Novos Algoritmos para a Compressão de Sequências Parcialmente Comutativas

Andresso da Silva, Francisco Marcos de Assis

***Resumo*— Neste artigo, são apresentados dois novos métodos para a compressão de sequências definidas em alfabetos com ordem parcial. Os métodos utilizam conceitos dos monoides de comutatividade e da forma normal de Foata associada com o LZ78 para proporcionar maiores taxas de compressão. São obtidos alguns limitantes para a complexidade de sequências. Os métodos propostos apresentam ganho na compressão de sequências nos exemplos considerados quando comparados o Sequitur comutativo e com o LZ78.**

***Palavras-Chave*— Compressão, Ordem Parcial, Forma Normal de Foata, LZ78, Monoide de Comutatividade.**

***Abstract*— In this paper, two new methods for the compression of sequences defined in partial order alphabets are presented. The methods use concepts of commutativity monoids and Foata normal form associated with LZ78 to provide higher compression rates. Some bounds for the sequence complexity are obtained. The proposed methods present gain in the compression of sequences in the considered examples when compared the commutative Sequitur and the LZ78.**

***Keywords*— Compression, Partial Order, Foata Normal Form, LZ78, Commutative Monoid.**

I. Introdução

Os métodos mais populares de compressão são baseados em alfabetos com ordem total [1], [2], [3], [4]. O mesmo pode ser observado em técnicas mais atuais de compressão [4], [5], [6], [7], [8], [9], [10], [11], [12]. Tais abordagens são mais apropriadas para comprimir sequências associadas aos sistemas sequenciais. Entretanto, é notável o aumento do número de sistemas concorrentes (*e.g.,* computadores, celulares, redes de computadores, GPU). Em sistemas onde há ordem parcial entre eventos, como nos concorrentes, técnicas de compressão que levam em consideração essa característica são mais apropriados, podendo levar a maiores taxas de compressão [13].

Alur *et al*. [14] foram os primeiros a apresentar o problema de como realizar a compressão de sequências definidas em um alfabeto parcialmente comutativo. Neste contexto, esses autores propuseram quatro algoritmos para compressão sem perdas que consideram tais alfabetos. Um dos algoritmos usa o Sequitur [4] como inspiração, chamado, doravante, de Sequitur comutativo. Eles obtiveram resultados experimentais que indicam melhorias na compressão. Apesar de fornecer resultados melhores que alguns dos algoritmos de compressão de ordem total, os algoritmos de Alur *et al*. [14] não são ótimos. Desta forma, não é garantido que alcancem a máxima taxa de compressão, principalmente o que se baseia Sequitur, um algoritmo sub-ótimo nesse sentido.

Savari [13] propôs utilizar a forma normal de Foata (FNF) [17] seguida de um método de compressão sem perdas. Na FNF, os símbolos que comutam são agrupados em blocos chamados de fatores de Foata. Esse agrupamento faz surgir padrões que favorecem o processo de compressão. A ideia descrita por Savari é mapear uma sequência em sua forma normal e substituir os fatores de Foata por elementos de um alfabeto. Com isso, espera-se obter uma representação mais compacta da sequência original. Entretanto, Savari não fornece limitantes e nem desenvolve mais profundamente a ideia de compressão usando a forma normal de Foata.

Reznik [15], [16] propôs um método de codificação de conjuntos de palavras não-ordenados, mostrando que a abordagem proposta por ele requer $\log(m!)$ menos bits do que uma abordagem de ordem total, na qual m é a cardinalidade do conjunto de palavras. A principal limitação dos trabalhos de Reznik [15], [16] é que se aplicam somente quando todos os símbolos podem comutar, sendo um caso particular da ordem parcial, assim como a ordem total também o é.

No presente trabalho, são apresentados dois novos métodos de compressão que consideram a ordem parcial, usando como inspiração as ideias de Savari [13]. O primeiro método é a formalização do esquema de compressão usando a forma normal de Foata apresentado em [13], seguido substituição dos fatores mais frequentes e da aplicação do LZ78. A formalização também inclui a definição dos limitantes superiores para as complexidades das sequências. O segundo método mapeia a sequência e em seguida aplica o LZ78. Os resultados para os dois métodos propostos são comparados com o Sequitur comutativo de [14] e com o LZ78.

Este artigo está organizado da seguinte maneira: na Seção II, são apresentados os conceitos fundamentais relacionados aos monoides de comutatividade e o algoritmo de Lempel-Ziv LZ78. Na Seção III, são apresentados os novos métodos de compressão propostos. Na Seção IV, são apresentados os resultados obtidos utilizando esses novos métodos. Por fim, na Seção V, são apresentadas as conclusões.

II. Fundamentação

A. Monoides de Comutatividade

Seja Σ um alfabeto finito e Σ^* seja o conjunto de todas as palavras finitas formadas pelos elementos de Σ, incluindo a palavra vazia ε. O conjunto Σ^* é chamado de monoide livre gerado por Σ. Além disso, seja Σ^n o conjunto de todas as palavras de comprimento n definidas no alfabeto Σ. No decorrer deste artigo, palavra e sequência são usadas como sinônimos.

Andresso da Silva, Departamento de Engenharia Elétrica, UFCG, Campina Grande-PB, e-mail: andresso.silva@ee.ufcg.edu.br; Francisco Marcos de Assis, Departamento de Engenharia Elétrica, UFCG, Campina Grande-PB, e-mail: fmarcos@dee.ufcg.edu.br; Esse artigo foi parcialmente financiado pelo CNPq e pela CAPES.

Seja $G = (V, E)$ um grafo simples não-orientado chamado *grafo de comutatividade*, no qual V é o conjunto de vértices e E é o conjunto de arestas. Por grafo simples, entende-se que não existe aresta saindo e chegando no mesmo vértice, além de que não possui arestas múltiplas. Os elementos de V estão associados a um alfabeto finito Σ por meio de uma função bijetiva $\lambda : V \mapsto \Sigma$.

Dois símbolos de Σ comutam quando há duas palavras $\mathbf{u} = \mathbf{l}xy\mathbf{m}$ e $\mathbf{v} = \mathbf{l}yx\mathbf{m}$ que representam execução equivalente de sistemas concorrentes, com $\mathbf{l}, \mathbf{m} \in \Sigma^*$ e $x, y \in \Sigma$. A relação de comutatividade parcial induz uma ordem parcial entre as palavras definidas no monoide. Esses tipos de palavras são conhecidos como traços de Mazurkiewicz [18].

Se os símbolos $\lambda(x), \lambda(y) \in \Sigma$ comutam, então os vértices $x, y \in V$ estão conectados por uma aresta, $(x, y) \in E$. Se dois vértices estão conectados por uma aresta, diz-se que eles são adjacentes. O complemento do grafo de comutatividade, $\overline{G} = (V, \overline{E})$, é chamado de grafo de não-comutatividade, no qual os vértices conectados por uma aresta estão associados aos símbolos que *não* comutam.

Se os símbolos $\lambda(x)$ e $\lambda(y)$ comutam, então a relação é representada por $xy \equiv_G yx$. Se eles não comutam, a relação é representada por $xy \not\equiv_G yx$. Outra maneira de representar isso é por meio de $xy \equiv_G yx \Leftrightarrow (x, y) \in E$. Por exemplo, se o *grafo de comutatividade* for $a - b - c$, então a comuta com b e b comuta com c, mas a não comuta com c. Portanto, nesse caso, $ab \equiv_G ba$, $bc \equiv_G cb$ e $ac \not\equiv_G ca$. Desta forma, pode-se definir o monoide de comutatividade.

Definição 1 (Monoide de Comutatividade): O monoide de comutatividade $\mathcal{M}(\Sigma, G) = \Sigma^* \backslash \equiv_G$ é o quociente do monoide livre Σ^* pela relação de congruência $\equiv_G$.

Duas palavras $\mathbf{u}, \mathbf{v} \in \mathcal{M}(\Sigma, G)$ são equivalentes de acordo com a relação $\equiv_G$ se for possível obter uma a partir da outra trocando as posições dos símbolos consecutivos que comutam. Seja $\mathcal{E}_G(\mathbf{u})$ o conjunto de palavras equivalentes a uma palavra $\mathbf{u} \in \mathcal{M}(\Sigma, G)$ de acordo com a relação $\equiv_G$. O conjunto $\mathcal{E}_G(\mathbf{u})$ é chamado de classe de equivalência de $\mathbf{u}$. Se duas palavras pertencem à mesma classe de equivalência, diz-se que elas são congruentes.

Considere, como exemplo, o grafo de não-comutatividade $\overline{G}$ apresentado na Fig. 1. Desta forma, o alfabeto é $\Sigma = \{a, b, c, d, e\}$. As palavras $\mathbf{u} = abcde$, $\mathbf{v} = cabed$ e $\mathbf{w} = adbec$ estão definidas em Σ^5. A classe de equivalência de $\mathbf{u}$ é igual a $\mathcal{E}_G(abcde)$ = $\{abcde, abced, acbed, acbde, cabde, cabed, acbed, acebd, caebd\}$. A palavra $\mathbf{u}$ é congruente a $\mathbf{v}$, mas não é congruente a $\mathbf{w}$.

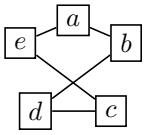


Fig. 1: Grafo de não-comutatividade $\overline{G}$.

Corolário 1 (Forma Normal de Foata): Seja uma palavra $\mathbf{u} \in \mathcal{M}(\Sigma, G)$. A forma normal de $\mathbf{u}$ é definida como uma decomposição única em fatores $\mathbf{u} = \mathbf{u}_1 \cdots \mathbf{u}_p$, em que os fatores $\mathbf{u}_i$ respeitam as seguintes propriedades:

1) cada $\mathbf{u}_i$ é uma palavra não-vazia em que as letras comutam entre si; e
2) se uma letra a de $\mathbf{u}_i$ não aparece em $\mathbf{u}_{i+1}$, então existe uma letra b de $\mathbf{u}_{i+1}$ que não comuta com a.

A forma normal de Foata (FNF) pode ser obtida por meio do algoritmo apresentado por Perrin [19] ou usando processo indutivo descrito:

1) se $\mathbf{u}$ é uma letra, então $\mathbf{u}$ já está na forma normal; e
2) se $\mathbf{u} = \mathbf{u}_1\mathbf{u}_2 \cdots \mathbf{u}_p$, então a forma normal de $\mathbf{u}a$, $a \in \Sigma$, é $\mathbf{u}_1\mathbf{u}_2 \cdots \mathbf{u}_p a$ se a não comuta com $\mathbf{u}_p$; ou $\mathbf{u}_1\mathbf{u}_2 \cdots \mathbf{u}'_i \cdots \mathbf{u}_p$, em que $\mathbf{u}'_i = \mathbf{u}_i a$ e a comuta com $\mathbf{u}_p, \mathbf{u}_{p-1}, \ldots, \mathbf{u}_i$, mas não com $\mathbf{u}_{i-1}$.

Considerando, mais uma vez como exemplo, o grafo de não-comutatividade da Fig. 1 e uma palavra $\mathbf{u} = abcbcdea$. Fazendo o processo indutivo, inicialmente $\mathbf{u} = a$, então já está na forma normal, sendo representado por $\mathbf{u} = (a)$. Adicionando b, como b não comuta com $\mathbf{u}_1$, então $\mathbf{u} = (a)(b)$. Adicionando agora c, tem-se que c comuta com b e com a, então $\mathbf{u} = (ac)(b)$. Continuando o procedimento, chega-se a $\mathbf{u} = (ac)(bc)(be)(ad)$.

Algumas considerações a respeito da forma normal de Foata (FNF) são que todas as palavras de uma classe de equivalência possuem a mesma forma normal e que os fatores $\mathbf{u}_i$ correspondem a um clique do grafo de comutatividade do monoide $\mathcal{M}(\Sigma, G)$.

B. LZ78

Um dos algoritmos da família Lempel-Ziv (LZ) foi apresentado em 1978 [2] e é conhecido como LZ78. O LZ78 é um algoritmo de compressão ótimo e sem perdas que, deterministicamente, gera um dicionário a partir de uma sequência de entrada. A compressão é obtida por meio da decomposição dessa sequência em frases (subsequências) e codificando-as. As frases são as menores sub-sequências ainda não observadas ao percorrer a sequência da esquerda para a direita.

Seja Σ um alfabeto com ordem total. Uma sequência $\mathbf{u} \in \Sigma^n$ é decomposta em frases m tais como $\mathbf{u} = \mathbf{s}_1\mathbf{s}_2 \ldots \mathbf{s}_m$. O dicionário é formado por m tuplas e cada uma delas codifica uma frase $\mathbf{s}_i, i \in [1, m]$. A codificação de uma frase $\mathbf{s}_i$ é obtida sabendo que $\mathbf{s}_i$ é composta de uma frase $\mathbf{s}_\kappa$ observada anteriormente (*i.e.*, $i > \kappa$) concatenada com um elemento $\sigma \in \Sigma$, ou ainda $\mathbf{s}_i = \mathbf{s}_\kappa\sigma$.

A complexidade de uma sequência $\mathbf{u}$ é então definida como o número de frases $C_{LZ78}(\mathbf{u}) = m$ em que ela é decomposta. Considere a sequência $\mathbf{u} = abbabbbaababaa$, como um exemplo. Assim, $\mathbf{u}$ será decomposta em $\mathbf{u} = (a)(b)(ba)(bb)(ab)(bba)(aba)(baa)$, tendo por complexidade $C_{LZ78}(S) = 8$.

III. Novos Métodos de Compressão Considerando Ordem Parcial

A. Compressão da Forma Normal de Foata (CFNF) + LZ78

O esquema de compressão proposto nessa seção tira proveito das propriedades da FNF para realizar a compressão de sequências utilizando um dicionário que é formado usando os cliques como regras. No Algoritmo 1 é apresentado o esquema

Algoritmo 1: Pseudo-código do CFNF.

```
Entrada: Sequência u
Saída: Gramática de Q = (S, P)
P ← ∅; Conjunto de regras
S ← ε; Regra principal
Decomposição de u em u_1 u_2 ··· u_p;
i ← 1 para cada fator u_i faça
    se u_i ∈ P e |u_i| > 1 então
        Adiciona regra ρ_i ← u_i a P;
        i ← i + 1;
    senão
        S ← S + ρ(u_i);
    fim
fim
```

de compressão. A função $\rho(\cdot)$ retorna o símbolo utilizado para representar a nova regra e ε é a palavra vazia.

Cabe observar que esse algoritmo é mais eficiente quando os cliques mais frequentes na forma normal possuem tamanhos maiores que 2. Para o caso de um grafo de comutatividade em que nenhum símbolo comuta, então, a compressão obtida seria nula.

Definição 2 (Complexidade de um dicionário): A complexidade de uma dicionário gerado a partir de uma sequência $\mathbf{u}$ é definida como

$$C(\mathbf{u}) = |S| + \sum_{\rho_i} C(\rho_i \leftarrow \mathbf{u}_i), \tag{1}$$

em que $|S|$ é o número de elementos em S e $C(\rho_i \leftarrow \mathbf{u}_i) = |\mathbf{u}_i|$

Como exemplo, considera-se um clique $\mathbf{u}_i$ de tamanho 5. Então, para uma sequência $\mathbf{u} = \mathbf{u}_i\mathbf{u}_i\mathbf{u}_i$, o dicionário gerado seria $P = \{\rho_1 \leftarrow \mathbf{u}_i\}$ e $S = 111$, tendo um complexidade igual a $5 + 3 = 8$, quase a metade de $|\mathbf{u}| = 15$.

Teorema 1: (Limite Superior para a complexidade do CFNF) Seja um grafo de comutatividade G, para todas as sequências $\mathbf{u} \in \mathcal{M}(\Sigma, G)$, a complexidades é limitada por

$$C_{CFNF}(\mathbf{u}, G) \leq p + |\mathcal{C}| - |\Sigma| \tag{2}$$

Demonstração: A complexidade do CFNF é formado pela complexidade da regra principal S e do conjunto das regras que associam um clique a um símbolo. A regra principal S terá no máximo p elementos, correspondendo ao número de fatores da forma normal de Foata. As regras que associam um símbolo a um clique terá no máximo um número de elementos igual à cardinalidade $|\mathcal{C}|$ do conjunto de cliques do grafo G. Como o CFNF só considera regras associadas a cliques de tamanho maior ou igual a 2, então os cliques de tamanho 1 (os vértices) são desconsiderados, correspondendo à subtração de $|\Sigma|$. ■

O método de compressão proposto, combina a compressão da forma normal de Foata (CFNF) com o LZ78. Nesse caso, inicialmente aplica-se o CFNF à sequência inicial, gerando o dicionário com S e P, e, posteriormente, aplica-se o LZ78 na regra principal S obtida. Considerando o exemplo de $P = \{\rho_1 \leftarrow \mathbf{u}_i\}$ e $S = 111$, aplica-se o LZ78 em $S = (1)(11)$, obtendo-se uma complexidade $5 + 2 = 7$, menor que a do CFNF aplicado sozinho.

Desta forma, pode-se obter o limite superior da complexidade desse método como apresentado no Teorema 2.

Teorema 2: (Limite Superior para a complexidade de CFNF + LZ78) Para todas as palavras $\mathbf{u} \in \mathcal{M}(\Sigma, G), \beta = |\Sigma|$, seja p o número de fatores da FNF de $\mathbf{u} = \mathbf{u}_1\mathbf{u}_2\cdots\mathbf{u}_p$, então a complexidade de $\mathbf{u}$ é dada por

$$C_{CFNF+LZ78}(\mathbf{u}, G) \leq \frac{p}{(1-\epsilon_p)\log_\beta p} + |\mathcal{C}| - |\Sigma| \tag{3}$$

em que $|\mathcal{C}|$ é o conjunto dos cliques de G e $\epsilon_p = 2\frac{1+\log_\beta \log_\beta(\beta p)}{\log_\beta(p)}$.

Demonstração: O primeiro fator da Eq. 3 é resultado direto da complexidade do LZ78 dado pelo Teorema 2 de [21, p.77]. Os outros fatores correspondem à complexidade das regras que associam os cliques aos símbolos de regras, de forma semelhante à demonstração do Teorema 1. ■

B. FNF + LZ78

Símbolos semelhantes contíguos favorece o processo de compressão, como pode ser observado em métodos como o Burrows-Wheeler [20, Cap.8]. Usando as características fornecidas pela FNF, o segundo método de compressão proposto utiliza a obtenção da FNF de uma sequência seguida de uma aplicação do LZ78 na FNF. Outros métodos de compressão poderiam ser utilizados no lugar do LZ78, mas foi escolhido esse último por ser ótimo.

IV. Resultados Experimentais

Nessa seção, são apresentados alguns resultados experimentais obtidos aplicando a comutação parcial para obter esquemas mais eficientes de compressão. São comparados os algoritmos Sequitur comutativo proposto por Alur *et al.* [14], o LZ78 [2] e os algoritmos propostos nesse artigo, FNF + LZ78 e CFNF + LZ78. A medida considerada para a complexidade do Sequitur comutativo foi a mesma da Definição 2.

Todas as sequências foram geradas considerando uma distribuição equiprovável dos símbolos do alfabeto e que os símbolos são independentes. Para cada comprimento considerado, foram geradas 15 sequências nas quais os algoritmos foram aplicados. Os valores de complexidades apresentados nas figuras são uma média aritmética considerando as 15 sequências.

A. Grafos de Comutatividade Aleatórios

Na Fig. 2, são apresentados dois grafos de comutatividade gerados aleatoriamente, o primeiro com 5 vértices (ver Fig. 2a) e o segundo com 6 vértices (ver Fig. 2b).

(a) G_1. (b) G_2.

Fig. 2: Grafos de comutatividade G_1 e G_2.

Os dois grafos da Fig. 2, possuem o mesmo número de cliques de tamanho 3 e um número parecido de cliques de

tamanho 1 e 2. Entretanto, como o número de vértices de G_2 é maior, então é esperado maiores complexidades.

Na Fig. 3 são apresentados os valores de complexidade obtidos considerando o grafo G_1 e na Fig. 4 são apresentados os valores de complexidade obtidos para o grafo G_2.

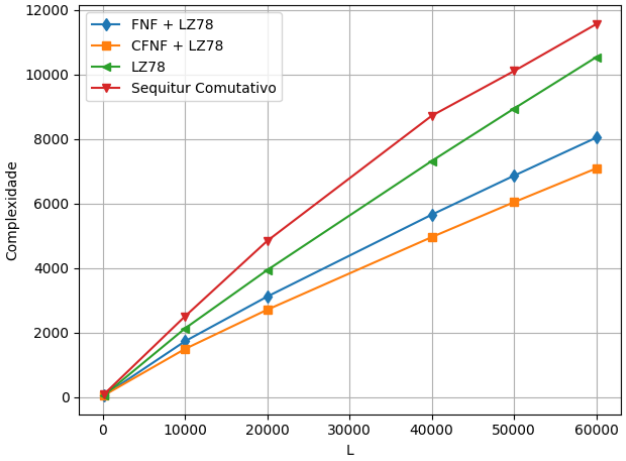


Fig. 3: Complexidades das sequências de comprimento L considerando G_1.

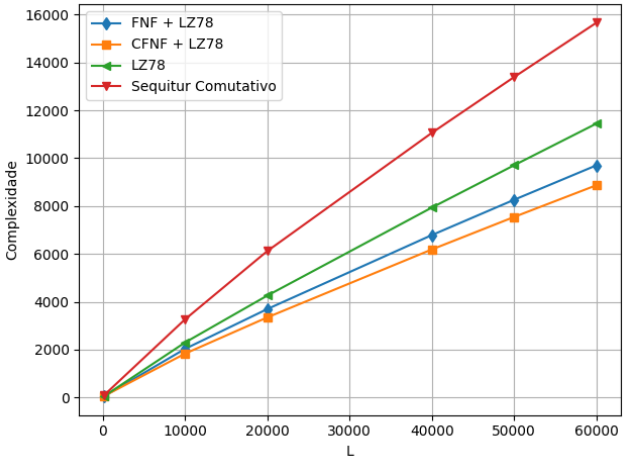


Fig. 4: Complexidades das sequências de comprimento L considerando G_2.

Considerando os resultados para G_1, o algoritmo que obteve melhor desempenho foi o CFNF + LZ78, seguido do FNF + LZ78, LZ78 e Sequitur comutativo, respectivamente. O Sequitur, originalmente, não é um algoritmo ótimo. O Sequitur comutativo também não o é, obtendo uma complexidade acima do LZ78, que por sua vez é um algoritmo de compressão para alfabetos totalmente ordenados. Os algoritmos CFNF + LZ78 e FNF + LZ78 apresentam um ganho considerável na compressão em comparação ao LZ78. É esperado que, conforme o grafo de comutatividade se aproxime de um grafo completo, melhor o desempenho dos dois.

Para o grafo G_2, o resultado foi semelhante ao observado para o G_1. Como o G_2 tem uma menor relação entre o número de arestas e o número de arestas possíveis em relação ao G_1, $|E(G_1)|/\binom{|V(G_1)|}{2} > |E(G_2)|/\binom{|V(G_2)|}{2}$, era esperado que a redução na complexidade em relação ao LZ78 fosse um pouco menor, pois está mais distante do grafo completo de 6 vértices. O algoritmo que apresentou menor compressão foi o Sequitur comutativo, assim como no G_1.

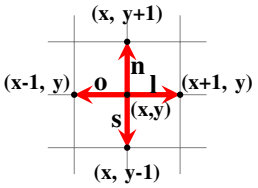
O CFNF + LZ78 foi o algoritmo que apresentou maior taxa de compressão para os dois casos considerados, oferecendo vantagem considerável em relação ao LZ78. O FNF + LZ78 teve um desempenho consideravelmente melhor ao LZ78, estando mais próximo ao CFNF + LZ78. O FNF + LZ78 pode ser uma boa alternativa quando o tempo é um fator crítico, pois apresenta uma taxa de compressão melhor que o Sequitur comutativo e o LZ78 e não possui procedimentos além da geração da forma normal e a aplicação do LZ78.

B. Comandos para Robôs

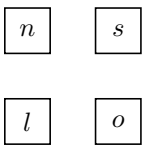
Considera-se um robô que move-se em um plano cartesiano bidimensional e recebe uma sequência de comandos que definem sua trajetória de um ponto A para um ponto B. Sendo seu ponto inicial (x, y), os comandos são definidos como:

- n: incrementa em um sua coordenada no eixo y;
- s: decrementa em um sua coordenada no eixo y;
- l: incrementa em um sua coordenada no eixo x;
- o: decrementa em um sua coordenada no eixo x.

A representação gráfica dos comandos é apresentada na Fig. 5a. Após alguma experimentação, é possível observar que a ordem dos comandos em uma determinada sequência não modifica o comportamento do robô ir do ponto A ao ponto B. Em termos de comutatividade, pode-se representar as relações como $no \equiv_{G_3} on$, $nl \equiv_{G_3} ln$, $sl \equiv_{G_3} ls$, $ns \equiv_{G_3} sn$, $so \equiv_{G_3} os$, $ol \equiv_{G_3} lo$. Consequentemente, o grafo de não-comutatividade será vazio (sem arestas), como apresentado na Fig. 5b. ’



(a) Representação dos comandos.



(b) $\overline{G}_3$.

Fig. 5: Comandos e Grafo de não-comutatividade.

Na Fig. 6 são apresentados os valores de complexidade obtidos pelos algoritmos quando utilizados em sequências aleatórias de comprimento L indicado. Como nos casos anteriores, os pontos correspondem a uma média da complexidade considerando 15 sequências aleatórias de mesmo comprimento L. Como nesse caso $|E(G_3)|/\binom{|V(G_3)|}{2} = 1$, é esperado um ganho considerável na compressão do CFNF + LZ78 e do FNF + LZ78.

Ao gerar as sequências de considerando os comandos como variáveis independentes e identicamente distribuídas, as formas normais de Foata das sequências geradas serão como $\mathbf{u} = (lnos)(lnos)\cdots(lnos)(\mathbf{u}_i)\cdots(\mathbf{u}_p)$. Para $L = |\mathbf{u}| =$

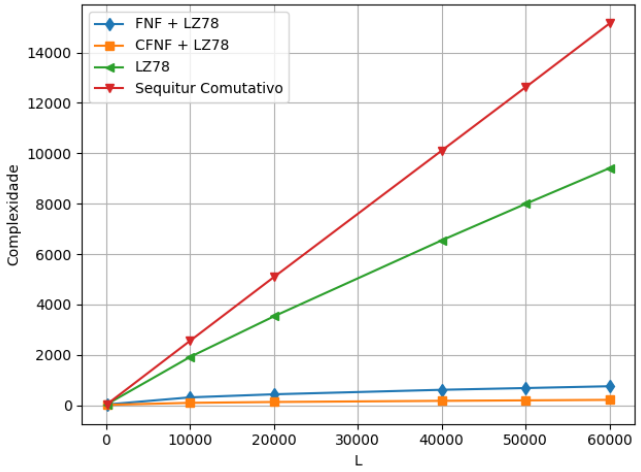


Fig. 6: Complexidades das sequências de comandos de comprimento L.

40000, o valor esperado do número de fatores é $p = 10000$ e $|\mathcal{C}| - |\Sigma| = 0$, então o limite superior esperado para a complexidade com $\beta = |\Sigma| = 4$ para o CFNF + LZ78 será $C(\mathbf{u}, G_3) \leq \frac{10000}{(1-\epsilon_{10000})\log_4 10000} \approx 3325,32$.

Mesmo o limite superior é menor que o valor de complexidade do LZ78 observado para o mesmo comprimento da sequência (ver Fig. 6). Mais uma vez, o CFNF + LZ78 foi o que obteve melhor desempenho. O FNF + LZ78 apresentou uma complexidade consideravelmente menor do que a do LZ78, reforçando que ele pode ser uma alternativa quando o fator tempo for crítico.

Os exemplos apresentados dão indicações dos possíveis ganhos na compressão ao considerar a ordem parcial entre símbolos.

V. Conclusões

Foram apresentados novos métodos para a compressão de sequências definidas em alfabetos com ordem parcial. Os algoritmos propostos utilizam a forma normal de Foata (FNF) para tirar proveito da ordem parcial das sequências e, assim, obter maiores compressões em relação às técnicas convencionais de compressão que consideram alfabetos com ordem total. O primeiro método, utiliza a chamada de compressão da forma normal de Foata (CFNF), que gera um dicionário a partir da FNF, seguido da utilização do LZ78. O segundo método utiliza diretamente a FNF seguido de uma compressão usando a versão LZ78 do algoritmo de Lempel-Ziv.

Os resultados obtidos indicam que os métodos propostos nesse artigo fornecem vantagens em relação ao LZ78 e ao o Sequitur comutativo. Foram obtidos ganhos consideráveis na compressão de sequências e os ganhos se tornaram mais evidentes conforme o número de elementos que comutam no alfabeto aumenta. Em todos os casos, o primeiro método proposto foi o que obteve maiores taxas de compressão, seguido do segundo método. Com as menores taxas de compressão, ficaram o Sequitur comutativo, seguido do LZ78.

Como atividades futuras, propõe-se a determinação teórica do número de fatores da forma normal de Foata e, assim, a determinação mais fácil dos limitantes superiores para as complexidades dos algoritmos propostos.

Agradecimentos

Este estudo foi financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código 88887.641221/2021-00 e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) sob o número de contrato 302704/2013-2 (PQ).

Referências

[1] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[2] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[3] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[4] C. Nevill-Manning and I. Witten, "Linear-time, incremental hierarchy inference for compression," in *Proceedings DCC '97. Data Compression Conference*, pp. 3–11, 1997.

[5] J. Arz and J. Fischer, "Lz-compressed string dictionaries," in *2014 Data Compression Conference*, pp. 322–331, 2014.

[6] K. J. Conrad and P. R. Wilson, "Grammatical ziv-lempel compression: Achieving ppm-class text compression ratios with lz-class decompression speed," in *2016 Data Compression Conference (DCC)*, pp. 586–586, 2016.

[7] W. Liu, F. Mei, C. Wang, M. O'Neill, and E. E. Swartzlander, "Data compression device based on modified lz4 algorithm," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 1, pp. 110–117, 2018.

[8] T. K. Tajul, S. R. Bhuiyan, and A. Habib, "Enhancement of lzap (lempel ziv all prefixes) compression algorithm," in *2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT)*, pp. 69–73, 2018.

[9] C. Y. Wu, "Improved lz77 compression," in *2021 Data Compression Conference (DCC)*, pp. 377–377, 2021.

[10] "An optimal text compression algorithm based on frequent pattern mining," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 3.

[11] M. Brehm and M. Thomas, "An efficient lossless compression algorithm for trajectories of atom positions and volumetric data," *Journal of Chemical Information and Modeling*, vol. 58, no. 10, pp. 2092–2107, 2018.

[12] G. Shrividhiya, K. S. Srujana, S. N. Kashyap, and C. Gururaj, "Robust data compression algorithm utilizing lzw framework based on huffman technique," in *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pp. 234–237, 2021.

[13] S. Savari, "Compression of words over a partially commutative alphabet," *IEEE Transactions on Information Theory*, vol. 50, no. 7, pp. 1425–1441, 2004.

[14] R. Alur, S. Chaudhuri, K. Etessami, S. Guha, and M. Yannakakis, "Compression of partially ordered strings," in *CONCUR 2003 - Concurrency Theory*, pp. 42–56, Springer Berlin Heidelberg, 2003.

[15] Y. Reznik, "Coding of sets of words," in *2011 Data Compression Conference*, pp. 43–52, 2011.

[16] Y. Reznik, "Codes for unordered sets of words," in *2011 IEEE International Symposium on Information Theory Proceedings*, pp. 1322–1326, 2011.

[17] C. Pierre and D. Foata, *Problèmes combinatoires de commutation et réarrangements*. Lecture notes in mathematics, Berlin Heidelberg New York: Springer-Verlag, 1969.

[18] A. Mazurkiewicz, "Concurrent program schemes and their interpretations," *DAIMI Report Series*, vol. 6, Jul. 1977.

[19] D. Perrin, "Words over a partially commutative alphabet," in *Combinatorial Algorithms on Words. NATO ASI Series (Series F: Computer and Systems Sciences)*, vol. 12, pp. 329–340, Springer, Berlin, Heidelberg, 1985.

[20] D. Salomon, *Data Compression: The Complete Reference*. second ed., 2004.

[21] A. Lempel and J. Ziv, "On the complexity of finite sequences," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 75–81, 1976.