua dari algoritma *Taboo Codes* adalah panjang keseluruhan berganda dan tidak terbatas. Tipe ini disebut tidak terbatas dan telah terbukti berhubungan dengan bilangan Fibonacci ke-n.

Tabel 1. Pola *Taboo Codes*

M	Code	M	Code	m	Code	M	Code
0	01 00	4	01 10 00	8	10 11 00	12	01 01 01 00
1	10 00	5	01 11 00	9	11 01 00	13	01 01 10 00
2	11 00	6	10 01 00	10	11 10 00	14	01 01 11 00
3	01 01 00	7	10 10 00	11	11 11 00	…	

Sumber: David Salomon, Giovanni Motta, 2010 [9]

2.5 Algoritma *Yamamoto's Recursive Code*

Yamamoto's Recursive Code adalah kode pengulangan untuk beberapa angka bulat positif di mana setiap tahap digunakan sebagai pembatas, tidak seperti kode universal lainnya menggunakan bit "0" sebagai pembatas seperti Elias Omega Code, Even-Rodeh Code, Stout Code, dll. Pembatas yang digunakan dalam algoritma *Yamamoto's Recursive Code* lebih pendek dari $log2$ * n dari semua angka bulat positif dibandingkan algoritma tersebut[3][5]. Langkah yang dilakukan untuk membentuk tabel kebenaran *Yamamoto's Recursive Code* mana nilai heksadesimal dari *file* teks akan dikompres adalah sebagai berikut:

a. Tentukan pembatas f-bit yang sederhana (di mana f tersebut bilangan integer positif), misalnya f = 2 dan pembatas 00.
b. Pada tabel Ba, f (n), dengan melakukan proses mengurutkan angka biner yang tidak diawali dengan pembatas, dalam hal ini diambil 00 sebagai contoh adalah 00.
c. Setelah diperoleh table Ba, f (n) kemudian definisikan table $B\bar{a}$, f (n) yaitu proses menghilangkan semua nilai pada Ba, f(n) yang bila digabungkan akan membentuk pemisah, dalam hal ini diperoleh B00, 2 (1) = 0.

Tabel 2. *Yamamoto's Recursive Codewords*

N	a= 00	a= 100
1	1 00	0 100
2	1 01 00	0 00 100
3	1 10 00	0 01 100
4	1 11 00	0 11 100
5	1 01 010 00	0 00 000 100
6	1 01 011 00	0 00 001 100
7	1 01 100 00	0 00 010 100
8	1 01 101 00	0 00 011 100
9	1 01 110 00	0 00 101 100
10	1 01 111 00	0 00 110 100
11	1 10 0100 00	0 00 111 100
12	1 10 0101 00	0 01 0000 100
13	1 10 0110 00	0 01 0001 100
14	1 10 0111 00	0 01 0010 100
15	1 10 1000 00	0 01 0011 100
16	1 10 1001 00	0 01 0100 100
17	1 10 1010 00	0 01 0101 100
18	1 10 1011 00	0 01 0110 100
19	1 10 1100 00	0 01 0111 100
20	1 10 1101 00	0 01 1010 100
21	1 10 1110 00	0 10 1011 100
22	1 10 1111 00	0 01 1100 100
23	1 11 0100 00	0 01 1101 100
24	1 11 0100 00	0 01 1110 100
25	1 11 01011 00	0 01 1111 100
26	1 11 01100 00	011 00000 100

Sumber: David Salomon, Giovanni Motta, 2010 [9]

2.6 Metode *Exponential*

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Metode perbandingan Exponential adalah metode untuk melakukan penentuan prioritas alternatif keputusan dengan berdasarkan beberapa kriteria. Dengan perhitungan exponential, perbedaan nilai antar kriteria dapat dibedakan menurut kemampuan seseorang dalam menilai[10][11][12].

$$TotalNilai(TN_i) = \sum_{j=1}^{m}(V_{ij})B_j \quad (5)$$

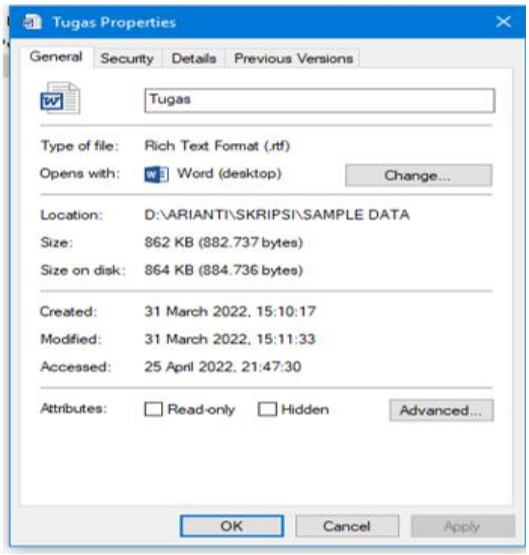
Keterangan:
Tni = Total Nilai Alternatif Ke-
Vij = Derajat kepentingan relatif kriteria ke-j pada keputusan ke-i, yang dapat dinyatakan dengan skala ordinal (1,2,3,4,5)
Bj = Derajat kepentingan kriteria keputusan, yang dinyatakan dengan bobot
m = Jumlah kriteria keputusan
n = Jumlah pilihan keputusan
j = 1, 2, 3, ...; m = jumlah kriteria
i = 1, 2, 3, ...; n = jumlah pilihan alternatif

3. HASIL DAN PEMBAHASAN

3.1 Analisa

Analisa masalah yang diselesaikan dalam penelitian ini adalah kinerja antar algoritma Taboo Codes dan algoritma Yamamoto's Recursive Code saat mengompresi file teks serta membandingkan kinerja dua algoritma berdasarkan parameter Ratio of Compression (RC), Compression Ratio (CR), Redudancy (RD), Space Saving (SS). Kompresi data menerapkan algoritma Taboo Codes dan algoritma Yamamoto's Recursive Code pada file teks berformat .rtf berukuan besar mengakibatkan proses pengiriman data memerlukan waktu yang cukup lama. Kedua algoritma adalah teknik kompresi lossless di mana data yang telah terkompresi dapat dikembalikan ke data asli, dengan tujuan mengetahui algoritma yang lebih efisien dan efektif yang digunakan untuk mengkompresi file teks.

Proses awal yang dilakukan penulis adalah menentukan *file* teks yang akan dikompres kemudian melakukan konversi ke nilai heksadesimal melalui aplikasi HxD. Setelah nilai *file* teks Hexadecimal didapat maka akan dilakukan proses kompresi dengan menggunakan algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code* dan diperoleh *file* hasil kompresi. Setelah hasil kompresi didapat maka dilakukan proses dekompresi dari algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code*. Dan dari hasil kompresi dilakukan proses perbandingan dengan menggunakan metode exponential dari *file* teks tersebut menggunakan parameter yang telah ditentukan.



Gambar 2. Sample *File* Teks

3.1.1 Penerapan Algoritma *Taboo Codes*

Berdasarkan sampel nilai hexadecimal, 20 sampel nilai hexadecimal akan diambil untuk menghitung file teks, seperti yang ditunjukkan pada tabel berikut:

Tabel 3. Nilai Bilangan Hexadecimal

31	32	35	32	5C	75	63	31	5C	61
64	65	66	66	30	5C	64	65	66	66

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Tabel di atas berguna untuk menghasilkan angka hexadecimal untuk *file* .rtf yang akan dihitung secara manual. Berikut nilai hexadecimal yang diurutkan berdasarkan frekuensi, nilai frekuensi tertinggi akan berada di tempat pertama.

Tabel 4. Nilai Hexadecimal Tidak Terkompresi

n	Nilai *Hexadecimal*	Nilai Biner	Bit	Freq	Bit x Freq
1	66	01100110	8	4	32
2	5C	01011100	8	3	24
3	65	01100101	8	2	16
4	64	00110100	8	2	16
5	31	00110001	8	2	16
6	32	00110010	8	2	16
7	35	00110101	8	1	8
8	75	01110101	8	1	8
9	63	01100011	8	1	8
10	61	01100001	8	1	8
11	30	00110000	8	1	8
Total Bit					160

Berdasarkan kemunculan dan nilai biner telah diperoleh, langkah selanjutnya adalah menghitung nilai bit dan melakukan pengurutan kode *Taboo Codes* dan mendapatkan *file* bit yang telah terkompresi. Proses kompresi *file* teks dapat dilihat pada tabel 5 dibawah ini:

Tabel 5. Tabel Karakter Algoritma *Taboo Codes*

n	Nilai *Hexadecimal*	Nilai Biner	Bit	Freq	Bit x Freq
0	66	0100	4	4	16
1	5C	1000	4	3	12
2	65	1100	4	2	8
3	64	010100	6	2	12
4	31	011000	6	2	12
5	32	011100	6	2	12
6	35	100100	6	1	6
7	75	101000	6	1	6
8	63	101100	6	1	6
9	61	110100	6	1	6
10	30	111000	6	1	6
Total Bit					102

Langkah selanjutnya adalah mengatur ulang bit string yang dihasilkan dari kompresi berdasarkan posisi dalam bilangan hexadecimal.

Tabel 6. String Bit Yang Telah Dikompresi Dengan Algoritma *Taboo Codes*

011000	011100	100100	011100	1000
101000	101100	011000	1000	110100
010100	1100	0100	0100	111000
1000	010100	1100	0100	0100

Dikarenakan 102 tidak habis dibagi 8 atau bukan kelipatan 8 maka dibentuk variabel yang disebut dengan padding dan flagging untuk penambahan bit data. Rumus padding ialah 7-n+"1" dan Rumus Flagging ialah 9-n.

Tabel 7. Penambahan *Padding* dan *Flagging* Algoritma *Taboo Codes*

Padding	*Flagging*
$102\ mod\ 8 = 6 = n$ $7 - 6 + "1"$ $7 - 6 + "1" = 01$	$9 - n$ $9 - 6 = 3 = 00000011$

Jadi bit string yang terbentuk setelah menambahkan padding dan flagging adalah: "01100001110010010001110010001010001011000110001000110100010100110001000100111000100000101001100 010001000100000011". Oleh karena itu, panjang total bit adalah 112. Kemudian dilanjutkan untuk membagi bit menjadi beberapa kelompok. Setiap grup terdiri dari 8 bit.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Tabel 8. Pengelompokan Bit Algoritma *Taboo Codes*

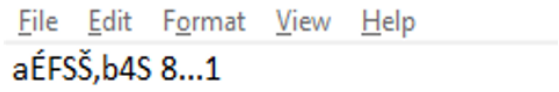
01100001	11001001	00011100	00101100
01100010	00110100	01010011	00111000
10000101	00110001	00010001	

Berdasarkan hasil pemecahan bit di atas, terbentuk 14 kelompok dengan nilai biner terkompresi baru serta nilai biner tambahan. Kemudian langkah selanjutnya adalah mengubah nilai biner ke nilai hexadecimal untuk menemukan karakter yang dihasilkan yang cocok dengan kode ASCII. Nilai hexadecimal terkompresi dapat dilihat di bawah ini.

Tabel 9. Hasil Karakter Terkompresi Algoritma *Taboo Codes*

No	*Codeword* Hasil Kompresi	*Decimal*	*Hexa Decimal*	Karakter	Keterangan
1	01100001	97	61	a	Huruf latin a kecil
2	11001001	201	C9	É	Latin capital letter E with acute
3	00011100	28	1C	FS	*File* separator
4	10001010	138	8A	Š	Latin capital letter S with caron
5	00101100	44	2C	,	Karakter koma
6	01100010	98	62	b	Huruf latin b kecil
7	00110100	52	34	4	Angka empat
8	01010011	83	53	S	Huruf latin S kapital
9	00010001	17	11		Device control 1 (tidak terlihat)
10	00111000	56	38	8	Angka delapan
11	10000101	133	85	…	Horizontal ellipsis
12	00110001	49	31	1	Angka satu
13	00010001	17	11		Device control 1 (tidak terlihat)
14	00000011	3	3		End of text (tidak terlihat)

Jenis *file* akan diubah menjadi tipe baru sesuai dengan format yang ditentukan sebagai pengenal *file* yang telah dikompresi format *file* telah ditentukan sebagai (.tc) dan jika *file* dibuka dengan aplikasi notepad, karakter akan muncul seperti gambar di bawah ini:



Gambar 3. Karakter Terkompresi *Taboo Codes*

Untuk mengetahui tingkat kinerja algoritma kompresi berikut parameter yang digunakan untuk mengukur kinerja Algoritma *Taboo Codes*:

a. *Ratio of Compression* (RC)

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$

$$RC = \frac{160}{102} = 1{,}569$$

b. *Compression Ratio (CR)*

$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$

$$CR = \frac{102}{160} \times 100\% = 63{,}75\%$$

c. *Redudancy (Rd)*

$$Rd = \frac{160 - 102}{160}\ x\ 100\% = 36{,}25\%$$

d. *Space Saving (Ss)*

$$S_s = 100\% - C_R$$

$$S_s = 100\% - 63{,}75\% = 36{,}25\%$$

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Dari perhitungan di atas, dapat disimpulkan bahwa rasio kompresi *file* teks dengan algoritma *Taboo Codes* adalah 36,25%.

3.1.2 Penerapan Algoritma *Yamamoto's Recursive Code*

Berdasarkan sampel nilai hexadecimal, 20 sampel nilai hexadecimal akan diambil untuk menghitung file teks, seperti yang ditunjukkan pada tabel berikut:

Tabel 10. Nilai Bilangan Hexadecimal

31	32	35	32	5C	75	63	31	5C	61
64	65	66	66	30	5C	64	65	66	66

Berdasarkan kemunculan dan nilai biner diperoleh, langkah selanjutnya adalah menghitung bit dan mengurutkan kode *Yamamoto's Recursive Code* dan mendapatkan *file* bit terkompresi. Proses kompresi *file* teks dapat dilihat pada Tabel 12 di bawah ini.

Tabel 11. Tabel Karakter Algoritma *Yamamoto's Recursive Code*

N	Nilai *Hexadecimal*	Nilai Biner	Bit	Freq	Bit x Freq
1	66	100	3	4	12
2	5C	10100	5	3	15
3	65	11000	5	2	10
4	64	11100	5	2	10
5	31	10101000	8	2	16
6	32	10101100	8	2	16
7	35	10110000	8	1	8
8	75	10110100	8	1	8
9	63	10111000	8	1	8
10	61	10111100	8	1	8
11	30	110010000	9	1	9
Total Bit					120

Langkah selanjutnya adalah mengatur ulang bit string yang dihasilkan dari kompresi berdasarkan posisi dalam bilangan hexadecimal.

Tabel 12. String Bit Hasil Kompresi Algoritma *Yamamoto's Recursive Code*

10101000	10101100	10110000	10101100	10100
10110100	10111000	10101000	10100	10111100
11100	11000	100	100	110010000
10100	11100	11000	100	100

Dari hasil jumlah bit x frekuensi didapatkan nilai 120, 120 habis dibagi 8, sehingga tidak perlu menambah jumlah bit. Jika jumlah bit habis dibagi 8 atau kelipatan 8, maka tidak perlu padding, tetapi perlu menambahkan flagging.

Tabel 13. *Flagging* Algoritma *Yamamoto's Recirsive Code*

Flagging
$9-n$
$9-0=9=00001001$

Oleh karena itu, urutan bit terbentuk setelah menambahkan flagging yaitu: "10101000101011001011000010101100101001011010010111000101010001010010111100111001100010010011001000010100111001100010010000001001" Panjang total bit adalah 128. Kemudian bagi bit menjadi beberapa kelompok. Setiap grup terdiri dari 8 bit.

Tabel 14. Pengelompokan Bit Algoritma *Yamamoto's Recursive Code*

10101000	10101100	10110000	10101100
10100101	10100101	11000101	01000101
00101111	00111001	10001001	00110010
00010100	11100110	00100100	00001001

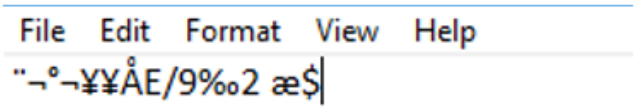
Berdasarkan hasil pemisahan bit di atas, 16 grup terbentuk dengan nilai biner terkompresi baru dan nilai biner ditambahkan. Kemudian ubah nilai biner ke nilai hexadecimal untuk menemukan karakter yang dihasilkan yang cocok dengan kode ASCII. Nilai hexadesimal terkompresi ditunjukkan di bawah ini.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

Tabel 15. Karakter Terkompresi Algoritma *Yamamoto's Recursive Code*

No	*Codeword* Hasil Kompresi	*Decimal*	*Hexadecimal*	Karakter	Keterangan
1	10101000	168	A8	¨	Spacing diaeresis – umlaut
2	10101100	172	AC	¬	Not sign
3	10110000	176	B0	°	Tanda derajat
4	10101100	172	AC	¬	Not sign
5	10100101	165	A5	¥	Tanda Yen
6	10100101	165	A5	¥	Tanda Yen
7	11000101	197	C5	Å	Latin capital letter A with ring above
8	01000101	69	45	E	Huruf latin E kapital
9	00101111	47	2F	/	Garis miring (slash)
10	00111001	57	39	9	Angka sembilan
11	10001001	137	89	‰	Per mille sign
12	00110010	50	32	2	Angka dua
13	00010100	20	14		Device control 4 (tidak terlihat)
14	11100110	230	E6	æ	Latin small letter ae
15	00100100	36	24	$	Tanda mata uang dolar
16	00001001	9	9		Horizontal tabulation

Jenis *file* akan diubah menjadi tipe baru sesuai dengan format yang ditentukan sebagai pengenal *file* yang telah dikompresi format *file* telah ditentukan sebagai (.yrc) dan jika *file* dibuka dengan aplikasi notepad, karakter akan muncul seperti gambar di bawah ini:



Gambar 4. Karakter Hasil Kompresi *Yamamoto's Recursive Code*

Untuk mengetahui tingkat kinerja algoritma kompresi berikut parameter yang digunakan untuk mengukur kinerja Algoritma *Taboo Codes*:

a. *Ratio of Compression* (RC)

$$RC = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$

$$RC = \frac{160}{120} = 1{,}333$$

b. *Compression Ratio (CR)*

$$CR = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi} \times 100\%$$

$$CR = \frac{120}{160} \times 100\% = 75\%$$

c. *Redudancy (Rd)*

$$Rd = \frac{160 - 120}{160}\ x\ 100\% = 25\%$$

d. *Space Saving (Ss)*

$$S_s = 100\% - C_R$$

$$S_s = 100\% - 63{,}75\% = 36{,}25\%$$

Dari perhitungan di atas, dapat disimpulkan bahwa rasio kompresi *file* teks dengan algoritma *Yamamoto's Recursive Code* adalah 25%.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

3.1.3 Metode Perbandingan Eksponential

Dalam metode perbandingan eksponential untuk menghitung dan membandingkan proses pencarian dari kedua algoritma tersebut adalah sebagai berikut:

a. Menentukan alternatif, untuk menganalisa perbandingan kecepatan antara algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code* saat melakukan proses kompresi, maka perlu ditentukan algoritma mana yang harus digunakan sebagai algoritma kompresi.
b. Menentukan kriteria, adapun kriteria tersebut ialah *Ratio of Compression* (RC), *Compression Ratio* (CR), *Redudancy* (Rd), *Space Saving* (*Ss*).
c. Menentukan bobot kriteria, penentuan bobot merupakan faktor yang sangat mempengaruhi nilai analisa.
d. Pemberian Nilai Pada Setiap Kriteria.
e. Menghitung Nilai, proses selanjutnya adalah melakukan perhitungan dengan menggunakan rumus metode perbandingan Eksponensial (MPE).
f. Menentukan Hasil Atau Prioritas Keputusan, berikut ini adalah tabel prioritas keputusan.

Tabel 16. Prioritas Keputusan

No.	Alternatif	Total Nilai	Rangking
1.	Algoritma *Taboo Codes*	8,641	2
2.	Algoritma *Yamamoto's Recursive Code*	7,203	1

Berdasarkan analisa tabel diatas, diambil kesimpulan bahwa algoritma *Yamamoto's Recursive Code* merupakan algoritma kompresi *file* teks tercepat dan paling efisien, karena semakin kecil total nilai prioritas yang diperoleh, maka semakin sedikit proses kompresi yang dilakukan pada algoritma tersebut.

4. KESIMPULAN

Setelah menerapkan algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code* untuk melakukan kompresi dan dekompresi *file* teks telah membuktikan bahwa kompresi *file* teks menggunakan algoritma *Taboo Codes* dengan hasil space saving sebesar 36,25% dan algoritma *Yamamoto's Recursive Code* dengan hasil space saving 25%. Tanpa metode perbandingan algoritma *Taboo Codes* menjadi algoritma yang lebih efektif dan efisien untuk melakukan kompresi *file* teks. Hasil perbandingan kompresi *file* teks antara algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code* tidak jauh berbeda hasilnya. Dengan melakukan penerapan metode perbandingan Exponential dengan 4 kriteria, hasil algoritma *Taboo Codes* memiliki total nilai 8,641 sedangkan algoritma *Yamamoto's Recursive Code* 7,203. Algoritma yang lebih efektif dan efisien dalam melakukan kompresi *file* teks dengan metode perbandingan exponential adalah algoritma *Yamamoto's Recursive Code*. Aplikasi yang dirancang mampu melakukan kompresi *file* teks dengan menggunakan algoritma *Taboo Codes* dan algoritma *Yamamoto's Recursive Code* serta melakukan perbandingan kedua algoritma tersebut dengan menggunakan metode exponential menggunakan Microsoft Visual Studio 2010 dan berjalan dengan sangat baik.

REFERENCES

[1] R. I. Borman and H. Fauzi, "Dalam Sistem Pendukung Keputusan Penerima Beasiswa," *CESS J. Comput. Eng. Syst. Sci.*, vol. 3, no. 1, pp. 17–22, 2018.
[2] M. A. Latif, S. D. Nasution, and P. Pristiwanto, "Analisa Perbandingan Algoritma Rice Codes Dengan Algoritma Goldbach Codes Pada Kompresi *File* Text Menggunakan Metode Exponential," *Maj. Ilm. INTI (Informasi dan Teknol. Ilmiah)*, vol. 13, no. 1, pp. 28–33, 2018, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/639.
[3] N. Aftikasyah, "Penerapan Algoritma Yamamoto ' s Recursive Code Untuk Mengkompresi *File* Dokumen," vol. 5, pp. 255–263, 2022, doi: 10.30865/komik.v5i1.3716.
[4] S. Simanjuntak, "Implementasi Metode Taboo Code Untuk Kompresi *File* Video," *Explorer (Hayward).*, vol. 2, no. 1, pp. 32–38, 2022, doi: 10.47065/explorer.v2i1.156.
[5] S. B. Ginting *et al.*, "Perbandingan Algoritma Yamamoto ' s Recursive Code Dan Additive Code Dalam Kompresi *File* Video," vol. 5, 2021, doi: 10.30865/komik.v5i1.3819.
[6] W. T. W. Simanjuntak, "Analisa Perbandingan Algortima Prediction By Partial Matching Dengan Sequitur Pada Kompresi *File* Teks," vol. 5, pp. 221–227, 2021, doi: 10.30865/komik.v5i1.3675.
[7] M. R. Irliansyah, S. D. Nasution, and K. Ulfa, "Penerapan Metode Deflate Dan Algoritma Goldbach Codes Dalam Kompresi *File* Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 1, no. 1, pp. 186–189, 2017.
[8] A. Apijuddin Kompresi and A. S. Codes, "Perancangan Aplikasi Kompresi *File* Teks Menggunakan Algoritma Stout Codes," vol. 9, pp. 183–188, 2021.
[9] D. Salomon and G. Motta, *Handbook of Data Compression 5th*, vol. 53, no. 9. 2019.
[10] L. V Simanjuntak, "Perbandingan Algoritma Elias Delta Code dengan Levenstein Untuk Kompresi *File* Teks," *J. Comput. Syst. ...*, vol. 1, no. 3, pp. 184–190, 2020, [Online]. Available: https://ejurnal.seminar-id.com/index.php/josyc/article/view/168.
[11] J. P. Informatika *et al.*, "Analisa Perbadingan Algoritma Goldbach Codes Dengan Algoritma Sequitur Pada Kompresi *File* Text," vol. 18, pp. 354–357, 2019.

ISSN 2597-4645 (media online)
ISSN 2597-4610 (media cetak)
Page: 140-150

[12] . Nainggolan, "Analisa Perbandingan Algoritma Goldbach Codes Dengan Algoritma Dynamic Markov Compression (DMC) Pada Kompresi *File* Teks Menggunakan Metode Eksponensial," *Maj. Ilm. INTI*, vol. 6, no. 3, pp. 395–399, 2019.