Indexação sob Demanda para a Compressão Referencial de Ficheiros de ADN

Fernando Alves[1], Vinicius Cogo[1] e Alysson Bessani[1]

[1] LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Resumo A compressão de ficheiros de ADN é uma prática comum em diversos fluxos de trabalhos em bioinformática. Por sua vez, a compressão referencial é uma das suas abordagens mais recentes e eficientes, que se baseia na alta probabilidade de quaisquer dois organismos da mesma espécie possuírem um alto grau de semelhança genética. Esta permite armazenar somente as diferenças entre cada organismo e um genoma de referência (que representa essa espécie) ao invés de guardarmos todos os nucleótidos de todos os organismos de um estudo ou colecção. Uma das suas principais vantagens é o alto rácio de compressão sem a perda de informações em genomas já alinhados. Porém, algumas ferramentas que a implementam gastam um tempo substancial na indexação de todo o genoma de referência logo no início da sua execução. Neste trabalho, propomos uma optimização aos algoritmos de compressão referencial de ficheiros de ADN, que consiste na redução do volume de dados utilizados na indexação da referência. Sucintamente, iniciamos a indexação de pequenas porções da referência apenas quando não conseguimos comprimir os dados através do uso de outras heurísticas mais simples. Tal optimização, que visa estritamente reduzir o tempo de compressão, é capaz de trazer ganhos de até uma ordem de magnitude em comparação a outros algoritmos comummente utilizados para compressão referencial de ficheiros de ADN.

Palavras-chave: Compressão de genomas, Compressão referencial, Indexação

1 Introdução

O custo do sequenciamento de ADN teve uma queda exponencial nos últimos anos [1], facto que está directamente relacionado ao aumento, nas mesmas proporções, da quantidade de genomas sequenciados em todo o globo [2]. A impressionante marca de $1000 pelo sequenciamento de um genoma humano completo foi alcançada no início de 2014 pela *Illumina Inc.* com a plataforma *HiSeq X Ten* [3,4]. Adicionalmente, as expectativas são de que tais custos continuem a cair nos próximos anos, a fim de se atingir as metas dos $100 [5,6] e eventualmente dos $30 [7] por genoma. Tal tendência tem uma implicação directa no aumento da complexidade de gestão da imensa quantidade de dados que está a ser gerada todos os dias.

No contexto informático, a compressão é um processo que visa diminuir o tamanho de um conjunto de dados, a fim de permitir que estes sejam transmitidos e armazenados eficientemente. Este processo tem e continuará a ter um papel muito importante nos fluxos de trabalhos em bioinformática, uma vez que atenua a diferença evolutiva das

tecnologias de sequenciamento de ADN e as de armazenamento. Porém, sabe-se de longa data que os algoritmos tradicionais de compressão de dados são ineficientes para a compressão de sequências genéticas [8].

As especificidades do ADN vão para além do facto do seu alfabeto poder ser frequentemente reduzido a um conjunto de apenas quatro letras (A, C, G e T). Os algoritmos de compressão para este tipo de sequência podem ser do tipo horizontal, quando se utiliza apenas a sequência a ser comprimida como base de conhecimento, ou do tipo vertical, quando se utiliza um conjunto de sequências biológicas para auxiliar a compressão de cada uma das sequências [9]. Outra forma de classificar os algoritmos de compressão é com base na sua abordagem, onde existem os seguintes grupos [9,10]: codificação binária ingénua, métodos baseados em dicionários, métodos estatísticos, e métodos de compressão referencial.

A compressão referencial de sequências genéticas [10,11] é a abordagem que vamos tomar como ponto de partida para este trabalho, uma vez que é a que tem obtido os melhores resultados práticos em termos de tempo e rácio de compressão. Porém, como esta abordagem só foi recentemente aplicada a genomas, a mesma ainda possui oportunidades de optimização de desempenho. Neste trabalho, apresentamos uma optimização que visa reduzir o tempo de compressão, em algoritmos que são considerados o estado-da-arte, em até uma ordem de magnitude através da minimização de uma das etapas internas do algoritmo, a qual é responsável pela indexação total do genoma de referência no início da execução.

O presente artigo segue dividido em cinco secções, sendo a primeira delas esta introdução. A segunda aborda em maior profundidade o tópico de compressão referencial de ficheiros de ADN, bem como apresenta uma das mais recentes soluções para tal, que servirá como base de comparação para este trabalho. A terceira contém uma discussão das alterações necessárias aos algoritmos de compressão referencial e detalha as contribuições deste artigo. Uma avaliação experimental para comprovar a viabilidade das principais ideias deste artigo é apresentada na quarta secção. Por fim, na quinta secção são apresentadas as nossas considerações finais, bem como alguns tópicos a serem trabalhados num futuro próximo.

2 Compressão Referencial de Ficheiros de ADN

A ideia básica da compressão referencial é, dado como entrada um genoma a ser comprimido e um genoma de referência, armazenar no ficheiro de saída apenas as diferenças do genoma em questão comparado à referência [11].[1] Esta abordagem tem como base a grande probabilidade de genomas de organismos da mesma espécie possuírem uma

[1] Durante todo este trabalho, sempre que mencionarmos o termo *genoma de referência*, estamos a nos referir ao genoma de referência humano fornecido pelo projecto 1000 Genomes Project [12] versão 37, disponível em http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/reference/human_g1k_v37.fasta.gz. Porém, pode ser empregado qualquer genoma de referência para a espécie humana ou outras, sempre a cargo do utilizador. Apenas chamamos à atenção que cada genoma deve ser comprimido e descomprimido com a mesma referência.

grande semelhança genética. Por isso, tanto é melhor o rácio de compressão quanto mais representativo for o genoma de referência para a espécie com que se está a trabalhar.

A compressão referencial é uma das formas mais eficazes para reduzir o tamanho de ficheiros de ADN, visto que os algoritmos que a implementam são capazes de obter rácios de compressão de até $400 : 1$ (quatrocentos para um), ou seja, o tamanho dos dados comprimidos chega a ser $400\times$ menor que o original. Em comparação, as implementações das outras abordagens são capazes de comprimir genomas apenas ao rácio máximo de $8 : 1$ [10].

Dentre as diversas implementações de compressão referencial para genomas [10], destacamos três delas, nomeadamente: RLZ [13], GDC [14] e FRESCO [15]. A primeira é uma das ferramentas pioneiras em compressão referencial e utiliza arrays de sufixos para a indexação da referência. A segunda é a ferramenta que possuiu o melhor desempenho em termos de tempo e rácio de compressão durante algum tempo, e utiliza uma pré-selecção da referência para cada conjunto de genomas. A terceira é a ferramenta que se tornou a mais rápida na compressão referencial ao superar em uma ordem de magnitude o tempo de compressão das outras. Porém este resultado corresponde somente a compressão propriamente dita, uma vez que não levou em consideração o tempo necessário para preparar as estruturas de dados de cada algoritmo. Esta ferramenta utiliza uma estrutura de dados chamada tabela de *k-mers*, a qual é semelhante a uma tabela *hash* com uma lista de valores para cada chave. O FRESCO apresenta-se como uma solução interessante para compressão de genomas em diversas espécies, assim como para a sua execução em sistemas com diferentes capacidades de memória [16]. Tendo em vista todos esses pontos, daremos continuidade a este trabalho a partir do uso da ferramenta FRESCO, a qual terá seu funcionamento discutido ainda nesta secção.

O FRESCO é uma ferramenta de compressão referencial sem perdas (*lossless*) para genomas que já se encontram alinhados e estão armazenados nos formatos RAW (somente os nucleótidos) ou FASTA (nucleótidos e linhas de comentários). Ela foi desenvolvida em C++ e é fornecida publicamente como um software de código aberto [17]. O fluxo de compressão do FRESCO consiste em comparar um genoma a ser comprimido a um genoma de referência, a fim de obter todas as suas diferenças para serem codificadas e armazenadas. A descompressão trata-se do processo inverso, onde se obtém o genoma completo original a partir do genoma de referência e das diferenças armazenadas no processo de compressão. A fim de analisarmos o funcionamento interno da ferramenta, podemos dividir o seu algoritmo de compressão em três etapas principais, que estão representadas nos círculos da Figura 1: (1) indexação, (2) compressão propriamente dita e (3) codificação.

(1) Indexação. A etapa de indexação é a primeira a ocorrer no fluxo de compressão do FRESCO. Esta recebe o genoma de referência como entrada e parte-o em entradas contíguas de tamanho K. A Figura 2 providência uma referência visual de como a indexação é executada. Uma estrutura de dados semelhante a uma tabela *hash* é inicializada e para cada um desses segmentos (chamados *k-mer*) é gerado um *hash* que será usado como a chave que o representa. A especificidade desta estrutura de dados é que para cada chave é possível armazenar uma lista de valores, que são os índices da referência onde cada segmento de tamanho K foi encontrado. Esta tabela é mantida em memória e é parte fundamental da etapa de compressão. Resumindo, o FRESCO

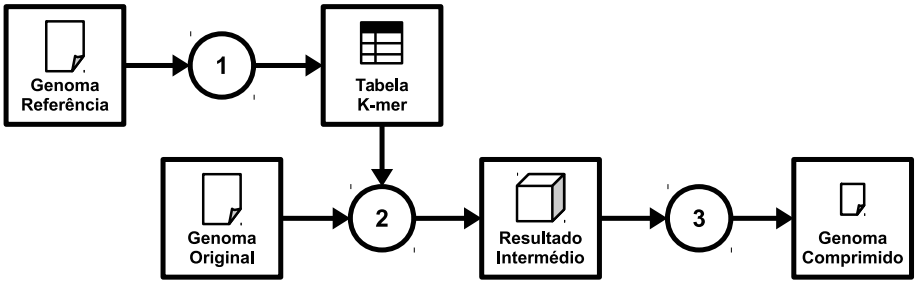

Figura 1. Modelo conceptual de compressão referencial através da ferramenta FRESCO. (1) Indexação do genoma de referência. (2) Compressão do genoma original, usando a referência indexada. (3) Codificação do resultado intermédio a fim de gerar o ficheiro final comprimido.

indexa toda a referência nesta estrutura de dados, a qual permite que as buscas sejam deterministas, ou seja, se uma correspondência existir temos a certeza de encontrar o(s) seu(s) índice(s) na referência.

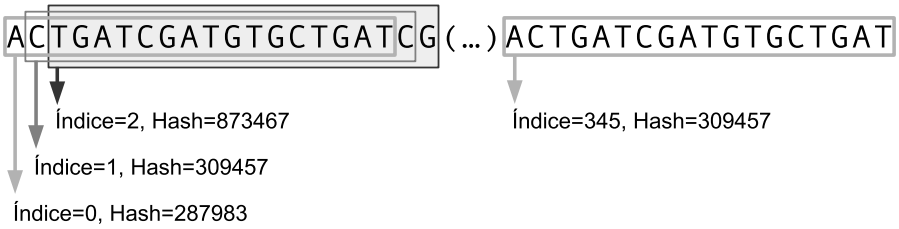

Tabela K-mer (K=18)

Hash	Índices
287983	{0}
	...
309457	{1,345}
	...
873467	{2}
	...

Figura 2. Modelo de indexação usando uma tabela de *k-mer*. A primeira coluna contém o *hash* associado a um ou vários índices, os quais são armazenados na segunda coluna.

(2) Compressão. Esta segunda etapa, que é a compressão propriamente dita, recebe como entrada o genoma a ser comprimido e a estrutura de dados criada com todo o genoma de referência. O genoma a ser comprimido é também partido em segmentos de tamanho K. Para cada um desses segmentos é gerado um *hash*, o qual é utilizado para fazer a busca (*lookup*) na estrutura de dados. O resultado dessa busca retorna uma lista

de índices da referência onde o segmento existe. Um apontador é utilizado para que o algoritmo saiba em que região do genoma se está a operar, dessa forma o algoritmo selecciona o índice mais próximo ao valor deste apontador. Após a correspondência ser encontrada, o algoritmo ainda procura se ela pode ser estendida por mais alguns nucleótidos até encontrar alguma diferença. A cada diferença, é adicionada uma entrada no resultado intermédio do ficheiro comprimido, assim como é adicionado o(s) nucleótido(s) divergente(s). A busca recomeça com o próximo segmento de tamanho K e o processo é repetido até que todo o genoma seja comprimido. O resultado intermédio é composto por índices para as partes que são iguais entre os ficheiros de entrada (o genoma a ser comprimido e o genoma de referência), e os nucleótidos diferentes. Note que tais correspondências e nucleótidos divergentes encontram-se sempre intercalados, independentemente do número de nucleótidos de diferença presentes em cada segmento.

(3) Codificação. Por fim, os bytes do resultado intermédio são novamente comprimidos. O FRESCO usa GZIP para comprimir o resultado intermédio e produzir o ficheiro de saída.

A Figura 3 apresenta os resultados de uma primeira análise do tempo total de compressão de um genoma (dividido pelos seus cromossomas) com o FRESCO original. Cada barra apresenta a parcela de cada uma das três etapas explicadas anteriormente no tempo total. A componente “Outros”contém os tempos de leitura e escrita de ficheiros, a etapa de codificação e outras operações, como por exemplo, a inicialização de variáveis e estruturas de dados. Esta figura permite-nos observar que a maior parte do tempo total de compressão é gasto na etapa de indexação. As etapas de compressão propriamente dita e codificação representam parcelas muito menores que a de indexação nos resultados apresentados.

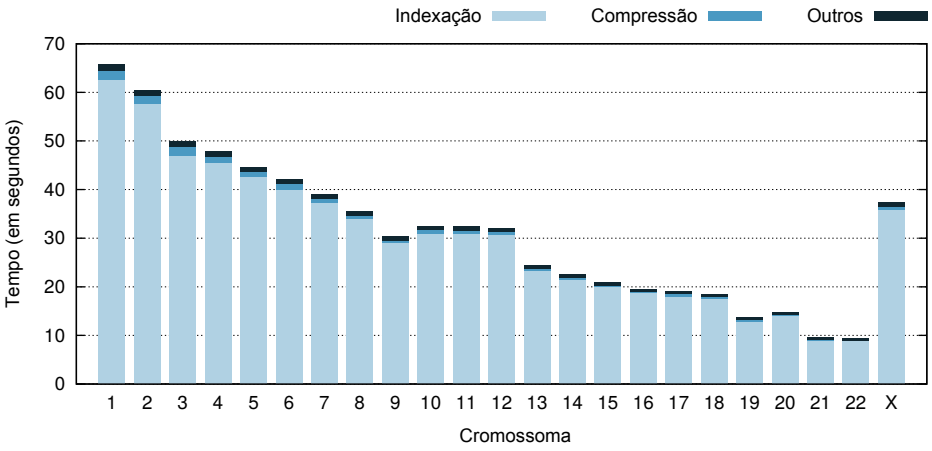

Figura 3. Tempo total de compressão do FRESCO original por cromossoma e as suas etapas internas.

É justamente a etapa de indexação que pretendemos optimizar, visto que é onde se concentra a maior parte do tempo gasto do algoritmo original do FRESCO. A ideia é reduzir drasticamente essa parcela de tempo. Na próxima secção vamos apresentar as possibilidades de optimização existentes para essa etapa.

Por fim, ainda existe a implementação do processo inverso à compressão, chamada de descompressão. Começa-se com a descodificação do GZIP, que vai traduzir o ficheiro comprimido para o resultado intermédio com todos os índices e nucleótidos divergentes entre o genoma original e o de referência. Para cada segmento do ficheiro comprimido, escrevem-se os nucleótidos da referência se o segmento for composto por um índice, e escrevem-se os nucleótidos divergentes se o segmento for composto por estes. Ao final, volta-se a obter o genoma original, uma vez que ele é composto pelas porções que são iguais ao genoma de referência, acrescido dos nucleótidos que compõem as variações genéticas do indivíduo. Não é nosso foco avaliar o tempo gasto na descompressão, uma vez que não existe indexação neste processo, logo não podemos optimizá-lo com a nossa abordagem.

3 Indexação sob Demanda

Conforme discutido na secção anterior, a etapa de indexação é a responsável pela maior parcela do tempo gasto em todo o processo de compressão. Na Figura 3 é possível confirmar que mais de 95% do tempo total de compressão é gasto pela indexação, enquanto que pouco mais de 2% do tempo é gasto na compressão propriamente dita e pouco menos de 3% em outras operações (nas quais a codificação está incluída). Nesta secção vamos discutir como pretendemos minimizar a parcela da etapa de indexação.

Dentre as diversas possibilidades de redução da etapa de indexação, duas delas aparecem como principais candidatas: (1) a utilização de paralelismo e (2) a substituição da indexação total do genoma de referência por pequenas indexações parciais quando necessárias. No primeiro caso, a construção da estrutura de dados na etapa de indexação pode ser paralelizada através do uso de sincronização na escrita dos índices relativos a um mesmo *hash*. Porém, o ganho será limitado pelas capacidades da máquina utilizada, e obteremos uma estrutura de dados enorme, que podemos não utilizar na sua totalidade. No segundo caso, não existe indexação do genoma de referência no início da execução, e inicia-se o processo de compressão com heurísticas simples de comparação de segmentos. Esta segunda opção é a que seguiremos no decorrer deste artigo, à qual demos o nome *indexação sob demanda*. Note que as duas alternativas apresentadas não são mutuamente exclusivas.

A *indexação sob demanda* baseia-se no facto de genomas de organismos da mesma espécie possuírem grande semelhança genética (por exemplo, 99,5% de semelhança nos humanos). A maioria das diferenças entre genomas da mesma espécie são consideradas pequenas, sendo maioritariamente variações de um único nucleótido (chamadas *single nucleotide polymorphisms*, ou SNPs), ou de poucos nucleótidos em pequenas substituições, inserções e eliminações. Procuras deterministas são capazes de facilmente identificar a maioria destas variações através da comparação directa de segmentos. Isto gera a expectativa de que a indexação total do genoma de referência pode não ser necessária. Daí que a nossa abordagem (descrita de seguida) remova a etapa

de indexação presente na Figura 1. No entanto, sabemos que este mecanismo pode ter limitações quando as variações genéticas são grandes substituições, inserções ou eliminações. Nestes casos, a indexação é a solução mais eficiente, e indexa-se pequenas porções do genoma de referência próximas ao índice em que se está a trabalhar.

O nosso mecanismo de compressão baseia-se na baixa frequência destas grandes diferenças, e é descrito a seguir. A compressão é feita partindo o ficheiro em blocos ou com o ficheiro inteiro em memória (se este for menor que o tamanho do bloco). Tanto a referência como o ficheiro a comprimir são tratados desta maneira. Cada bloco é comprimido em separado, por isso não existe ligação entre blocos, mesmo que sejam do mesmo ficheiro. Se a compressão por blocos é usada, apenas um ficheiro é gerado, que contém os blocos comprimidos. Para encontrar correspondências, o algoritmo segue os seguintes passos:

1. Testamos deterministicamente se existe um SNP.
2. Se um SNP não é encontrado, então faz-se uma procura local de Δ pares-base, que cobre a maioria das pequenas diferenças. Δ é igual a $6 \times K$, onde K é o tamanho de *k-mer* usado.
3. Se nenhuma correspondência é encontrada, então indexamos uma parte da referência e procuramos a posição actual do genoma original na tabela.

Se após estes passos não é encontrada nenhuma correspondência, então escreve-se um par-base do ficheiro de input e repete-se o processo. Descobrimos que é mais eficiente indexar apenas um pouco da referência (até 200 pares-base) de cada vez pois se existe uma grande inserção no ficheiro de input, estaríamos a indexar a referência sem haver utilidade para tal. Relembramos que não existe indexação inicial, o nosso algoritmo começa directamente na etapa de compressão. Depois de uma correspondência ser encontrada (de qualquer tipo), procuramos de forma determinística o comprimento máximo dessa correspondência (para aumentar o rácio de compressão), e depois escrevemos o resultado. O ficheiro de input é processado para encontrar correspondências com a referência até todo o ficheiro ser comparado.

Implementámos em Java o algoritmo descrito nesta secção, juntamente com a ideia de indexação sob demanda e as partes do algoritmo original do FRESCO que permaneceram inalteradas. Sabemos que se implementássemos a nossa ferramenta em C++ provavelmente os tempos de execução seriam menores que os que serão apresentados a seguir, mas optamos pelo Java devido à sua interoperabilidade e facilidade de integração com as outras ferramentas desenvolvidas pelo nosso grupo. Por fim, vamos analisar se as modificações propostas são suficientes para atingirmos o objectivo traçado inicialmente: reduzir drasticamente a parcela de tempo gasto na indexação. As alterações devem reduzir a quantidade de dados que obrigatoriamente precisamos indexar. Na Figura 4 são apresentadas as percentagens do genoma de referência que são indexadas na abordagem do FRESCO original e na nossa abordagem. Note que o FRESCO sempre indexa todo o genoma de referência, logo a sua percentagem nesta figura é sempre de 100%.

Os resultados apresentados na Figura 4 indicam que definitivamente não é necessário indexarmos todo o genoma de referência, visto conseguirmos comprimir os genomas usando os mecanismos do FRESCO original modificados pela indexação sob

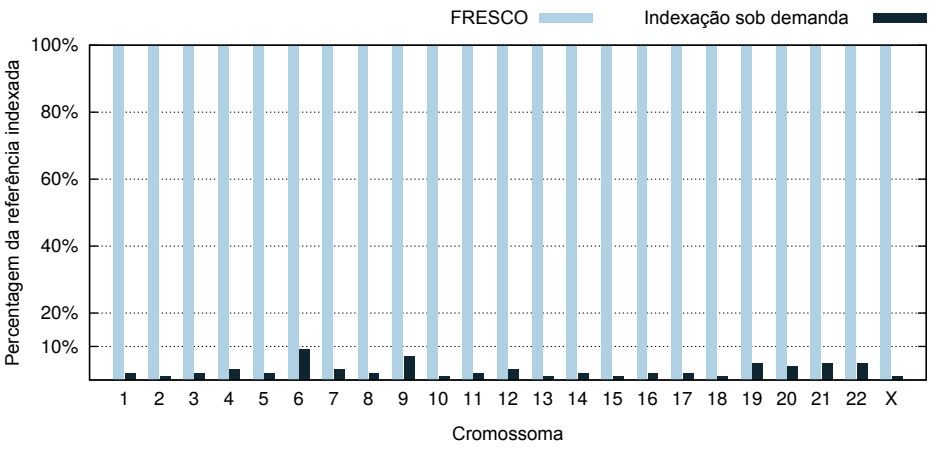

Figura 4. Percentagens da referência que são efectivamente indexadas pelo algoritmo original do FRESCO e pela nossa abordagem com a indexação sob demanda.

demanda. Além disso, indica também que mais de 90% da referência é indexada sem necessidade, ou seja, que precisamos indexar menos de 10% da referência para resolver os casos mais complicados que não são solucionados pelas heurísticas apresentadas nesta secção. Desta forma fechamos a discussão da viabilidade da nossa optimização com base na redução da necessidade de indexação, o que na próxima secção será traduzida para valores de tempo de execução, os quais esperamos reduzir proporcionalmente aos valores apresentados na Figura 4.

4 Avaliação Experimental

A presente avaliação tem como foco principal analisar o tempo total de compressão de um genoma (dividido pelos seus cromossomas) e comparar o desempenho do algoritmo original do FRESCO com o da nossa implementação (com a indexação sob demanda). A expectativa é de que o tempo de compressão seja reduzido proporcionalmente às reduções da quantidade de dados do genoma de referência a serem indexados durante todo o processo (ver a Fig. 4). Através de uma breve interpretação desta imagem, definimos como meta sermos capazes de comprimir cada cromossoma pelo menos $10\times$ mais rápido que o FRESCO original.

A metodologia utilizada neste teste consiste em lançar 10 execuções da compressão em ambas implementações (FRESCO original e a nossa abordagem) para cada genoma que utilizamos e calcular a média de todas as execuções. Seleccionámos 10 genomas aleatoriamente dos 1092 disponíveis no 1000 Genomes Project [12]. Os 10 genomas obtidos foram aqueles identificados pelos números NA19788, HG00173, NA20810, HG00339, HG00619, NA20339, HG00475, HG01390, NA19449 e NA12546. O genoma de referência utilizado é o mesmo apresentado na nota de rodapé da Secção 2.

O nosso ambiente de teste conta com 1 (uma) máquina Dell PowerEdge R420 equipada com dois processadores Intel Xeon E5520 (2.27GHz), 32GB de memória RAM

(1066 MHz) e um disco rígido de 146GB (15000 RPM). O sistema operativo instalado na máquina é o Ubuntu Server 10.04 (64-bit, kernel 2.6.32-21-server). A versão do C++ utilizada é a 4.4.3 (para a implementação original do FRESCO), e a do Java é a 1.7.0 (para a nossa implementação). A versão do FRESCO utilizada nesta avaliação é a 0.2 [17]. Utilizamos a ferramenta *time* do próprio sistema operativo para obter as marcações de tempo de cada execução completa da compressão, uma vez que este é o tempo na óptica do utilizador.

A Figura 5 contém os resultados dos testes especificados anteriormente. Os resultados da nossa primeira implementação da abordagem com indexação sob demanda já apresentam uma melhoria significativa em relação ao tempo gasto para comprimir cada um dos cromossomas. Por exemplo, enquanto o FRESCO precisa de mais de 1 minuto para comprimir o cromossoma 1 (o maior dos cromossomas humanos), a nossa implementação precisa apenas de cerca de 5 segundos para concluir a mesma tarefa. No outro extremo, o FRESCO leva cerca de 9 segundos para comprimir o cromossoma 22 (o menor dos cromossomas humanos), enquanto que a nossa ferramenta não chega a utilizar 2 segundos.

Os resultados da nossa abordagem têm como média o tempo de 3,7 segundos, enquanto a média com o FRESCO é de 31,4 segundos. Separadamente, os resultados representam melhorias entre $5\times$ e $12\times$ em comparação ao algoritmo original do FRESCO, que indexa todo o genoma de referência logo no início da sua execução.

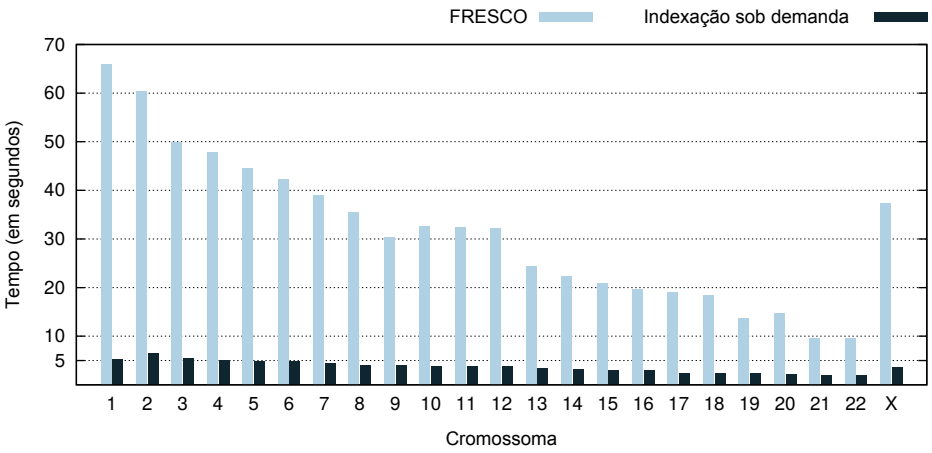

Figura 5. Tempo total de compressão (em segundos) por cromossoma ao utilizarmos o FRESCO original e a nossa abordagem com indexação sob demanda.

Note que este trabalho não pretende colocar em causa os tempos de compressão do FRESCO, uma vez que a etapa de compressão propriamente dita representa menos de 3% do tempo total de execução (ver Figura 3). O que queremos chamar a atenção é que não existe a necessidade de indexar todo o genoma de referência logo no início da execução para realizarmos a compressão referencial. Devido ao alto grau de semelhança

genética entre genomas de uma mesma espécie, podemos substituir a maior parte das indexações por heurísticas simples, como as apresentadas na Secção 3.

Esta redução de tempo tem um impacto maior nos fluxos de trabalho curtos (em que utilizador interage) do que em fluxos de várias horas. Visto que muitos passos dos fluxos de trabalho em bioinformática são lançados em tempo de execução (não havendo estado guardado entre execuções de cada fluxo), uma redução no tempo de execução da ordem de magnitude que obtivemos poderá ser benéfica para os utilizadores de fluxos de trabalho curtos que utilizam compressão/descompressão. No futuro iremos medir os tempos de transferência de um genoma sem compressão, e do fluxo compressão-transferência-descompressão (usando a nossa ferramenta na compressão/descompressão). Esperamos que este teste seja positivo para a nossa ferramenta, sendo mais um ponto que valida a nossa abordagem de compressão com indexação sob demanda.

A abordagem do FRESCO é mais vantajosa se houver um fluxo de trabalho específico em que uma grande quantidade de genomas é comprimido e transferido ou armazenado. Neste caso a indexação completa da referência trará ganhos no tempo de execução, pois a tabela de indexação seria gerada uma vez e usada para todos os genomas. Sabemos também que a descompressão é uma parte importante destes fluxos, e o nosso trabalho procurará obter tempos de descompressão semelhantes ao do FRESCO de modo a manter a viabilidade completa da nossa ferramenta.

Por fim, os resultados obtidos com a primeira implementação da nossa abordagem permitem inferir a viabilidade das melhorias propostas neste artigo. Resta para as próximas etapas analisar e optimizar a solução para que esta tenha uma eficiência igual ou superior que a do FRESCO original em termos de rácio de compressão e a quantidade de memória utilizada pela ferramenta de compressão. Apesar de ainda não termos resultados conclusivos acerca destas propriedades, esperamos obter valores semelhantes visto que o conceito da etapa de compressão propriamente dita é semelhante em ambos os algoritmos. Esperamos ainda reduzir a quantidade de memória utilizada pela ferramenta de compressão, devido ao facto de também termos reduzido a quantidade de dados a serem indexados.

5 Considerações Finais

Neste artigo apresentamos uma optimização viável aos algoritmos de compressão referencial de ficheiros de ADN. Chamamos-lhe indexação sob demanda, uma vez que indexamos pequenas porções do genoma de referência apenas quando não conseguimos realizar a compressão com outras heurísticas mais simples. Enquanto algumas soluções existentes indexam todo o genoma de referência logo no início da execução, comprovamos que mais de 90% deste processo é desnecessário. Nós aplicámos a nossa abordagem, a qual foi implementada em Java, ao algoritmo da ferramenta chamada FRESCO. A nossa optimização traz ganhos de desempenho de até uma ordem de magnitude em termos do tempo de execução. Mesmo sendo estes resultados preliminares, já podem ser considerados significativos visto que o FRESCO é uma ferramenta no estado-da-arte e que ainda existe a possibilidade de se inserir paralelismo no processo de compressão. Tão importante quanto os ganhos obtidos nesta primeira fase com a nossa abordagem será obter uma eficiência igual ou superior que a do FRESCO ori-

ginal no rácio de compressão e na quantidade de memória utilizada pela ferramenta de compressão. Nós pretendemos analisar e apresentar novos resultados num futuro próximo com as comparações e melhorias mencionadas, assim como continuar a melhorar a nossa implementação e aplicá-la a casos reais de armazenamento e transmissão de dados a fim de comprovar a praticabilidade da nossa abordagem.

Por fim, note novamente que este trabalho não pretende colocar em causa os tempos de compressão do FRESCO. Queremos apresentar a ideia de que não existe necessidade de indexar todo o genoma de referência para realizarmos a compressão referencial. Isto significa que, assim como foi aplicada no FRESCO, outras soluções existentes também podem beneficiar da nossa abordagem.

Agradecimentos

Agradecemos a todas as pessoas que de uma forma ou de outra contribuíram com este trabalho, em especial ao Ulf Leser (HU-Berlin) e ao Sebastian Wandelt (HU-Berlin) pelas discussões e críticas construtivas a respeito dos tópicos abordados neste artigo. Este trabalho foi parcialmente financiado pela Fundação para a Ciência e a Tecnologia, através do LaSIGE (PEst-OE/EEI/UI0408/2014), e pelo Sétimo Programa Quadro (FP7) da Comissão Europeia, através do projecto BiobankCloud (ICT-317871).

Referências

1. Wetterstrand, K.: DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). Disponível em: www.genome.gov/sequencingcosts. Acedido em 20/06/2014.
2. Kahn, S.D.: On the future of genomic data. Science(Washington) **331**(6018) (2011) 728–729
3. Illumina Inc.: Illumina introduces the HiSeq X Ten sequencing system. Disponível em: http://investor.illumina.com/phoenix.zhtml?c=121127&p=irol-newsArticle. Acedido em 20/06/2014.
4. Hayden, E.C.: Is the $1,000 genome for real? Nature News (2014)
5. Singer, E.: The $100 genome. Disponível em: http://www.technologyreview.com/news/409919/the-100-genome/. Acedido em 20/06/2014. (2008)
6. McMorrow, D.: The $100 genome: Implications for the DoD. Technical report, DTIC Document (2010)
7. Singer, E.: The $30 genome. Disponível em: http://www.technologyreview.com/news/419258/the-30-genome/. Acedido em 20/06/2014. (2010)
8. Grumbach, S., Tahi, F.: A new challenge for compression algorithms: genetic sequences. Information Processing & Management **30**(6) (1994) 875–886
9. Giancarlo, R., Scaturro, D., Utro, F.: Textual data compression in computational biology: a synopsis. Bioinformatics **25**(13) (2009) 1575–1586
10. Wandelt, S., Bux, M., Leser, U.: Trends in genome compression. Current Bioinformatics (2013)
11. Brandon, M.C., Wallace, D.C., Baldi, P.: Data structures and compression algorithms for genomic sequence data. Bioinformatics **25**(14) (2009) 1731–1738
12. The 1000 Genomes Project Consortium: 1000 Genomes Project: A deep catalog of human genetic variation. Disponível em http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/. Acedido em 20/06/2014.

13. Kuruppu, S., Puglisi, S.J., Zobel, J.: Optimized relative Lempel-Ziv compression of genomes. In: Proceedings of the Thirty-Fourth Australasian Computer Science Conference - Volume 113. ACSC '11, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2011) 91–98
14. Deorowicz, S., Grabowski, S.: Robust relative compression of genomes with random access. Bioinformatics **27**(21) (2011) 2979–2986
15. Wandelt, S., Leser, U.: FRESCO: Referential compression of highly similar sequences. Computational Biology and Bioinformatics, IEEE/ACM Transactions on **10**(5) (Sept 2013) 1275–1288
16. Wandelt, S., Leser, U.: Adaptive efficient compression of genomes. Algorithms for Molecular Biology **7**(30) (2012) 1–9
17. Wandelt, S., Leser, U.: FRESCO: A framework for referential sequence compression. Disponível em https://github.com/hubsw/FRESCO. Acedido em 20/06/2014.