Modifikasi Algoritme *J-Bit Encoding* untuk Meningkatkan Rasio Kompresi

Johanes K.M. Lobang[1], Pranowo[2], Suyoto[3]

***Abstract*—J-bit encoding is a lossless data compression algorithm that manipulates each bit of file data in order to minimize the size by dividing data into two outputs and combining data into two outputs. This research proposes a modification of the J-bit Encoding algorithm by eliminating zero and one symbols of the first output. As a result, the first output will contain the original data without zero and one symbols and the second output will contain the value of two bits that describe the position of zero, one, and byte besides zero and one. The two algorithms are compared by testing four scheme combination algorithms, which are (i) Burrows-Wheeler transformation, Move to Front, J-Bit Encoding, and arithmetic coding, (ii) Burrows-Wheeler transformation, Move to Front, modification of the J-bit Encoding, and arithmetic coding, (iii) Burrows-Wheeler transformation, Move One From Front, J-Bit encoding, and arithmetic coding, (iv) Burrows-Wheeler transformation, Move One From Front, modification of the J-bit Encoding, and arithmetic coding. By using the Calgary Corpus and Canterbury Corpus data sets, the test results show that the best compression ratio is obtained by using a second scheme on average. On the other hand, by using four image files, the test results show that the best compression ratio is obtained by using a fourth scheme on average.**

***Intisari*—*J-bit encoding* merupakan algoritme kompresi data *lossless* yang memanipulasi setiap bit data dalam *file* untuk meminimalkan ukuran dengan cara membagi data menjadi dua keluaran, kemudian dikombinasikan kembali menjadi satu keluaran. Makalah ini mengusulkan modifikasi algoritme *J-bit encoding* dengan cara mengeliminasi simbol nol dan satu dari keluaran pertama, sehingga keluaran pertama akan berisi data asli selain nol dan satu (dalam ukuran *byte*) dan keluaran kedua akan berisi nilai dua bit yang menjelaskan posisi *byte* nol, *byte* satu, dan *byte* selain nol dan satu. Perbandingan unjuk kerja kedua algoritme ini dilakukan dengan menggunakan empat skema kombinasi algoritmem yaitu (i) transformasi Burrows-Wheeler, *Move to Front*, *J-bit encoding*, dan pengkodean aritmatika, (ii) transformasi Burrows-Wheeler, *Move to Front*, algoritme hasil modifikasi, dan pengkodean aritmatika, (iii) transformasi Burrows-Wheeler, *Move One From Front*, *J-bit encoding*, dan pengkodean aritmatika, (iv) transformasi Burrows-Wheeler, *Move One From Front*, algoritme hasil modifikasi, dan pengkodean aritmatika. Dengan menggunakan *data set* Calgary *Corpus* dan Canterbury *Corpus*, hasil pengujian menunjukkan bahwa rata-rata rasio kompresi terbaik diperoleh dengan menggunakan skema kedua. Sedangkan dengan menggunakan empat *file* gambar, hasil pengujian menunjukkan bahwa rata-rata rasio kompresi terbaik diperoleh dengan menggunakan skema keempat.**

***Kata Kunci*—kompresi data, burrows-wheeler *compression algorithm*, *j-bit encoding*, kombinasi algoritme.**

I. PENDAHULUAN

Kompresi data telah dan masih memainkan peran penting dalam berbagai area di kehidupan sehari-hari [1]. Sejak Claude E. Shannon merumuskan teori informasi yang memberikan cara formal untuk mengukur informasi dan ketidakpastian yang merupakan dasar untuk kompresi data [2], hingga saat ini telah banyak algoritme kompresi data yang dikembangkan, seperti Huffman *coding*, *arithmetic coding*, Lempel-Ziv *coding*, *wavelet coding*, *discrete cosine transform*, dan sebagainya. Secara umum, algoritme-algoritme tersebut dapat dikelompokkan ke dalam dua jenis, yaitu kompresi *lossless* dan kompresi *lossy*. Pengelompokan ini didasarkan pada hilang atau tidaknya informasi saat data yang terkompresi dikembalikan ke bentuk aslinya. Pada kompresi *lossless*, data yang terkompresi dapat dipulihkan atau dikembalikan sama seperti data aslinya. Sebaliknya, pada kompresi *lossy*, data yang terkompresi tidak dapat dipulihkan sama seperti data aslinya, tetapi dianggap setara dengan data asli atau sesuai dalam pengamatan manusia [3].

Apabila ditinjau dari efektivitasnya, dapat dinyatakan bahwa kompresi *lossy* lebih efektif jika dibandingkan dengan kompresi *lossless*. Rasio kompresi yang dapat dicapai oleh kompresi *lossy* umumnya berkisar antara 100:1 - 200:1. Sedangkan pada kompresi *lossless* rasio kompresi yang dapat dicapai umumnya hanya berkisar antara 2:1 – 8:1, sehingga untuk meningkatkan rasio kompresi, umumnya pada kompresi *lossless*, beberapa teknik kompresi dikombinasikan bersama untuk membentuk sebuah algoritme kompresi baru. Sebagai contoh, mesin faks menggunakan kombinasi *Run-Length Encoding* (RLE) dan pengkodean Huffman untuk mencapai rasio kompresi sekitar 10:1. Algoritme *Deflate* yang diperkenalkan oleh Phil Katz pada tahun 1993 dan dimplementasikan dalam aplikasi pkzip menggunakan kombinasi algoritme LZ77 dan pengkodean Huffman [4], [5].

Selain kedua algoritme tersebut, salah satu kombinasi algoritme kompresi *lossless* yang terkenal adalah Burrows-Wheeler *Compression Algorithm* (BWCA). BWCA merupakan metode kompresi data untuk tujuan umum yang menggabungkan empat teknik kompresi data, yaitu Burrows-Wheeler *Transform* (BWT), *Global Structure Transform* (GST), *Run-Length Encoding* (RLE), dan *Entropy Coding* (EC) untuk meningkatkan rasio kompresi [6].

Pada tahun 2012, Suarjaya mempublikasikan algoritme *J-bit encoding* (JBE) yang dimaksudkan untuk digabungkan

[1]*Mahasiswa, Program Studi Magister Teknik Informatika, Program Pasca Sarjana Universitas Atma Jaya Yogyakarta, Jl. Babarsari No. 43, Yogyakarta 55281, INDONESIA (e-mail: johaneslobang@gmail.com)*

[2,3]*Dosen, Program Studi Magister Teknik Informatika, Program Pasca Sarjana Universitas Atma Jaya Yogyakarta, Jl. Babarsari No. 43, Yogyakarta 55281, INDONESIA (e-mail: pran@staff.uajy.ac. id, suyoto@staff.uajy.ac.id)*

dengan algoritme kompresi data lainnya untuk mengoptimalkan rasio kompresi [7]. Ide utama algoritme ini adalah membagi data masukan menjadi dua keluaran. Keluaran data pertama memuat data asli selain *byte* nol dan keluaran data kedua memuat nilai bit yang menjelaskan posisi *byte* nol dan *byte* selain nol [8], [9].

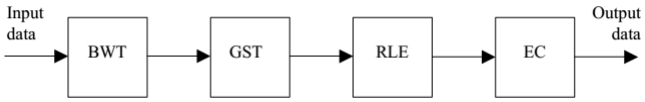
Sebagai algoritme yang tergolong dalam jenis kompresi *lossless* [9]-[12], JBE dapat diterapkan untuk semua jenis *file*. Kelebihan dari algoritme JBE adalah dapat menghasilkan rasio kompresi yang tinggi apabila data masukan memuat persentase *byte* nol yang tinggi. Sebaliknya, apabila persentase *byte* nol dari data masukan rendah, maka algoritme JBE akan menghasilkan rasio kompresi yang dapat mendekati nilai satu, bahkan bisa melebihi satu (data terekspansi).

Kelemahan algoritme JBE ini dapat diatasi dengan mengkombinasikannya dengan BWT dan *recency ranking scheme* seperti *Move to Front* (MTF), karena data keluaran hasil transformasi kedua algoritme tersebut sering didominasi oleh simbol nol (umumnya sekitar 60%) [3]. Selain didominasi oleh simbol nol, data keluaran BWT dan MTF juga menghasilkan keluaran simbol satu dengan persentase yang cukup tinggi (umumnya sekitar 10-15%).

Penulisan makalah ini diatur sebagai berikut. Bagian kedua memuat penjelasan singkat mengenai BWCA, bagian ketiga memuat pembahasan mengenai modifikasi algoritme JBE yang diusulkan, bagian keempat membahas analisis perbandingan algoritme JBE dan algoritme modifikasi berdasarkan perhitungan rasio kompresi, bagian kelima membahas perbandingan unjuk kerja algoritme yang diusulkan dan algoritme asli, serta bagian keenam menyajikan kesimpulan dari hasil pengujian.

II. BURROWS-WHEELER *COMPRESSION ALGORITHM* (BWCA)

Skema umum BWCA terdiri atas empat tahap sebagaimana dapat dilihat pada Gbr. 1 [13].



Gbr. 1 Skema umum BWCA.

Untuk proses kompresi, tahap-tahap tersebut diproses secara berurutan dari kiri ke kanan. Sebaliknya untuk proses dekompresi tahap-tahap tersebut diproses dari kanan ke kiri.

A. Transformasi Burrows-Wheeler (BWT)

Apabila data yang akan dikompresi dikelompokkan terlebih dahulu berdasarkan simbolnya secara terurut, maka rasio kompresi yang dapat diperoleh ketika data tersebut dikompresi akan tinggi. Namun cara pengurutan ini tidak *reversible*. Pada tahun 1994, Burrows dan Wheeler mempublikasikan makalah mereka yang menunjukkan cara pengurutan *reversible* mirip seperti ilustrasi di atas, yang dikenal sebagai Transformasi Burrows-Wheeler atau disingkat BWT [14], [15].

BWT bekerja dengan cara membaca sebuah blok data dengan panjang n sebagai sebuah string (S = S1..Sn), kemudian melakukan rotasi (*cyclic shift*) sebanyak n-1 kali, sehingga menghasilkan matriks M berukuran n x n dan mengurutkan matriks tersebut secara leksikografi. Hasil transformasi ini berupa semua elemen pada kolom terakhir matriks (L) dan indeks dari posisi string aslinya [16]. Tabel I dan Tabel II memberikan gambaran proses BWT dari masukan string "RAKSASA".

TABEL I
PROSES ROTASI BWT PADA STRING "RAKSASA"

Indeks	**Masukan dirotasi**						
	F						**L**
0	R	A	K	S	A	S	A
1	A	K	S	A	S	A	R
2	K	S	A	S	A	R	A
3	S	A	S	A	R	A	K
4	A	S	A	R	A	K	S
5	S	A	R	A	K	S	A
6	A	R	A	K	S	A	S

TABEL II
PROSES *SORTING* PADA STRING "RAKSASA"

Indeks	**Rotasi diurutkan**							**Hasil**
	F						**L**	
0	A	K	S	A	S	A	R	R
1	A	R	A	K	S	A	S	S
2	A	S	A	R	A	K	S	S
3	K	S	A	S	A	R	A	A
4	R	A	K	S	A	S	A	A
5	S	A	R	A	K	S	A	A
6	S	A	S	A	R	A	K	K

B. Global Sructure Transform (GST)

GST adalah proses pengubahan masukan data yang memiliki konteks lokal menjadi keluaran data yang memiliki konteks global. Umumnya keluaran dari BWT mengandung distribusi probabilitas lokal, yaitu banyak fragmen simbol dengan konteks yang sama muncul dengan distribusi probabilitas yang berbeda. Sebagai tahap kedua dalam BWCA, GST mengubah konteks lokal hasil keluaran BWT ini menjadi konteks global dengan menggunakan *recency ranking scheme* seperti MTF [13].

Untuk meningkatkan rasio kompresi dari BWCA, banyak penelitian yang mengembangkan algoritme untuk memproses keluaran BWT dengan memodifikasi atau menggantikan MTF seperti *Move One From Front* (M1FF), *Move One From Front Two*, *Incremental Frequency Count*, *Distance Coding*, dan *Inversion Frequencies*.

M1FF merupakan modifikasi dari MTF yang diusulkan oleh Balkenhol, et al. [17]. Berbeda dengan MTF yang mengubah simbol masukan dari posisi yang tinggi ke posisi awal, M1FF bekerja dengan mengubah simbol masukan dari posisi kedua dalam *list* dipindahkan ke posisi pertama dan masukan dari posisi yang lebih tinggi dipindahkan ke posisi kedua [18]. Sebagai contoh, diasumsikan bahwa *list* MTF dan M1FF diinisialisasi dengan empat simbol secara berurutan, yaitu [A, K, S, R]. Apabila keluaran BWT dari *string* "RAKSASA" yaitu "RSSAAAK" dikodekan dengan MTF, maka akan dihasilkan keluaran "3302003" seperti ditunjukkan

pada Tabel III, sedangkan jika dikodekan dengan M1FF, maka keluaran yang akan dihasilkan adalah "3311003", sebagaimana ditunjukkan pada Tabel IV.

TABEL III
PROSES *ENCODING* MTF

Simbol yang akan dikodekan	Daftar simbol MTF				Kode MTF
	0	1	2	3	
RSSAAAK	A	K	S	(R)	**3**
SSAAAK	R	A	K	(S)	3**3**
SAAAK	(S)	R	A	K	33**0**
AAAK	S	R	(A)	K	330**2**
AAK	(A)	S	R	K	3302**0**
AK	(A)	S	R	K	33020**0**
K	A	S	R	(K)	3302003

TABEL IV
PROSES *ENCODING* M1FF

Simbol yang akan dikodekan	Daftar simbol M1FF				Kode M1FF
	0	1	2	3	
RSSAAAK	A	K	K	(R)	**3**
SSAAAK	A	R	K	(S)	3**3**
SAAAK	A	(S)	R	K	33**1**
AAAK	S	(A)	R	K	331**1**
AAK	(A)	S	R	K	3311**0**
AK	(A)	S	R	K	33110**0**
K	A	S	R	(K)	3311003

C. Run-Length Encoding (RLE)

RLE merupakan teknik kompresi klasik yang sering digunakan atau dikombinasikan dengan teknik kompresi data lainnya dalam mengompresi data. Ide dari teknik ini adalah mengkodekan perulangan *string* dengan dua buah simbol, yaitu panjang dari *string* dan simbol dari *string* tersebut, ataupun sebaliknya. Sebagai contoh *string* "abbaaaaabaabbbaa" dengan panjang 16 *byte* apabila dikodekan dengan RLE akan menjadi "1a2b5a1b2a3b2a" atau "a1b2a5b1a2b3a2" dengan panjang 14 *byte* [19].

Dalam publikasinya, Burrows dan Wheeler mengusulkan pendekatan RLE yang berbeda, yang disebut RLE0. RLE0 bekerja dengan menjumlahkan simbol selain nol dengan satu dan kemudian mengkodekan perulangan dari nol [14].

D. Entropy Encoding (EC)

Tahap akhir dalam BWCA adalah EC, yang mengompresi *stream* hasil keluaran RLE menjadi lebih kecil dengan menggunakan metode statistik. Dua algoritme yang umumnya digunakan dalam tahapan ini adalah pengkodean Huffman dan pengkodean aritmatika. Diantara kedua metode ini, pengkodean aritmatika merupakan metode yang paling optimal. Namun, pengkodean aritmatika lebih lambat dibandingkan dengan pengkodean Huffman, karena pada proses kompresi dan dekompresi, pengkodean aritmatika menggunakan operasi pembagian dan perkalian [20].

III. MODIFIKASI ALGORITME JBE

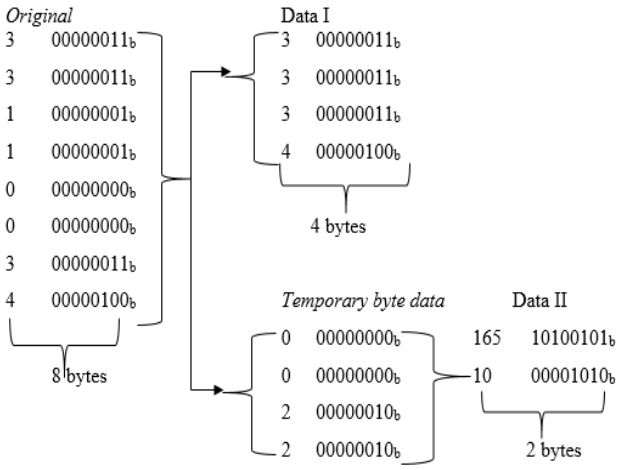
Makalah ini mengusulkan modifikasi algoritme JBE yang bekerja sama seperti algoritme JBE yaitu membagi data masukan menjadi dua keluaran, kemudian dikombinasikan kembali menjadi satu keluaran. Perbedaaannya yaitu pada JBE keluaran pertama akan berisi data asli yang bukan nol (dalam ukuran *byte*) dan keluaran kedua akan berisi nilai bit yang menjelaskan posisi *byte* nol dan *byte* selain nol. Sedangkan pada modifikasi algoritme JBE, keluaran pertama akan berisi data asli selain nol dan satu (dalam ukuran *byte*) dan keluaran kedua akan berisi nilai dua bit yang menjelaskan posisi *byte* nol, *byte* satu, dan *byte* selain nol dan satu.

Langkah-langkah proses kompresi dan dekompresi dari modifikasi JBE dapat dilihat pada *pseudocode* berikut.

A. Proses Kompresi

a. Membaca masukan per *byte*.
b. Jika *byte* yang dibaca bernilai nol maka dua bit '00' ditulis ke dalam *temporary byte* data. Jika *byte* yang dibaca bernilai satu maka dua bit '01' ditulis ke dalam *temporary byte* data. Jika *byte* yang dibaca bernilai lebih dari satu maka dua bit '10' ditulis ke dalam *temporary byte* data dan dituliskan juga *byte* tersebut ke dalam data II .
c. Jika *temporary byte* data telah terisi dengan 8 bit, maka nilai *byte* tersebut ditulis ke dalam data II dan *temporary byte* data dihapus.
d. Mengulangi langkah a - c hingga akhir *file* dicapai.
e. Jika diikuti oleh algoritme kompresi lain, data I dan data II dapat dikompresi secara terpisah sebelum digabungkan (opsional).
f. Menggabungkan data I dan II
   - Menulis panjang data I.
   - Menulis data I.
   - Menulis panjang data II.
   - Menulis data II.

Untuk lebih memperjelas proses kompresi dari algoritme yang diusulkan, Gbr. 2 menyajikan contoh proses tersebut.



Gbr. 2 Gambaran proses kompresi algoritme hasil modifikasi.

B. Proses Dekompresi

a. Memisahkan data I dan II
   - Membaca panjang data I.
   - Menulis data I.
   - Membaca panjang data II.
   - Menulis data II.

b. Jika dikompresi dengan algoritme kompresi lain, data I dan data II didekompresi (opsional).
c. Membaca Data II per dua bit.
d. Jika yang dibaca adalah '10', maka data I dibaca dan ditulis ke keluaran. Jika yang dibaca adalah '01', maka *byte* satu ditulis ke keluaran. Jika yang dibaca adalah '00' maka *byte* nol ditulis ke keluaran.
e. Mengulangi langkah c dan d hingga panjang masukan data asli dicapai.

Adapun pertimbangan sehingga simbol satu juga dieliminasi dari data I dan diusulkan sebagai modifikasi algoritme JBE yaitu:

a. Dengan mengeliminasi simbol nol dan satu dari data I, maka maksimum terdapat 253 simbol pada data I. Sedangkan untuk algoritma JBE, dengan hanya mengeliminasi simbol nol dari data I, maka maksimum terdapat 254 simbol pada data I.
b. Kombinasi bit-bit data dari *temporary byte* data menjadi data II untuk algoritme hasil modifikasi maksimum menghasilkan 81 simbol, sedangkan untuk algoritme JBE maksimum menghasilkan 255 simbol.
c. Simbol satu hasil keluaran kombinasi BWT dan MTF/M1FF umumnya sekitar 10-15%.

Ketiga pertimbangan di atas mensyaratkan bahwa algoritme hasil modifikasi akan bekerja maksimum apabila sebelum algoritme ini diproses, terlebih dahulu dijalankan proses BWT dan MTF/M1FF, dan setelah algoritme ini dijalankan, proses kompresi dilanjutkan dengan algoritme kompresi lainnya. Oleh karena itu, pada makalah ini empat skema kombinasi algoritme kompresi data diuji dan dibandingkan untuk mengetahui unjuk kerja algoritme JBE dan algoritme hasil modifikasi . Keempat skema kombinasi tersebut yaitu:

a. BWT, MTF, JBE, dan pengkodean Aritmatika,
b. BWT, MTF, algoritme hasil modifikasi, dan pengkodean Aritmatika,
c. BWT, M1FF, JBE, dan pengkodean Aritmatika, dan
d. BWT, M1FF, algoritme hasil modifikasi, dan pengkodean Aritmatika.

IV. ANALISIS PERBANDINGAN

Umumnya rasio kompresi dari algoritme kompresi dapat dihitung apabila data masukan telah dikompresi. Berbeda dengan algoritme lainnya, algoritme JBE memiliki kelebihan, yaitu dengan diketahuinya frekuensi simbol nol dan panjang data masukan, sebelum data tersebut dikompresi rasio kompresi yang akan dicapai sudah dapat dihitung.

Jika dianggap X merupakan panjang *file* pertama, Y merupakan panjang *file* kedua, N merupakan panjang *file* masukan, dan Fs merupakan frekuensi simbol nol, maka rasio kompresi (Z) dari algoritme JBE dapat dihitung dengan rumus (1) sampai (3).

$$X = N - Fs \quad (1)$$

$$Y = N / 8 \quad (2)$$

$$Z = (X + Y) / N \quad (3)$$

Sebagai pengembangan dari algoritme JBE, maka rasio kompresi yang akan dicapai oleh algoritme hasil modifikasi juga dapat dihitung sebelum data dikompresi. Dengan diketahuinya frekuensi simbol nol, frekuensi simbol satu, dan panjang data masukan, maka rasio kompresi dapat dihitung menggunakan (3).

Untuk algoritme JBE, nilai Fs pada (1) merupakan nilai frekuensi simbol nol, sedangkan untuk algoritme hasil modifikasi, nilai Fs pada (1) merupakan jumlah frekuensi simbol nol dan simbol satu. Pada (2), nilai Y untuk algoritme JBE diperoleh dari hasil bagi panjang *file* asli dengan delapan, sedangkan nilai Y untuk algoritme hasil modifikasi diperoleh dari hasil bagi panjang *file* asli dengan empat.

TABEL V
PERBANDINGAN RASIO KOMPRESI

Algoritme JBE	**Algoritme Hasil Modifikasi**
File book1 X= 768.771– 382.507 = 386.264 Y= 768.771/8 = 96.097. Z = (386.264 + 96.097) / 768.771 = 0,627444	File book1 X= 768.771– 500.534 = 268.237 Y= 768.771/4 = 192.193 Z = (268.237 + 192.193) / 768.771 = 0,598917
File book2 X= 610.856 – 371.490 = 239.366 Y= 610.856/8 = 76.357 Z = (239.366 + 76.357) / 610.856 = 0,516853	File book2 X= 610.856 – 448.161 = 162.695 Y= 610.856/4 = 152.714 Z = (162.695 + 152.714) / 610.856 = 0,516339
File pertama X= 8.000 – 8.000 = 0 Y= 8.000/8 = 1.000 Z = (0 + 1.000) / 8.000 = 0,125	File pertama X= 8.000 – 8.000 = 0 Y= 8.000/4 = 2.000 Z = (0 + 2.000) / 8.000 = 0,250
File kedua X= 8.000 – 0 = 8.000 Y=8.000/8 = 1.000 Z = (8.000 + 1.000) / 8.000 = 1,125	File kedua X= 8.000 – 8.000 = 0 Y= 8.000/4 = 2.000 Z = (0 + 2.000) / 8.000 = 0,250
File ketiga X= 8.000 – 0 = 8.000 Y=8.000/8 = 1.000 Z = (8.000 + 1.000) / 8.000 = 1,125	File ketiga X= 8.000 – 0 = 8.000 Y= 8.000/4 = 2.000 Z = (8.000 + 2.000) / 8.000 = 1,250

Tabel V menyajikan contoh perhitungan rasio kompresi algoritme JBE dan algoritme hasil modifikasi pada hasil

keluaran BWT dan MTF dari *file* book1 dan book2 yang terdapat dalam *data set* Calgary *Corpus*, dan pada tiga *file* masukan yang masing-masing berukuran 8.000 *byte*, dengan *file* pertama semua *byte*-nya bernilai nol, file kedua semua *byte*-nya bernilai satu, dan file ketiga semua *byte*-nya bernilai lebih dari satu.

Perbandingan rasio kompresi algoritme JBE dan algoritme hasil modifikasi yang disajikan pada Tabel V untuk *file* book1 dan *file* book2 menunjukkan bahwa algoritme hasil modifikasi dapat meningkatkan rasio kompresi. Peningkatan ini ditunjukkan dengan semakin kecilnya nilai rasio kompresi.

Maksimum rasio kompresi terbaik untuk algoritme JBE adalah 0,125, yang dapat dicapai apabila semua *byte* dalam data masukan bernilai nol. Sedangkan maksimum rasio kompresi terbaik untuk algoritme hasil modifikasi, yaitu 0,250, dapat dicapai ketika semua *byte* dalam data masukan bernilai nol atau satu.

Rasio kompresi terburuk untuk algoritme JBE adalah 1,125, yang dihasilkan apabila data masukan tidak memuat *byte* nol. Sedangkan rasio kompresi terburuk untuk algoritme hasil modifikasi adaah 1,250, yang akan dihasilkan apabila data masukan tidak memuat *byte* nol atau *byte* satu.

V. HASIL DAN PEMBAHASAN

Pengujian unjuk kerja algoritme kompresi *lossless* umumnya menggunakan *data set* uji standar dari Calgary *Corpus* dan Canterbury *Corpus* [21]-[23]. Dalam makalah ini, selain digunakan kedua *dataset* tersebut, pengujian unjuk kerja keempat skema kombinasi algoritme yang telah disebutkan sebelumnya juga dilakukan pada empat buah *file* gambar berwarna, yaitu airplane.tiff, lena.tiff, mandrill.tiff, dan peppers.tiff.

Pengujian dilakukan menggunakan laptop Acer Aspire E11 dengan spesifikasi processor Intel celeron N2840 2,1 GHz, Memori DDR3 2 Gb, dan Hardisk 500 GB.

A. Analisis Hasil Berdasarkan Rasio Kompresi

Perbandingan rasio kompresi pada *data set* Calgary *Corpus* dan Canterbury *Corpus* dari keempat skema kombinasi yang diuji ditampilkan pada Tabel VI dan Tabel VII. Untuk *file* gambar berwarna, perbandingan rasio kompresi keempat skema kombinasi yang diuji ditampilkan pada Tabel VIII. Adapun ukuran blok memori yang digunakan untuk melakukan proses transformasi Burrows-Wheeler adalah 1 MB.

Sebagaimana yang ditampilkan pada Tabel VI dan Tabel VII, penggunaan algoritme MTF dan M1FF sebelum algoritme JBE mempengaruhi pencapaian rasio kompresi. Kombinasi BWT, MTF, JBE, dan pengkodean aritmatika menghasilkan rasio kompresi lebih baik untuk data berkapasitas kurang dari 200 KB. Sebaliknya kombinasi BWT, M1FF, JBE, dan pengkodean aritmatika menghasilkan rasio kompresi lebih baik untuk data berkapasitas lebih dari 200 KB. Berbeda dengan algoritme JBE, algoritme hasil modifikasi lebih cocok menggunakan skema kombinasi BWT, MTF, algoritme hasil modifikasi, dan pengkodean aritmatika dibandingkan dengan menggunakan skema kombinasi BWT, M1FF, algoritme hasil modifikasi, dan pengkodean aritmatika.

TABEL VI
PERBANDINGAN RASIO KOMPRESI *DATA SET* CALGARY *CORPUS*

No	*File*	Ukuran (*byte*)	Skema kombinasi			
			1	2	3	4
1	bib	111.261	0,267	0,267	0,270	0,268
2	book1	768.771	0,321	0,321	0,317	0,317
3	book2	610.856	0,274	0,276	0,273	0,274
4	geo	102.400	0,624	0,626	0,623	0,623
5	news	377.109	0,330	0,332	0,332	0,333
6	obj1	21.504	0,556	0,543	0,553	0,541
7	obj2	246.814	0,325	0,327	0,329	0,329
8	paper1	53.161	0,336	0,329	0,340	0,331
9	paper2	82.199	0,328	0,324	0,328	0,324
10	paper3	46.526	0,369	0,361	0,370	0,361
11	paper4	13.286	0,460	0,428	0,460	0,429
12	paper5	11.954	0,478	0,444	0,478	0,447
13	paper6	38.105	0,353	0,342	0,358	0,344
14	pic	513.216	0,110	0,113	0,105	0,110
15	progc	39.611	0,346	0,335	0,350	0,337
16	progl	71.646	0,238	0,233	0,242	0,236
17	progp	49.379	0,240	0,233	0,247	0,237
18	trans	93.695	0,203	0,201	0,213	0,206
Rata-rata :			0,342	0,335	0,344	0,336

TABEL VII
PERBANDINGAN RASIO KOMPRESI *DATA SET* CANTERBURY *CORPUS*

No	*File*	Ukuran (*byte*)	Skema kombinasi			
			1	2	3	4
1	alice29.txt	152.089	0,303	0,301	0,301	0,299
2	asyoulik.txt	125.179	0,339	0,337	0,337	0,335
3	cp.html	24.603	0,356	0,341	0,358	0,343
4	fields.c	11.150	0,347	0,314	0,354	0,318
5	grammar.lsp	3.721	0,511	0,449	0,500	0,455
6	kennedy.xls	1.029.744	0,192	0,187	0,191	0,187
7	lcet10.txt	426.754	0,270	0,270	0,268	0,269
8	plrabn12.txt	481.861	0,324	0,325	0,321	0,322
9	ptt5	513.216	0,110	0,113	0,105	0,110
10	sum	38.240	0,375	0,366	0,379	0,368
11	xargs.1	4.227	0,567	0,506	0,555	0,510
12	bible.txt	4.047.392	0,223	0,227	0,222	0,225
13	e.coli	4.638.690	0,249	0,248	0,248	0,248
14	world192.txt	2.473.400	0,207	0,211	0,207	0,211
Rata-rata :			0,312	0,300	0,310	0,300

TABEL VIII
PERBANDINGAN RASIO KOMPRESI EMPAT *FILE* GAMBAR BERWARNA

No	File	Ukuran (*byte*)	Skema kombinasi			
			1	2	3	4
1	airplane.tiff	786.572	0,642	0,639	0,640	0,638
2	lena.tiff	786.572	0,758	0,758	0,758	0,758
3	mandrill.tiff	786.572	0,920	0,919	0,919	0,919
4	peppers.tiff	786.572	0,832	0,833	0,830	0,831
Rata-rata :			0,788	0,787	0,787	0,786

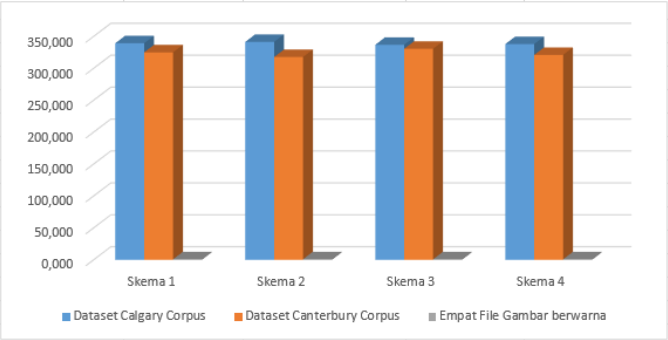
Apabila ditinjau berdasarkan tipe *file*-nya, maka dapat dinyatakan bahwa algoritme hasil modifikasi menghasilkan rasio kompresi lebih baik untuk *file* bertipe teks, sedangkan algoritme JBE menghasilkan rasio kompresi lebih baik untuk tipe *file* selain teks. Jika ditinjau berdasarkan kapasitas data yang dikompresi, maka dapat dinyatakan bahwa algoritme hasil modifikasi bekerja lebih baik untuk *file* berkapasitas

kurang dari 200 KB, sedangkan algoritme JBE bekerja lebih baik untuk *file* berkapasitas lebih dari 200 KB.

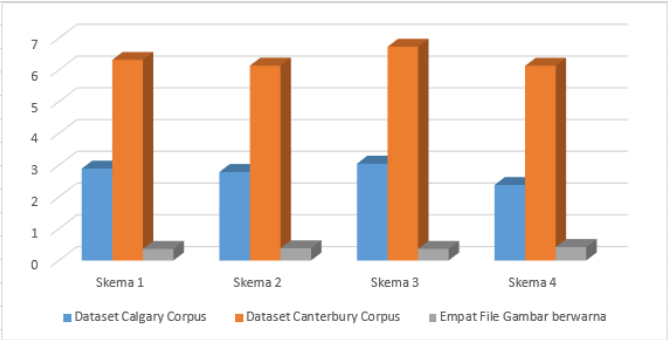
Tabel VIII menunjukkan bahwa keempat skema kombinasi algoritme yang diuji kurang cocok diterapkan pada kompresi *file* gambar berwarna karena rata-rata rasio kompresi yang dihasilkan adalah 0,78.

B. Analisis Hasil Berdasarkan Kecepatan

Dalam makalah ini, kecepatan kompresi dan dekompresi diukur dalam satuan detik. Total waktu proses kompresi untuk semua *file* pada *data set* Calgary *Corpus* dan Canterbury *Corpus* serta keempat *file* gambar berwarna ditunjukkan pada Gbr. 3, sedangkan total waktu proses dekompresi ditunjukkan pada Gbr. 4.



Gbr. 3 Perbandingan waktu proses kompresi (dalam detik).



Gbr. 4 Perbandingan waktu proses dekompresi (dalam detik).

Gbr. 3 menunjukkan bahwa efisiensi waktu terbaik untuk proses kompresi pada *data set* Calgary *Corpus* dicapai dengan menggunakan skema kombinasi BWT, MTF, algoritme hasil modifikasi, dan pengkodean aritmatika, sedangkan efisiensi waktu terbaik untuk proses kompresi pada *data set* Canterbury *Corpus* dicapai dengan menggunakan skema kombinasi BWT, M1FF, JBE, dan pengkodean aritmatika. Untuk empat *file* gambar berwarna, efisiensi waktu terbaik untuk proses kompresi dicapai dengan menggunakan kombinasi algoritme BWT, MTF, JBE, dan pengkodean aritmatika.

Untuk proses dekompresi sebagaimana yang ditampilkan pada Gbr. 4, efisiensi waktu terbaik untuk kedua *data set* dicapai dengan menggunakan skema kombinasi BWT, M1FF, algoritme hasil modifikasi, dan pengkodean aritmatika, sedangkan untuk empat *file* gambar berwarna, efisiensi waktu terbaik untuk proses dekompresi dicapai menggunakan skema kombinasi BWT, MTF, JBE, dan pengkodean aritmatika.

VI. KESIMPULAN

Makalah ini mengusulkan modifikasi algoritme JBE yang bekerja seperti algoritme JBE yaitu membagi data masukan menjadi dua keluaran, kemudian dikombinasikan kembali menjadi satu keluaran. Berdasarkan pengujian dengan menggunakan empat skema kombinasi algoritme, dapat dinyatakan bahwa algoritme hasil modifikasi dapat meningkatkan rasio kompresi karena rata-rata rasio kompresi terbaik dari *data set* Calgary *Corpus* dan *data set* Canterbury *Corpus* diperoleh dengan menggunakan skema kombinasi BWT, MTF, algoritme hasil modifikasi, dan pengkodean aritmatika, yaitu 0,335 untuk *data set* Calgary *Corpus* dan 0,300 untuk *data set* Canterbury *Corpus*. Sedangkan untuk empat *file* gambar berwarna, rata-rata rasio kompresi terbaik diperoleh dengan menggunakan skema kombinasi BWT, M1FF, algoritme hasil modifikasi, dan pengkodean aritmatika, yaitu 0,786.

REFERENSI

[1] B. Winduratna, A. Susanto dan R. Hidayat, "Penyandi Persepsi Isyarat Audio Berdasar pada Model," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. V, no. 3, pp. 213-221, 2016.

[2] C. Steinruecken, "Lossless Data Compression," University of Cambridge, Cambridge, 2014.

[3] P. Fenwick, "Burrows–Wheeler Compression: Principles and Reflections," *Theoretical Computer Science*, vol. CCCLXXXIX, no. 3, pp. 200-219, 2007.

[4] Y. Rathore, M. K. Ahirwar dan R. Pandey, "A Brief Study of Data Compression Algorithms," *International Journal of Computer Science and Information Security*, vol. XI, no. 10, pp. 86-94, 2013.

[5] H. Al-Bahadili dan S. M. Hussain, "An adaptive character wordlength algorithm for data compression," *Computers and Mathematics with Applications*, vol. LV, no. 6, p. 1250–1256, 2008.

[6] S. David , *Data Compression The Complete Reference*, 4th penyunt., London: Springer–Verlag, 2006.

[7] I. M. A. D. Suarjaya, "A New Algorithm for Data Compression Optimization," *International Journal of Advanced Computer Science and Applications*, vol. III, no. 8, pp. 14-17, 2012.

[8] S. Ambadekar, K. Gandhi, J. Nagaria dan R. Shah, "Advanced Data Compression Using J-bit Algorithm," *International Journal of Science and Research*, vol. IV, no. 3, pp. 1366-1368, 2015.

[9] N. Sharma, J. Kaur dan N. Kaur, "A Review on various Lossless Text Data Compression Techniques," *Research Cell: An International Journal of Engineering Sciences*, vol. XII, no. 2, pp. 58-63, 2014.

[10] A. T. Sadiq, M. G. Duaimi dan R. S. Ali, "Large Dataset Compression Approach Using Intelligent Technique," *Journal of Advanced Computer Science and Technology Research*, vol. III, no. 1, pp. 1-20, 2013.

[11] T. Singla dan R. Kumar, "ECG Signal Monitored Under Various Compression Techniques and Transmission Environments - A Survey Approach," *International Journal of Applied Research in Computing*, vol. II, no. 4, pp. 69-74, 2014.

[12] S. E. Adewumi, "Character Analysis Scheme for Compressing Text Files," *International Journal of Computer Theory and Engineering*, vol. VII, no. 5, pp. 362-365, 2015.

[13] J. Abel, "Post BWT Stages of The Burrows-Wheeler Compression Algorithm," *Software Practice and Experience*, vol. XL, no. 9, p. 751–777, 2010.

[14] M. Burrows dan D. J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," Digital Systems Research Center Report, Palo Alto, California, 1994.

[15] S. Chatterjee, "Block Based Data Compression Using Burrows-Wheeler Transform," Department Of Information Technology Jadavpur

University, Kolkata-India, 2013.

[16] S. Mantaci, A. Restivo, G. Rosone dan M. Sciortino, "An extension of the Burrows–Wheeler Transform," *Theoretical Computer Science*, vol. CCCLXXXIX, no. 3, p. 298–312, November 2007.

[17] B. Balkenhol , S. Kurtz dan Y. M. Shtarkov, "Data Compression Conference, 1999. Proceedings. DCC '99," *Modifications of the Burrows and Wheeler data compression algorithm*, pp. 188-197, March 1999.

[18] E. Syahrul, J. Dubois dan V. Vajnovszki, "Combinatorial Transforms : Application in Lossless Image Compression," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. XIV, no. 2, pp. 129-147, 2011.

[19] S. Shanmugasundaram dan R. Lourdusamy, "A Comparative Study Of Text Compression Algorithms," *International Journal of Wisdom Based Computing*, vol. I, no. 3, pp. 68-76, 2011.

[20] L. Sasilal dan V. K. Govindan, "Arithmetic Coding-A Reliable Implementation," *International Journal of Computer Applications*, vol. LXXIII, no. 7, pp. 1-5, 2013.

[21] R. Dorrigiv, . A. López-Ortiz dan J. I. Munro, "An Application of Self-Organizing Data Structures to Data Compression," dalam *International Symposium on Experimental Algorithms*, Berlin Heidelberg, 2009.

[22] J. Chen, J. Zhou dan K.-W. Wong, "A Modified Chaos-Based Joint Compression and Encryption Scheme," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. LVIII, no. 2, pp. 110 - 114, 2011.

[23] S. Shanmugasundaram dan R. Lourdusamy, "IIDBE: A Lossless Text Transform for Better Compression," *International Journal of Wisdom Based Computing*, vol. I, no. 2, pp. 1-6, 2011.