UNIVERSIDADE DO MINHO
Escola de Engenharia, Departamento de Informática

Sérgio Alípio Domingues Deusdado

Análise e Compressão de Sequências Genómicas

Orientação Científica: Professor Doutor Paulo Martins de Carvalho

Braga, 2008

À minha família.

AGRADECIMENTOS

Todos os que passaram pela experiência de percorrer com seriedade o longo caminho de investigação conducente ao grau de Doutor sabem que, durante os anos envolvidos, para além da intensa carga de trabalho, tiveram de abdicar de muita coisa importante. Ainda assim, quando devotados à ciência e investigação, cremos como importante que, para além da nossa evolução académica, podemos contribuir modestamente para a acumulação de conhecimento de um povo, de um país ou do planeta. Nesse enquadramento, podemos tornar-nos merecedores dos benefícios que o conhecimento anteriormente desvendado nos proporcionou simplesmente pelo reconhecimento. A todas as mulheres e homens de ciência, que dedicaram as suas vidas ou parte delas à nobre tarefa de investigar e descobrir, devem pertencer as primeiras palavras de agradecimento.

Por toda uma vida de dedicação, quero agradecer à minha família pelo continuado apoio e incentivo. É neles que reside a minha confiança, a minha coragem e a minha fé. É com eles que faz sentido o que fiz, faço e farei. São a minha inspiração, especialmente o meu filho Tiago, e depositários da maioria dos meus agradecimentos. Faz todo o sentido agradecer-lhes pessoalmente. Assim, por ordem de aparição na minha vida, OBRIGADO Celina, Lázaro, Paulo, Leonel, Sandra e Tiago. Uma menção especial à minha esposa Sandra pela sua inspiradora capacidade de lutar e de amar.

Ao meu orientador científico, Prof. Doutor Paulo Carvalho, um grande e merecido agradecimento, pois acompanha-me desde longa data (há quase uma década!), guiando-me com mestria inicialmente no mestrado, agora no doutoramento e em diversas publicações científicas, estando consciente que a excelente relação de trabalho e amizade perdurará. Destaco e agradeço a sua qualidade científica e o bom ambiente de trabalho. Agradeço também a ajuda da Prof. Doutora Paola Bignone, investigadora do LRF Lymphoma Antigens Group, Universidade de Oxford, na revisão científica em matéria de biologia molecular.

A todos os meus amigos, que sempre acreditaram em mim e valorizam os meus esforços uma palavra de agradecimento. Agradeço igualmente ao programa PRODEP o apoio financeiro concedido.

Resumo

A informação dos códigos genéticos sequenciados é na actualidade, provavelmente, a fonte mais inspiradora para o estudo e avanço das teorias da informação e da codificação. Algoritmos eficientes para a sua compressão antevêm-se essenciais para a optimização do armazenamento e comunicação da informação genómica. A compressão de informação genómica é um caso particular da compressão de informação. A entropia das sequências de ADN é elevada, contudo variável. Ao nível intra-genómico é maior nas regiões codificantes e menor nas regiões não codificantes. Ao nível inter-genómico é maior nos seres procarióticos e menor nos eucarióticos. Na base da redução da entropia estão as regularidades que perfazem as regiões repetitivas do ADN. As regiões repetitivas compõem-se sobretudo de padrões aproximados, que incluem pontualmente mutações, delecções, inserções ou *gaps*. Os padrões exactos são menos relevantes e geralmente apresentam-se em numerosas repetições adjacentes. A redundância do ADN também tem manifestações estatísticas e probabilísticas. As redundâncias das sequências de ADN são a fonte de recursos de compressão, as grandes repetições indicam-se para a compressão substitucional com recurso a dicionário, enquanto que as evidências estatísticas e probabilísticas permitem modelar e predizer parcialmente a sucessão de símbolos (bases), utilizando compressores estatísticos para capitalizar esse potencial de compressão. Considerando a entropia máxima para o ADN, a sua codificação corresponde a 2 bits por base. Em média, os melhores compressores disponíveis, concebidos para a especificidade do ADN, alcançam os 1,7 bits/base, o que corresponde a uma taxa de compressão de apenas 15%, valor que é demonstrativo da dificuldade inerente.

O trabalho realizado corresponde a um *framework* de análise e compressão de sequências de ADN, cuja aplicação principal corresponde ao DNALight. O DNALight é uma solução híbrida para compressão de informação genómica baseada na cooperação de várias metodologias vocacionadas para absorver ocorrências das diferentes tipologias de redundâncias presentes nas cadeias de nucleótidos. De facto, a compressão não é possível sem análise. É na completa análise que reside a obtenção dos recursos que permitirão reduzir a entropia. Para a análise de sequências de ADN desenvolveram-se algoritmos inovadores para a pesquisa de padrões exactos (GRASPm) e aproximados

(SimSearch) que alcançam desempenhos que superam destacadamente o estado da arte. Estes algoritmos intervêm na primeira fase do DNALight que aproveita o potencial dos padrões mais representativos para a compressão substitucional baseada em dicionário de padrões exactos e aproximados. Para maximizar as captações de padrões, a pesquisa é exaustiva e efectuada multi-nível, ou seja, na sequência normal 5'-3', na complementar natural 3'-5', e também nas duas restantes complementares artificiais. Na segunda fase do DNALight, que procura fazer o aproveitamento das redundâncias desconsideradas pela captação da primeira fase, são construídos modelos probabilísticos de linguagem compactos com bases nas regiões menos repetitivas que transitam para esta fase, e que constituem o *input* para esta metodologia complementar. Em concorrência, os modelos geram predições sustentadas nas apreciações probabilísticas de modelos de linguagem globais e locais. As predições acertadas ou aproximadas permitem codificações mais económicas pois criam maior desequilíbrio no modelo probabilístico de codificação, beneficiando o desempenho da codificação aritmética que encerra o processo. O processo de descompressão é similar mas reverso ao descrito para a compressão. Os resultados experimentais colocam o DNALight como novo integrante do estado da arte em compressão de sequências de ADN, superando consistentemente, mas em pequena escala, os seus antecessores.

Abstract

Genetics is nowadays, probably, the most inspiring source for coding theory study and developments. Efficient compression algorithms are essential to optimise genomic data storage and communication. Genomic data compression is a particular case of data compression. The entropy present in DNA sequences is high, however variable. At intra-genomic level, it is higher in coding regions and lower in non-coding regions. At inter-genomic level, it is higher in the prokaryotes and lower in eukaryotes. DNA entropy reduction is achieved by coding more efficiently the repetitive regions of the ADN. Repetitive regions are mainly composed of inexact patterns. Patterns' errors are caused by biological processes and DNA dynamics including mutations, deletions, insertions or gaps. Exact patterns are less relevant and generally are presented in tandem repetitions. DNA redundancies have also statistical and probabilistic manifestations. The redundancies of DNA sequences are the most proficuous source of compression resources, the larger repetitions are indicated for substitucional compression based on a dictionary, whereas the statistical and probabilistic evidences allow to model and predict the succession of symbols (bases) in the sequence, using statistical compression to capitalize this compression potential. Considering the maximum DNA entropy, its codification cost corresponds to 2 bits per base. On average, the best available compressors, conceived accordingly DNA data specificities, reach 1,7 bits/base, which corresponds to a compression rate of only 15%, and this value is demonstrative of the inherent difficulty.

The developed work corresponds to a framework for the analysis and compression of DNA sequences, being DNALight the most representative application. DNALight is a hybrid solution for DNA compression based on the cooperative integration of complementary methodologies to absorb the different redundancies present in DNA sequences. In fact, compression is not possible without analysis. Gathering resources for compression relies mostly in analysis, and the emerged recurrences will allow to reduce the entropy. Innovative algorithms were developed for exact pattern-matching (GRASPm) and approximate and exact pattern discovery (SimSearch) and their performance notoriously surpasses the state of the art. These algorithms play an important role in the first phase of the DNALight to implement substitucional

compression based on dictionary of exact and approximated repeats. To maximize pattern recollection, the searching is performed multi-level, i.e., in normal sequence 5' - 3', in natural complementary sequence 3' - 5', and also in the two remaining artificial complementary sequences. In the second phase of DNALight, focused on taking advantage of the missed redundancies in the first phase, probabilistic language models are built based on the less repetitive regions as they constitute the input of this complementary methodology. In competition, the models generate predictions supported in the probabilistic analysis of global and local language models. Accurate or approximated predictions allow compact codifications as they provide a more disproportional probabilistic model for codification, benefiting the arithmetic coding performance that encloses the process. The decompression process is similar, but reverse when compared with compression. The experimental results place DNALight as a new constituent of the state of the art in DNA sequences compression, surpassing consistently, but in small scale, its predecessors.

Índice

Índice de Ilustrações

Figuras

Tabelas

LISTA DE ACRÓNIMOS E TERMINOLOGIA BIOINFORMÁTICA

ADN – Ácido Desoxirribonucleico

Alinhamento – O resultado da comparação da composição de duas ou mais sequências biológicas para determinar o seu grau de similaridade, homologia ou correlação

ARN – Ácido Ribonucleico

ARNm – Ácido Ribonucleico mensageiro

ARNt – Ácido Ribonucleico transportador

BLAST – Basic Local Alignment and Search Tool, a aplicação bioinformática mais popular, disponível on-line no NCBI e EMBL

Codão – Sequência de três bases adjacentes que codifica um amino-ácido

Codão START – Significa o início da codificação de uma proteína, marca um ORF

Codão STOP – Significa o fim da codificação de uma proteína

Dupleto – Segmento de 2 bases consecutivas, também designado por dinucleótido

EMBL – European Molecular Biology Laboratory

Exon – Região codificante do ADN nos seres eucariotas

FASTA – Fast All – Alinhamentos rápidos para toda a informação biológica, disponível on-line no NCBI e EMBL

Frame – Existem 3 frames (quadros de leitura) em cada ordem das sequências (normal e complementar), cada frame significa uma possibilidade de leitura de codões sucessivos e não sobrepostos depois de um ORF

Gap – Um conjunto de bases sucessivas intercaladas que perfazem a diferença entre duas sequências

GenBank – Base de dados de informação biológica de acesso público disponibilizada pelo NCBI

Gene – Um segmento de uma sequência de ADN que contém, nos exons que a compõem, informação para codificar uma proteína bem como a informação de regulação necessária

Genoma – A sequência de ADN completa de um ser vivo

Hit – Cada uma das ocorrências detectadas de uma seed ou padrão numa sequência

Indel – Insertion/Deletion, operações complementares de edição de sequências necessárias para lidar com padrões aproximados

Intron – Região não codificante do ADN, ocorre entre os exons

Motif – Uma característica reconhecida, identificada pela presença de um segmento específico, contida numa sequência genómica ou proteómica

NCBI – National Center for Biotechnology Information

Nucleótido – O mesmo que base, existindo quatro bases diferentes no alfabeto genómico: (A) Adenina; (C) Citosina; (G) Guanina e (T) Timina

ORF – Open Reading Frame, quadro aberto de leitura, situados entre o codão START e o codão STOP

Pattern-matching – Identificação de ocorrências de um determinado padrão conhecido numa sequência

PD – Programação Dinâmica

PH – PatternHunter, ferramenta para análise de sequências baseada em *seeds* espaçadas

Seed – (Semente) – Um conjunto de bases sucessivas usadas para detectar um foco de similaridade no alinhamento de sequências

Sensitivity – (Sensibilidade) – Mede a capacidade de detecção de similaridades por parte de um algoritmo de análise de similaridades, os algoritmos óptimos baseados em PD atingem 100% de sensibilidade e servem de referência para avaliar os algoritmos baseados em heurísticas

SSEARCH – Implementação mais popular do algoritmo de Smith-Waterman para alinhamento local óptimo, recorre a PD

Threshold – Um limite máximo definido para as discrepâncias admissíveis entre dois alinhamentos

Tradução – O processo de formação de uma proteína com base no ARN

Transcrição – O processo de formação do ARNm com base na expressão de um gene inscrito numa cadeia de ADN

UniProt – Universal Protein Resource, Base de dados mundial de sequências proteómicas

"Tenho pensamentos que, se conseguisse realizá-los e torná-los vivos, acrescentariam uma nova luz às estrelas, uma nova beleza ao mundo e um maior amor ao coração dos homens."

Fernando Pessoa

Capítulo 1. Introdução

O desenvolvimento das ciências é necessariamente cumulativo e osmótico. Primeiro formou-se uma base onde a matemática e as ciências da natureza preponderaram e, com estes conhecimentos inferiram-se princípios de exploração de muitas outras ciências, que por sua vez originaram a necessidade de investigar novas ciências. Neste contexto, não tardará muito até a Bioinformática produzir descendência. Contudo, a magna revolução científica ocorrerá, não pelo domínio das ciências de base mas, quando se lograr a sua automatização, tornando-as omnipotentes e omnipresentes, fazendo com que o desenvolvimento de novas extensões possa ser acelerado. Este estádio de base científica automatizada ainda está em construção, e sustenta-se com a maturidade da informática, vista como tecnologia de processamento e comunicação, e das suas aplicações. Dispor de poderio computacional, atingindo uma quase infinidade de cálculos e respectiva visualização 3D, permite-nos simular quase tudo, criar experiências espacio-temporais há pouco tempo julgadas impossíveis, e até criar novos mundos virtuais que mimetizam a própria evolução da vida, minimizando custos temporais, materiais ou éticos.

Em 1953, James Watson e Francis Crick determinaram a estrutura do ácido desoxirribonucleico (ADN) [1], e com isso germinaram uma nova extensão da ciência, a biologia molecular, que não parou de evoluir porque encontrou aliados poderosos, os computadores, e as suas capacidades de processamento e interligação, onde se estribou para a obtenção do seu marco maior, a sequenciação do genoma humano. Actualmente

somos detentores dos códigos da vida e de meios de computação poderosos. Assim, torna-se possível projectar novas proteínas e novas formas de vida que interferirão necessariamente com a de todos nós. Vamos subindo degraus na pirâmide evolucional das ciências e a bioinformática [2] assume-se cada vez mais como uma ciência de ponta, que assenta sobretudo nos espaços de intersecção entre a biologia molecular, a informática, as bases de dados, a matemática biológica e a estatística.

O ADN constitui o meio físico onde todas as propriedades dos organismos vivos são codificadas, um conjunto de quatro bases (Adenina, Citosina, Guanina e Timina) constitui o alfabeto para uma sequência de nucleótidos interligados entre si e entre os seus complementares numa estrutura de dupla hélice. A sub-cadeia …AACTGTTGTTGTTAGAA… poderá ser o exemplo de uma fracção de um "texto" que, dependendo da complexidade do organismo, poderá atingir os biliões de caracteres e que encerra o código genético de um indivíduo. Nem toda a extensão do ADN está vocacionada para a codificação de proteínas. A codificação ocorre em segmentos específicos, os genes [3], e dentro desses segmentos selectivamente em fracções não necessariamente adjacentes dessa sub-sequência de nucleótidos, os exões ou exons. Nos eucariotas, a maioria dos genes tem regulação individual, os mecanismos de regulação operam com base em informação associada aos genes igualmente inscrita no código genético, tais inscrições denominam-se genericamente por *motifs*.

A tarefa de sequenciar o código genético de todas as espécies, incluindo aquelas que já se extinguiram, implica armazenar uma quantidade de informação colossal. Em jeito de futurismo seria como começar a escrever o inventário genómico da futura arca de Noé que será necessário levar para colonizar planetas distantes, e note-se que a natureza levou muitos milhões de anos a produzir este património. Actualmente existem em bases de dados distribuídas por todo o mundo, *gigabytes* sem fim de informação correspondente a sequências de nucleótidos, que perfazem o ADN e ARN, bem como de sequências de aminoácidos, os elementos das proteínas. Esta proliferação em massa de informação genómica implica a necessidade de algoritmos de compressão que optimizem e racionalizem o seu armazenamento e comunicação, ao nível das sequências individualmente consideradas e das próprias bases de dados [4].

A análise das sequências genómicas, quer ao nível intrínseco, no tocante às suas regularidades internas, nomeadamente o conteúdo GC, segmentação e outras características, combinada com a análise extrínseca, que se incumbe de proceder a comparações com outras sequências de origens diversas, é fundamental para o avanço da genómica funcional, mas não se fica por aí. De facto, a análise das sequências de nucleótidos é tema de investigação de físicos, estatísticos, matemáticos, teóricos da informação e de uma pletora de disciplinas que concorrem para a descoberta dos segredos do ADN [5-8], considerando-o muito para além de um código que raramente denota a função de codificar proteínas.

Uma vez que a sequência de ADN é não aleatória, o conhecimento das suas especificidades e regularidades intrínsecas e filogenéticas, torna-se fundamental para atingir resultados satisfatórios na compactação dessa codificação. A compressão das sequências de nucleótidos torna-se assim um móbil científico que motiva uma comunidade cada vez maior de cientistas, que se digladiam com a enorme interdisciplinaridade da bioinformática, no intuito de contribuir para que o desenvolvimento de novo conhecimento, potenciador de novas conquistas numa faceta tão crítica das nossas vidas como é a saúde, floresça. Os algoritmos clássicos para a compressão universal de informação [9, 10] não obtêm resultados substantivos na compressão das sequências de ADN, em casos extremos o seu comprimento vê-se inclusivamente alargado face à codificação linear de 2 bits por base.

A compressão das sequências de ADN é uma tarefa altamente aliciante na área da compressão de informação, em boa medida pelas peculiaridades da composição e organização das sequências biológicas, nas quais os algoritmos clássicos de compressão, que evoluíram pelos e para os textos e imagens, falham na compressão da informação genómica. Como o volume de informação biológica, mormente aquele que é produto das sequenciações genómicas, se multiplica a um ritmo muito elevado, torna-se cada vez mais premente o desenvolvimento de algoritmos que permitam compactar esse volume de informação por forma a optimizar o seu armazenamento e comunicação em redes de dados.

Curiosamente, o estudo e o êxito da compressão do ADN tem por base as regularidades específicas e já conhecidas da composição da molécula, mas poderá eventualmente

revelar outras [11], dilatando o conhecimento que possuímos dessas regularidades, fazendo-as emergir como efeito secundário mas, de grande valia científica. Em concreto, através das técnicas de compressão podem inferir-se similaridades filogenéticas, determinar o nível de entropia de uma sequência, realizar a identificação e contabilização de padrões e *motifs*, etc.

Apesar de existirem actualmente algoritmos muito avançados na área da compressão de ADN [12-15], quer ao nível da compressão pura, quer ao nível da rapidez em que a executam, podemos considerar que esta área está ainda na sua infância, isto porque os primeiros algoritmos especialistas surgiram apenas na década de noventa do século passado.

1.1. Motivação e considerações prévias

A crescente produtividade da biologia molecular no que concerne à acumulação de informação biológica, amparada nas modernas técnicas de sequenciação e assistida pela bioinformática, providencia a uma vasta comunidade científica uma quantidade de genomas disponíveis nas bases de dados públicas em expansão exponencial. Em poucos anos o volume de informação biológica disponível ultrapassará as Terabases, e admitindo que a sequenciação das espécies dará lugar à sequenciação dos indivíduos, então esse valor parecer-nos-á irrisório. Assim, a eficiente compressão deste tipo de informação é uma necessidade premente para optimizar o seu armazenamento e comunicação [4]. Como passo inicial, cumpre analisar e mensurar a complexidade da informação contida nas sequências genómicas [16].

Do ponto de vista estritamente matemático, compressão implica compreensão e entendimento da informação comprimida [17]. Quanto mais soubermos das propriedades da informação que pretendemos comprimir maior será a probabilidade de sucesso na obtenção de compressão [18]. Paradoxalmente, o conhecimento das propriedades da informação também pode advir da aplicação de técnicas de compressão cujo intuito inicial era captar indistintamente as regularidades e redundâncias, o que acaba por expor, confirmando ou descobrindo, as propriedades da informação [19, 20].

Assim, a motivação primária deste trabalho consiste em desenvolver o conhecimento acerca da informação biológica, mormente das sequências de nucleótidos.

A era pós-genoma acarreta novos desafios. Cumpre realizar a competente análise dos dados biológicos. Essa função cumpre fundamentalmente à genómica funcional, adjuvada pela bioinformática. No essencial, a tarefa que os investigadores em bioinformática têm pela frente é passar da informação biológica ao conhecimento [21]. A pesquisa e análise de regularidades do ADN, entendidas como padrões exactos ou aproximados, é fundamental na maioria das aplicações bioinformáticas orientadas para genómica funcional. A predição de genes, descoberta de *motifs*, alinhamento de sequências, compressão de sequências ou o estudo de correlações filogenéticas têm o seu cerne na pesquisa e análise de padrões. Em todas estas aplicações, o desempenho dos algoritmos de pesquisa de padrões, ao nível da eficiência e sensibilidade, é determinante na qualidade das soluções que fornecem.

Neste trabalho procurou-se situar a investigação ao nível da bioinformática e não apenas usar algoritmos independentes da biologia usando dados de origem biológica. Assim, os algoritmos desenvolvidos usam o conhecimento das sequências biológicas revelado pela investigação em todos os quadrantes, desde a linguística à estatística do ADN. Especificamente, a abordagem à análise das sequências de ADN é feita com as balizas das propriedades intrínsecas conhecidas das sequências, desde a distinção de regiões de codificantes e não codificantes, de *motifs*, padrões repetitivos, etc. É crença do autor que os algoritmos que desvalorizem a especificidade da informação biológica, ignorando a sua lógica ou sendo incapazes de comportamento adaptativo perante genomas de diversas complexidades e densidades génicas, sem consciência dos funcionalismos genómicos, terão poucas possibilidades de êxito na tarefa de compressão de informação biológica. Tal conjectura ficou provada pelas tentativas de aplicação pouco frutuosas que aconteceram aquando da transposição de algoritmos universais de compressão ou de compressão de linguagens humanas para o ADN.

É comum aludir aos símbolos do ADN pela sua ordem alfabética (A,C,G,T). Neste trabalho optou-se, quer na dissertação, quer nos algoritmos desenvolvidos, por uma ordenação baseada em critérios biológicos. Assim, intercalam-se bases complementares, iniciando-se pela base A. O resultado é (A,C,T,G).

1.2. Objectivos

A informação dos códigos genéticos sequenciados é na actualidade, provavelmente, a fonte mais inspiradora para o estudo e avanço da teoria da informação, teoria da codificação e estudo da entropia informacional. Analisar uma variedade e quantidade de informação biológica tão extensa requer uma comunidade de investigadores cada vez mais numerosa e especializada. Os ramais de acesso ao conhecimento alargado dos códigos genómicos são variados mas, têm em comum a necessidade incontornável de mais investigação científica. Neste trabalho procura-se desenvolver o conhecimento dos códigos genómicos pela via da análise orientada para a compressão da informação, deste desiderato deverão resultar alternativas de codificação compacta que tornarão mais eficiente o armazenamento e a comunicação da informação genómica.

Desenvolver, testar e validar metodologias inovadoras para a análise e compressão da informação genómica são os principais objectivos deste trabalho. Num nível elevado de abstracção estes deverão ser os objectivos explícitos dos trabalhos de doutoramento, contudo será muito difícil, senão impossível, atingir esses objectivos sem antes ter conseguido avanços nas metodologias e componentes parcelares que cumulativamente contribuem para a compressão final. No caso presente partiu-se de um modelo conceptual, ilustrado na Figura 1.1, que engloba, sobretudo a montante do processo clássico de compressão de informação, um conjunto de pré-tratamentos específicos da informação genómica que procuram optimizar a compressão. Tais procedimentos, como sejam a análise de regularidades e a segmentação implícita terão de ser necessariamente muito eficientes e muito céleres pois é expectável que a compressão de um ficheiro se processe praticamente de forma instantânea. A inclusão da pesquisa multinível e da integração cooperativa de metodologias substitucionais e estatísticas é certamente inovadora e acarreta um conjunto de dados importantes para suportar decisões a tomar nas fases ulteriores do processo de compressão idealizado.

O modelo conceptual em que este trabalho se baseia preconiza que a compressão seja produzida por cooperação de metodologias nas diferentes regiões da sequência, implicitamente segmentadas pela separação dos elementos redundantes mais significativos. O que se pretende significar é a diferenciação de comportamento do

algoritmo perante as diferentes regiões da sequência com base na entropia presente. Após um reconhecimento inicial do genoma a tratar, cuja classificação não terá necessariamente que obedecer aos cânones da biologia, mas antes aos da informação biológica, a abordagem subsequente será no sentido de segmentar a sequência a comprimir. Esta segmentação far-se-á com o intuito de homogeneizar as parcelas de informação a analisar, i.e., aplicar a cada fracção da sequência o tratamento mais profícuo em termos de obtenção de recursos para a compressão, tais recursos baseiam-se maioritariamente em elementos redundantes ou estatisticamente previsíveis. Uma apreciação mais detalhada do modelo conceptual terá melhor cabimento no capítulo alusivo ao trabalho desenvolvido.

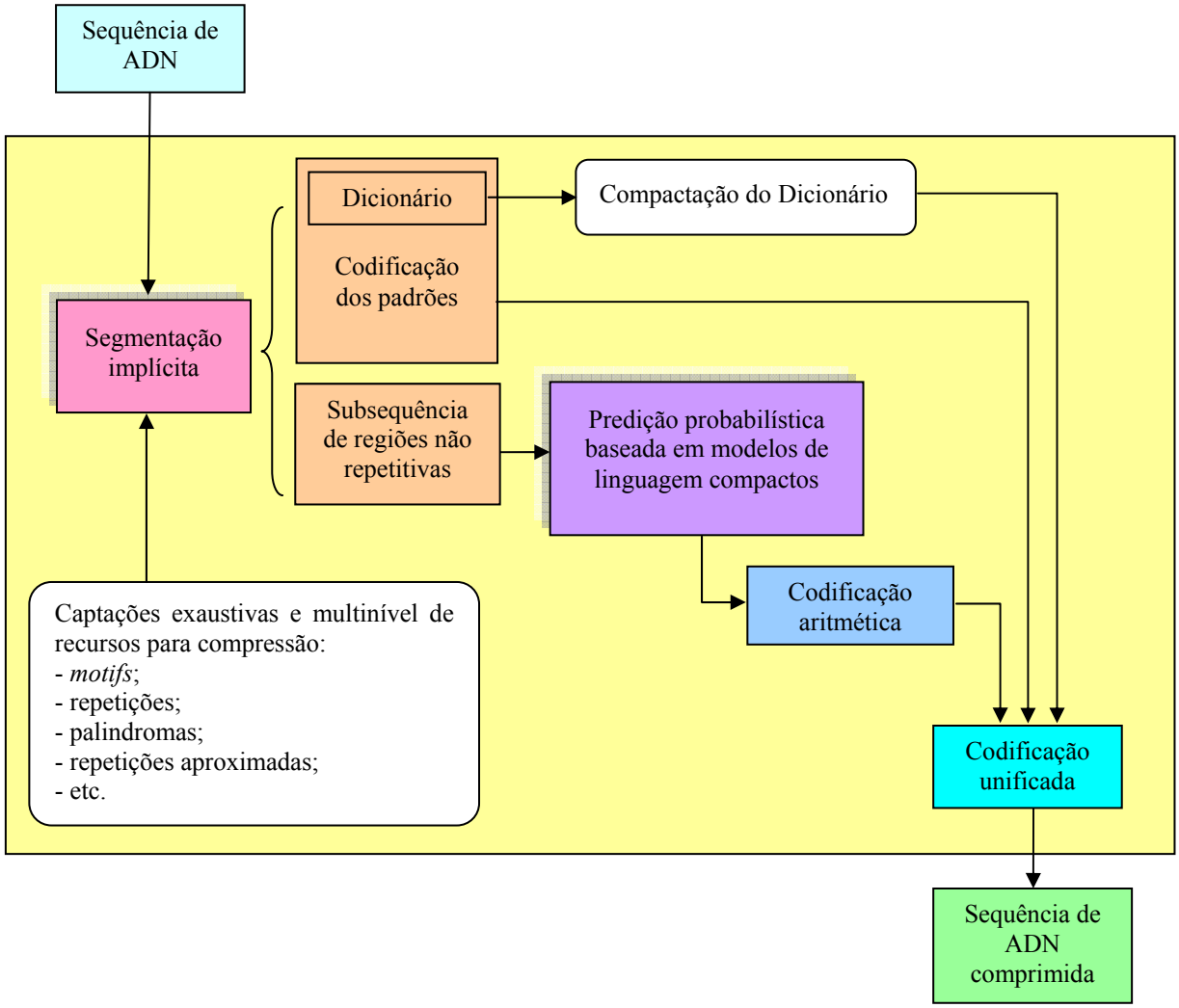


Figura 1.1 - Modelo conceptual de base para a aplicação de compressão a desenvolver.

Os constituintes do ficheiro comprimido final serão o dicionário compacto, a codificação dos padrões intercalada com as extensões de intervalo de ADN não repetitivo, expressas num código auto-delimitado, e finalmente a codificação aritmética das predições emitidas pelos modelos probabilísticos de linguagem compactos.

Para sistematizar os objectivos deste trabalho reuniram-se os seus enunciados em duas grandes categorias:
- atingir progresso científico ao nível da eficiência na análise e pesquisa de padrões exactos e aproximados em sequências genómicas pela inovação nas metodologias a empregar;
- integrar em cooperação as metodologias de pesquisa de similaridades, compressão substitucional e compressão por modelos probabilísticos num *framework* inovador, orientado para a compressão eficiente de informação genómica, e dele obter resultados que superem o estado da arte.

1.3. Contribuições

Pretende-se com este trabalho contribuir com soluções inovadoras para a análise e compressão de informação biológica, preferencialmente informação genómica. Dada a natureza modular do trabalho desenvolvido, é possível identificar nos seus módulos – a maior parte deles passíveis de utilização independente – um conjunto de contribuições relevantes nas áreas de pesquisa de padrões, pesquisa de similaridade, estimação de entropia e compressão de infomação biológica, como sejam:
- pesquisa de padrões exactos em sequências genómicas mais eficiente, em média com ganho de 20%, dependentemente do tamanho dos padrões;
- pesquisa de padrões exactos em sequências de aminoácidos mais eficiente, melhorada em 15% em média;
- resultaram novas abordagens em pesquisa de padrões exactos em linguagens naturais como efeito colateral das contribuições anteriores;
- descoberta de similaridade intra-genómica e inter-genómica melhorada, introduzindo uma nova metodologia com sensibilidade optimizada, comparável com a programação dinâmica, e de elevado desempenho, comparável com as soluções baseadas em heurísticas.

- análise informacional de sequências genómicas multi-nível para maximização de aproveitamento de elementos redundantes na compressão da informação;
- compressão de informação genómica melhorada, possibilitando o seu armazenamento e comunicação de forma mais eficiente;
- recolecção exaustiva de elementos repetitivos do ADN providenciado uma estimação de entropia mais objectiva. Alguns desses elementos redundantes possuem significância biológica conhecida, os outros ficam assim disponibilizados à comunidade científica da especialidade para estudo e análise funcional.

1.4. Organização da tese

A tese organiza-se em seis capítulos.

O capítulo introdutório pretende ser breve e apresentar em traços muito gerais e basilares os princípios que conduziram e motivaram este trabalho de investigação. Enunciam-se os objectivos e apresenta-se o modelo conceptual seguido para a consecução dos mesmos nos trabalhos de investigação realizados, sendo ponto central a justificação da eleição das metodologias de compressão nele integradas. Sumariam-se as contribuições para o desenvolvimento científico na área onde se insere e resumem-se as principais publicações científicas decorrentes do trabalho de investigação desenvolvido.

O segundo capítulo reúne uma síntese da teoria da biologia molecular considerada essencial para suportar os desenvolvimentos subsequentes. Interessa, sobretudo, caracterizar os processos onde intervém a informação biológica bem como entender a sua organização e finalidade. Importa igualmente, contextualizar e enquadrar a temática do trabalho - a compressão da informação genómica - no panorama e abrangência da bioinformática, uma nova área científica multi-diciplinar que se encontra em grande expansão e se entende como fundamental no suporte ao desenvolvimento das biociências. O resto do capítulo está destinado à compreensão e caracterização da informação genómica, com ênfase na análise informacional pura e não subjugada à genómica funcional. À compressão da informação interessa a análise integral, numa perspectiva abstracta, com o intuito de numa fase subsequente poder capitalizar toda a riqueza de padrões presentes na sequência a comprimir, independentemente de ser ou

não conhecida a sua funcionalidade genómica. A entropia das sequências genómicas é, no plano teórico, analisada à luz da teoria da informação.

No terceiro capítulo são apresentadas as metodologias usadas para a análise de similaridade e pesquisa de padrões nas sequências genómicas, desde as óptimas, as probabilísticas e as baseadas em heurísticas. Complementa-se a revisão com a análise dos algoritmos de eleição na pesquisa de padrões exactos e aproximados em informação genómica, debatendo as suas especificidades, vantagens e desvantagens. Na segunda metade do capítulo apresenta-se o estado da arte na área da compressão da informação genómica, analisando em maior detalhe os algoritmos mais eficientes, os quais servirão para análise comparativa com os propostos por esta tese. Descrevem-se as diferentes abordagens e metodologias de compressão de informação, para seguidamente seleccionar, com fundamentação, as mais adequadas à informação genómica, que se baseia em sequências de símbolos derivados do alfabeto quaternário $\Sigma=\{a,c,t,g\}$.

No quarto capítulo apresentam-se três algoritmos (GRASPm, DC e SimSearch) desenvolvidos para suporte à compressão da informação genómica, mas que extravasam largamente esse desiderato inicial, sendo igualmente úteis para análise genómica, proteómica e filogenética generalizada. A análise de complexidade desses algoritmos também foi abordada. Na área do *pattern-matching*, expõem-se os algoritmos desenvolvidos para pesquisa de padrões exactos e aproximados. Esta área científica caracteriza-se, no básico, pela resolução do problema de identificar repetições de um padrão inicialmente conhecido ao longo de uma sequência alvo fornecida. Têm igualmente cabimento neste capítulo, os desenvolvimentos efectuados na área de *pattern discovery*, que, latamente, corresponde à análise de similaridade entre as sub-regiões de uma sequência com vista à identificação de zonas repetitivas com potencial para a compressão de informação - nucleares na composição de um dicionário de padrões. No caso dos algoritmos de *pattern-matching* apresentam-se duas versões orientadas para a informação biológica, a primeira concebida e optimizada para as sequências genómicas (GRASPm) e a segunda mais vocacionada para alfabetos maiores (DC), com ênfase para as sequências proteómicas. O terceiro algoritmo apresentado, o SimSearch, destina-se à descoberta de similaridade inter e intra-genómica, identificando padrões exactos e aproximados. Trata-se de um algoritmo exaustivo, implementando a lógica da programação dinâmica, mas recorrendo a novas heurísticas que lhe

proporcionam mais polivalência e melhor desempenho. Este algoritmo será mais tarde aplicado às várias apresentações ou níveis da bioinformação genómica, desde a sequência natural às complementares, com vista à obtenção de regularidades para formação do dicionário.

O quinto capítulo está centrado na descrição da implementação e funcionamento do algoritmo híbrido de compressão de informação genómica desenvolvido, o DNALight. Analisam-se em detalhe todas as fases do algoritmo, as metodologias seleccionadas para a composição e compressão do dicionário, bem com para a compressão dos segmentos não incluídos no dicionário. Seguem-se as análises das fases de compressão e descompressão. Finalmente, apresentam-se resultados comparativos do desempenho da aplicação desenvolvida, caracterizando a complexidade temporal e espacial envolvida.

No sexto capítulo discutem-se os resultados, expõe-se a análise crítica do trabalho desenvolvido e elencam-se as conclusões, para finalizar com a relação dos temas em aberto que emergem do trabalho e que estão na base da previsão de trabalhos e desenvolvimentos futuros.

1.5. Publicações científicas decorrentes da investigação realizada

No decurso deste trabalho foi concedido tempo e importância à divulgação de resultados parciais, bem como à submissão do trabalho desenvolvido à apreciação da comunidade científica local e global. Para além das várias presenças nos Simpósios Doutorais promovidos pelo Departamento de Informática da Escola de Engenharia da Universidade do Minho, foram efectuadas comunicações nos seguintes fóruns e eventos:

- A convite do Prof. Doutor António Macedo, do Departamento de Informática da Universidade do Minho, foi ministrada uma sessão alusiva à compressão de sequências genómicas aos alunos do 4º ano de Engenharia Biomédica, da Universidade do Minho em Braga em Março de 2006.

- Palestra sobre Bioinformática, integrada na Semana da Ciência e Tecnologia, realizada na ESAB do Instituto Politécnico de Bragança e promovida pela FCT - Fundação para a Ciência e Tecnologia, 26 de Novembro de 2006.

- Duas comunicações orais, a primeira alusiva às ferramentas bioinformáticas, a segunda focando a pesquisa de padrões exactos e aproximados em sequências biológicas, integradas no Curso de Bioinformática, edição de 2007, realizado na ESAB do Instituto Politécnico de Bragança, em 19 e 20 de Abril de 2007.

Concomitantemente, foram submetidos artigos para publicação em revistas internacionais indexadas ou em conferências internacionais. Segue-se a sua listagem:

Sérgio A. D. Deusdado, Paulo M. M. Carvalho, ***GRASPm: An efficient algorithm for exact pattern-matching in genomic sequences***, aceite para publicação no International Journal of Bioinformatics Research and Applications, Inderscience, 2008.

Sérgio A. D. Deusdado, Paulo M. M. Carvalho, ***Highly efficient and flexible exact pattern-matching algorithm***, submetido para publicação na revista internacional Software: Practice and Experience, Wiley, 2008.

Sérgio A. D. Deusdado, Paulo M. M. Carvalho, ***A new variant of dynamic programming for optimal and near-optimal similarity discovery in genomic sequences based on distance series***, aceite para apresentação e publicação no 2nd International Workshop on Practical Applications of Computational Biology & Bioinformatics, Salamanca, Spain, October 2008.

Capítulo 2. Contexto: A informação genómica

Importa neste capítulo contextualizar a temática e os objectivos deste trabalho e desta dissertação. Dada a multidisciplinaridade patente será útil inscrever, de forma propedêutica, algum conhecimento fundamental para a plena compreensão do estado da arte da bioinformática, mormente nas extensões desta que mais intervêm nos domínios abordados neste trabalho. A caracterização da linguística do ADN, sobretudo no tocante às especificidades dos elementos repetitivos que integram em grande proporção as sequências genómicas, é igualmente tema central deste capítulo, estatuto justificado pela importância que têm como recurso magno da compressão.

2.1. Bioinformática

No passado a biologia computacional abriu o caminho para a actual bioinformática. Com o advento da Internet, uma miríade de aplicações bioinformáticas estão acessíveis on-line para os investigadores, vão desde as poderosas ferramentas exploratórias das bases de dados genómicas às representações das estruturas 3D das proteínas. Os acessos diários a estes serviços provam o imenso interesse destas aplicações bem como a fervilhante actividade investigadora nestes domínios [2]. A bioinformática é uma área da ciência altamente multi-disciplinar e assim o são, para já, os seus intervenientes, na sua maioria biólogos que fizeram a aprendizagem posterior das vertentes informáticas

ou bem informáticos com o percurso inverso, se os primeiros deram os primeiros passos, os segundos estão a potenciar novos estádios de desenvolvimentos para a bioinformática [22].

Actualmente a bioinformática ocupa-se maioritariamente da organização, análise, simulação e extracção de conhecimento dos dados produzidos pelos recentes avanços da biologia molecular e da genética. Sendo a bioinformática, em senso lato, a ciência multidisciplinar amparada na biologia, matemática, bioestatística e informática, que procura dar solução aos problemas baseados em bioinformação, são as biosequências (ADN, ARN e proteínas), em grande expansão, que acaparam actualmente os maiores enfoques das aplicações da bioinformática.

Com a proliferação das bases de dados biológicas e a investigação biomédica com recurso à bioinformática a rivalizarem pela liderança na ocupação dos recursos de computação actualmente existentes, podemos identificar alguns dos vectores onde a bioinformática procura com maior intensidade estabelecer-se. São eles:
- assemblagem de fragmentos de ADN;
- pesquisa de genes;
- análise e comparação de sequências biológicas;
- identificação de *motifs* e padrões;
- pesquisa de homologias ou similaridades;
- inferência filogenética;
- simulação de processos genéticos;
- inferência de função e estrutura de proteínas;
- desenho de proteínas;
- determinação de locais de conexão de proteínas com moléculas de fármacos;
- etc.

A bioinformática fornece algoritmos e aplicações que resolvem, cada vez com maior eficiência, os problemas acima listados. A sua evolução torna-se crucial pois o volume e a complexidade da bioinformação está em expansão acelerada e a procura de soluções para, por exemplo, encontrar a cura para um determinado tipo de cancro, passa necessariamente pelos avanços na bioinformática.

O paralelismo entre a forma como funciona um programa de computador e os processos biológicos de codificação e replicação de informação génica já foi estabelecido por muitos dos teóricos que abordaram a bioinformática [23, 24]. De facto, os princípios estabelecidos por Turing, von Newmann e outros teóricos da computação aproximam-se de forma muito considerável dos processos que regem a própria vida no interior das células. Se o ADN constitui um repositório da informação fundamental tal como acontece com o disco rígido principal de um computador, o ARN pode equiparar-se à memória RAM, através da qual os dados são mais directamente disponibilizados ao processamento, tal como acontece nos processos biológicos de maturação do ARNm, preliminar à formação das proteínas.

No sentido de encerrar esta secção com algumas garantias de não defraudar os interesses dos leitores, ficam aqui algumas referências bibliográficas que poderão completar e complementar de forma mais enciclopédica as noções aqui aventadas sobre bioinformática [2, 21, 25-27].

2.2. ADN

Cada célula de um ser vivo contém um ambiente dinâmico constituído por moléculas, reacções químicas e uma cópia do genoma do organismo. Embora o ADN seja exactamente o mesmo em todas as células de um dado organismo, a expressão desse código pode ser radicalmente diferente em cada uma delas, originando desta forma células especializadas em diferentes órgãos do organismo. É no núcleo da célula que se encontra o ADN, o ácido desoxirribonucleico onde se escreve o código da vida.

Dentro da célula, o ADN é organizado numa estrutura chamada cromossoma e o conjunto de cromossomas de uma célula forma o cariótipo. Antes da divisão celular os cromossomas são duplicados através de um processo chamado replicação. Eucariontes como animais, plantas e fungos têm o seu ADN dentro do núcleo enquanto que procariontes como as bactérias o tem disperso no citoplasma.

A Figura 2.1 ilustra a estrutura do ADN tal como hoje a conhecemos devido, em grande medida, aos trabalhos de Watson e Crick [1], que lhes valeram o prémio Nobel da Fisiologia e Medicina em 1962.

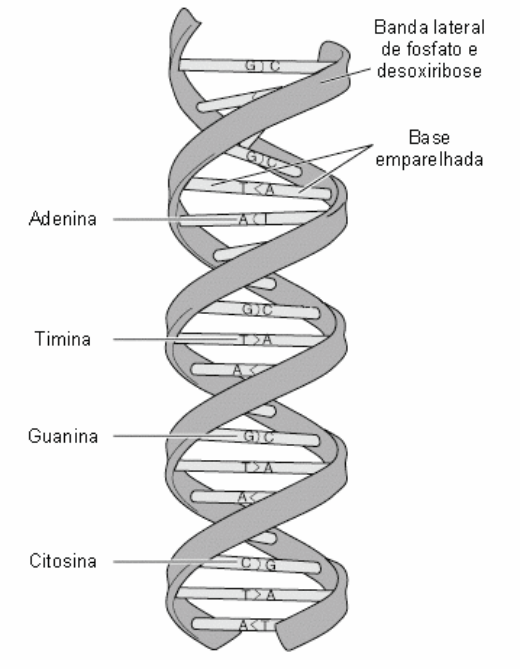


Figura 2.1 - Estrutura de dupla hélice do ADN.

O ADN é uma longa molécula constituída por duas hélices de nucleótidos interligadas por pontes de hidrogénio. Cada um dos nucleótidos tem na sua constituição três componentes: um grupo fosfato, uma pentose e uma base azotada (Figura 2.2). No ADN consideram-se quatro categorias de nucleótidos que são designados pela base azotada que entra na sua constituição, a saber: adenina (A), guanina (G), citosina (C) e timina (T). A ligação das duas hélices de nucleótidos segue a regra do emparelhamento de bases, isto é, A emparelha sempre com T e C emparelha sempre com G. As duas hélices de ADN são desta forma estruturalmente complementares.

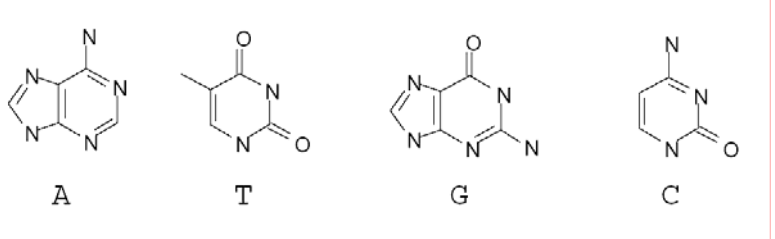


Figura 2.2 – A química dos quatro nucleótidos que compõem a molécula de ADN.

O ADN de todos os organismos vivos contém associações das quatro bases de nucleótidos. Cada uma das bases possui a sua complementar pelo que a codificação genética aparece duplamente inscrita, pensa-se que tal redundância se justifique para fomentar a isenção de erros aquando de replicações ou transcrições.

Para formalizar a complementaridade do ADN e usando uma notação matemática temos que, sendo N um nucleótido e $\overline{N}$ o seu complementar:

$$\overline{A} = T, \overline{T} = A, \overline{C} = G, \overline{G} = C \qquad \text{e} \qquad \overline{\overline{N}} = N$$

…	A	A	C	T	G	T	…
…	T	T	G	A	C	A	…

Figura 2.3 – Subsequência de ADN com os pares complementares de bases.

De um ponto de vista computacional, a informação genómica corresponde a uma sequência unidimensional de símbolos, um alfabeto de quatro símbolos para o ADN e o ARN. Para codificar as proteínas são necessários 20 símbolos, para outros tantos aminoácidos.

A sequência de ADN divide-se em cromossomas, nestes encontram-se distribuídos os genes que geralmente correspondem a uma fracção diminuta do material genético. Os genes são formados por uma sequência de bases, geralmente não excedendo as kilobases, que nalguns casos se localiza na sequência 5'-3' e noutros na sequência complementar 3'-5' [3]. Cada gene contém nas suas bases as instruções de como produzir as proteínas. As partes do ADN que contêm genes têm a denominação de áreas de codificação enquanto que a porção remanescente corresponde às áreas de não codificação. Apesar das áreas que não contêm genes ainda não estarem suficientemente estudadas para aferir a sua verdadeira importância, há mesmo quem as refira como "junk DNA", estas áreas correspondem à imensa maioria do nosso material genético e não devem ser ignoradas aquando da sequenciação, análise e posterior compressão e armazenamento.

Em Abril de 2003 declarou-se concluída a sequenciação do genoma humano, a grande bandeira do Projecto Genoma Humano (HGP) [28] iniciado em 1990, que ficará na história como um marco cuja magnitude ainda estamos a descobrir. O genoma humano contém cerca de 3,1647 biliões de pares de bases. Dessa informação apenas 3% correspondem às porções dos genes codificadores de proteínas, e cerca de metade são subsequências mais ou menos repetitivas, muitas de origem viral [29, 30]. Actualmente são sequenciados (terminados) genomas de diversos seres vivos quase todos os dias e as bases de dados de genomas sequenciados duplicam a cada ano, o que significa um crescimento exponencial de informação genómica.

Segundo a analogia de Ridley [31], o genoma (dê-se como exemplo o humano) é uma espécie de livro de tamanho gigantesco: tem «23 capítulos, chamados cromossomas; cada capítulo contém vários milhares de histórias, chamadas genes; cada história é feita de parágrafos, chamados exões (*exons*) que são interrompidos por anúncios chamados intrões (*introns*); cada parágrafo é feito de palavras, chamadas codões; e cada palavra é escrita com três letras, chamadas bases.» Todos os seres vivos, quer se trate de plantas, animais ou insectos, partilham código genético, o que revela um passado comum entre todas as espécies de seres vivos.

A informação genética armazenada no genoma apresenta uma organização espacial que deriva de alguns processos conhecidos e estudados, necessariamente susceptíveis de análise estatística, eis alguns dos mais relevantes:

- Introns e Exons : Genes eucarióticos são geralmente compostos de exons e introns.

  - Introns: A quase totalidade das sequências genómicas nos procariotas é informativa, mas nos eucariotas superiores somente cerca de 5% do genoma é expresso. A maior parte dos genes estruturais conhecidos contêm segmentos não informacionais (introns) que se intercalam nas sequências informacionais. Introns são portanto, sequências não codificantes de ADN localizadas num gene. Existem duas hipóteses principais sobre a origem dos introns, chamadas: "introns-early" e "introns-late". A primeira, "introns-early", assume que os introns já estavam presentes nos genes do progenoto (ancestral comum entre os seres procarióticos e eucarióticos) e foram perdidos nos procarióticos na redução

dos genomas devido à pressão selectiva de replicação rápida do ADN. Em contraste, a hipótese "introns-late" acredita que os introns foram inseridos mais tarde durante a evolução, após a divergência de procarióticos e eucarióticos. Actualmente pensa-se que estas regiões não codificantes desempenham funções de controlo e regulação do RNAt dos genes e contribuem significativamente para a variabilidade de fenotipos observados nas doenças humanas [32].

- Exons: Apenas parte dos genes contém informação codificadora de proteínas, essas subsequências denominam-se de exons, em oposição ao introns, que perfazem as regiões não codificantes.

- Rearranjos genómicos: São basicamente translocações das bases numa subsequência que anteriormente se verificara noutra porção da sequência, onde podem incluir-se os palindromas.

- Codões: Sequências de três nucleótidos que codificam um aminoácido ou têm uma função de sinalização na síntese das proteínas.

- *Isochores*: a composição base da sequência genómica dos vertebrados não é homogénea, porém existem extensas regiões codificadas uniformemente com as bases G+C, que se denominam de *isochores*.

- *Chirochores*: Sabe-se que o sistema de codificação no ADN da maioria dos seres é simétrico, contudo, a composição das duas cadeias da dupla hélice de ADN difere na maioria dos genomas das bactérias. As regiões homogéneas em termos da composição base da cadeia única de ADN são chamadas *chirochores*.

- Elementos transponíveis: trata-se da mobilidade de determinadas regiões contidas nos ADNs genómicos procarióticos e eucarióticos, ou seja, a capacidade dessas regiões (genericamente denominadas elementos móveis ou transponíveis) se transporem ("saltarem") para novos locais no próprio genoma onde residem ou para outros genomas e se reverterem e sofrerem delecções e/ou amplificações. Interessa igualmente estudar os mecanismos pelos quais estes rearranjos se processam e são regulados, bem como as consequências biológicas que deles advêm e a forma como se desenvolvem de acordo

com as mais variadas estratégias genéticas específicas, o que permite ou poderá permitir a exploração dos elementos transponíveis como potentes ferramentas de investigação fundamental e aplicada. Principalmente, a maioria dos genomas eucarióticos possuem quantidades substantivas de subsequências de elementos transponíveis que são repetidas desde as dezenas às várias centenas de vezes.

O dogma central da genética molecular, esquematizado na Figura 2.4, consiste nos processos que originam as proteínas a partir do ADN, desde a codificação na sequência de ADN, à transcrição para o ARN (ácido ribonucleico), à tradução e finalmente ao produto péptidico [33]. O ARN mensageiro, cuja sequência provém do ADN com a substituição da timina (T) pelo uracil (U), será traduzido em proteínas pelo ribossoma. Após o codão START, cada codão subsequente do ARNm implica a adição de uma amino-ácido à cadeia péptidica, a cadeia finaliza com um codão de STOP.

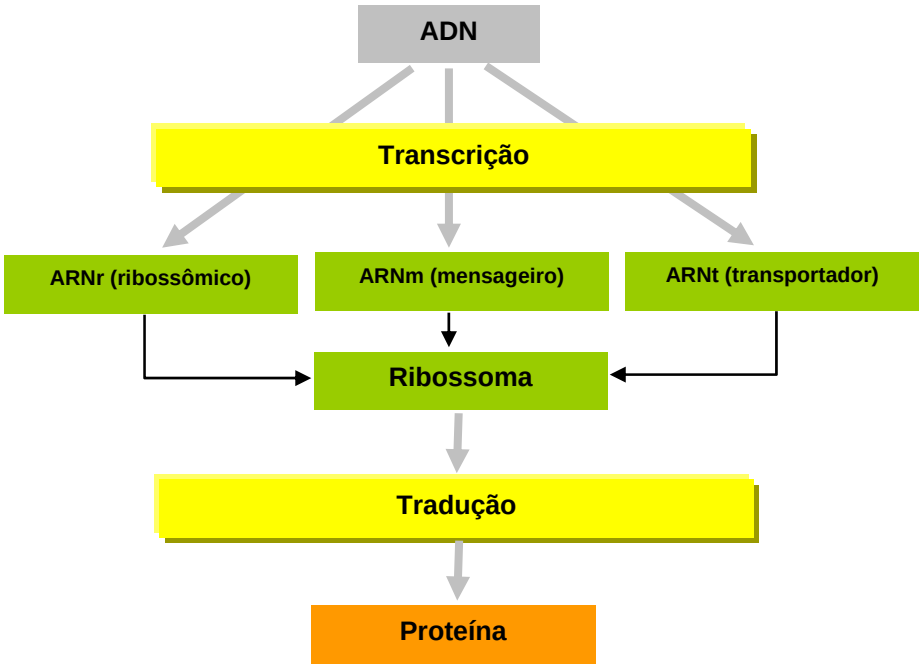


Figura 2.4 – O dogma central da genética molecular, os processos que originam as proteínas.

2.2.1. Do ADN aos amino-ácidos e às proteínas

Cada gene contém uma subsequência do ADN que inclui além dos promotores e outros elementos de regulação, um conjunto de codões que sequencialmente, mas não necessariamente de forma adjacente, codificam os elementos que será necessário sintetizar para formar uma determinada proteína. Em rigor, os amino-ácidos produzidos derivam da cadeia de ARNm, porém esta é um reflexo fiel do ADN que lhe deu origem, pelo que é usual representar a tabela de amino-ácidos com os codões de ADN como acontece na Tabela 2.1.

		Codões Sinónimos (CS)						
		Codão	**Codão**	**Codão**	**Codão**	**Codão**	**Codão**	**Nº CS**
A	Alanina	GCA	GCT	GCC	GCG			4
R	Arginina	AGA	AGG	CGA	CGT	CGC	CGG	6
N	Asparagina	AAC	AAT					2
D	Ácido Aspártico	GAC	GAT					2
C	Cisteína	TGC	TGT					2
E	Glutamato	GAA	GAG					2
Q	Glutamina	CAA	CAG					2
G	Glicina	GGA	GGT	GGC	GGG			4
H	Histidina	CAT	CAC					2
I	Isoleucina	ATA	ATT	ATC				3
L	Leucina	CTA	CTT	CTC	CTG	TTA	TTG	6
K	Lisina	AAA	AAG					2
M	Metionina (START)	ATG						1
F	Fenilalanina	TTC	TTT					2
P	Prolina	CCA	CCT	CCC	CCG			4
S	Serina	TCA	TCT	TCC	TCG	AGT	AGC	6
T	Trionina	ACA	ACT	ACC	ACG			4
W	Triptofan	TGG						1
Y	Tirosina	TAT	TAC					2
V	Valina	GTA	GTT	GTC	GTG			4
STOP		TGA	TAA	TAG				3
		Combinações Totais						64
		Combinações para Amino-ácidos						61

Tabela 2.1 - O significado dos codões.

Das 64 combinações possíveis com 3 bases, apenas 61 codificam amino-ácidos, as restantes 3 são codões sinónimos para a terminação de uma sequência péptidica, os codões STOP. É de notar que a maioria dos codões sinónimos varia apenas na última base. Existem apenas dois codões unívocos, o codão da metionina (M), que representa o codão START, i.e, o codão que assinala o início da codificação da cadeia péptidica que constituirá a proteína e o codão do triptofan (W), curiosamente este último partilha o dupleto inicial com um dos codões de STOP.

2.2.2. Introns e Exons - o caso dos eucariotas

O código genético é ele próprio parte integrante da vida e como tal está em permanente evolução. As proteínas, por inerência, estão igualmente em evolução e incremento da sua diversidade. O genoma evoluiu de um sistema auto-replicativo baseado em ARN, o mundo ARN [34], para um sistema complexo de genes multi-exon codificantes de proteínas multi-modulares. Durante este processo de evolução foram introduzidas ou modificadas muitas propriedades e funções sem, no entanto, erradicar mecanismos ancestrais. Apesar da actual complexidade, o sistema de codificação genético mantém-se flexível e ao mesmo tempo robusto.

A unidade básica da informação genética não tem entendimento consensual entre os cientistas, se para uns se trata do codão, para outros a unidade principal é o gene. Se o codão pode representar um amino-ácido, um gene codifica uma proteína. O gene pode ser visto como um módulo auto-contido com uma interface bem definida. Na extensão de um gene, que pode variar entre as dezenas e os milhares de pares de bases, está distribuída toda a informação necessária para gerar uma proteína. O termo "distribuída" quer significar, neste caso, que a informação codificante não se encontra numa subsequência ininterrupta mas antes em segmentos, denominados exons, intercalados por outros segmentos não codificantes denominados de introns, tal como ilustrado na Figura 2.5.

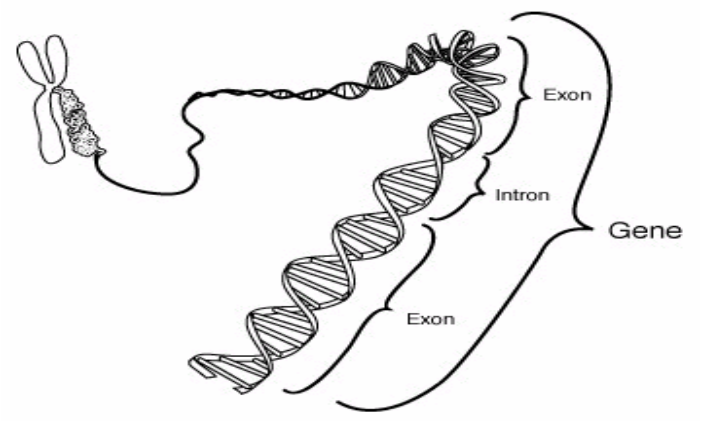


Figura 2.5 – As regiões de codificação dos genes não são contínuas, são intercaladas pelos introns [3].

Os introns serão excisados aquando da formação do ARNm (ver Figura 2.6), essa excisão assenta em informação localizada exclusivamente no intron, os sítios de reconhecimento de uniões (ver Figura 2.7). Desta forma e apenas atendendo a esta função básica, os introns demonstram-se imprescindíveis pois é através desta informação que os exons são aglutinados na sequência de ARNm que originará a proteína.

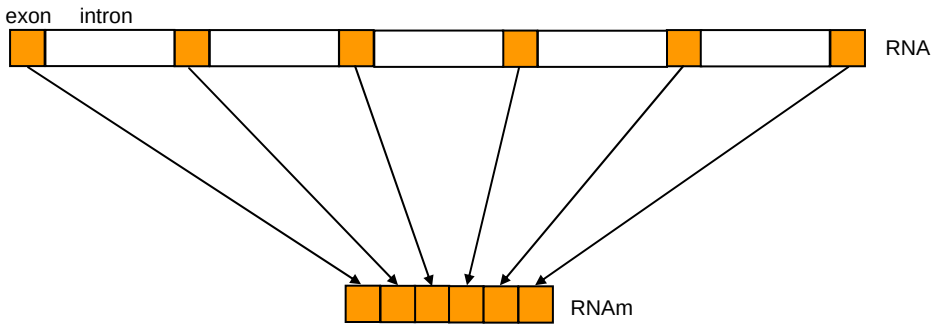


Figura 2.6 – Excisão dos introns aquando da maturação do RNA.

Uma dependência introduzida pelos introns resulta certamente numa dificultação do movimento e troca de sequências codificantes entre genes. Contudo, uma extensiva recombinação de exons por *exon shuffling* (mistura) é considerada com tendo um papel importante na criação da diversidade genética [35, 36]. Prevendo a inserção de um módulo do exon numa sequência aleatória de nucleótidos, então este módulo deveria comportar-se como independente e auto-contido. Esta independência pode ser adquirida

pela conservação das sequências dos flancos do intron que delimitam o exon. Se os delimitadores são incluídos como parte integrante do módulo do exon, permitindo-lhe funcionar como módulo de codificação independente, estaremos perante um módulo de exon ancestral.

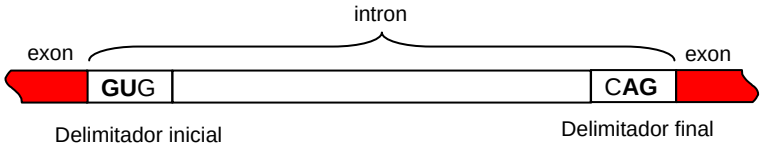


Figura 2.7 – Os delimitadores presentes nos introns.

O termo "expressão génica" refere-se basicamente à produção de ARN por um determinado gene. A expressão é condicionada pelo estado da célula, desde as condições básicas de nutrição aos comportamentos anormais. Recentemente descobriu-se que um determinado gene pode gerar diferentes cadeias de ARN, o que equivale a dizer que pode produzir diferentes tipos de proteínas, tais mecanismos são actualmente assunto de pesquisa [37].

O termo "*splicing*" significa, de forma simplista, a remoção dos introns para formar em contínuo a região codificante [38]. Aquando da sua expressão, os genes evidenciam os pontos de descontinuidade expressiva, correspondendo a zonas expressivas numa determinada transcrição inseridas no seio de zonas inexpressivas, na prática trata-se de distinguir entre introns e exons, rejeitando os introns na formação do mRNA, que será uma sequência completamente co-linear com a proteína a produzir.

A regra GT-AG indica que estes dinucleótidos são os delimitadores base para introns e exons [3]. Nesta convenção os introns começam com o dinucleótido GT, e terminam com o dinucleótido AG. Para o caso do ARN, a regra funcionará obviamente com GU-AG (Figura 2.7).

2.2.3. Segmentação de sequências de ADN

As sequências de ADN não são homogéneas. A heterogeneidade reflecte-se a vários níveis, desde o conteúdo G+C, as ilhas CpG, a assimetria das hélices em A-T, em regiões codificantes e não codificantes, existência de *motifs*, etc. Esta heterogeneidade é ela própria heterogeneamente distribuída em diversos genomas. A segmentação do ADN pode, portanto, estar dirigida para estas e outras propriedades do ADN, e fazer-se com recurso à análise fundamental ou à estatística [39].

A segmentação mais básica implica a predição de genes ou regiões codificantes e, dentro destes, distinguir exons de introns. As metodologias desenvolvidas neste trabalho proporcionarão como efeito colateral a distinção entre regiões codificantes e não codificantes, dado que a entropia presente em cada uma das regiões permitirá a sua distinção.

2.3. Análise informacional das sequências genómicas

Concorrem para a genética contemporânea uma nova vaga de investigadores interessados, não na análise física do genoma – esta mais clássica –, mas sim na análise informacional das sequências biológicas. A análise informacional está direccionada para a compreensão da organização do código do ADN, usando as contribuições da informática [40], ao nível da teoria de informação de Shannon [41] e ao nível dos algoritmos de caracterização, segmentação e mapeamento genómico, os modelos matemáticos, estatísticos e probabilísticos. Por outro lado, existem abordagens gráficas que incluem representações pictóricas da informação do ADN bem como compreensões da informação genómica à luz da teoria dos fractais [42-44].

Convém neste estádio de descrição dos termos do trabalho, diferenciar similaridade e homologia, no tocante à análise de sequências. A homologia é uma similaridade com fundamento funcional, é deduzida pela similaridade e pelos conhecimentos reunidos pelas anotações de genomas. A similaridade é computada, trata-se de uma análise puramente informacional sem intuito primário de determinar identidade funcional ou filogenética. Neste trabalho é a similaridade que mais interessa, será pela sua análise

que emanarão os padrões exactos e aproximados que integrarão o dicionário, sendo seguramente a componente mais crítica no sucesso da compressão.

2.3.1. ADN repetitivo: *motifs* e padrões

Tomando como exemplo o genoma humano, mais de metade da sua composição consiste, do ponto de vista informacional, em subsequências repetitivas. Se algumas dessas subsequências que se repetem ao longo das sequências possuem uma função bem determinada, como por exemplo os promotores TATA, então tais elementos são denominados de *motifs*. Por outro lado, existem inúmeras repetições cuja funcionalidade não está directamente relacionada e podem derivar, por exemplo, da saturação resultante de um código com apenas 4 símbolos, ou 64 codões, nestes casos vamos denominar estas repetições de padrões. As regiões não codificantes são ricas em padrões e linguisticamente diferentes das regiões codificantes [45, 46]. Se inicialmente se devotavam todos os meios e atenção na análise das regiões de codificação, actualmente procura-se estudar a sequência integralmente. De facto, o interesse científico das regiões não codificantes ainda é secundário e o seu estudo incipiente. A aparente redundância dessas regiões que apresentam ADN repetitivo tem elevado interesse dentro da temática deste trabalho, mas o seu interesse não se esgota na teoria da informação como o atestam os estudos [47-50]. Existem muitas variantes de sequências repetitivas, e.g., satélites, ALUs, repetições adjacentes (*tandem repeats*), repetições dispersas (*interspersed repeats*), etc. As repetições estão equitativamente distribuídas pelos genomas e espécies e constituem uma fracção muito considerável de todo o ADN dos organismos. As repetições podem ser exactas ou aproximadas, assumindo estas últimas, claramente, mais importância na composição da informação biológica.

2.3.1.1. Repetições não adjacentes – elementos transponíveis

A maioria das repetições não adjacentes são originadas por um processo de transposição, grosso modo, são segmentos de ADN que são replicados noutro local da sequência. Esses segmentos denominam-se de *transposons*.

Os *transposons* podem ser de dois tipos, os *transposons* do ADN e os *retrotransposons*. Estes últimos são os mais abundantes, e derivam de cópias de ARN transcritas reversamente em ADN, que são inseridas no genoma numa outra determinada localização, estes segmentos são maioritariamente inactivos. Se estes elementos concorrem para a plasticidade do ADN, também podem ser responsáveis pelo surgimento de doenças quando interferem com a normal actividade génica. Podem ser de dois tipos quanto à forma como se propagam, designando-se por replicativos os que acumulam várias cópias, como é o caso dos *retrotransposons*, e por não-replicativos, os que se movem sem disseminações, como é o caso dos *transposons*, a esquemática usada está descrita na Figura 2.8.

A. *Transposons* que se movem por transposição replicativa são duplicados no processo de salto, num mecanismo de copiar-e-colar.

B. *Transposons* não-replicativos movem-se por um mecanismo recorta e cola. Nenhuma cópia é feita.

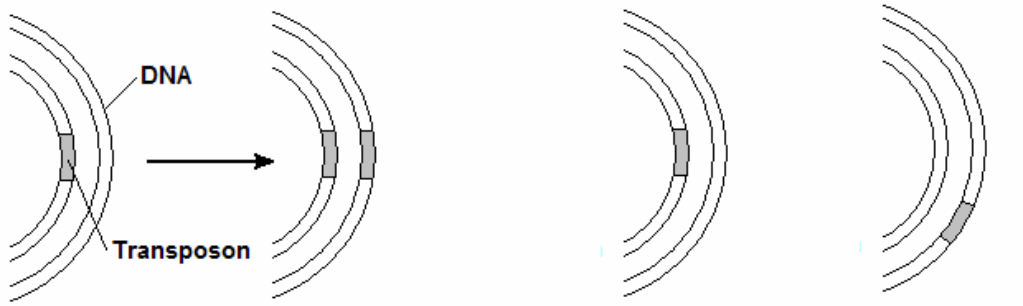


Figura 2.8 – *Transposons* replicativos (A) e não-replicativos (B).

Os elementos retrotransponíveis são categorizados em autónomos e não-autónomos no que concerne à auto-suficiência para a mobilidade. Dos elementos autónomos contam-se os LTRs (*Long Terminal Repeats*) e os não-LTRs, estes últimos são vulgarmente representados pelos LINEs (*Long Interspersed Nuclear Elements*). Os LINEs são longas repetições não adjacentes e compreendem cerca de 21% do genoma humano, os SINEs (*Short Interspersed Nuclear Elements*), que correspondem aos elementos não-autónomos, constituem-se em curtas repetições não adjacentes, tipicamente com um número de bases inferior a 500. Os ALUs estão incluídos nos SINEs, estes elementos existem em número elevado no genoma humano, mais de um milhão e perfazem mais de 11% da sua composição. Um maior detalhe na função destes elementos repetitivos está fora do âmbito deste trabalho, contudo apontam-se alguns trabalhos na área [51-

53]. Nos mamíferos, as repetições dispersas, representam cerca de uma terço do material genético [54], são obviamente de enorme importância para a compressão da informação.

2.3.1.2. Repetições adjacentes

As repetições em tandem são normalmente associadas a doenças humanas, apresentam importantes relações com a evolução, bem como, interferem decisivamente nos processos de regulação da célula [55]. Este tipo de repetições pode, em certos casos, aparecer com alguma degeneração, ou seja, as unidades repetitivas podem conter eventualmente algum erro relativamente à unidade de referência. As repetições de maior tamanho são denominadas de satélites, variam em tamanho desde os 100kb até 1 Mb. No genoma humano, um exemplo bem conhecido é o ADN *alphoid*, que se localiza perto do centrómero e se caracteriza por repetições sucessivas de 171 bases, perfazendo em cada cromossoma cerca de 3 a 5% da sua composição genómica.

As repetições adjacentes curtas são denominadas de minisatélites (na ordem das kb) e as mínimas (normalmente 1-5 pb) são denominadas de microsatélites. Os microsatélites raramente excedem as centenas de repetições, a família de repetições mais comum deste tipo são as de duas bases, entre as quais o dinucleótido CA prevalece em frequência. O termo satélite parece aqui estranhamente aplicado, mas obviamente tem a sua explicação, que assenta nos resultados do processo de centrifugação de ADN, onde os fragmentos repetitivos adjacentes tendiam a formar uma espécie de banda satélite e por isso receberam esse nome. Como exemplo, refira-se que os telómeros são mormente constituídos por pequenas repetições sucessivas das bases GGGTTA. Para um maior desenvolvimento neste tipo de repetições genómicas, sobretudo ao nível da sua pesquisa por meios bioinformáticos, sugerem-se os seguintes trabalhos [56-59].

2.3.1.3. Palindromas

Nas sequências de ADN, os símbolos ‘A’ e ‘T’ são complementares um do outro, de igual modo, ‘G’ e ‘C’ são complementares um do outro. Uma *string* y_1^n é a reversa

complementar de x_1^n se x_i e y_{n+1-i} são complementares mútuos para $1 \leq i \leq n$. Ao par de *strings* y_1^n e x_1^n dá-se o nome de palindroma [60]. Basicamente, trata-se de repetições de sub-sequências invertidas e com as bases representadas pelas correspondentes complementares. Por exemplo, a reversa complementar de "AAACGT*"* é "ACGTTT*"*.

Existem exemplos curiosos de sequências de ADN com forte presença de palindromas. Para mencionar alguns, refira-se o caso do cromossoma III de uma levedura denominado por "CHMPXX", composto por 121024 símbolos, que apresenta um palindroma de 10000 bases. Outro caso relevante acontece num gene de um vírus denominado de "VACCG", com uma extensão de 191737 símbolos, que contém um palindroma de 8000 símbolos. Estes casos, pela sua representatividade, fazem parte de bases de dados usadas em *benchmarking* de algoritmos de compressão e análise entropia de sequências de ADN. Estas redundâncias constituem, juntamente com as demais repetições já apresentadas nesta secção, recursos importantes na compressão da informação genómica.

A complexa modularidade da organização dos genomas será, por muito tempo, um tema em aberto mas, nessa modularidade as regiões repetitivas desempenham certamente um papel fundamental [61].

2.4. Análise estatística das sequências de ADN

A análise estatística das sequências de ADN constitui procedimento primordial imediatamente após a assemblagem mais estruturada da sequenciação. Analisar as frequências dos nucleótidos, das famílias químicas - as purinas (A e G) e as pirimidinas (C e T) -, dos codões ou de *motifs* conhecidos contam-se entre as primeiras caracterizações a realizar [62]. Sucedem-se análises mais aturadas em relação a distribuições, estatística comparada com outras sequências [63], caracterização de correlações [64] e periodicidades [8], riqueza GC e modelação estatística [65], diferenciação estatística entre regiões codificantes e não codificantes [66], entre outras. Portanto, a análise estatística do ADN [67] é contemporânea da sequenciação dos primeiros genomas e tem acompanhado a sua evolução. É de tal modo importante, que a

(bio)estatística é um dos pilares da bioinformática, juntamente com a biologia molecular e a informática.

A análise estatística é fundamental na acumulação de conhecimento das peculiaridades que caracterizam as sequências, quer a nível intra-genómico, quer a nível inter-genómico, desde eucariotas a procariotas, desde regiões codificantes a regiões conservadas. Tal conhecimento é decisivo quando se constroem, por exemplo, modelos probabilísticos de predição de sequências para o desígnio da compressão da informação. A análise estatística das ocorrências da sequência de símbolos que compõem um gene ou um genoma (ver Figura 2.9), podem ser organizadas para construir modelos probabilísticos de predição. Nesses casos, a análise estatística é realizada on-line e a predição dos símbolos é baseada num historial, que corresponde à base de conhecimento do modelo probabilístico de predição. Sendo o trabalho desenvolvido baseado na cooperação de metodologias para a obtenção de maior eficiência na compressão das sequências de ADN, esta abordagem é naturalmente parte integrante e as metodologias desenvolvidas neste domínio serão devidamente explanadas mais adiante no Capítulo 5. A teoria dos modelos probabilísticos de predição é apresentada no capítulo seguinte, na secção 3.5.2.

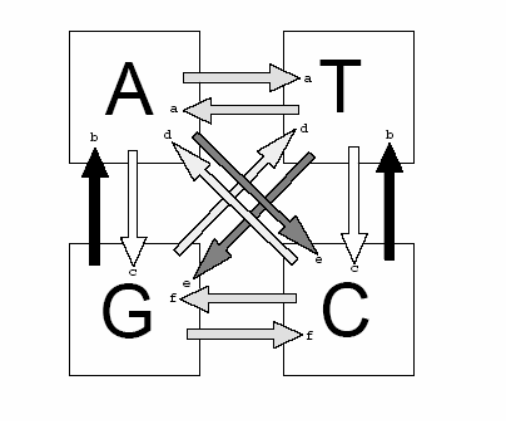


Figura 2.9 – Possibilidades de transições entre nucleótidos do ADN [68].

2.5. Teoria da informação: informação biológica

A molécula de ADN codifica informação que, por convenção, é representada como uma sequência de bases, representadas por caracteres pertencentes ao alfabeto *{A,C,G,T}*.

Assumindo que cada nucleótido, aqui visto como caracter, ocorre com a mesma probabilidade dos demais e que todos os símbolos da sequência são independentes, podemos recorrer à teoria da informação elementar [69, 70] para afirmar que a codificação informática de uma sequência necessitaria de 2 bits por caracter. Este seria o caso da entropia máxima.

Entropia Máxima = $\log_2(n)$, onde n é o número de estados possíveis.

Exemplos:

- Representação dos resultados de lançar uma moeda ao ar $\rightarrow \log_2(2)=1$ bit
- Representação de um nucleótido numa cadeia $\rightarrow \log_2(4)=2$ bits
- Representação de uma amino-ácido numa cadeia $\rightarrow \log_2(20)=4.32$ bits

Na realidade a molécula de ADN deve ser entendida como altamente ordenada, objectivamente desenhada, e fruto de uma depuração por evolução. Assim, é legitimamente expectável a existência propriedades estatísticas, regularidades, similaridades, interdependências, etc., que nos permitam aliviar a entropia na sua representação e qualificar as sequências naturais de ADN como não aleatórias.

Um pressuposto comum mas errado assenta na presunção de que as regiões codificantes do ADN são por natureza mais compressíveis, isto porque são formadas por codões, portanto 64 combinações de três nucleótidos, que se destinam a codificar apenas 20 amino-ácidos, sendo certo que acrescem os codões de *start* e *stop*, três no total. Assim, teremos que são 61 os codões que dão conteúdo aos exons, e calculando o nível máximo de entropia por nucleótido para esta realidade temos que $\log_2 61/3 \approx 1.977$ bits. De facto, as regiões codificantes são menos compressíveis que as regiões não codificantes (introns), justificando-se em grande medida pelas abundantes repetições ou repetições aproximadas presentes nos introns, assim se verifica em diversos estudos [45, 46].

A complexidade de uma sequência pode ser definida como a riqueza do seu vocabulário. Importa determinar quantas palavras diferentes de tamanho *i* aparecem na sequência. Esta abordagem à complexidade linguística foi introduzida por Trifonov em [71].

As regiões de baixa complexidade das sequências de ADN [72, 73] são de elevada importância para a problemática da compressão [74], isto porque encerram maior redundância qualitativa ou quantitativa, onde se destacam as repetições simples ou inversas, exactas ou aproximadas.

A compressão de sequências de ADN é um desafio interessante mas de grande dificuldade, nos últimos anos, os algoritmos de maior sucesso superaram os seus antecessores em menos de 0,01 bits por base, o que é demonstrativo da dificuldade inerente a esta empresa.

2.6. Entropia na codificação de sequências genómicas

Foi em 1972, que Gatlin publicou o livro seminal [75] para a introdução da análise de entropia nas sequências de ADN. Nesse trabalho, explorou-se a relação entre a teoria da informação e a informação biológica, bem como a aplicabilidade dos conceitos de entropia na análise de sequências de ADN. Seguiram-se diversos estudos que exploraram as aplicações e potencialidades desta nova via de investigação, desde a predição de genes, a segmentação, a compressão da informação biológica ou o estudo da importância das regiões repetitivas do ADN na estimação da sua entropia [76-79]. Em suma, todos estes trabalhos procuram caracterizar a complexidade do código genético, porém dada a multiplicidade de factores que concorrem para essa complexidade, como é o caso do codões sinónimos, as replicações, as delecções, os palindromas, etc. O trabalho é árduo e os avanços muito lentos.

A entropia, a redundância, a diferenciação probabilística, a previsibilidade, a complexidade e a compressão são conceitos relacionados. Estudar estes conceitos tendo como campo de aplicação as sequências de ADN é um desafio, basta pensar que as sequências de péptidos apresentam em média 99% de imprevisibilidade [80, 81], pelo que ainda não se entende praticamente nada da complexidade da sua linguagem, ou que o ADN dos organismos menos complexos como vírus e bactérias apresenta níveis de entropia muito próximos dos 2 bits por base [82]. Esta subtileza do código da vida é uma verdadeira lição, sujeita a milhões de anos de evolução, de como desenhar códigos

simples, fáceis de replicar, com capacidade para muita informação e razoavelmente resistentes a erros, é um balanço virtuoso entre fidelidade e erro.

Uma via para comprimir a informação do ADN natural usa a obtenção de um estimador de entropia que propicie a predição com um grau de acerto elevado de uma determinada sequência baseando-se na porção conhecida com antecedência [83]. O ADN natural inclui repetições aproximadas, um facto bem conhecido e documentado assim como as propriedades estatísticas dessas ocorrências [84].

2.6.1.1. Entropia segundo a teoria de Shannon

Recorrendo à teoria da informação [70] podemos enunciar o suporte teórico para a compreensão e importância da estimação da entropia.

Seja S uma fonte discreta emitindo símbolos X_i. Os símbolos X_i são modelados como variáveis aleatórias com realizações no alfabeto finito X de comprimento $|X|=L$. O *output* da fonte é uma sequência de variáveis aleatórias $X_1,X_2,...,X_n$ e pode ser modelado como um processo estocástico com entropia:

$$H(S) = \lim_{n\to\infty} \frac{1}{n} H(X_1,...,X_n) \qquad (2.1)$$

Um limite superior para a entropia pode ser definido por estimativa assumindo uma fonte estacionária, desprovida de memória, com um alfabeto de símbolos uniformemente distribuídos. No caso do ADN, esse limite superior pode corresponder aos 2 bits por símbolo, e atendendo a que estamos a lidar com o código da vida, no qual é expectável alguma ordem, é presumível que a entropia real possa ser inferior [85]. O teorema fundamental de Shannon [41] na compressão de informação estabelece que a informação de qualquer fonte pode ser comprimida sem perdas na proporção da sua taxa de entropia. Portanto, a taxa de compressão em bits por símbolo conseguida por um algoritmo de compressão óptimo para uma mensagem s gerada por uma fonte S, é uma boa aproximação à sua verdadeira taxa de entropia, definida pela Equação 2.2, onde |.| representa o tamanho do conjunto em bits ou símbolos.

$$H(S) \approx \frac{|comp(s)|}{|s|} \qquad (2.2)$$

A entropia e a compressão de informação são conceitos necessariamente interligados. Estimar a entropia de uma sequência de ADN [45] com acuidade permite-nos aventar assertivamente o grau de compressão augurável na recodificação compacta da sequência.

Se o código ADN fosse uma *string* puramente aleatória então a entropia seria máxima e a melhor maneira de a representar seria usando dois bits por cada um dos quatro diferentes símbolos, porém existem regularidades, propriedades específicas da sequência estatisticamente comprováveis, que provam a existência de entropia redutível [86]. O grau de redutibilidade da entropia do ADN está ainda por descobrir, abrindo caminho a uma investigação que facultará um melhor conhecimento das razões pelas quais a Natureza usa tão peculiar código, conduzindo a descobertas genómicas/filogenéticas e necessariamente à compressão da informação.

Para avaliar a dificuldade de compressão das sequências de nucleótidos podem-se efectuar as seguintes simulações, tendo em conta análises da entropia prevista. Porém, é sabido que entropia e compressão estão directamente relacionadas, mas efectivamente, por muito bom que o compressor seja, nem sempre consegue atingir a compressão prevista teoricamente pela análise de entropia.

Recordando a fórmula da entropia da informação de Shannon [41], tem-se na Equação 2.3:

$$H = -\sum_{i}^{n} p_i \log_2 p_i \qquad (2.3)$$

Analisando uma sequência de n símbolos, a entropia (H) é o somatório negativo das probabilidades de ocorrência de um símbolo p_i multiplicadas pelo binómio ($log_2\ p_i$). Para clarificar o conceito seguem-se alguns exemplos elucidativos. Considerando os lançamentos de uma moeda "justa", as probabilidades para cara e coroa seriam as

mesmas (50%), assim pela aplicação da fórmula de Shannon, presente na Equação 2.3, ter-se-ia:

$$- ((0.5)(-1) + (0.5)(-1)) = 1 \text{ bit}$$

Considerando uma moeda "viciada", na qual as ocorrências de cara fossem caracterizadas por uma probabilidade de 75% e a face da coroa ocorresse com 25% de probabilidade, então a entropia seria mais reduzida:

$$- ((0.75)(-0.415) + (0.25)(-2)) = 0.81 \text{ bits}$$

No caso do ADN, considerando o alfabeto quaternário $\sum=\{a,c,t,g\}$, e considerando uma sequência aleatória, sem diferenças probabilísticas entre as quatro bases teríamos:

$$- ((0.25)(-2) + (0.25)(-2) + (0.25)(-2) + (0.25)(-2)) = 2 \text{ bits}$$

Considerando que a probabilidade de ocorrência de A ou T é de 90%, sendo de apenas 10% a probabilidade de C ou G, então a entropia seria reduzida para:

$$- (2(0.45)(-1.15) + 2(0.05)(-4.32)) = 1.47 \text{ bits}$$

Pois bem, nas sequências naturais de ADN, as probabilidades de ocorrência das bases são muito similares, de facto, as diferenças entre elas não ultrapassam geralmente os 5%, pelo que se compreende o elevado grau de entropia associado e como tal o elevado grau de dificuldade para efectuar a sua compressão.

2.6.1.2. Complexidade algorítmica, a entropia segundo Kolmogorov

A entropia, na visão de Kolmogorov, entendida como a entropia algorítmica de um objecto finito define-se como sendo a dimensão do menor programa de computador que permita gerar completamente esse objecto [87]. Assim, considerando uma string x de tamanho l e uma máquina de Turing universal U, usando o programa p, a complexidade K é dada por:

$$K(x) = \min\{l(p) : U(p) = x\} \qquad (2.4)$$

Esta metodologia para a avaliação de entropia é necessariamente dependente da evolução dos computadores e das linguagens de programação dos mesmos. *K* é incomputável e a sua determinação pode apenas ser aproximada. É norma avaliar esta entropia algorítmica por aproximação usando as contribuições das melhores ferramentas informáticas de compressão disponíveis. Apenas no limite superior é possível determinar o valor de *K*, empregando para tal a seguinte expressão:

$$K(n) \underset{-k}{\overset{+}{\leq}} \log_2 n + 2\log_2 \log n \qquad (2.5)$$

As sequências de ADN são passíveis de análise com recurso à teoria da complexidade de Kolmogorov [88]. Estudos de análise de similaridade para a determinação de sequências ortólogas[1] ou de correlações filogenéticas demonstraram que a aferição e comparação de entropia é um factor válido e robusto para determinar similaridade [89-91].

Em suma, neste trabalho terão de conviver as análises das sequências de ADN quer pela perspectiva da teoria de informação de Shannon quer pela teoria algorítmica da informação de Kolmogorov, que no fundo apontam para uma mesma realidade, apenas vista de ângulos diferentes. A primeira do particular para o geral e a segunda no sentido reverso. Importa referir que a teoria de Kolmogorov carece ainda do desenvolvimento e formalismo que a teoria de Shannon, por via da sua maior antiguidade, já conquistou.

2.6.2. Estimadores de entropia em sequências de ADN

Se bem que os conceitos se confundem amiúde na bibliografia, existe em termos conceptuais uma diferença subtil entre o programa estimador de entropia e o compressor. É igualmente comum estimar-se a entropia baseando-se em compressores mas, como antes referido, no caso particular das sequências de ADN, alguns compressores excedem na sua representação a entropia máxima de código do ADN,

[1] Denominam-se ortólogas as sequências ou regiões do genoma de diferentes espécies que evoluíram de um ancestral comum ou que manifestam a mesma função biológica.

pelo que a estimação feita por esta via é falaciosa. O estimador de entropia não tem como objectivo último produzir um ficheiro compacto. Assim, pode descurar a componente de codificação dos elementos redundantes, centrando-se na descoberta desses mesmos elementos redundantes. Obviamente, em termos científicos a estimação da entropia será a fase mais importante da compressão pois providencia a "matéria-prima" necessária a jusante. Depois de completada essa fase, utilizam-se uma série de heurísticas juntamente com códigos, normalmente auto-delimitados, para recodificar a informação na forma compacta. Neste trabalho, a maior ênfase é colocada na estimação da entropia, no Capítulo 5 são descritas as metodologias desenvolvidas quer para a estimação e avaliação da entropia quer para a (re)codificação de sequências genómicas.

Apresentam-se a seguir menções a alguns dos estudos mais representativos que foram conduzidos para avaliar e estimar a entropia das sequências de ADN:

• Em 1995, Farach, M. *et al.* [46] propuseram um novo método para estimar a entropia do ADN chamado de *match length entropy estimator*. Este método foi usado para comparar as diferenças entre exons e introns e, ao contrário do esperado, depararam-se com o factos de que a entropia dos exons era 73% das vezes maior do que a dos introns e que a variabilidade dos introns era 80% das vezes maior do que a dos exons.

• Em 1999, Lowenstern, D. e Yianilos, P. [83] criaram o programa CDNA para estimar a entropia de uma sequência de ADN. A observação básica utilizada por eles foi a de que sequências de ADN contêm muito mais repetições aproximadas (*near repeats*) do que seria normalmente expectável. O CDNA utiliza dois parâmetros para capturar as repetições aproximadas: (w) o comprimento da substring e (h) a distância de Hamming [92].

• Em 2000, Lanctot, K., Li, M., e Yang, E.H. [86] baseados nas ideias de Kieffer e Yang [93, 94], publicadas pouco antes, sobre códigos baseados em gramáticas que reconhecem repetições e reversões complementares de sequências de DNA, desenvolveram um programa para estimação da entropia de sequências de DNA chamado GTAC (*Grammar Transform Analysis and Compression*).

Resumindo, neste capítulo fez-se um enquadramento dos objectivos deste trabalho na tipologia da informação biológica, procurando reunir os aspectos teóricos e os avanços na investigação que permitem caracterizar a natureza da informação presente nas sequências de ADN. Comprimir é antes de mais compreender, e em suma, neste capítulo pretendeu-se dar a compreender as especificidades das sequências de ADN abordando-as desde a perspectiva biológica até à perspectiva da teoria geral da informação. A ênfase foi colocada nos elementos repetitivos, que em grande maioria compõem o ADN dos eucariotas, e que são a fonte para a obtenção do sucesso na compressão, que se pretende atingir reduzindo a entropia por via da recodificação mais eficiente destas redundâncias.

Capítulo 3. Metodologias de análise e compressão de sequências genómicas

Qualquer metodologia de compressão de informação recorre à identificação e redução de redundâncias. A primeira parte, a da identificação é tão importante como a segunda, que corresponde à codificação reduzida, mas não redutora. Neste capítulo, primeiramente, serão expostas as metodologias para análise, identificação e pesquisa sistemática de elementos redundantes, que por simplicidade serão denominados de padrões. As regiões repetitivas do ADN adquirem esse estatuto por apresentarem recorrências de padrões, exactos ou inexactos, de forma adjacente ou dispersa.

Neste capítulo figurarão igualmente, e em sucessão lógica, as metodologias de compressão mais vocacionadas para lidar com a especificidade da linguagem e do alfabeto genómico. Como se explicitou no modelo conceptual que serve de base ao trabalho desenvolvido, a metodologia de compressão adoptada é híbrida, tal escolha pretende tirar o melhor partido das regiões repetitivas e não repetitivas do ADN. Se nas primeiras se adequa melhor uma técnica baseada em dicionário, já nas segundas, um método probabilístico de predição, baseado em modelos de linguagem, pode obter melhores resultados. Sem desprimor para os demais métodos, será concedida maior ênfase às metodologias adoptadas, justificando a eleição.

3.1. Pesquisa de padrões em sequências biológicas

A periodicidade de subsequências de nucleótidos é recorrente no ADN natural, especialmente nos seres eucariotas. As correlações variam em dimensão, desde os codões [91, 95] (3 bp) que codificam os aminoácidos das proteínas até às correlações longas [5, 96] (~10^6 bp) que ocorrem nas regiões não codificantes.

No domínio da biologia molecular, a comparação de sequências biológicas e a aferição de padrões recorrentes, forneceu pistas e respostas a muitas questões relacionadas, por exemplo, com a evolução e as doenças. Dum ponto de vista mais informacional, os padrões são fundamentais na inferência filogenética, na compressão da informação biológica, na segmentação das sequências ou na descoberta de *motifs*. As metodologias utilizadas na comparação de sequências estão orientadas, sobretudo, para duas realidades concretas. A pesquisa de padrões exactos e a pesquisa de padrões aproximados.

A pesquisa de padrões exactos [97-99], teoricamente mais simples e de menor importância relativa em bioinformática, prende-se com a identificação de segmentos de uma sequência, que se replicam com exactidão noutros locais da mesma sequência ou noutras sequências relativas a outros seres da mesma espécie ou de diferentes espécies. No âmbito da bioinformação, as repetições exactas também podem aparecer nas formas inversa ou em palindromas, no primeiro caso as bases repetem-se mas de forma simétrica, no segundo caso, além de se apresentarem na ordem inversa estão substituídas pelas bases complementares. A pesquisa de padrões exactos é parte integrante, na fase inicial, da pesquisa de seeds em algoritmos de pesquisa de similaridade e homologia como é o caso da ferramenta bioinformática BLAST [100].

A pesquisa de padrões aproximados [101-103] é crucial em bioinformática, recorrente em vários domínios de investigação desde a biomedicina à filogenia. Trata-se de identificar segmentos das sequências biológicas que apresentam réplicas aproximadas ou cópias imperfeitas ou mutadas noutras regiões do genoma ou em diferentes genomas. As repetições aproximadas incluem, igualmente, as repetições aproximadas de padrões escritos inversamente ou em palindromas. Para avaliar a similaridade entre duas *strings*

torna-se necessário estabelecer a definição de similaridade (estabelecer as condições a verificar) e concomitantemente descrever as medidas que quantificam as distâncias de edição que permitem igualar as *strings* aproximadas.

3.1.1. Localizar padrões exactos

Apesar dos padrões exactos serem de menor relevância que os inexactos na análise de sequências genómicas, ainda assim, constituem um recurso importante na análise de similaridade. Para além da identificação de padrões exactos nas sequências, por exemplo a maioria dos *motifs* e os ETR (*exact tandem repeats*), de facto, as *seeds* (sementes) que alguns algoritmos de alinhamento de sequências utilizam para determinar alinhamentos similares, são padrões exactos que devem verificar-se nas sequências a alinhar. Sendo a partir dessas sementes que os alinhamentos se estendem pelas bases que as ladeiam dentro de determinado limite de discrepância ou *threshold*. O caso mais proeminente é o BLAST [100], o algoritmo mais utilizado e citado em alinhamentos locais de sequências biológicas, que recorre a sementes exactas e contínuas.

A função do *pattern-matching* é pesquisar as ocorrências do padrão p, de tamanho m, numa sequência x, de tamanho n, sendo $n \geq m$. As *strings* p e x são produtos de um alfabeto finito Σ, composto por σ símbolos, normalmente entendidas como justaposições dos símbolos de Σ.

A pesquisa de padrões exactos é uma recorrência dentro da computação passada, presente e seguramente futura. É essencial na pesquisa de informação na *web*, na edição e processamento de textos, na exploração de bases de dados, na análise anti-vírica exercida pelo software de segurança, etc. Apesar da sua necessidade ser contemporânea da criação dos computadores electrónicos, foi apenas no final dos anos setenta que os algoritmos de pesquisa de padrões exactos se tornaram sub-lineares no seu desempenho.

Em boa medida, a introdução de uma fase de pré-processamento foi a inovação decisiva para atingir a sublinearidade, já que conhecer e estudar a composição do padrão pode resultar em conhecimento decisivo para pesquisas mais selectivas, logo mais eficientes.

Em 1977 Knuth, Pratt e Morris (KMP), propuseram um algoritmo de pesquisa de padrões exactos inovador [104], que efectuava um pré-processamento ao padrão no sentido de compor uma tabela de avanços que utilizaria no caso de falho na comprovação de ocorrência num qualquer tentativa ou iteração, usando o conhecimento da causa da falha. No mesmo ano, Boyer e Moore, apresentaram um algoritmo muito eficiente em alfabetos de linguagens naturais [97]. Era baseado em duas heurísticas, cuja avaliação efectuada em pré-processamento permitia deslocar a janela de pesquisa no caso de ocorrer um falho ou a descoberta de uma ocorrência do padrão, usando a heurística mais vantajosa. Na década de 80, outros investigadores propuseram variantes optimizadas desse algoritmo que ainda hoje se mantêm com desempenhos competitivos, sendo mesmo usadas em múltiplas aplicações do nosso dia a dia.

Por ser uma base de compreensão do funcionamento da generalidade dos algoritmos de pesquisa de padrões exactos, importa descrever em maior detalhe o algoritmo de Boyer-Moore. A estratégia de pesquisa deste algoritmo inicia-se pelo alinhamento do padrão com o início da sequência, essa será a primeira janela de pesquisa. As janelas de pesquisa usadas no algoritmo BM são de tamanho m. Em cada janela as comparações de caracteres para descoberta de instâncias do padrão são desenvolvidas da direita para a esquerda, ou seja a primeira comparação é a verificação do último caracter do padrão com o último caracter da janela de pesquisa. As comparações sucedem-se até ao máximo de m se não ocorrer qualquer falha, nesse caso ter-se-á descoberto uma réplica do padrão p. Normalmente, os padrões são dispersos e o caso mais frequente é uma tentativa falhada num qualquer caracter onde a correspondência não se verifica. Seja qual for o caso, a seguir é necessário avançar a janela de pesquisa para continuar o percurso de x.

O algoritmo de BM incorpora duas heurísticas para fazer o avanço seguro da janela para a próxima iteração. O avanço seguro implica uma deslocação máxima da janela sem, no entanto, perder qualquer ocorrência do padrão. As duas heurísticas utilizadas são: (*i*) a regra do bom sufixo; e (*ii*) a regra do mau caracter. As duas regras são concorrentes no sentido em que apenas é usada a que, para determinadas circunstâncias de término de uma iteração, providenciar o maior avanço seguro.

A regra do bom sufixo faz uso de uma tabela calculada na fase de pré-processamento que prevê o avanço seguro no caso de se verificar um dos possíveis sufixos parciais ou o total, por total entenda-se uma ocorrência do padrão. Esta regra baseia-se no facto de que, tendo-se conhecimento de que uma fracção do padrão na sequência correcta se encontra presente em determinada localização de *x*, então será possível determinar se uma ocorrência de *p* é possível com essa presença, ou se essa presença torna impossível uma ocorrência de *p* partilhando o mesmo espaço do sufixo verificado. Para clarificar esta regra veja-se a Tabela 3.1, construída com base nesta regra para os sufixos do padrão *p*=”*atgat*”. Exemplificando, após uma ocorrência do padrão, a janela pode deslocar-se três bases/caracteres pois o último dupleto do padrão pode vir a ser o primeiro da próxima ocorrência.

sufixo	*atgat*	*tgat*	*gat*	*at*	*t*	
caracteres verificados no sufixo	5	4	3	2	1	0
avanço seguro	3	3	3	3	5	1

Tabela 3.1 - Tabela resultante do pré-processamento da regra do bom sufixo.

A regra do mau caracter é mais simples, basicamente prevê para cada caracter cuja verificação de correspondência falhou, um avanço seguro baseado nesse facto. As verificações falham por um de dois motivos, ou dado caracter que faz parte do padrão está mal situado ou o caracter encontrado nem sequer faz parte do padrão. Por exemplo, se a verificação falhou porque foi encontrado um caracter que não integra o padrão, então é seguro afirmar que uma nova ocorrência do padrão apenas pode iniciar-se depois desse caracter. Para armazenar o conhecimento a aplicar por esta regra constrói-se uma tabela de avanço por cada símbolo do alfabeto. Para o padrão anteriormente utilizado *p*=”*atgat*”, os valores de avanço por símbolo constam na Tabela 3.2.

a	*c*	*t*	*g*
1	5	3	2

Tabela 3.2 - Tabela resultante do pré-processamento da regra do mau caracter.

Munido destas duas tabelas, o algoritmo de BM parte para a fase da pesquisa iterativa dos padrões com os recursos necessários para fazer com que a pesquisa seja mais

eficiente. No melhor caso, o algoritmo de BM apresenta um complexidade temporal de *O(n/m)* e no pior caso de *O(nm)*. Em média este algoritmo apresenta uma complexidade inferior a *O(n)*, pelo que é considerado sublinear. Em 1980, Horspool produziu um versão simplificada do algoritmo de BM que na prática apresentava melhor desempenho [105]. Assim, o algoritmo BMH, descarta a regra do bom sufixo e aplica apenas a regra do mau caracter. Desse modo, o avanço usado nem sempre será o máximo possível mas, evita-se o *overhead* introduzido pela comparação e selecção do maior avanço de entre os fornecidos pelas duas heurísticas como acontece no algoritmo original. O algoritmo de BMH constitui ainda hoje uma referência de simplicidade e desempenho na pesquisa de padrões exactos em textos de linguagem natural ou outros de alfabetos maiores.

Em 1998, Navarro e Raffinot apresentaram uma solução híbrida, com recurso a operações de *bitwise*, aproveitando o paralelismo possível nas operações com bits usando o comprimento de palavra permitido pela arquitectura dos computadores, usualmente limitado a 32 ou 64 bits. Combinaram essa faceta, com a simulação de autómato finito não determinístico para a identificação de sufixos do padrão a pesquisar. Se o sufixo fosse total, então teria sido descoberta um instância do padrão, caso fosse apenas parcial, então serviria para determinar qual o avanço seguro da janela de pesquisa para iniciar, mais adiante na sequência, novo exame de ocorrência do padrão. Denominaram o seu algoritmo de BNDM [98], tendo concluído que ostentava um desempenho muito flexível em todos os alfabetos, revelando-se como mais eficiente em alfabetos de menor dimensão. A supremacia, segundo os seus autores, cifrava-se num ganho de eficiência entre os 10 e 40 % em média, se comparado com os melhores concorrentes, que correspondem às variantes optimizadas do algoritmos de BM, mais concretamente o algoritmo BMH [105] e o algoritmo Quick Search de Sunday [106].

Em 2003, o algoritmo BNDM foi optimizado, no sentido da simplificação, por Peltola e Tarhio e publicado em [107]. A simplificação correspondeu igualmente a um aumento de desempenho. A nova variante, denominada SBNDM, apresenta um acréscimo de eficácia sobre o seu precursor de 10%.

Os algoritmos de pesquisa de padrões exactos são normalmente centrados em pesquisas e comparações elementares, correspondendo normalmente a um caracter. Porém, por forma a maximizar o desempenho, existem versões destes algoritmos que efectuam a

pesquisa baseada em sub-segmentos de múltiplos caracteres contíguos, os denominados *q-grams, n-gramas* ou *sub-strings* com tamanho reduzido, normalmente entre 2 e 6 caracteres, sendo que essa dimensão depende fortemente do tipo de aplicação em causa.

3.1.1.1. O recurso a *q*-gramas para elevar o desempenho

Qualquer algoritmo de identificação de padrões recorre a comparações. O senso comum poderá induzir a ideia de que, comparando um caracter de cada vez se evitam pesquisas redundantes imediatamente a seguir ao caracter que dita a impossibilidade de existência do padrão na janela de pesquisa. Na prática isto nem sempre se verifica, de facto quando envolvidos alfabetos curtos utilizar *q*-gramas como base de comparação na fase de filtragem demonstra ser rentável. Basicamente, *q*-gramas são *sub-strings* de tamanho *q*, e a razão de os usar será a de artificialmente aumentar o tamanho do alfabeto ganhando com isso maior capacidade de decisão eliminatória de janelas. Na grande maioria das pesquisas, a frequência com que se identificam padrões é muito baixa, sendo que a grande maioria das janelas se há-de de declarar infrutífera Utilizar filtros baseados em *q*-gramas é rentável pois abrevia os testes de despistagem necessários. De igual modo, os casos em que se torna necessária uma verificação caracter a caracter depois de o filtro ter detectado uma possível ocorrência reduzem-se, o que evita um conjunto significativo de instruções para testar, no pior dos casos (para o desempenho), integralmente a janela.

A ideia de usar *q*-gramas não é nova, de facto já Boyer e Moore em 1977 [97] se referiram a essa possibilidade, desenvolvida mais tarde por Baeza-Yates em 1989 [108] e continuada por vários outros em anos recentes [109]. Usar *q*-gramas em *pattern-match* não é, no entanto, a solução universal. Em alfabetos maiores a sua utilidade dilui-se [110], e mesmo considerando pesquisas em textos gerados de pequenos alfabetos, o valor de *q* terá de ser reduzido, normalmente entre *2* e *6* face às pesquisas de padrões mais usuais, i.e., com padrões de comprimento *m* com $2 \leq m \leq 255$. Obviamente um algoritmo de *pattern-matching* baseado em *6*-gramas não será apto para detectar padrões com $m<6$, pelo que essa é uma limitação. Desenhar um algoritmo adaptativo para o tamanho de *q* a empregar pode ser uma solução, contudo tal adaptabilidade constituirá sempre uma sobrecarga para o desempenho.

Existem diversos estudos cujo cerne se baseia na reengenharia de metodologias mais clássicas de pesquisa de padrões [108, 110, 111], adaptando-as para funcionarem com *q*-gramas, pesquisando, por assim dizer, baseando-se em super-alfabetos. Algoritmos anteriormente descritos na secção anterior como o de BM ou o BNDM foram testados com *q*-gramas e o seu desempenho acentuou-se em situações de pesquisa de textos originados por alfabetos curtos, tal como acontece com as sequências de ADN. Recentemente e com resultados que superaram o estado-da-arte em casos pontuais, Lecroq publicou resultados interessantes na adaptação do algoritmo de Wu e Manber [112] - um descendente do Shift-Or [113] orientado para pesquisa de padrões múltiplos - para incorporar um filtro baseado em *q*-gramas com $3 \leq q \leq 8$. Este algoritmo, denominado de WML [110], demonstrou ser o mais rápido considerando alfabetos e padrões curtos (tais como código binário ou ADN), superando inclusivamente os algoritmos que recorrem a técnicas de *bitwise*. O WML executa um pré-processamento do padrão de modo a conhecer todos os avanços passíveis de aplicar na fase seguinte tendo em com o tamanho da *sub-string* usada como filtro. Assim, é realizada uma operação de *hashing*, através de uma função *h*, usando valores entre 0 e 255 (para alfabetos com, no máximo, esta dimensão). Mais tarde na fase de pesquisa se o avanço possível perante uma janela de pesquisa for zero então o padrão é testado caracter a caracter, de outro modo é aplicado imediatamente o valor do avanço pré-computado para a ocorrência do *q*-grama encontrado. Trata-se de um algoritmo simples mas ao mesmo tempo de elevado desempenho.

Actualmente as aplicações bioinformáticas recorrem a implementações de BNDM com *q-grams*, para obter desempenho elevado na identificação de padrões exactos. É exemplo o software PatMatch [114] de 2005.

3.1.2. Localizar padrões aproximados

A pesquisa de padrões aproximados em sequências biológicas é nuclear. Por definição as sequências biológicas são mutáveis nos processos biológicos que as regem e que as propagam. Os codões sinónimos indicam-nos que a mutação numa base pode não ter afectação na proteína codificada. Existem várias degenerações aceitáveis na biologia molecular que não põem em causa similaridades e homologias. Deste modo, a pesquisa

de padrões aproximados é de extrema relevância na análise funcional de sequências. No âmbito da compressão de informação biológica, o salto qualitativo que marcou a evolução recente dos compressores de genomas foi a adopção de padrões aproximados.

Os algoritmos conhecidos para a pesquisa de padrões aproximados dividem-se e caracterizam-se pela metodologia utilizada, tais como:
- programação dinâmica;
- baseados em heurísticas ou de filtragem;
- métodos probabilísticos.

As secções seguintes estão devotas às diversas metodologias de pesquisa de padrões aproximados, que em bioinformática de designam habitualmente por análise de similaridade. Se a similaridade é dentro da mesma sequência trata-se de similaridade intra-genómica, se for entre várias sequências designa-se por inter-genómica.

3.2. Metodologias para a análise de similaridade inter e intra-sequências biológicas

No âmbito da bioinformática, similaridade e homologia não são sinónimos. A análise de homologias ocupa-se essencialmente de descortinar estruturas funcionais ou domínios conservados recorrentes em diferentes genomas. Por outro lado, a similaridade entre duas sequências não encerra apenas o cariz funcional ou evolutivo, é uma perspectiva mais abrangente, que extravasa os *motifs* ou os genes, comparando integralmente, de um ponto de vista informacional, regiões de uma sequência, ou várias sequências. Neste trabalho, importa primordialmente estudar a similaridade intra-sequências, no intuito de identificar exaustivamente toda e qualquer réplica, exacta ou aproximada, de segmentos repetitivos do ADN, sem restrições ou limitações impostas pela lógica funcional da biologia molecular. O objectivo maior é, considerando o desiderato da compressão da informação biológica por dicionário de recorrências, construir o dicionário mais vantajoso possível para maximizar a compressão, abstraindo-nos da especificidade da informação e atendendo apenas à teoria da informação abstracta. Seguramente, alguns dos padrões que irão constar no dicionário poderão, à luz do conhecimento genómico

actual não receber uma leitura funcional, mas de futuro, com as descobertas vindouras, a percentagem de padrões considerados aleatórios irá ser reduzida.

Embora as sequências genómicas sejam replicadas e passadas às gerações seguintes com elevada fidelidade, ocorrem erros e mutações, que evitam que as cópias sejam exactas para o bem (evolução) e para o mal (doenças). Assim sendo, a similaridade entre sequências deve ser entendida pela ideia de proximidade, que obedece a um conjunto de métricas de espaço:
- positividade: a distância entre duas sequências é sempre maior ou igual a zero (será zero se forem iguais);
- simetria: a distância da sequência *x* para a sequência *y* é a mesma que existe entre *y* e *x*;
- desigualdade triangular: se duas sequências são similares a uma terceira, então não podem ser dissimilares entre si.

A análise de similaridade e alinhamento inter e intra-genómico é fulcral em bioinformática [115]. Corresponde igualmente a um problema complexo que motiva uma comunidade cada vez maior de investigadores na procura de soluções com melhor desempenho, temporal e espacial, que permitam inferir com maior rapidez e acuidade homologias no ADN na era pós-genómica.

A programação dinâmica, nomeadamente o algoritmo de Smith-Waterman [116] provê uma solução algorítmica exacta para o problema de determinar alinhamentos locais óptimos. Contudo, a complexidade quadrática ($O(n^2)$) associada, motivou a criação de heurísticas mais rápidas, com concessões ao nível da sensibilidade é certo, mas garantindo ainda sim, soluções aproximadas satisfatórias na maioria dos casos. A ideia chave para reduzir racionalmente a informação a analisar consiste na filtragem dessas sequências, apurando as regiões que evidenciam sinais de similaridade, ou acertos (*hits*), que correspondem a repetições de padrões primários. Estes padrões primários, normalmente de dimensão reduzida (até 25 nucleótidos), vulgarmente designados por sementes (*seeds*), repetem-se inter e intra-genomicamente. A recolecção destas *seeds* deve ser realizada com critérios de selectividade de forma a não incorrer em *hits* redundantes. Numa fase posterior estas *seeds* podem confirmar-se ou não como reveladoras de regiões mais ou menos extensas de similaridade. As metodologias

diversas de pesquisa e selecção de *seeds* [117], bem como a configuração e os modelos das próprias *seeds*, são elementos distintivos dos algoritmos de análise de alinhamentos de sequências que usam filtragem. As metodologias implementadas no BLAST [100], PatternHunter [118, 119] e LAGAN [120], constituem exemplos relevantes de algoritmos de análise de sequências baseados em filtragem com heurísticas.

O termo sensibilidade compreende a avaliação dos alinhamentos detectados pelos métodos heurísticos face aos óptimos. Se determinado método atinge sensibilidade de 70%, corresponde a que identifica 70% das regiões similares realmente existentes, comprovadas por um algoritmo exaustivo ou óptimo (que identifica 100%) baseado em análise integral da sequência recorrendo a programação dinâmica.

3.2.1.1. Distâncias e medidas de similaridade

A análise de similaridade assenta essencialmente na distância de edição para alinhar duas sequências. A distância de edição corresponde às operações de edição a aplicar para que ambas sequências se correspondam. Entre as operações de edição mais usadas em bioinformática, contam-se as inserções, delecções/substituições, e a abertura de intervalos de diferença (*gaps*). As distâncias de Hamming, bem como as de Levenshtein, são vulgarmente usadas como métrica para a análise de similaridade exprimindo a distância de edição.

A distância de Hamming é denotada por d_H e definida pela seguinte recorrência [92]:

$$d_H(X_{1..n}, Y_{1..m}) = \begin{cases} 0 & \text{se } n = 0 \text{ e } m = 0 \\ d_H(X_{1..n-1}, Y_{1..n-1}) + \alpha(X_n, Y_n) & \text{se } n = m \text{ e } n, m > 0 \\ \infty & \text{se } n \neq m \text{ e } n, m > 0. \end{cases}$$

A distância de Levenshtein, vulgo distância de edição, é denotada por d_L e definida pela seguinte recorrência [121]:

$$d_L(X_{1..n}, Y_{1..m}) = \begin{cases} \max\{n, m\} & \text{se } n = 0 \text{ ou } m = 0 \\ \min \begin{cases} d_L(X_{1..n-1}, Y_{1..m-1}) + \alpha(X_n, Y_m) \\ d_L(X_{1..n}, Y_{1..m-1}) + 1 \\ d_L(X_{1..n-1}, Y_{1..m}) + 1 \end{cases} & \text{se } n, m > 0. \end{cases}$$

A distância entre duas cadeias pode ser calculada, para as funções apresentadas, através de programação dinâmica, com uma complexidade temporal de $O(n^2)$.

Quando se comparam sequências, com vista à análise de similaridade, podem empregar-se duas metodologias no que respeita à extensão da análise. A análise de alinhamentos das sequências pode realizar-se global (ver Figura 3.1) ou localmente (ver Figura 3.2).

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||  |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

Figura 3.1 – Alinhamento global.

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Figura 3.2 – Alinhamento local.

Os alinhamentos globais são destinados à comparação de genomas, enquanto que os alinhamentos locais procuram identificar segmentos localizados de sequências ou sub-sequências que apresentam maior grau de semelhança. Como referido anteriormente, este trabalho incide na análise intra-sequência, e cada padrão encontrado corresponde a um alinhamento local. As técnicas de análise de similaridade intra ou inter-sequências dividem-se em três grupos: (*i*) – exaustivas ou óptimas; (*ii*) – heurísticas; e (*iii*) – probabilísticas.

3.2.2. Métodos óptimos ou exaustivos de análise de similaridade em sequências biológicas

Tratando-se de um problema de optimização, o alinhamento óptimo de sequências, usa os algoritmos da programação dinâmica. Com a introdução dos meios computacionais esta técnica recebeu um forte avanço e proporcionou solução a problemas complexos em tempo útil. Contudo, trata-se de uma técnica que assenta na análise exaustiva de

todas as combinações possíveis de alinhamentos, pelo que se caracteriza por processamento intensivo e moroso, agravado pelas extremas necessidades de recursos de memória. Para as especificidades dos alinhamentos globais ou locais existem variantes de algoritmos de programação dinâmica. Na secção seguinte fornecem-se, de forma sucinta, descrições dos fundamentos desses algoritmos.

3.2.2.1. Programação dinâmica

O interesse da programação dinâmica (PD) reside no facto de permitir uma solução eficiente (polinomial) de certos problemas de optimização. A PD é útil em caso de determinação de alinhamentos simples em sequências biológicas, tornando-se proibitiva nos casos de alinhamentos múltiplos. A programação dinâmica [122-125] e o princípio da optimalidade têm origem em 1957, e são devidos a R. Bellman. A técnica de programação dinâmica consiste em resolver uma instância de um problema usando soluções já obtidas para instâncias menores do mesmo problema. Alguns problemas de optimização admitem esta abordagem. Na prática, a programação dinâmica resolve todos os sub-problemas e armazena as soluções numa tabela. Aquando da computação da solução global, baseada na articulação regressiva das soluções parciais, é possível a recuperação dessas soluções, sem que tenham de ser recomputadas.

A programação dinâmica foi adoptada pela bioinformática em 1970, enquadrada no âmbito da optimização de alinhamentos de sequências de forma global, por Needleman & Wunsch [126], secundados por Sellers [127] em 1974, e mais tarde, em 1981, por Smith & Waterman [116], que ajustaram o algoritmo para análise de alinhamentos locais. A análise global de alinhamentos é adequada para comparar sequências inteiras que, de antemão se reconhecem como similares. O alinhamento local permite identificar sub-regiões comuns em diferentes sequências, tendo como objectivo encontrar sub-domínios recorrentes mesmo em sequências com baixa similaridade.

A programação dinâmica é implementada empregando uma relação de recorrência e uma tabela para armazenar os resultados dos sub-problemas. Na tabela assinalam-se os resultados das comparações elementares, que são valorizadas por um esquema de pontuação (*scoring scheme*) que, normalmente, premeia as correspondências e penaliza

as diferenças e as operações de edição que as suplantam. Finalmente, por uma análise regressiva (*traceback*), maximizando um percurso pelas maiores pontuações (*scores*) obtidas, obtêm-se os alinhamentos óptimos, representados pelos nodos desse caminho e pelas operações de edição neles contidas. Quando as sequências são demasiado longas para serem processadas por inteiro, podem subdividir-se em segmentos que são comparados por este método e numa fase ulterior os resultados parciais são analisados para conformação da solução global.

O algoritmo de Smith-Waterman é o que mais nos interessa no âmbito deste trabalho pois corresponde à determinação de alinhamentos locais óptimos. Para descrever com detalhe o seu funcionamento é observado e comentado um exemplo de comparação de duas sequências *S1* e *S2* através da Tabela 3.3, usando a relação de recorrência (3.1) com *d*=*gap* e o esquema de pontuação seguinte.

Esquema de pontuação usado:

Match:2

Mismatch:-1

d (Gap):-1

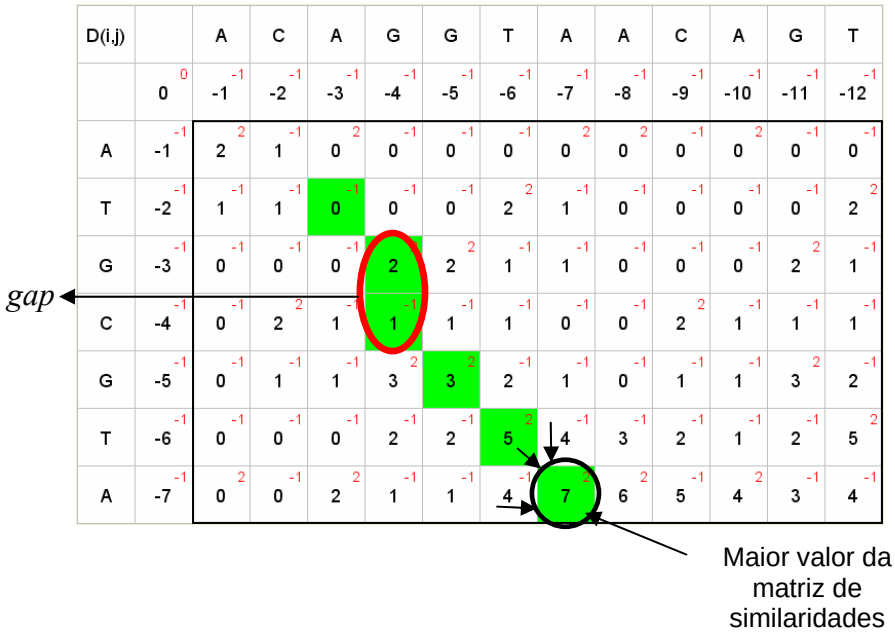
$$F(i,j)=\max\begin{cases}0,\\ F(i-1,j-1)+s(x_i,y_j),\\ F(i-1,j)+d,\\ F(i,j-1)+d.\end{cases} \tag{3.1}$$

A primeira fase ocupa-se do preenchimento da matriz de pontuações, presentes na Tabela 3.3 no canto superior direito de cada célula. Esse preenchimento decorre célula a célula, se houver correspondência (*match*) então é atribuída à célula a pontuação 2, caso contrário é atribuída a pontuação -1. Para o caso da linha e coluna zero são atribuídos os valores definidos para o *gap* (-1) excepto a célula inicial que recebe o valor 0. Os valores da primeira fase correspondem aos números sitos no canto superior direito da tabela do exemplo.

Na segunda fase é feito, com base nos valores obtidos na fase anterior e o valor do *gap (d)*, o cálculo dos valores de similaridade de cada célula da matriz segundo a relação de

recorrência apresentada para o algoritmo de Simth-Waterman (ver Equação (3.1). Em suma, trata-se de apurar o valor máximo proveniente de três possíveis fontes para cada célula: a célula diagonal superior esquerda mais a pontuação da célula, ou a célula à esquerda mais o valor do *gap (d),* ou a célula acima mais o valor do *gap (d)*. Veja-se o exemplo do máximo valor da tabela onde se colocaram setas indicando as três fontes.

A terceira fase corresponde a analisar o caminho de alinhamento óptimo que se inicia na célula com maior coeficiente, usando os coeficientes máximos e as diagonais (*matches*) para chegar à célula (0,0). O caminho percorrido corresponde ao alinhamento local óptimo.



D(i,j)		A	C	A	G	G	T	A	A	C	A	G	T
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
A	-1	2	1	0	0	0	0	0	0	0	0	0	0
T	-2	1	1	0	0	0	2	1	0	0	0	0	2
G	-3	0	0	0	2	2	1	1	0	0	0	2	1
C	-4	0	2	1	1	1	1	0	0	2	1	1	1
G	-5	0	1	1	3	3	2	1	0	1	1	3	2
T	-6	0	0	0	2	2	5	4	3	2	1	2	5
A	-7	0	0	2	1	1	4	7	6	5	4	3	4

Tabela 3.3 – Matriz de similaridades preenchida pelo método de Smith-Waterman.

Melhor alinhamento local pelo algoritmo de Smith-Waterman:

```
Sequência 1 - ACAG-GTAACAGT
                   | |||
Sequência 2 - ATTGCGTA
```

Existem dezenas de variantes da programação dinâmica que procuram resolver outros tantos problemas em bioinformática [128], desde a assemblagem de sequências à análise

de homologias e inferência filogenética. Neste trabalho, confere-se maior atenção às técnicas de análise comparativa de sequências com ênfase no alinhamento local.

3.2.3. Métodos probabilísticos

É possível estudar o historial de uma sequência ao nível das probabilidades de ocorrência dos símbolos que as compõem. Em posse desse historial é possível aplicá-lo a outra sequência e reconhecer quando o historial de uma se repete na outra. Desta forma é possível realizar análise de similaridade com recurso a métodos probabilísticos. Ao nível local é igualmente possível e profícuo estudar regularidades aplicando tal metodologia.

3.2.3.1. Modelos ocultos de Markov

Os modelos ocultos de Markov (HMMs - *Hidden Markov Models*) são amplamente usados na análise de sequências biológicas [129], porém tiveram a sua aplicação anterior no reconhecimento de voz e ainda aí desempenham um papel fundamental.

Os HMMs [130] podem ser entendidos como variantes dos transdutores probabilísticos ou estocásticos de estado finito. O autómato altera o seu estado de acordo com a entrada (*input*) de símbolos que vão sendo examinados. Num determinado estado, o autómato pode inclusivamente exteriorizar um símbolo. Havendo, portanto, um estado inicial, o autómato assumirá estados intermédios e final condicionados por relações de causalidade probabilística ou estocástica.

Os modelos de Markov podem ser usados sempre que se quer modelar a probabilidade de uma sequência linear de eventos. Podem ser representados por um diagrama de estados, como mostra a Figura 3.3, no qual cada estado é nomeado e as transições possíveis entre os estados são representadas por setas identificadas com a probabilidade da transição. As probabilidades dos arcos que saem de cada estado devem somar 1.

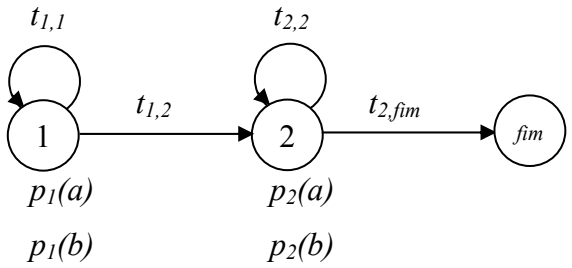


Figura 3.3 – Diagrama de estados e transições modelados por um processo Markoviano.

Assim, o modelo de Markov pode ser visto como um autómato finito (não determinístico), com probabilidades junto a cada arco. Cada estado tem uma tabela com *probabilidades de emissão* de símbolos, e existem *probabilidades de transição* para a movimentação de um estado para outro. O caminho através do modelo começa de um estado inicial, e o próximo passo é escolher um novo estado com uma probabilidade de transição (por exemplo, ficar no estado 1 com probabilidade de transição $t_{1,1}$ ou mover para o estado 2 com probabilidade $t_{1,2}$). Depois, é gerado um resíduo com uma probabilidade de emissão específica daquele estado (escolher um "a" com probabilidade p(a), por exemplo). Este processo é repetido até que o estado final seja alcançado. Como resultado, tem-se uma sequência oculta de estados percorridos (a qual não é observada) e uma sequência de símbolos (que é observada).

De acordo com o modelo (M) apresentado na Figura 3.3, para a sequência de símbolos de saída *a – b – a* (*x*), uma sequência possível de estados ocultos é 1 – 1 – 2 (π). Assim, a probabilidade P da sequência *x* percorrer o modelo M pelos estados π é o produto das probabilidades encontradas pelo caminho, logo,

$$P(x, \pi \mid M) = t_{1,1} * p_1(a) * t_{1,2} \, x \, p_1(b) * t_{2,end} * p_2(a)$$

Resultando que, dada uma sequência de saída, não se sabe a sequência de estados pela qual passa o modelo, mas somente uma função probabilística deste caminho, e por isso o modelo é chamado Modelo Oculto de Markov.

Um Modelo Oculto de Markov (HMM) é, geralmente, uma tripla M=(s, P, B) onde se consideram:

- um conjunto específico O_k de observações do tipo k que resultam de experimentação;
- um conjunto X de estados x_i, onde em cada estado x_i é possível realizar uma observação $b_i(k)$, com i =1.... N e $k \in O_k$;
- uma distribuição de probabilidade para o estado inicial dada pelo vector s = $[s_i]$, onde $s_i = Pr(x_i(0))$;
- uma distribuição de probabilidade para as transições de estados dadas pela matriz P = $[p_{ij}]$, onde $p_{ij} = Pr(x_j(t+1) \mid x_i(t))$;
- uma distribuição de probabilidade para as observações em cada estado dada pela matriz B = $[b_j(k)]$, onde $b_j(k) = Pr(O_k \mid x_j)$.

Em biologia molecular, a utilização dos HMM destina-se à obtenção de um "perfil probabilístico" de uma família de proteínas ou de padrões de ADN, chamado *HMM profile*. É usado para procurar proteínas homólogas em bases de dados, e para descobrir o alinhamento múltiplo de sequências [131]. Também pode ser usado para predição de genes em sequências de ADN [132], tarefa na qual vários sinais genéticos devem ser reconhecidos e combinados para discernir a predição de exons e introns. Na compressão de informação biológica também existem aplicações dos HMMs [45], para tal faz-se o aproveitamento da capacidade de predição probabilística que os HMMs providenciam para estimação da entropia.

3.2.4. Métodos heurísticos

Como já foi abordado, a programação dinâmica é um método óptimo de análise de similaridade, porém tal propriedade, apesar de desejável, não é compatível com as dimensões dos genomas a comparar atendendo aos meios computacionais actuais, mesmo considerando os poderosos *clusters* de computação paralela. Por essa razão, desenvolveram-se métodos alternativos, empregando heurísticas que fornecem soluções incompletas mas satisfatórias, dentro de um tempo de resposta razoável. Comummente, estas soluções designam-se por filtragem, pois resumem as pesquisas de similaridade nas regiões que apresentam maior probabilidade de a conter. A probabilidade é apurada pela presença contígua de um número restrito (entre 5 e 25) de bases de similares,

denominadas de sementes (*seeds*), nalguns casos considerando apenas a identidade (exacta) noutros a similaridade (aproximada). As soluções de filtragem são incompletas porque apresentam apenas uma parte das similaridades realmente existentes, há portanto uma menor sensibilidade envolvida pois os crivos "mais porosos" não retêm as similaridades menos evidentes ou de menor intensidade.

Desde a publicação do primeiro trabalho versando uma solução heurística e rápida [133] na prospecção de similaridades em informação biológica em 1983, muito caminho já foi percorrido. Ainda assim, a crescente demanda em análise de bioinformação cada vez mais volumosa aumenta a pressão para o desenvolvimento de algoritmos de pesquisa de similaridades cada vez mais eficientes. Os métodos heurísticos socorrem-se de técnicas alternativas às exaustivas e providenciam soluções satisfatórias para a maioria dos problemas num tempo de resposta considerado útil. Para o propósito da compressão do ADN estas técnicas necessitam de reforço da capacidade de detecção dos padrões mais imperceptíveis, porém o tempo de processamento para atingir a compressão é um factor crucial. Para perceber as metodologias de referências no domínio da análise de similaridade com recurso a métodos heurísticos vejam-se os seguintes exemplos.

3.2.4.1. FASTA e BLAST, recorrendo a *seeds* exactas

O princípio no qual se baseiam o FASTA [134] e o BLAST (*Basic Local Alignment Search Tool*) [100], assenta no facto de que, se houver um bom alinhamento aproximado, então deverá haver vários sub-alinhamentos exactos envolvidos. Este princípio fica dependente da determinação daquilo a que se chama de sub-alinhamento exacto, ou simplesmente de *seed*. Assim, se o tamanho requerido para *seed* for de 4 bases exactas consecutivas então o seguinte alinhamento (ver Figura 3.4) não seria detectado, isto apesar de haver 75% de similaridade.

```
A C T T T T G A A A G G
|   |   |   |   |   |   |   |   |
A C T G T T A A A A T G
```

Figura 3.4 – Alinhamento ignorado por uma *seed* de tamanho maior que 3.

Por defeito o BLAST usa *seeds* de 11 bases, o que agrava ainda mais a falta de sensibilidade. É certo que se pode reduzir o tamanho da *seed*, e assim recuperar a sensibilidade, porém tal medida afecta sobremaneira o desempenho do algoritmo em termos de tempo de processamento, tornando-o inviável para comparar múltiplas sequências de comprimento superior às centenas de Mb. A técnica usada no BLAST, recorrendo a *seeds* exactas, é útil quando à partida se impõem limites para a classificação de alinhamento. Isto é se se considerarem alinhamentos apenas os casos em que haja no mínimo 80% se similaridade, então estará garantida a presença de *seeds* maiores. Ainda assim, em casos particulares poderá suceder que as *seeds* exactas não sejam exactamente as mesmas em todas as sequências como se mostra a seguir (ver Figura 3.5).

```
A C T T T T A A A A G G
|   |   |   |   |       |   |   |       |
A C T T T T G A A A T G
|   |   |       |   |   |       |   |   |   |
A C T C T T G T A A T G
```

Figura 3.5 – Alinhamento múltiplo sem *seeds* comuns a todas as sequências.

No caso apresentado a terceira sequência não estaria directamente relacionada com a primeira, mas apenas indirectamente por apresentar uma *seed* de dimensão 4 em comum com a segunda, que por sua vez sim está relacionada com a primeira mas por outra *seed* diferente.

O BLAST detecta então *seeds* exactas de determinada dimensão e faz a análise de continuidade de similaridade quer à direita da *seed* quer à esquerda, estendendo assim, sob regras de limitação de tolerância a discrepâncias (*threshold*), a região de similaridade. Essa análise faz-se por programação dinâmica bandeada, sendo possível detectar e assimilar, delecções, inserções, substituições e, nas versões mais recentes, também *gaps*.

Recorrendo a um sistema de pontuação (*scoring system*), que latamente premeia as correspondências e penaliza ligeiramente as não correspondências, e mais acentuadamente os *gaps*, é possível valorizar as regiões de eventual similaridade

detectadas. No BLAST, estas regiões são denominadas de HSPs (*High-Scoring Segment Pairs*), quando duas sequências aleatórias de tamanho *m* e *n* são comparadas, o número de HSPs expectável, com uma pontuação de pelo menos *S*, é dado pela Equação 3.2.

$$E = Kmn \cdot e^{-\lambda S} \qquad (3.2)$$

A esse número dá-se o nome de *E-value* para o valor *S*. K e λ são parâmetros estatísticos dependentes do sistema de pontuação utilizado, conjuntamente com a estatística de frequência dos símbolos que constituem as sequências. Basicamente o que se pretende com a interposição da estatística, é discernir em avanço a quantidade e a qualidade dos alinhamentos locais, que podem variar desde coincidências fortuitas de ocorrências de *seeds* até vastas regiões de similaridade.

Depois de seleccionados os HSPs mais promissores, estes são estendidos por programação dinâmica bandeada e a rigorosa medida de similaridade é apurada, sendo mais tarde apresentados estes resultados como possíveis alinhamentos locais.

A *seed* por defeito no BLAST é de tamanho 11 e corresponde à palavra binária (11111111111), onde cada "1" significa uma correspondência necessária. Cada vez que a *seed* se reencontra no percurso da sequência a analisar é assinalado um *hit*. Cada *hit* pode ser um falso positivo, e por isso contribuir negativamente para a eficiência do algoritmo. Adicionalmente, o facto de as *seed* serem compostas de correspondências exactas consecutivas leva a que sejam detectadas várias *seeds* consecutivas, e como tal, redundantes, pois levam ao mesmo alinhamento. Esta faceta é também uma fuga de recursos que deveria ser minimizada. O rigor do BLAST é também um aspecto negativo, pois a estatística usada, associada com o tamanho (excessivo) da *seed* pode ignorar alinhamentos significativos para a genómica funcional [135]. O BLAST tem vindo a ser melhorado gradualmente, por exemplo através de uma nova versão que está preparada para lidar com alinhamentos com *gaps* [136].

3.2.4.2. *Spaced Seeds*, o algoritmo *PatternHunter*

Desde a sua génese, há cerca de 20 anos, a ferramenta BLAST [100], analisada na secção anterior, é rotineiramente utilizada por cientistas de todo o mundo na pesquisa de similaridades e homologias inter e intra-genómicas ou proteómicas. Recentemente, com a sequenciação de genomas de seres eucariotas mais complexos, como é o caso do Homem, do rato, ou da galinha, correspondendo a genomas mais extensos, na ordem das Gigabases (10^9 bases), a ferramenta BLAST, mesmo com as melhorias introduzidas ao longo dos anos [136, 137], demonstra graves limitações ao nível do tempo de resposta se necessitarmos de elevada sensibilidade, e denota fraca sensibilidade, se pretendermos a resposta em tempo útil. Tornava-se premente desenvolver novos algoritmos e aplicações que permitissem que o avanço da genómica funcional não fosse entorpecido pelo fraco desempenho das aplicações de alinhamento e análise de similaridades de sequências genómicas, quer ao nível da sensibilidade, quer ao nível to tempo de resposta. O PatternHunter [118, 119], uma ferramenta de análise de similaridade de última geração, deu um grande contributo nesse sentido.

O algoritmo denominado de PatternHunter (PH) introduziu na análise de similaridade de sequências uma nova heurística, baseada no conceito de sementes espaçadas (*spaced seeds*) [138, 139]. Na primeira versão da ferramenta [118], a metodologia recorria a *spaced seeds* isoladas e na segunda evolui-se para a utilização de *spaced seeds* múltiplas e cooperativas [119]. O funcionamento algorítmico do PatternHunter é similar ao funcionamento do BLAST, realçando-se porém, diferenças fundamentais ao nível das *seeds* utilizadas com repercussão evidente nos resultados obtidos. Assim, em oposição às *seeds* exactas do BLAST, o PatternHunter usa *seeds* espaçadas, as quais intercalam caracteres especiais, denominados de *don't care*, que permitem maior flexibilidade e sensibilidade à *seed*.

O BLAST usa *seeds* exactas de tamanho 11, por defeito, obrigando a ocorrências de segmentos de *exact pattern-matching* de, no mínimo, 11 bases consecutivas (11111111111), e a partir dessas *seeds* a zona de similaridade pode ser alargada para a esquerda e direita através da prospecção de similaridade bandeada usando programação dinâmica. O PatternHunter emprega *seeds* binárias do tipo 111010010100110111 (*seed* por defeito do PH), onde os 1s correspondem a bases que devem ser correspondidas e os

0s a qualquer base. A *seed* apresentada contém tamanho 18, mas igualmente 11 bases exactas ainda que descontínuas, contudo pode detectar mais zonas de similaridade pois apresenta maior probabilidade de correspondência nas sequências a analisar. Para comprovar esta afirmação aporta-se, pela Figura 3.6, a análise comparativa da sensibilidade conseguida usando *seeds* contínuas de 10 e 11 bases, versus *spaced seeds* de 11 bases precisas [118]. De realçar que o melhor desempenho das *spaced seeds* ocorre para sequências com similaridades entre os 60 e 90%.

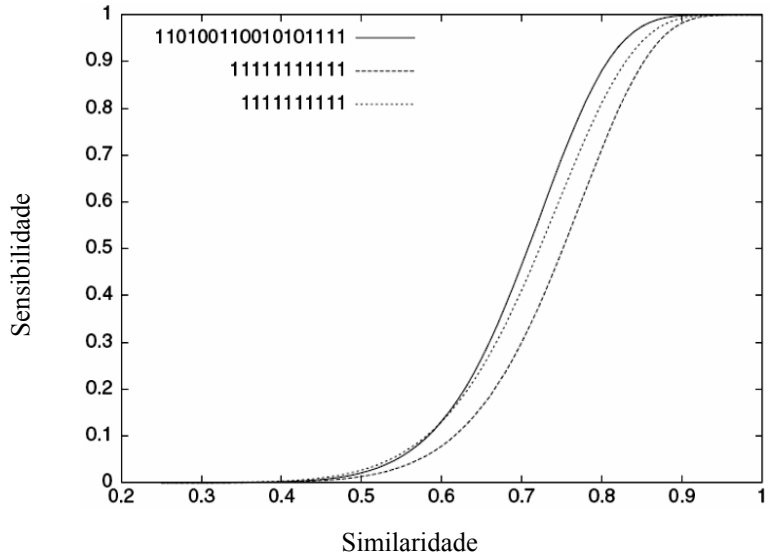


Figura 3.6 – Sensibilidade das *spaced seeds* versus *seeds* exactas.

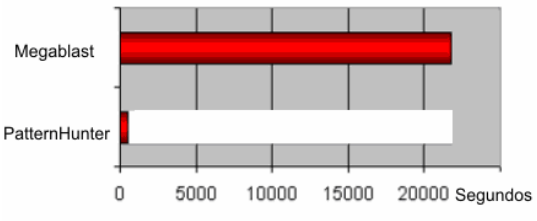
A área de desenho de *seeds* eficientes é uma área de investigação muito activa [138-142], utilizar as melhores *seeds* permite mais sensibilidade e prospecção em menor tempo. Assim, pode considerar-se fundamental na eficiência dos algoritmos que usam as *spaced seeds* como recurso nuclear.

A probabilidade de *hit* (ocorrência da *seed*), revelando uma potencial região similar, é computada no PH, recorrendo a um algoritmo baseado em programação dinâmica. As ocorrências das *seeds* são registadas em estruturas de dados de elevado desempenho, tabelas de *hash* neste caso, que constituem o repositório das HSP. Posteriormente esse repositório é avaliado no sentido de o depurar de elementos redundantes. Finalmente, as HSP mais relevantes são analisadas nas suas imediações, por programação dinâmica localizada, de forma a obter a extensão completa da região de similaridade.

A versão mais recente do PatternHunter, referida por PH-II [119], utiliza o conceito de "*spaced seed* optimizada", envolvendo múltiplas *seeds* em cooperação. A pesquisa das *seeds* utiliza algoritmos de pesquisa de padrões múltiplos, que pode ser feita em paralelo. Assim, o varrimento das sequência a analisar é virtualmente realizado em várias camadas, correspondendo cada camada ao estado da detecção de uma *seed*. Finalmente, as várias camadas proporcionam uma visão integrada das regiões de similaridade evidenciadas, aumentando, com esta redundância, o grau de sensibilidade do algoritmo, sendo que, no dizer dos seus autores, o desempenho conseguido se situa nos 98% de sensibilidade. Recorde-se que para obter os 100% de sensibilidade são necessárias as técnicas de programação dinâmica, porém impraticáveis em genomas da ordem das Gigabases. Importa igualmente referir que o tempo de resposta é afectado negativamente pelo acréscimo de *seeds* a pesquisar. Outro factor importante é a complexidade associada à obtenção do conjunto de *seeds* óptimas [143] ajustadas a cada sequência a analisar, neste caso as soluções óptimas são igualmente proibitivas, correspondendo a problemas de tipo NP, o PH-II usa uma solução de tipo ganancioso (*greedy algorithm*) para reduzir a complexidade do problema sem depauperar a qualidade da solução. Se bem que esta computação só é necessária uma vez para cada genoma a analisar, o tempo de processamento envolvido, pode, para os casos mais exigentes, prolongar-se por vários dias de CPU.

Para concluir a secção apresentam-se, na Figura 3.7, gráficos de desempenho comparado, em termos de tempo de processamento e recursos de memória utilizados, pelo PH e Megablast (a versão mais rápida do tradicional BLAST) na análise de similaridade entre os cromossomas 2 e 4 da *Arabidopsis*, usando os parâmetros mais favoráveis para o Megablast e os parâmetros *standard* do PH. Os referidos gráficos são originários do estudo de *benchmark* disponibilizado no sítio da Internet da empresa que desenvolve e comercializa o PH, em http://www.bioinformaticssolutions.com. O estudo aludido foi realizado num PC equipado com um processador Pentium III – 700MHz, 1GB de memória RAM, com o sistema operativo Linux Redhat 7.1. São patentes as diferenças de desempenho, mais impactantes ao nível do tempo necessário, onde a diferença, a favor do PH, é superior a factor 20.

Tempo de processamento necessário para comparar os cromossomas 2 e 4 da *Arabidopsis*



Recursos de memória necessários para comparar os cromossomas 2 e 4 da *Arabidopsis*

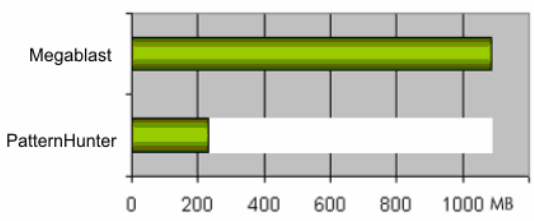


Figura 3.7 – Desempenho comparado entre o PH e o Megablast.

Utilizando 4 *spaced seeds* o PH-II ganha uma sensibilidade, em média, na ordem dos 85-95%, muito superior à do BLAST (em média 65-75%), consumindo o mesmo tempo para a análise de similaridade.

3.2.4.3. *Data mining*

As ferramentas de *data mining* contêm o potencial para extrair conhecimento da informação biológica [144-146], através das suas ferramentas podem-se realizar estudos de prospecção de padrões e similaridades que permitam entender melhor os processos que levam à expressão génica, descoberta de vias metabólicas, contribuir para melhorar o desenho e engenharia de novos biofármacos ou resolver os problemas emergentes da genómica e da proteómica [147, 148].

Uma das aplicações mais relevantes na área de *data mining* em literatura biológica é o Weka [149], esta aplicação de uso alargado está vocacionada para a classificação automática, regressão, *clustering* e selecção de características em informação, aqui instanciada na informação biológica, o que constitui a base da maioria das necessidades desta área. Contém uma colecção de algoritmos na categoria da aprendizagem máquina, métodos de pré-processamento de informação, complementados com interfaces gráficas para a exploração da informação e análise comparativa de diferentes técnicas de aprendizagem máquina para o mesmo problema. Como principais contribuições oferece a possibilidade de extrair conhecimento útil de bases de dados (biológicas), assistindo o utilizador na escolha consciente e avalizada da melhor metodologia a empregar com a finalidade de gerar um modelo predictivo fiável.

As bases de dados biológicas estão em acelerada expansão, não só em quantidade como em qualidade. Desde as primárias bases de dados de nucleótidos e proteínas, como o GenBank ou a Uniprot, assistimos hoje à aparição de bases de dados dedicadas, desde genomas de determinada espécie, vias metabólicas, anotações, *microarrays*, *motifs*, etc. Esta panóplia de informação necessita de integração, pois em bom rigor está toda relacionada, mas faltam as ferramentas que nos permitam com proficuidade analisar relacionalmente um volume enorme de informação distribuída e remota. As ferramentas de *data mining* cumprirão certamente essa função num futuro mais ou menos próximo. No âmbito deste trabalho o interesse centra-se na exploração da informação intra-genómica, e em capitalizar o potencial das regiões informacionalmente redundantes para o fim da compressão de informação, pelo que as ferramentas de *data mining* são apenas aludidas superficialmente.

3.3. A linguística do código do ADN como recurso auxiliar da compressão

Uma parte da comunidade que estuda as sequências de ADN está mais interessada nas propriedades da linguagem biológica dos nucleótidos. Em rigor, os objectivos passam por estabelecer uma caracterização linguística [150], a diferentes níveis, a saber: lexical, gramatical, matemático e estatístico da linguagem do ADN. Esta investigação reveste-se de importância para este trabalho no sentido de preparar a abordagem à análise das sequências. Com os conhecimentos adequados é possível efectuar uma prospecção de padrões com maior razoabilidade, empregando menos e menores meios e obter, de forma mais rápida, melhores resultados. Ao nível da linguística do ADN existem diversos estudos que suportam as ideias aventadas, e.g., [7, 151, 152]. Uma das conclusões principiais destes estudos postula que a linguagem do ADN nem é comparável à linguagem natural humana nem tampouco se pode considerar aleatória, assim trata-se de uma linguagem peculiar cujas propriedades intrínsecas vão sendo progressivamente capturadas pela investigação. Contudo, é muito difícil encontrar um modelo universal e consensual para o ADN, isto porque basta distinguir entre regiões codificantes e não codificantes para perceber que a composição linguística varia

substancialmente. Se, noutro plano, considerarmos as propriedades linguísticas do código do ADN nos seres eucariotas e procariotas então acentuam-se as divergências.

Os modelos de linguagens, introduzidos em [153], são uma área da investigação que já produziu resultados satisfatórios na compressão e predição de linguagens naturais, neste trabalho interessa estudá-los e aplicá-los na linguagem do ADN, com o intuito de implementar um modelo probabilístico de predição de sequências de ADN que possa contribuir positivamente na compressão das regiões menos repetitivas das sequências, que por esse mesmo motivo não possuem representação no dicionário.

3.3.1. Modelos de linguagem

Os modelos estatísticos de análise de linguagem estimam a distribuição probabilística das componentes (palavras ou caracteres) que integram uma sequência de expressão de uma linguagem determinada, com ênfase na linguagem natural [153-155]. Hoje em dia são usados por todos nós quando digitamos uma mensagem SMS no telemóvel aproveitando as sugestões fornecidas pelo software para terminar as palavras da mensagem.

Os modelos de linguagem são inferidos a partir de um corpus de texto, que funciona como objecto de treino para a formação do modelo, e sem necessitar de qualquer outra informação adicional à partida, podem depois ser usados por um algoritmo na predição de palavras [156]. Esta possibilidade é relevante e de interesse para a compressão da informação em geral, aqui analisada na perspectiva da informação biológica, mais concretamente das sequências de nucleótidos.

Se se considerarem W sequências de palavras então pode-se assumir que nos modelos de língua de *Ngramas*, P(W) reflecte a distribuição de probabilidade de ocorrência das sequências de palavras W. Por exemplo, num modelo que descreve a linguagem de um gene, pode-se ter P(*cct*)=0.01, pois provavelmente um em cada cem codões é "*cct*".

P(W) pode ser decomposto em:

$$P(W) = P(w_1, w_2, ..., w_n)$$

$$= P(w_1)\, P(w_2 \mid w_1)\, P(w_3 \mid w_1, w_2) \ldots P(w_n \mid w_1, w_2, ..., w_{n-1})$$

$$= \prod_{i=1}^{n} P(w_i \mid w_1, w_2, ..., w_{i-1})$$

Onde $P(w_i \mid w_1, w_2, ..., w_{i-1})$ é a probabilidade de ocorrência w_i dada a sequência de palavras $w_1, w_2, ..., w_{i-1}$ aparecer previamente.

A qualidade da predição de símbolos numa sequência de ADN é reveladora da qualidade do modelo bem como da entropia da sequência analisada. Na avaliação dos modelos de linguagem, a perplexidade corresponde ao grau de diferenciação das probabilidades das palavras recolhidas no modelo. Por exemplo, um modelo onde todas as palavras tenham a mesma probabilidade possui 100% de perplexidade e, nessa ordem, torna-se absolutamente inútil. A seguir importa analisar bigramas, que correspondem a duas palavras sucessivas, podendo dessa forma atingir níveis de probabilidade diferenciada que aportem algum potencial à predição. No geral, a perplexidade de um modelo de linguagem é igual à média geométrica da probabilidade inversa das palavras de referência, medida na informação de treino [157]:

$$\sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 \ldots w_i - 1)}} \qquad (3.3)$$

A perplexidade e a entropia estão logicamente relacionadas, entendendo a entropia como a quantidade mínima de bits, em média, para codificar um símbolo/palavra, podemos dizer que a perplexidade de um modelo é igual a 2^H, onde H corresponde à entropia. Da mesma relação inferimos que a entropia é igual a $\log_2$(perplexidade). Por exemplo, para um modelo com 68 de perplexidade temos uma entropia correspondente a $\log_2(68)$=6,09 bits por palavra.

Na análise de ADN, as palavras correspondem normalmente aos codões (3 nucleótidos sucessivos), mas para construir o modelo podemos optar por diferente *ómeros*, de dimensão 4,5, 6, ou até maiores. Sendo certo que, modelos baseados em palavras muito curtas podem resultar numa perplexidade elevada e modelos pouco extensos, e palavras muito compridas podem resultar em baixa perplexidade mas modelos muito extensos e dificilmente completos.

O objectivo principal dos modelos probabilísticos de linguagem é providenciar informação estatística para que, nas predições efectuadas, as sequências de palavras mais recorrentes tenham maior probabilidade do que as menos prováveis. Deste modo é possível melhorar a precisão e reduzir o espaço de procura no reconhecimento da linguagem. A utilização de modelos de linguagem é mais comum na análise de linguagem e discurso naturais, se bem que tem vido a ser adoptada para a análise da linguística do ADN [158], desde a área de controlo de qualidade na sequenciação [159], até à compressão de sequências [160]. Pela utilização de modelos de linguagens, constata-se que as palavras anteriores condicionam as palavras que são sugeridas e também o número de palavras anteriores existentes condicionam quais os modelos de *Ngramas* a utilizar. Para um vocabulário de dimensão v existem v^{i-1} diferentes hipóteses e para especificar completamente $P(w_i \mid w_1, w_2, ..., w_{i-1})$, têm de ser estimados v^i valores. Mesmo para valores moderados de i é impossível estimar todos os valores pois a maior parte deles só ocorrem uma vez ou muito poucas vezes. A solução prática é restringir o número de palavras anteriores, levando aos modelos de *Ngramas*:

- os modelos de língua *unigramas* $P(w_1)$ apenas consideram a frequência de ocorrência da palavra sem considerar as palavras anteriores;
- os modelos de língua *bigramas* $P(w_i \mid w_{i-1})$ consideram apenas a palavra anterior no cálculo da frequência de ocorrência;
- os modelos de língua *trigramas* $P(w_i \mid w_{i-2}, w_{i-1})$, consideram as duas palavras anteriores no cálculo da frequência de ocorrência.

Desta forma, os modelos estatísticos de língua, através dos *Ngramas*, permitem representar um determinado corpus de texto e realizar a sugestão das palavras com base na frequência de ocorrência.

Um dos problemas existentes nos modelos de *Ngramas* deve-se ao facto de, mesmo utilizando corpus de treino grandes, existirem *Ngramas* que são válidos em termos de linguagem mas não existem no corpus de treino e por isso a sua ocorrência é nula. Este problema é tanto maior quanto maior for o número de palavras anterior que se quiser considerar.

3.3.1.1. *Smoothing* (Alisamento)

Existem técnicas que permitem lidar com este problema e permitem atribuir frequências diferentes de 0 aos *Ngramas* que não registam ocorrências. Esta técnica é designada por alisamento (*smoothing*). O primeiro método para realizar o alisamento consiste em considerar que todos os grupos de palavras têm pelo menos uma ocorrência, incluindo aqueles que não ocorrem nenhuma vez. Este método designa-se por alisamento "adicionar um".

Como exemplo, e considerando para o caso dos *bigramas*, em primeiro lugar é realizada uma matriz de ocorrência em que se considera que todos os pares de palavras vão ocorrer pelo menos uma vez. Depois actualizam-se os pares de ocorrências com os valores de ocorrência que existem no corpus. Seguidamente adiciona-se 1 a todos os valores de ocorrência da matriz, incluindo os nulos. Depois normalizam-se os valores, actualizando o número total de ocorrências com os valores acrescentados. Este método, no entanto, não é muito utilizado pois não apresenta bons resultados em termos de utilização e pode provocar uma grande diferença em relação às frequências originais quando se realiza o alisamento. O principal problema reside no facto de adicionar o valor 1 às contagens. Se fosse adicionado um valor mais pequeno, o problema era atenuado.

O segundo método de alisamento designa-se por desconto de Witten-Bell [161]. Este método, embora mais complexo que o anterior, baseia-se no conceito de modelar a primeira ocorrência dos *Ngramas* para estimar a ocorrência dos que ainda não ocorreram. Assim, a probabilidade de ocorrência de um *ngrama* que ainda não ocorreu é modelada com a ajuda dos *Ngramas* que apenas ocorreram uma vez. Se se considerar o exemplo dos *bigramas*, o valor de estimação de ocorrência dos *bigramas* que não têm

nenhuma ocorrência é dado pela contagem dos *bigramas* que ocorreram apenas uma vez. O valor final é normalizado com o número de *bigramas* observados e dividido por todos os possíveis *bigramas* que não têm ocorrência. Este método faz com que o cálculo da probabilidade de ocorrência seja dependente do histórico de ocorrência de palavras. Também as palavras que ocorrem em menos combinações de *bigramas* tendem a ter menos *bigramas* não vistos que as que entram em mais combinações de *bigramas* diferentes.

O terceiro método de alisamento designa-se por desconto de Good-Turing [162]. Este método, embora mais complexo que o anterior, tem como ideia principal a de voltar a calcular as probabilidades de ocorrência para atribuir valores aos *Ngramas* que são nulos e aos que têm valores muito baixos com base no número de *Ngramas* com elevados valores de probabilidade de ocorrência. A ideia é calcular as frequências de ocorrência dos valores de ocorrência c, ou seja, calcula-se para c=1 quantos tipos de *Ngramas* apenas ocorrem uma vez. Para c=i calcula-se quantos *Ngramas* diferentes têm esse valor de ocorrência i. Constrói-se assim uma tabela de frequências de ocorrência em que para c=0 tem-se a contagem de *Ngramas* com frequência de ocorrência nula. Como se espera, quanto menor é o c maior é a frequência de ocorrência. Se se considerar para o exemplo de *bigramas*, a contagem revista dos *bigramas* que nunca ocorreram é calculado dividindo o número de *bigramas* que ocorreram uma vez pelo número total de *bigramas* que nunca ocorreram. Na prática, este desconto não é aplicado para todos os valores de c. As frequências maiores, onde c>k (em que k representa o valor a partir do qual não são recalculados) são assumidas como fiáveis. Katz [153] sugere que k tome o valor 5.

Os métodos de desconto vistos permitem lidar com os *Ngramas* que não têm nenhuma ocorrência no modelo. No entanto, existem outros processos que permitem lidar com o facto de não existir um *ngrama* específico e calcular a sua frequência de ocorrência com base nos *Ngramas* de ordem *n-1*. Se se considerar o caso dos *trigramas*, o cálculo é efectuado com base nos *bigramas*. Para o caso de existir um *bigrama* que não tenha ocorrências, então a sua frequência de ocorrência é baseada na frequência de ocorrências das palavras simples. Existem duas formas de aplicar este processo por interpolação apagada ou *backoff*.

O quarto método de alisamento, designado por *backoff*, é um método não linear introduzido por Katz em 1987 [153]. Para salientar a diferença deste processo, e se se considerar o exemplo em que se tem *trigramas* com contagem diferente de 0, então apenas se tem em consideração a frequência de ocorrência dos *trigramas*. Se se pretender calcular a ocorrência de um trigrama que não tem nenhuma ocorrência consideram-se os *bigramas*. Se se pretender calcular um *bigrama* que não tenha ocorrência consideram-se as ocorrências das palavras simples.

3.3.1.2. *Skiping*

À medida que avançamos para maiores *Ngramas*, a probabilidade de correspondência exacta do contexto vai diminuindo, porém, a probabilidade de associar contextos semelhantes, que contenham várias palavras comuns, aumenta. De facto, considerando um contexto *5-grama*, o subconjunto $P(w_i | w_{i-4}\ w_{i-3}\ w_{i-1})$ pode ser semelhante a $P(w_i | w_{i-4}\ w_{i-2}\ w_{i-1})$. É nesta propriedade que se baseia o *skiping*, que aproveita a similaridade parcial desconsiderando eventuais e pontuais falhos.

3.3.1.3. *Clustering*

Trata-se de criar conjuntos lógicos de palavras que apresentam igual probabilidade de suceder a outra(s). Por exemplo o texto "no mês de", a palavra sucessiva pode ser com igual probabilidade "Janeiro", "Fevereiro", "Março", …, "Dezembro". Deste modo, criam-se classes de palavras que irão funcionar como palavras especiais.

3.3.1.4. *Caching*

Trata-se de memorizar as palavras encontradas prevendo que se podem repetir no futuro próximo. Por exemplo, no discurso de muitas pessoas existem palavras que se repetem frequentemente como é o caso do "portanto".
Na informação genómica, os *tandem repeats* ou repetições adjacentes, podem ocorrer em pequenas subsequências que se repetem exactamente várias vezes sucessivamente, por outro lado existem as repetições dispersas que se repetem de forma exacta e

aproximada por regiões não equidistantes. Assim sendo, e atendendo ao âmbito deste trabalho, esta técnica de aperfeiçoamento dos modelos de linguagem é particularmente interessante para o modelo a desenvolver.

As técnicas apresentadas podem ser usadas em conjunto para melhorar o desempenho do modelo. Contudo, dada a especificidade das várias linguagens, nem todas as linguagens requerem o mesmo tratamento e nalguns casos certas técnicas podem atentar contra o desempenho, pelo que o desenho do modelo deve ter em linha de conta as peculiaridades da linguagem a que se destina.

Existem *toolkits* para construção de modelos de linguagem que podem ser adaptados às necessidades de cada projecto. O mais representativo é o CMU [163], da Universidade de Carnegie Mellon. Face às especificidades da informação genómica optou-se neste trabalho por desenvolver um modelo de linguagem próprio, no Capítulo 5 dar-se-á a conhecer esse modelo de linguagem desenvolvido para o DNALight, a aplicar como último contribuinte para a obtenção de compressão no modelo conceptual adoptado.

3.4. Compressão de informação biológica

A compressão, entendida no sentido lato, permite aproveitar todo o conhecimento que detemos de um determinado tipo de informação para criar uma representação optimizada dessa mesma informação original sem perdas de conteúdo ou de acessibilidade. A compressão de sequências biológicas é uma área de investigação modelar da multidisciplinaridade, sendo basilar para a sua consecução a teoria fundamental da informação, esta terá de ser complementada com as especificidades do código da vida que ainda estamos a descobrir [164].

Quase invariavelmente, as tarefas de análise, interpretação e compressão de dados estão subtilmente interligadas. As sequências biológicas, mormente o ADN, devem ser analisadas nesta perspectiva [165].

A sequência de ADN contém níveis elevados de entropia, contudo existem regularidades e especificidades que podem ser exploradas para melhor codificar a

versão comprimida [83, 86, 165]. A peculiaridade mais saliente reside no facto de várias subsequências se repetirem. Para além destas cópias simples ou directas, existem os palindromas, que significam as cópias dos complementos de subsequências anteriores invertidas. Existem quasi-repetições que resultam muitas vezes dos erros cometidos pela célula nos processos de cópia, onde as bases são substituídas, inseridas ou eliminadas das sequências. Obviamente são estes os recursos a explorar na compressão das sequências genómicas [166].

A fase percursora da compressão, a pesquisa de elementos redundantes, é de grande utilidade na recolecção de características relevantes de um determinado genoma ou sequência. De facto, pode considerar-se um método robusto, aquele que utiliza a compressibilidade de duas sequências para as comparar. Dois genomas que partilham a mesma taxa de compressão têm maiores probabilidades de apresentarem homologias.

Para sistematizar as propriedades das sequências de ADN que maior peso apresentam nos resultados até agora obtidos pelos compressores de ADN mais eficazes, reveladas e confirmadas por anos de investigação, atentemos na lista seguinte:

- as sequências de ADN contêm subsequências que se repetem exacta e frequentemente;
- contêm subsequências que são repetições exactas mas representadas em forma de complementares reversas, os denominados palindromas;
- nas sequências genómicas proliferam as repetições aproximadas, ou repetições com erros. O termo erros, neste contexto, pretende significar inserções, delecções, *gaps* ou substituições de bases, que em conjunto perfazem as distâncias para a subsequência de referência.

Antes de entrar em matéria, convém diferenciar os algoritmos de compressão em dois tipos básicos, por um lado os que comprimem sem perdas (*lossless*) ou integralmente reversíveis [167], e por outro os que comprimem com perdas controladas de realidade (*lossy*), obviamente no tipo de aplicação genómica interessam preferencialmente os primeiros. Para além desta divisão quanto à integridade da informação original, pode estabelecer-se uma divisão funcional. Neste campo, podemos considerar os algoritmos baseados em dicionário e os baseados em predição por análise estatística e

probabilística. Na secção seguinte abordam-se em maior detalhe as metodologias mais vocacionadas para a compressão da informação biológica, mormente o ADN.

3.5. Revisão de metodologias de compressão adequadas à bioinformação

A maioria dos métodos usados para a compressão de sequências de ADN podem subdividir-se em duas grandes categorias. A primeira inclui as metodologias de índole substitucional, nas quais, basicamente, as "palavras" mais extensivamente usadas são substituídas com vantagem para a compressão por um símbolo mais curto. Esta engloba os esquemas baseados em dicionário, nos quais radica uma recolecção das subsequências com interesse para a compressão e substituem-se na reconstituição da sequência por informação que aponta para esses índices. A segunda categoria corresponde ao aproveitamento das recorrências estatística e probabilisticamente modeláveis para predizer a sequência. Este trabalho comunga das duas categorias, existindo alguma hibridação. A metodologia principal corresponde à substitucional sendo complementada com contributos de compressão estatística e probabilística.

Metodologias baseadas em estatísticas como a codificação aritmética e o CTW (*Context Tree Weighting*) [168, 169], demonstraram sucesso na compressão do ADN, conseguindo resultados abaixo dos 2 bits por base. Por outro lado, os códigos de Huffman [170] revelaram-se ineficazes, pois dado o facto de existirem apenas quatro símbolos diferentes, agregado ao facto de as frequências serem similares para a maior parte das sequências, tal metodologia não encontra sustentação na compressão da informação genómica.

Para referir alguns exemplos concretos de tecnologias, metodologias e algoritmos usados na compressão do ADN [169], apresenta-se a seguinte lista:
- baseados em dicionário de repetições, quasi-repetições e palindromas por métodos heurísticos [12, 166, 171];
- baseados em modelos probabilísticos de linguagem, recorrendo a predição com suporte probabilístico ou recorrendo às figuras dos peritos "*experts*" [14];
- baseados em modelos probabilísticos de Markov [172];
- usando a transformada de Burrows-Wheeler [173];
- usando substituição textual *greedy-offline* [174];
- baseado em dicionário exaustivo de padrões descobertos por programação dinâmica [13];
- baseado no modelo NML (*Normalized Maximum Likelihood*) [175, 176];
- baseados em métodos estatísticos, como a codificação aritmética ou CTW [177];
- baseados em gramáticas [93].

Em jeito de revisão importa caracterizar e referenciar com maior cuidado as metodologias que são empregues no trabalho desenvolvido, mais concretamente a compressão por dicionário, a compressão por predição probabilística e a compressão estatística por codificação aritmética.

3.5.1. Compressão por dicionário

A estratégia de compressão por dicionário enquadra-se na categoria da compressão substitucional. A constatação da existência de elementos redundantes, permite a composição de um dicionário de termos recorrentes, favorecendo-se a compressão sempre e quando a codificação recorre com frequência ao dicionário, substituindo as ocorrências dos termos por índices que as ligam à sua representação extensa no dicionário.

Devem-se a Lempel e Ziv os trabalhos seminais, que através de técnicas alternativas para a construção do dicionário, nomeadamente o LZ77 [10] e o LZ78 [9] desenvolvidos nos anos 70, nos facultaram os algoritmos, hoje mais desenvolvidos, que servem de base à maioria dos compressores de linguagem natural mais difundidos,

como sejam o ZIP e o GZIP, bem como na área da codificação de imagem como é o caso do formato GIF. Resumidamente, o LZ77 constrói intrinsecamente o dicionário, à medida que vai lendo o texto/sequência a comprimir, usando a metodologia de janela deslizante, vão sendo inseridos apontadores para ocorrências anterior de termos. O dicionário encontra-se embebido na codificação, e misturado com ela.

O LZ78 difere do seu predecessor em relação à metodologia de constituição do dicionário, nesta proposta o dicionário é separado da restante codificação, sendo inseridos nos locais da substituição índices que invocam entradas do dicionário. A temática da compressão por dicionário é muito mais vasta, abarcando o pré-processamento que maximize a obtenção de padrões, a representação compacta do dicionário, entre outros, abordados em [178].

No âmbito de aplicação em que este trabalho se insere, a implementação de um esquema de compressão por dicionário não pode seguir linearmente as técnicas aventadas pois não estão preparadas para tirar partido de padrões aproximados. De facto, a aplicação destas metodologias sem qualquer adaptação tem o efeito contrário à compressão, pois expandem as sequências genómicas acima dos 2 bits por base. Assim, estas técnicas terão de ser adequadas à especificidade da informação genómica, no sentido de garantir que o dicionário de padrões seja mais flexível, incorporando uma espécie de padrões maleáveis, que possam incorporar alterações mínimas, que permitam transformá-los em variantes que correspondam a várias formas similares de aparição. Concomitantemente os índices usados não serão simples, mas antes compostos, incorporando as edições a aplicar ao padrão para convertê-lo no termo adequado.

Por outro lado, torna-se mais vantajoso separar claramente o que é redundância (aqui apenas considerada a redundância usável com vantagem, i.e., rentável em termos compressivos) do que é não redundância, em termos práticos significa que o dicionário não deve ser total, mas apenas das regiões repetitivas aproveitáveis. A intenção será destacar a porção não comprimida para servir de *input* a metodologias alternativas não exclusivamente focalizadas no aproveitamentos de longas repetições exactas ou inexactas, como sejam as abordagens estatísticas e/ou probabilísticas.

O facto de em bioinformática preponderarem os padrões aproximados leva a que seja necessário avaliar os padrões para decidir pela sua utilização ou não em função da sua rentabilidade para a compressão. Atendendo a que se torna necessário incorporar operações de edição para tornar o padrão utilizável em todas as suas ocorrências, essas operações de edição representam um custo importante que tem de ser considerado. Por exemplo se se considerar que uma substituição pode custar 9 bits (2-operação, 5-localização, 2-nova base), não fará sentido utilizar padrões de comprimento 6, com apenas 5 bases correctas e um erro, já que a correcção desse erro envolve 9 bits e o código para índice pode representar no mínimo 3 bits, ou seja, seriam necessário 12 bits para representar 12 bits o que não corresponde a qualquer compressão, considerando 2 bits por base como adquirido. A Figura 3.8 mostra para a sequência HUMGHCSA, de cerca de 65Kb, que corresponde a um gene humano, e para a sequência VAACG que corresponde a um genoma de um vírus com cerca de 190Kb, a quantidade e qualidade dos padrões que uma aplicação típica de compressão de ADN por dicionário pode considerar rentáveis [171].

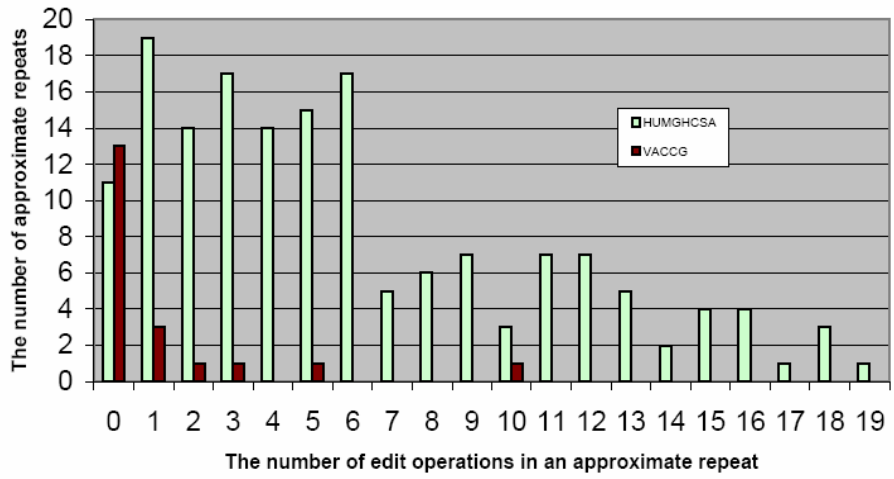


Figura 3.8 – Padrões considerados rentáveis para um algoritmo de compressão por dicionário em duas sequências exemplo.

3.5.2. Compressão por predição probabilística

Esta metodologia enquadra-se na categoria dos métodos estatísticos, a função de ganho centra-se na possibilidade de apreender comportamentalmente a sucessão dos símbolos

numa determinada sequência, sustentando-se das probabilidades e estatísticas aferidas, e construindo um modelo que mimetize com a máxima fidelidade possível o mesmo comportamento na predição dos símbolos desconhecidos com base nos anteriormente vistos ou com base em especificidades do código genético bem conhecidas, como é o caso, por exemplo, das repetições exactas ou aproximadas e dos palindromas. Quanto maior for a capacidade de predição do modelo, sendo o modelo intrínseco mais compacto que a porção por ele predita com sucesso, maior será a compressão alcançada.

Basicamente, a predição de um novo símbolo baseia-se no conhecimento e análise probabilística de uma série de símbolos predecessores, imediatos ou não. Ao conjunto de símbolos antecessores estudado dá-se o nome de contexto, um contexto finito neste caso, e são eles que constituem os modelos em que as predições se baseiam. Assim, podemos estabelecer vários modelos como contribuintes para a próxima predição, sendo que cada modelo providencia um palpite, ao qual estará associado um grau de certeza dependente da grandeza probabilística que o suporta. O número de símbolos usados designa-se por k e a ordem correspondente é a ordem k. Por exemplo, suponha-se que utilizamos uma análise das sucessões dos dois símbolos precedentes (ordem 2) para averiguar no passado quais foram os símbolos que, com maior frequência, os sucederam. Adicionalmente poderíamos alargar a ordem 2 para ordem 5 e, nesse caso estaríamos interessados em analisar probabilisticamente as ocorrências de segmentos iguais aos 5 anteriores símbolos e quais os sucessores mais prováveis. Esta análise providencia um perfil probabilístico para o próximo símbolo, do qual sairá uma predição. Em princípio uma predição baseada em ordem 5 será mais fidedigna do que ordem 2 mas, pode ocorrer não haver história de ocorrências de ordem 5 para a actual predição pelo que se deverá recuar e utilizar uma ordem menor que nos forneça alguma pista. Essa predição terá de ser analisada para validação ou correcção, sendo esta distância da predição o símbolo que entra para a codificação final usando por exemplo codificação aritmética.

Os modelos usados são normalmente adaptativos, o que corresponde à sua actualização iterada após cada predição, isto de forma a disporem de todo o conhecimento acumulado e dessa forma sair beneficiada a próxima predição. As ordens maiores são preferidas em relação a ordens menores, e no limite uma ordem 0 deverá existir quando

aparece um caracter desconhecido que torne a próxima predição impossível de prever apresentando-se todos os símbolos como equiprováveis.

As implementações mais relevantes nesta categoria são os algoritmos PPM [69, 179, 180], que foram durante a década de 90 a referência pelo seu desempenho na compressão sem perdas de informação. Destacam-se também, como sendo dos poucos clássicos que comprimem efectivamente as sequências de ADN, com resultados médios abaixo, ainda que muito ligeiramente, dos referenciais 2 bits por base. Recentemente, surgiram adaptações destes princípios para a especificidade do código genético [14], introduzindo-se inovações como a variabilidade do deslocamento do intervalo predecessor, ou seja os predecessores não precisam de ser os imediatos, já que as repetições mais frequentes no ADN ocorrem distancias de centenas ou milhares de bases, ou por exemplo a possibilidade do modelo predizer cadeias reversas complementares ou palindromas, algo igualmente frequente na linguagem do ADN que contrasta com as linguagens naturais. Essas especificidades foram conferidas a um painel de peritos virtuais que manifesta a sua predição e reforça ou reduz a sua preponderância face à taxa de sucesso alcançada. Mais detalhes deste trabalho são fornecidos na secção 3.6.

Os algoritmos baseados em predição probabilística, nomeadamente o PPM superam em ganho de compressão os baseados em dicionário, porém requerem mais tempo e espaço de processamento. A construção dos modelos, a sua recorrente consulta e o seu armazenamento em estruturas de dados de tipo *trie*[2] ou tabelas de *hashing* torna-se efectivamente ponderoso em termos computacionais.

3.5.3. Compressão por codificação aritmética

A codificação aritmética foi inicialmente desenvolvida e proposta por Abramson em [181], em 1963. A necessidade de desenvolver uma técnica que superasse as limitações dos códigos de Huffman levou a que investigadores como Rissanen e Pasco, já na década de 70, aportassem novos contributos à codificação aritmética [182, 183], porém

[2] Esrutura de dados de tipo árvore ordenada, usada para guardar um *array* associativo onde as chaves correspondem normalmente a *strings*.

esta metodologia viria a firmar-se definitivamente pelos contributos de Witten *et al.*, reunidos na sua publicação [177], de 1987.

Na codificação aritmética, uma mensagem é representada por um intervalo de números reais entre 0 e 1. À medida que a mensagem se vai tornando mais comprida, o intervalo necessário para a representar diminui, e o número de bits necessários para especificar esse intervalo cresce. Os sucessivos símbolos da mensagem reduzem o tamanho do intervalo de acordo com as probabilidades de ocorrência desses símbolos geradas pelo modelo associado. Quanto mais prováveis forem os símbolos menos reduzem o intervalo e portanto menos bits serão necessários à codificação, símbolos mais aleatórios implicam maior dispêndio de bits.

Inicialmente o intervalo é [0,1), constituindo-se entre $0 \leq x < 1$. À medida que cada símbolo vai sendo processado, o intervalo vai sendo reduzido, mantendo a porção que encerra o novo símbolo dentro do intervalo anterior. Por exemplo, supondo o alfabeto quaternário do ADN, e o modelo de probabilidades associado presente na Tabela 3.4, a sequência *ACTT* poderia ser codificada como a seguir se descreve. O primeiro símbolo (A) levaria a uma redução do intervalo para [0-0,29), o segundo símbolo (C) implicaria uma nova redução do intervalo na proporção da sua probabilidade e na localização especifica do símbolo, ou seja entre 26% de 0,29 e 55% de 0,29, logo o novo intervalo seria [0,0754-0,1595].

Símbolo	Probabilidade	Intervalo
A	0,29	[0-0,29)
C	0,26	[0,29-0,55)
T	0,21	[0,55-0,76)
G	0,24	[0,76-1)

Tabela 3.4 – Modelo de probabilidades dos símbolos para codificação aritmética.

Para melhor compreender a subdivisão do intervalo inicial acompanha-se esta explicação do esquema da Figura 3.9, onde estão representados todos os passos seguidos na codificação do exemplo apresentado.

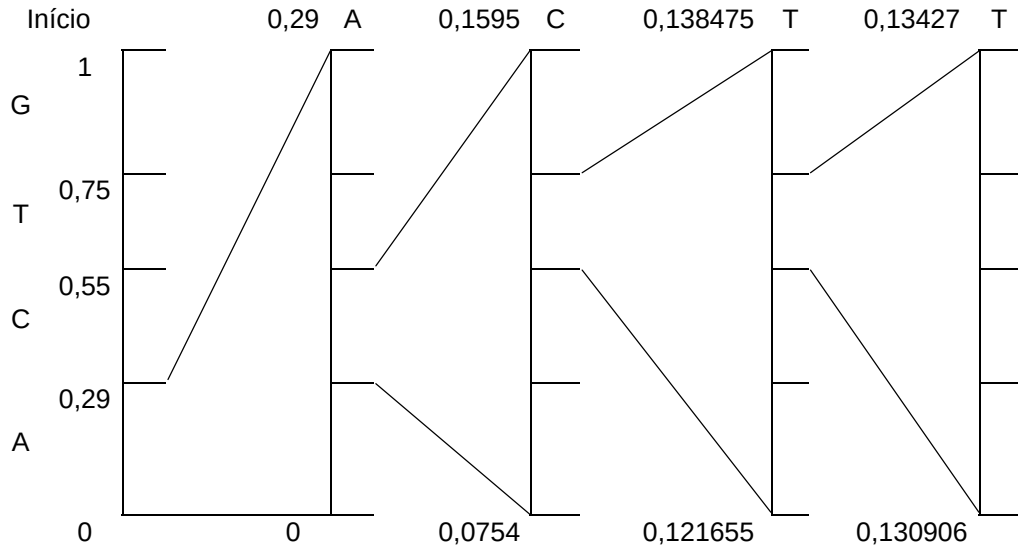


Figura 3.9 – Esquema de subdivisão sucessiva do intervalo em face do símbolo a codificar.

O terceiro símbolo (T) reduziria o intervalo a [0,121655-0,138475), que corresponde à porção entre 55% e 75% do intervalo anterior. Pela mesma lógica atingiríamos o fim da codificação com novo símbolo (T) e nova redução do intervalo para [0,130960-0,13427).

A codificação aritmética necessita de um símbolo especial para assinalar o final da sequência em codificação (EOF). Finalmente, apenas um número, o mais vantajoso, compreendido no intervalo final [0,130960-0,13427), por exemplo o 0,133, é suficiente para representar univocamente a sequência codificada. A descodificação baseia-se simplesmente na reconstituição dos passos seguidos na codificação. Assim, se o valor é 0,133, então o primeiro símbolo terá que ser necessariamente *A*, pois está inserido no intervalo [0-0,29), a seguir o único intervalo que comporta o valor conhecido é o que corresponde ao *C*, e seguidamente desvendam-se os intervalos seguintes até obter a sequência completa *ACTT*.

A eficiência da codificação aritmética é em boa medida devida à correcta modelação das probabilidades de ocorrência dos símbolos. Contudo, nas sequências genómicas naturais a diferenciação probabilística de ocorrência das bases é, em geral, inferior a 5%, ou seja, estão quase uniformemente distribuídas. Deste modo, a codificação aritmética não constitui a primeira opção na compressão de ADN, já que o seu melhor desempenho se situa entre 1,90 e 1,99 bits por base. Ainda assim, e dada a apertada

competição por ganho de compressão, esta técnica é usada como último recurso na porção remanescente que é, apesar de tudo, significativa, quantificando-se entre 40% e 80% por cento da sequência a comprimir.

Modernamente, a codificação aritmética poderá estar muito perto do seu limite de eficiência, já que nas últimas melhorias apresentadas [184], a diferença do desempenho efectivo para o óptimo teórico é de apenas 0,006 bits.

3.6. Breve historial de desenvolvimentos na compressão de ADN

Alguns dos algoritmos com melhores avaliações para a compressão de sequências biológicas incluem-se na categoria dos baseados em dicionário, aproveitando as regularidades exactas ou aproximadas das sequências biológicas. A maioria dos algoritmos baseados em dicionário, desenvolvem-se em duas fases, a primeira centra-se na prospecção de padrões que se coleccionam no dicionário, a segunda fase prende-se com a codificação da sequência na versão comprimida com recurso a referências de indexação mais curtas que os padrões referenciados. Actualmente, os algoritmos probabilísticos acaparam maiores atenções e esforços de desenvolvimento, pelo que os resultados obtidos por esta metodologia já se equiparam ou mesmo superam os mais clássicos.

Em 1993 surge a primeira ferramenta de compressão de sequências de ADN, trata-se do *Biocompress* [185], baseado nos princípios de Ziv e Lempel associando detecções de factores e palindromas. A primeira versão usava apenas as repetições exactas, nas versões subsequentes introduziram-se as repetições aproximadas o que representou um importante avanço, contudo este processamento revela-se bastante moroso e consumidor de recursos de processamento.

Em [171] está descrito o algoritmo GenCompress que reclamou, no início deste século, o estatuto de mais avançado. A principal inovação foi a introdução da utilização das repetições aproximadas, que com se sabe são o principal recurso na compressão de sequências de ADN. Para caracterizar as repetições aproximadas foram consideradas

três operações de edição: inserções (*I, posição, base*), substituições (*R, posição, base*) e delecções (*D, posição*). Portanto, os *gaps* foram desconsiderados pelo custo extra envolvido na sua prospecção e codificação.

Em [173] descreve-se a investigação de compressão de ADN baseada em dicionário *off-line* usando a transformada de Burrows-Wheeler (BWT) para capturar elementos redundantes exactos e aproximados nas sequências de ADN. Nesse trabalho usou-se um modelo probabilístico baseado em blocos. A transformada de BW é utilizada para uniformizar blocos pela transladação controlada dos símbolos, este processo é reversível, o que se utiliza para a descodificação. Quando o bloco transformado está mais uniforme é possível representá-lo mais compactamente, por exemplo usando *Run Length Encoding* (RLE), que significa codificar não exaustivamente, mas apenas a qualidade e a quantidade dos símbolos consecutivos que aparecem a seguir na sequência.

Em [175] descreve-se a implementação e os resultados da aplicação de um modelo NML (*normalized maximum likelihood*) na compressão de ADN sem perdas. A metodologia empregue é baseada num modelo probabilístico de predição, baseado num modelo de regressão discreta NML adequado à modelação das repetições aproximadas do ADN, em conjunção com um modelo de Markov de primeira ordem.

Em [12] apresenta-se o DNACompress, um algoritmo de referência na compressão do ADN. Funciona, numa primeira fase, com uma aplicação de prospecção de padrões genómicos que representa o estado da arte nesta matéria – o PatternHunter - [118], quer pela sensibilidade, quer pela rapidez que patenteia, funcionando com recurso a *spaced seeds* [139]. Esses padrões são posteriormente organizados num dicionário mais compacto que será posteriormente indexado para atingir a compressão.

Em [186] expõe-se um algoritmo rápido e simples para a compressão de sequências de ADN, reclama o estatuto de mais rápido e económico em termos de recursos de processamento. Em regra, os compressores mais sofisticados existentes para o ADN necessitam de vastas quantidades de memória principal para guardar estruturas de dados altamente complexas. A solução apresentada neste trabalho reclama superioridade em

relação à concorrência na rapidez e na possibilidade de trabalhar com sequências de dimensões elevadas, com centenas de MBs.

Em [187], o desafio da compressão de ADN foi revisitado e nele é dado a conhecer o algoritmo *DNAPack*, que a seguir se compara aos restantes de referência fazendo uso da Tabela 3.5. Este algoritmo é baseado em programação dinâmica localizada na fase de prospecção de padrões, assim, a programação dinâmica é apenas aplicada na imediações de *seeds* exactas detectadas.

sequence	length	BioCompress-2	GenCompress	CTW-LZ	DNACompress	DNAPack
CHMPXX	121024	1.6848	1.6730	1.6690	1.6716	**1.6602**
CHNTXX	155844	1.6172	1.6146	1.6120	1.6127	**1.6103**
HEHCMVCG	229354	1.8480	1.8470	1.8414	1.8492	**1.8346**
HUMDYSTROP	33770	1.9262	1.9231	1.9175	1.9116	**1.9088**
HUMGHCSA	66495	1.3074	1.0969	1.0972	**1.0272**	1.039
HUMHBB	73308	1.8800	1.8204	1.8082	1.7897	**1.7771**
HUMHDABCD	58864	1.8770	1.8192	1.8218	1.7951	**1.7394**
HUMHPRTB	56737	1.9066	1.8466	1.8433	1.8165	**1.7886**
MPOMTCG	186609	1.9378	1.9058	1.9000	**1.8920**	1.8932
PANMTPACGA	100314	1.8752	1.8624	1.8555	1.8556	**1.8535**
VACCG	191737	1.7614	1.7614	1.7616	**1.7580**	1.7583
Average	—	1.7837	1.7428	1.7389	1.7254	**1.7148**

Tabela 3.5 - Os algoritmos de compressão de ADN de referência e a sua comparação de desempenho com o *DNAPack*.

O *DNAPack* codifica por codificação aritmética ou CTW, a porção não repetitiva, pelo que simula a compressão alcançada por cada um dos métodos acabando por escolher o mais vantajoso. Assim, consegue alguma vantagem, mas essa ligeira vantagem tem o custo do processamento acrescido. Neste algoritmo foi colocada acentuada ênfase nos aspectos da codificação de toda a estrutura do ficheiro compacto, a estratégia de codificação dos padrões, bem como a codificação dos índices são muito racionais.

Muito recentemente, em 2007, uma nova abordagem [14] do problema da compressão de sequências de ADN, baseada na utilização de um painel de peritos na predição de ADN [188], compostos por peritos em replicações, palindromas, probabilidades baseadas em análise local, probabilidades baseadas em análise global, combinadas com codificação aritmética, conseguiu um ligeiro avanço neste domínio. Trata-se de uma abordagem por predição probabilística, porém, existem várias predições por entidades especializadas e baseadas em especificidades das sequências de ADN, sendo que cada

entidade vai reforçando ou perdendo o valor das suas predições caso os acertos no passado recente sejam ou não continuados. Deste modo, foi possível de forma simples, tirar partido deste modelo de *experts* (*XM – eXpert Model*) no sentido de comprimir as sequências de ADN do corpus usual, utilizado na maioria das publicações neste domínio, numa proporção diminuta mas, significativa dada a dificuldade da empresa. Os resultados são apresentados na Tabela 3.6.

Sequence	BioC	GenC	DNAC	DNAP	CDNA	GeMNL	XM
CHMPXX	1.6848	1.6730	1.6716	1.6602	-	1.6617	**1.6577**
CHNTXX	1.6172	1.6146	1.6127	1.6103	1.65	1.6101	**1.6068**
HEHCMVCG	1.8480	1.8470	1.8492	**1.8346**	-	1.8420	1.8426
HUMDYSTROP	1.9262	1.9231	1.9116	1.9088	1.93	1.9085	**1.9031**
HUMGHCSA	1.3074	1.0969	1.0272	1.039	**0.95**	1.0089	0.9828
HUMHBB	1.8800	1.8204	1.7897	1.7771	1.77	-	**1.7513**
HUMHDAB	1.8770	1.8192	1.7951	1.7394	1.67	1.7059	**1.6671**
HUMHPRTB	1.9066	1.8466	1.8165	1.7886	**1.72**	1.7639	1.7361
MPOMTCG	1.9378	1.9058	1.8920	1.8932	**1.87**	1.8822	1.8768
MTPACG	1.8752	1.8624	1.8556	1.8535	1.85	**1.8440**	**1.8447**
VACCG	1.7614	1.7614	1.7580	1.7583	1.81	**1.7644**	**1.7649**
Average	1.7837	1.7428	1.7254	1.7148	-	-	**1.6940**

Tabela 3.6 - Resultados comparados, em bits por base, das metodologias de compressão anteriores com o *XM*.

Neste mesmo trabalho aproveitou-se o painel de peritos para analisar cadeias de aminoácidos de proteínas tendo atingido igualmente resultados promissores.

3.6.1. Comentário ao estado da arte

A compressibilidade das sequências de ADN naturais não é linear, algumas denotam grande entropia e pouco melhor se consegue que os 2 bits/base, porém, em média, os valores de compressão que resultam da aplicação dos melhores algoritmos apenas se aproximam dos 1,7 bits/base, o que reflecte claramente a dificuldade da tarefa. Avaliando o estado da arte em termos de desempenho pelos resultados da Tabela 3.6, podemos considerar que uma liderança pouco destacada do *XM* e uma vice-liderança partilhada pelo *DNACompress* e pelo *DNAPack*. O *XM* é, em média, o que apresenta a melhor taxa de compressão (sensivelmente abaixo de 1,7 bits por base), o *DNACompress* é muito mais rápido com resultados muito semelhantes. O *DNApack*

apresenta boa sensibilidade na detecção de padrões conferida pelo recurso à programação dinâmica, contudo essa pesquisa mais exaustiva torna-o mais lento. Para contrapor esse défice foram introduzidas heurísticas com recurso a *seeds* para acelerar a prospecção de similaridades.

Em termos de metodologias, fica claro que a compressão por dicionário de padrões e a compressão por predição probabilística partilham as preferências e os créditos. A codificação aritmética de ordem 2 é, quase unanimemente, escolhida como técnica de apoio para a porção sobrante, cuja entropia é demasiado elevada para ser aproveitada pelas técnicas antes salientadas.

Neste trabalho, que procura contribuir para o estado-da-arte da compressão de sequências de DN, optou-se por uma solução tripartida, híbrida portanto, que serializa o processo de compressão pelas três técnicas. Numa primeira fase recolhe e aproveita os padrões por recurso a um dicionário, numa segunda faz a predição baseada num modelo probabilístico de linguagem construído atendendo às especificidades do ADN, e finalmente, para a informação predita e os seus desvios da real, conclui com a codificação aritmética.

Capítulo 4. Algoritmos bioinformáticos desenvolvidos para pesquisa de padrões exactos e aproximados

Como emerge dos capítulos anteriores, a análise das sequências do ADN é fundamental para uma boa parte das aplicações da bioinformática, onde se inclui igualmente a compressão de informação genómica. Neste capítulo revelam-se os detalhes de implementação dos algoritmos inovadores desenvolvidos para o propósito da pesquisa exacta e aproximada de padrões em sequências biológicas, com ênfase nas sequências de nucleótidos. Estes algoritmos serão determinantes na captação de recursos para a primeira fase de compressão, baseada em dicionário, bem como para a própria compactação do dicionário.

4.1. GRASPm: um novo algoritmo de pesquisa de padrões exactos orientado para a linguagem genómica

Basicamente um algoritmo de pesquisa de padrões exactos procura encontrar a totalidade das instâncias de uma subsequência padrão p, de tamanho m numa sequência x de tamanho n, sendo $n \geq m$. As sequências m e n são produtos de um alfabeto finito, de dimensão σ, sendo neste caso usado o alfabeto quaternário do ADN, $\Sigma=\{a,c,t,g\}$.

A maioria dos algoritmos que realizam a pesquisa de padrões exactos eficientemente operam em duas fases. A primeira fase corresponde ao pré-processamento, na qual se apreendem as especificidades do padrão, e eventualmente do texto, de forma a capitalizar esse conhecimento no sentido de obviar a pesquisa dos padrões na fase subsequente. A segunda fase, denominada de processamento, destina-se a percorrer iterativamente o texto, em busca de padrões exactos, fazendo-o da forma mais célere possível, sem no entanto, perder nenhuma ocorrência. Tal só é possível baseando-se no conhecimento reunido na fase primária do pré-processamento. Assim, a importância do pré-processamento é fundamental, mesmo crítica, pois compensa despender alguma atenção ao padrão para rentabilizar esse conhecimento na segunda fase. Atendendo a que o pré-processamento ocorre apenas uma vez, mas que a sua valia se aplica a milhares de iterações, fica mais fácil compreender a sua importância.

Uma revisão do trabalho relacionado neste campo foi apresentada na secção 3.1.1, nesta secção procurar-se-á descrever o novo algoritmo desenvolvido e compará-lo com os melhores competidores, pelo que apenas a estes se fará referência.

O algoritmo desenvolvido para pesquisa de padrões genómicos exactos foi denominado de GRASPm (*Genomic-oriented Rapid Algorithm for String Pattern-match*) pela forma como apreende e capitaliza o conhecimento do padrão, e pelo facto de colocar uma ênfase decisiva no pré-processamento.

4.1.1. Principais ideias do GRASPm

O GRASPm utiliza uma série de conceitos, inovadores nuns casos, renovados noutros que perfazem uma metodologia inovadora e eficiente para a pesquisa de padrões exactos em sequências genómicas. Destacam-se os seguintes conceitos:

- centrado em 2-gramas sobrepostos;
- ênfase no pré-processamento;
- utilização de janelas de pesquisa alargadas;
- dupla filtragem: primeiro filtro baseado na verificação do dupleto central da janela, segundo filtro baseado na verificação das bases que flanqueiam o dupleto central e na compatibilidade destas com o dupleto central;

- avanço da janela de pesquisa maximizado pelo somatório de duas componentes, uma fixa e outra variável.

Nas subsecções seguintes abordam-se os conceitos fundamentais do GRASPm.

4.1.1.1. 2-gramas como elemento de pesquisa

O algoritmo GRASPm utiliza os dupletos como unidades de pesquisa elementares, ou seja, lê e analisa tuplos de bases ao invés de bases. São usados 2-gramas sobrepostos, i.e., cada caracter marca o início de um 2-grama e num padrão de tamanho *m* existem *m-1* 2-gramas. Desta forma, consegue-se um bom compromisso entre tamanho do alfabeto, desempenho e complexidade espacial no que toca às estruturas necessárias para armazenar o conhecimento resultante do pré-processamento. Geralmente o elemento de pesquisa é o caracter/base, ou 1-grama. A ideia de aumentar artificialmente a dimensão do alfabeto não é nova, de facto já Boyer e Moore [97] se referem à possibilidade de usar n-gramas maiores que 1, inclusivamente há implementações com recurso a n-gramas orientadas à literatura do ADN [108, 189], e mais recentemente surgiram abordagens de *pattern-matching* com recurso a super-alfabetos [109]. Há que referir contudo que, as abordagens com n-gramas de grandes dimensões impediriam a pesquisa de padrões curtos, como codões por exemplo, que são vulgares na bioinformática, impossibilitariam ainda, a classificação do algoritmo como generalista, no sentido de pesquisar qualquer tamanho de padrão, precisamente os de tamanho inferior ao n-grama usado, enquanto que o GRASPm recorrendo a 2-gramas é generalista, apresentando solução para qualquer *m*. Mais gravosamente, utilizar n-gramas de grandes dimensões como forma de aumentar o alfabeto pode levar a estruturas de dados com dimensão proibitiva que ponham em causa o desempenho do próprio algoritmo ou do próprio sistema, em particular, quando se consideram sistemas de computação móvel eventualmente com recursos de memória limitados.

4.1.1.2. Janela de pesquisa alargada e estratégia de pesquisa

O GRASPm é, portanto, um algoritmo maioritariamente baseado em 2-gramas ou dupletos, ou até dinucleótidos, numa terminologia mais próxima da genómica. Outra

característica peculiar do GRASPm é o facto de, ao contrário da generalidade dos algoritmos que utilizam uma janela de pesquisa do tamanho do padrão para analisar iterativamente a possibilidade de ocorrência no texto, este utilizar uma janela de pesquisa substancialmente maior, quase o dobro, devido a uma estratégia de pesquisa diferente. Essa estratégia tem por objectivo estabelecer janelas maiores que o padrão, numa dimensão exacta de *2(m-1)* caracteres/bases, no intuito de operar uma primeira filtragem usando uma simples verificação do dupleto central da janela. Esta estratégia tem por base o Lema 4.1.

Lema 4.1

> Uma janela de pesquisa *i* de dimensão *2(m-1)* pode acomodar *m-1* alinhamentos de um padrão de tamanho *m*. Em qualquer eventual ocorrência de *p* na janela de pesquisa o dupleto central da janela *i*, cd_i, constituído, no caso da primeira janela, pelas bases *x[m-1]* e *x[m]*, deverá ser sempre um dupleto do padrão *p*. De igual modo, se o dupleto central (cd_i) não estiver contido no padrão *p*, não há ocorrências de *p* na janela *i*.

Prova

> Dando como exemplo um padrão de extensão 10 bases, e uma janela de pesquisa de extensão 2(10-1)=18, qualquer que seja a situação ou alinhamento do padrão no interior da janela, o dupleto central da janela, constituído pelas bases das posições 9 e 10, será sempre um dupleto do padrão. Vejam-se os casos extremos na Figura 4.1. Testando apenas o dupleto central de uma janela podemos, com segurança, afirmar se ela contém ou não pelo menos uma ocorrência do padrão.

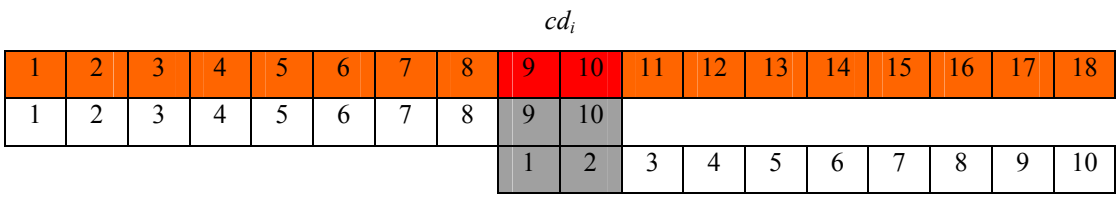


Figura 4.1 – Casos extremos de alinhamentos de ***p*** na janela de pesquisa.

A estratégia de usar janelas de pesquisa maiores, com avanços fixos, já tinha sido experimentada por outros investigadores [190], que conseguiram demonstrar a sua eficiência, porém os resultados apesar de melhores que os obtidos com janelas de tamanho *m*, não foram suficientes para normalizar esse procedimento e continua-se hoje a assistir à vigência dos algoritmos da família de Boyer Moore como sendo de referência na pesquisa exacta de padrões em linguagem natural. Kim em [190] enunciou uma regra similar à apresentada no Lema 4.1 denominando-a de regra da ocorrência.

Resultado de outras abordagens, aproveitando a rapidez com que os computadores executam operações de *bitwise*, surgiram os algoritmos de *bit parallelism*, que se apresentam como muito competitivos em todos os alfabetos, sendo considerados dos mais rápidos em alfabetos curtos ($\sigma \leq 8$), como é o caso do ADN. São baseados em autómatos não determinísticos, que analisam por *bit parallelism*, a extensão dos sufixos encontrados em cada janela, sendo que uma ocorrência é um sufixo total. Cada sufixo encontrado, de extensão 0 a *m*, permite calcular a deslocação segura da janela. A criação do algoritmo BNDM [191], em 2000, catapultou esta abordagem para a ribalta, reclamando os seus autores o estatuto de algoritmo mais rápido de sempre na altura de publicação, visto que superava largamente o desempenho dos melhores competidores para pesquisas em cadeias de ADN. Mais tarde, em 2003, surgiram evoluções e simplificações dessa abordagem que refinaram a lógica e o desempenho, sendo que o SBNDM [107] se apresenta hoje como um algoritmo de referência para aplicações em sequências de ADN. O SBNDM, segundo os seus autores supera o BNDM em 10%, em média.

Uma limitação deste tipo de algoritmos reside no facto de trabalharem com um comprimento de palavra (ω) igual ao da arquitectura da máquina em que são executados, sendo a norma 32 ou 64 bits, pelo que a sua performance é óptima quando o padrão a pesquisar se aproxima do máximo comprimento de palavra permitido pela arquitectura. Caso o padrão seja de extensão superior a ω então o algoritmo requer outra implementação pois terá de utilizar várias palavras para segmentar o padrão e organizar verificações repartidas e concertadas. Este constrangimento afecta claramente o desempenho e, de uma forma geral, este tipo de algoritmos aparece nas publicações

como sendo orientados para padrões de dimensão $\leq\omega$, o que é bastante limitativo para a bioinformática onde são usuais padrões com centenas de bases.

Na secção 3.1.1.1 descreveu-se o algoritmo WML [110], apresentado em 2007, que faz uso de n-gramas de maior dimensão ($3<n\leq8$) empregando, em tempo de pré-processamento, uma metodologia de *hashing* de todos os n-gramas possíveis para determinar, nessa fase, os avanços a incutir à janela de pesquisa nos casos que mais tarde, na fase de processamento, se venham a apresentar perante qualquer n-grama lido. O WML é um algoritmo da família Shift-Or [113] de última geração, adaptado da versão de Wu e Manber [112], que fora desenhada para a pesquisa de padrões múltiplos. Os resultados de desempenho do WML face aos melhores concorrentes demonstraram a utilidade dos n-gramas em pesquisas de dados baseados em alfabetos curtos, como é o caso das sequências genómicas. No dizer do seu autor, o WML demonstrou os melhores resultados em pesquisas em ADN, superando inclusivamente as melhores metodologias baseadas em *bit parallelism*. Note-se que o WML não tem limitações de comprimento do padrão pois não depende de ω, mas depende da utilização de n-gramas de grandes dimensões, o que lhe retira generalismo e o impede de pesquisar padrões de dimensão $<n$.

4.1.1.3. Filtragem, a introdução de filtragem suplementar pela regra da compatibilidade

O GRASPm é um algoritmo isento de limitações de tamanho de padrão, baseado em heurísticas, que numa combinação inovadora integra um novo conceito de selectividade que lhe permite um desempenho superior. A selectividade acrescida é resultante da integração de uma nova heurística a que se deu o nome de *regra da compatibilidade*. Esta nova regra é transversal a todo o algoritmo, tem início no pré-processamento, fase na qual são capturadas as características dos alinhamentos possíveis do padrão, e com a subsequente classificação desses alinhamentos como compatíveis ou incompatíveis face ao entorno onde serão testados mais tarde na fase de processamento. O potencial desta nova regra é enorme, e fica mais patente quando existem vários alinhamentos possíveis a testar, o que é mais vulgar quanto maior for o padrão, permitindo verificações em paralelo com um consumo de recursos de processamento mínimo. O princípio geral em que esta regra assenta enuncia-se como se segue.

Se determinado dupleto central da janela de pesquisa existir no padrão, então todos os alinhamentos possíveis dentro do padrão com esse dupleto deverão ser testados pois, algum ou todos podem coincidir com uma ocorrência do padrão no texto. Contudo, é possível estabelecer uma diferenciação, em tempo de pré-processamento, para as condições necessárias para que determinado alinhamento tenha sucesso, no sentido de corresponder a uma ocorrência efectiva. Numa abordagem simplista, o sucesso de determinado alinhamento pode ditar necessariamente o insucesso de outro(s) e como tal não vale a pena ser(em) verificado(s) nessa circunstância. Assim, na sua fase inicial a regra de compatibilidade define as condições em que cada possível alinhamento é classificado como compatível, e para todas as restantes circunstâncias ele será considerado incompatível e como tal inócuo, logo inibindo-se a sua verificação como forma de ganhar eficiência.

Tomando como exemplo o padrão *p="atgtgtgcat"*, com *m=10* e constituído por *m-1* dupletos (*at, **tg**, gt, **tg**, gt, **tg**, gc, ca, at*), pode antever-se o que sucederá caso o dupleto central da janela de pesquisa na fase de processamento seja igual a "*tg*", o que corresponderá à necessidade de verificar 3 possíveis alinhamentos, tantos quantos as ocorrências de "*tg*" no padrão. A Tabela 4.1 representa esses possíveis alinhamentos e através dela podem compreender-se as condições de compatibilidade.

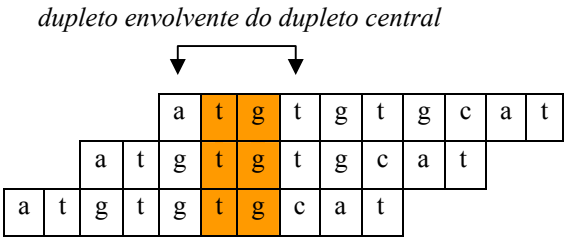


Tabela 4.1 – Análise das condições de compatibilidade dos alinhamentos com o dupleto "***tg***" no padrão.

Analisando os alinhamentos de "*tg*" pode afirmar-se que qualquer um deles, a verificar-se, exclui os demais. Considerando "*tg*" um futuro dupleto central (cd_i), basta atender à base antecessora desse dupleto central para poder emanar uma regra que diferencie a compatibilidade dos alinhamentos de "*tg*". Assim, pode pré-estabelecer-se que se a base anterior a cd_i for igual a "*a*" então o primeiro alinhamento é compatível e os restantes

incompatíveis, caso seja "*g*" então o segundo e o terceiro serão compatíveis e o primeiro incompatível. Se a base anterior a dc_i for outra base, "*c*" ou "*t*", todos os alinhamentos são incompatíveis e como tal, com uma simples verificação de um caracter são dispensados três custosos procedimentos de verificação de alinhamento. Esta simples regra permite, sobretudo em padrões de maiores dimensões, economias de processamento muito significativas e pode-se afirmar, por comprovação experimental (secção 4.1.4), que a regra da compatibilidade é o núcleo do algoritmo, decisiva e responsável maior pelo ganho de eficiência em relação a outras abordagens.

O algoritmo GRASPm emprega a regra da compatibilidade, não apenas com a base antecessora mas com o dupleto envolvente do dupleto central ($idcd_i$), formado pelas duas bases que flanqueiam o dupleto central dc_i. Esta opção insere-se na filosofia do algoritmo, que é baseado em dupletos e permite, ainda mais, acentuar as diferenciações de compatibilidade, já que a compatibilidade fica dependente da verificação de duas bases, o que torna menos prováveis as condições de compatibilidade que levam a processamentos mais dispendiosos. Nestas escolhas ponderaram os resultados do trabalho experimental, do qual emergiu o melhor compromisso para o favorecimento do desempenho do algoritmo. Enquanto a casos particulares da regra de compatibilidade, refira-se que caso o dupleto central coincida com o dupleto inicial ou final do padrão, a regra de compatibilidade ficará reduzida a uma base, respectivamente a antecessora ou a sucessora do dupleto central da janela.

Na fase de processamento, são iterativamente definidas janelas de pesquisa que sondam todo o texto, na análise a cada uma das sucessivas janelas de pesquisa apenas os alinhamentos compatíveis são testados em detalhe. Aqui, por detalhe, quer-se significar que são verificadas as bases remanescentes do alinhamento que ainda não foram verificadas, caso todas coincidam com o padrão então declara-se a ocorrência do padrão nas posições verificadas. Convém explicitar que a verificação exaustiva do alinhamento se faz no prefixo e no sufixo, considerando como divisória o dupleto central, sendo que se o dupleto central coincidir com o final do padrão não há sufixo e caso o dupleto central coincida com a início do padrão não há prefixo para verificar. O dupleto envolvente do dupleto central ($idcd_i$) é verificado pela regra de compatibilidade dispensando, assim, verificações adicionais.

4.1.1.4. Avanço maximizado da janela de pesquisa

O avanço da janela de pesquisa é outro aspecto importante do algoritmo, o GRASPm utiliza cumulativamente duas componentes de avanço da janela em cada iteração. A primeira é fixa, corresponde a *m-1* bases e decorre da estratégia de pesquisa adoptada e do Lema 4.1, do qual se pode inferir que esgotadas todas as possibilidades de ocorrências do padrão numa determinada janela, a próxima ocorrência se encontrará no mínimo uma base adiante do alinhamento do extremo direito da janela anterior (ver Figura 4.1). Assim, justifica-se a componente constante correspondente a *m-1* bases, pois o alinhamento do extremo esquerdo da nova janela permitirá testar sem descontinuidades os alinhamentos por verificar. Incorporando a heurística de avanço empregue no algoritmo de BMH [105] (ver secção 3.1.1), adaptada no GRASPm para funcionar com dupletos, esta pode ser utilizada como uma componente suplementar de avanço variável. Esta componente é dependente da composição do padrão. O somatório das componentes fixa e variável representa o número total de bases de avanço de uma janela para a próxima. No máximo serão *2(m-1)* bases, que resultam da primeira componente fixa de *m-1* bases, mais *m-1* bases caso o dupleto indicado provisoriamente pelo primeiro avanço como novo dupleto central não integre o padrão. De referir que nos algoritmos concorrentes o avanço é no máximo de *m* caracteres, mas em sequências de ADN com uma alfabeto quaternário, esse valor é raramente obtido pela distribuição mais ou menos uniforme dos símbolos.

Depois de descrito de forma geral o funcionamento do novo algoritmo GRASPm, cumpre analisar o funcionamento das duas fases, pré-processamento e processamento, revelando os respectivos pormenores de implementação. No intuito de aumentar a riqueza da explicação, a descrição das fases será realizada em paralelo com a análise de um exemplo concreto, evidenciando a selectividade e a eficiência do GRASPm.

No imediato e de forma a dar uma visão global do algoritmo, integrando esquematicamente os conceitos aventados que regem o funcionamento do algoritmo em todas as suas fases e processos apresenta-se, na Figura 4.2, a representação do diagrama de fluxo do GRASPm.

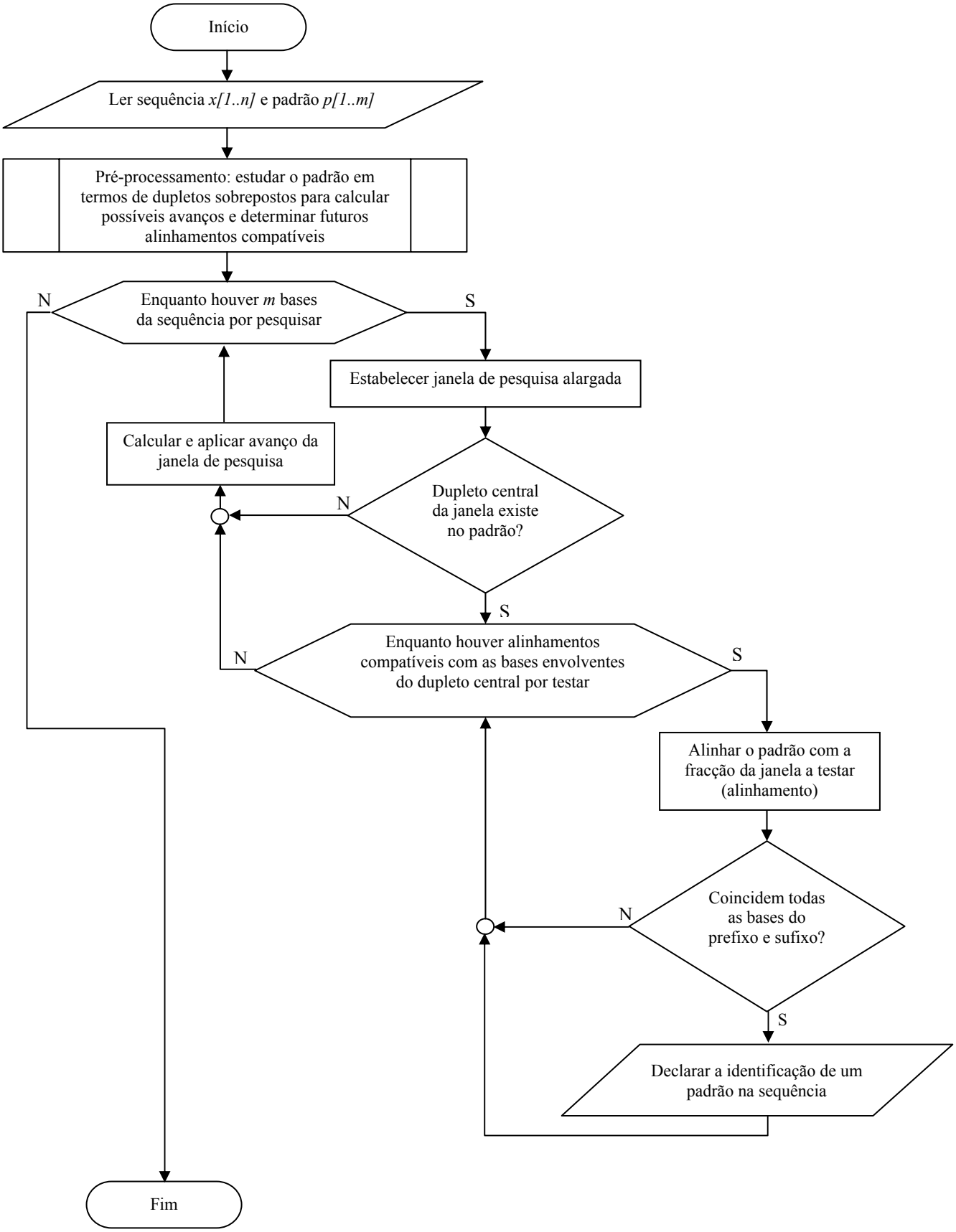


Figura 4.2 – O fluxograma do GRASPm.

4.1.2. Fase de pré-processamento

A fase de pré-processamento do GRASPm é mais substantiva do que na maioria dos algoritmos concorrentes, a regra da compatibilidade obriga a um pré-processamento mais aturado, e o facto de se utilizarem 2-gramas como unidades elementares também intensifica o pré-processamento. A tal corresponde um maior esforço de processamento e mais recursos envolvidos, contudo este investimento inicial é no final rentabilizado, já que no decurso da fase de pesquisa a maior selectividade propicia melhor desempenho. Antes de entrar efectivamente na análise, há que criar as estruturas de dados necessárias para concretizar o pré-processamento e garantir a disponibilidade do conhecimento adquirido para a fase subsequente, a de pesquisa ou processamento.

4.1.2.1. Estruturas de dados

Como referido, o GRASPm analisa dupletos em todas as fases. Havendo 16 diferentes dupletos, para os referenciar cada dupleto é convertido num código de 1 a 16. Assim, carece de uma estrutura de dados que lhe permita ler as bases que perfazem um dupleto e adquirir eficientemente o seu código ou índice. Para este desiderato, usa-se uma tabela bidimensional (ver Tabela 4.2), 4*4, denominada *ituplos* que permite através da intersecção linha-coluna, com as diferentes bases atribuídas a cada linha ou coluna, determinar o código do dinucleótido ou dupleto.

0		1	2	3	4
		a	c	t	g
aa 1	1 a	aa 1	ac 2	at 3	ag 4
ca 5	2 c	ca 5	cc 6	ct 7	cg 8
ta 9	3 t	ta 9	tc 10	tt 11	tg 12
ga 13	4 g	ga 13	gc 14	gt 15	gg 16

Tabela 4.2 – Tabela *ituplos*, para obter o índice de um dupleto.

Chama-se a atenção para o facto de na coluna 0 se repetirem os códigos da coluna 1 como forma de assegurar que, caso o último dupleto central usado na pesquisa a uma sequência coincida com o último dupleto dessa sequência e a última base do dupleto envolvente ($idcd_i$), usado na regra da compatibilidade, não exista, esse facto não comprometa a regra da compatibilidade, pelo que se atribui um dupleto envolvente com a segunda base igual à primeira.

Para obter o índice de cada base, de 1 a 4, recorre-se simplesmente a um vector (*ibases*) como o da Tabela 4.3, que atribui um código diferenciador a cada base em função do respectivo código ASCII do caracter. Deste modo, e contando com a tabela *ituplos*, pode-se obter, por exemplo, o código do dupleto "*gt*" através de uma simples expressão (exemplificada em linguagem C): *ituplos [ibases['g']][ibases['t']]*

a		*c*		*g*		*t*
97	…	99	…	103	…	116
1	…	2	…	4	…	3

Tabela 4.3 – Vector de códigos das bases em função do ASCII do respectivo caracter.

Preparadas as condições de base, as incumbências do pré-processamento são três, todas elas destinadas a estudar as especificidades do padrão e a organizar o conhecimento adquirido. As três tarefas são a análise dos dupletos do padrão, a antevisão e definição das condições de compatibilidade dos alinhamentos e por fim, o cálculo da tabela suplementar de avanços. A ordem pode ser considerada semi-arbitrária, já que apenas há dependência sequencial entre o estudo dos dupletos do padrão e a definição das condições de compatibilidade dos alinhamentos, assim sendo, elegeu-se começar pelo estudo dos dupletos do padrão. Para acompanhar os desenvolvimentos do algoritmo nas facetas subsequentes incorpora-se um exemplo representativo de pesquisa de um padrão *p* numa sequência *x* com as seguintes especificações:

Padrão *p="atgtgtgcat"*,
com *m=10* e constituído por *m-1=9* dupletos (*at, tg, gt, tg, gt, tg, gc, ca, at*)
Sequência *x*="*accgtatcattgcccatgtgtgcatgtgccaattctcgagtaccc*",
com *n*=45

4.1.2.2. Estudo dos dupletos do padrão

A análise do padrão principia pela recolha e anotação das características dos dupletos que o compõem. Cada dupleto no padrão é caracterizado pela posição (número da base) em que ocorre e pelo número total de ocorrências.

O resultado da análise do padrão *p* que serve de exemplo está exposta na Tabela 4.4, que deverá ser preservado para a análise em pré-processamento da regra da compatibilidade. Servirá igualmente para rapidamente saber se determinado dupleto existe no padrão, o que corresponde ao primeiro filtro da fase de pesquisa.

Índice	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dupleto	*aa*	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
Ocorrências	0	0	2	0	1	0	0	0	0	0	0	3	0	1	2	0
Índices das ocorrências			1		8							2		7	3	
			9									4			5	
												6				

Tabela 4.4 – Estudo dos dupletos do padrão.

4.1.2.3. Análise das condições de compatibilidade dos dupletos/alinhamentos

O objectivo desta análise é preencher adequadamente a tabela de compatibilidades. Dita tabela é uma tabela tridimensional, com as seguintes dimensões (para o caso de tamanho máximo de padrão=256 bases): 16*16*256. Serão 16 sub-tabelas, uma por cada dupleto existente e que virá, eventualmente, a igualar-se ao dupleto central. Em cada uma das sub-tabelas regista-se por cada dupleto, que eventualmente constituirá o dupleto envolvente do dupleto central ($idcd_i$), quais os alinhamentos (pelas suas posições de alinhamento) compatíveis, o que equivale a dizer, em quais será necessário proceder a verificações exaustivas. Os valores iguais a 0 para um determinado $idcd_i$ significam que o alinhamento é incompatível e portanto determina-se inibido. Vejam-se as sub-tabelas de compatibilidade para o exemplo em curso aglutinadas na Tabela 4.5. Note-se que apenas se fornecem as tabelas dos dupletos que constituem o padrão, pois serão as únicas que serão consultadas durante a fase de processamento.

"at"

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
0	0	0	1	9	9	9	9	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

"tg"

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
0	0	2	0	0	0	0	0	0	0	0	0	0	6	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

"gt"

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

"gc"

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

"ca"

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Tabela 4.5 – Sub-tabelas de compatibilidades dos alinhamentos do padrão.

4.1.2.4. Cálculo da tabela suplementar de avanços

A componente suplementar de avanço é avaliada após exercida a primeira componente fixa (*m-1* bases) que indica a posição preliminar para a nova janela, mais concretamente para o próximo dupleto central cd_{i+1}. Se considerarmos $s=m-1$ como a primeira componente do avanço, então o dupleto iniciado em cd_i+s será o argumento da função de avanço suplementar. É possível através da análise desse dupleto promulgar um valor

de avanço seguro para a nova iteração. Esta heurística corresponde à heurística do "mau caracter" introduzida por Boyer e Moore, aqui adaptada para lidar com dupletos. Basicamente, indica-se para cada dupleto, em que medida é que a sua presença como eventual dupleto central, considerando a sua posição na composição do padrão, permite avançar a janela de pesquisa sem correr o risco de perder ocorrências. A Figura 4.3 ilustra as situações que poderão ocorrer e a actuação respectiva da regra de avanço variável baseada no "mau dupleto".

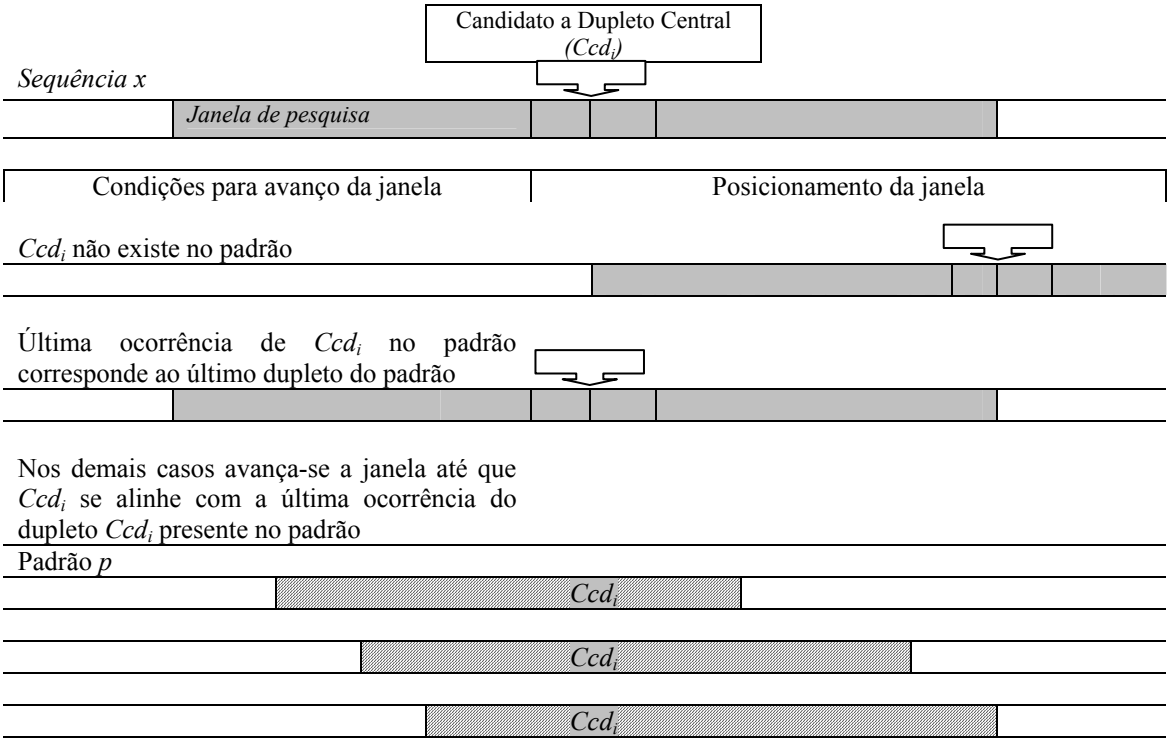


Figura 4.3 – Esquema ilustrativo do funcionamento da regra do "mau dupleto".

Analisando o exemplo em decurso, como o dupleto "*aa*" não ocorre no padrão *p*, caso o dupleto central preliminar cd_i+s seja "*aa*", então a janela pode avançar mais *m-1* bases pois é garantido que só poderá existir uma nova ocorrência do padrão depois de cd_i+s. Por outro lado, caso cd_i+s fosse "*gt*", sendo que esse dupleto ocorre duas vezes em *p*, a última na posição 5, para não correr o risco de perder uma ocorrência do padrão que o integre, a componente variável ou suplementar do avanço poderia ser, neste caso, no máximo 4, de modo a que todo o alinhamento correspondente ficasse ainda dentro da nova janela. O último dupleto no padrão corresponde a um avanço suplementar nulo, qualquer avanço por mínimo que seja excluiria esse alinhamento e uma eventual

ocorrência do padrão ficaria por assinalar. A regra de cálculo pode então ser descrita como a diferença da posição da última ocorrência do dupleto cd_i+s no padrão para a posição do último dupleto no padrão, portanto no caso de "*gt*" seria, como visto, 9-5=4. Veja-se na Tabela 4.6 o resultado do cálculo da regra do avanço extra ou suplementar para todo os dupletos do padrão *p*.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
aa	*ac*	*at*	*ag*	*ca*	*cc*	*ct*	*cg*	*ta*	*tc*	*tt*	*tg*	*ga*	*gc*	*gt*	*gg*
9	0	9	9	1	9	9	9	9	9	9	3	9	2	4	9

Tabela 4.6 – Tabela de avanços suplementares variáveis.

4.1.3. Fase de processamento

A fase de processamento consiste em definir iterativamente o posicionamento da janela de pesquisa e procurar no seu interior ocorrências do padrão de forma selectiva. A selectividade é conferida por dois filtros, o primeiro corresponde à identificação e verificação do dupleto central da janela de pesquisa (cd_i) consultando a informação detida na tabela resultante do estudo dos dupletos do padrão (ver Tabela 4.4). Caso tal dupleto central seja um dos dupletos do padrão inicia-se a segunda filtragem, caso contrário a iteração está terminada, com base no Lema 4.1, e pode proceder-se para a próxima iteração.

A segunda filtragem corresponde à aplicação da regra da compatibilidade. A regra usa como referência o dupleto envolvente do dupleto central ($idcd_i$). Procurando as condições de compatibilidade para cd_i quando envolvido por $idcd_i$ na tabela de compatibilidades (ver Tabela 4.5), é possível saber se há alinhamentos compatíveis casuisticamente, e caso os haja dispor das posições para alinhar o padrão para proceder às verificações caracter a caracter. Principia-se pelo sufixo, caso necessário segue-se o prefixo, caracter a caracter, por ordem reversa, até que haja uma falha ou se consume uma correspondência de *m* caracteres.

Como ficou demonstrado, as comparações exaustivas só acontecem em última análise, nos casos duplamente filtrados, onde a probabilidade de existência de uma réplica do

padrão é significativa, normalmente de 4 em *m*, pois duas bases do cd_i mais duas do $idcd_i$ já foram verificadas.

Analisam-se agora as iterações que o algoritmo GRASPm necessita para pesquisar ocorrências exactas de *p* em *x*. Assim, considerando a primeira iteração, que se inicia com a janela de pesquisa inicialmente alinhada com o início da sequência *x*, temos que o primeiro dupleto central cd_1=*x[9]+x[10]=”at”*. Esta iteração inicial está retratada na Figura 4.4 a seguir.

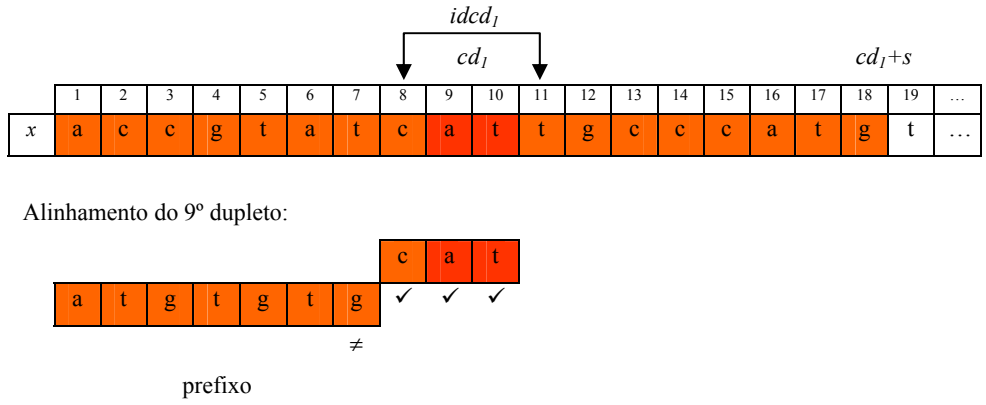


Figura 4.4 – Primeira iteração para o exemplo proposto.

Observando e descrevendo o comportamento do algoritmo, o dupleto central “*at*” corresponde a um dupleto do padrão pelo que a possibilidade de ocorrência do padrão ainda não pode ser descartada. Entra em acção a regra da compatibilidade para definir que, atendendo ao dupleto envolvente do dupleto central $idcd_1$ =”*ct*”, e consultando a tabela de compatibilidades (ver Tabela 4.5) para o binómio (cd_1, $idcd_1$), tem-se que apenas um alinhamento (apesar de haver dois para “*at*” no padrão) necessita verificação, especificamente o que alinha o dupleto central com o 9º dupleto do padrão. O alinhamento é realizado, e não havendo sufixo, as verificações caracter a caracter iniciam-se no prefixo, neste caso na 7ª base do padrão. A primeira verificação falha e, como não há mais alinhamentos compatíveis, com isso terminam as verificações na primeira iteração. A segunda iteração deve ser preparada, pelo que a nova localização da janela de pesquisa deve ser calculada, somando o avanço fixo *s=m-1* com o avanço variável para cd_1+s. Recorrendo à tabela de avanços suplementares (ver Tabela 4.6) o avanço previsto para o dupleto cd_1+s=”*gt*” é de 4 posições, pelo que o avanço total é de *9+4=13* bases, o que desloca o próximo dupleto central para a $9+13=22^a$ base de *x*.

A ilustração das componentes do avanço no final da 1ª iteração é fornecida na Figura 4.5.

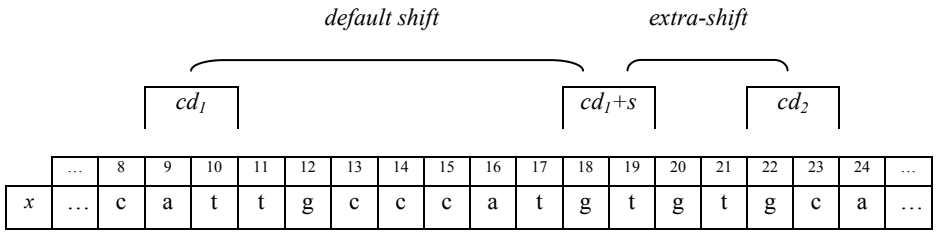


Figura 4.5 – Avanço da janela de pesquisa no final da 1ª iteração.

Note-se que para os casos em que se testam múltiplos alinhamentos, no sentido de descobrir as eventuais ocorrências do padrão pela ordem de aparição correcta, a tabela de compatibilidade possui os alinhamentos a testar ordenados por ordem de aparição, ou seja, alinhamentos com dupletos em posições avançadas acontecem antes de alinhamentos com dupletos em posições recuadas.

Na segunda iteração (ver Figura 4.6) o dupleto central cd_2="*gc*" existe no padrão, possui um alinhamento compatível com o $idcd_2$="*ta*", portanto as comparações de caracteres depois de efectuado o alinhamento (pelo 7º dupleto do padrão) são necessárias. Verificados com sucesso os caracteres do sufixo e do prefixo pode afirmar-se a existência de uma ocorrência do padrão.

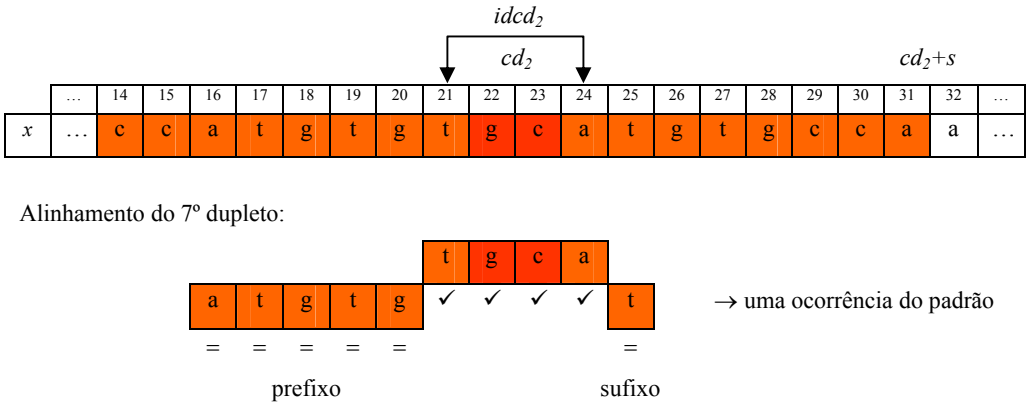


Figura 4.6 – Estado do algoritmo no decurso da 2ª iteração.

Sendo o único alinhamento compatível dão-se por terminadas as verificações na 2ª iteração e procede-se ao cálculo do avanço para a próxima iteração (ver Figura 4.7). Assim, temos que cd_2+s="*aa*", e sendo "*aa*" um dupleto que não integra o padrão, o

avanço extra para "*aa*" será máximo e igual a *m-1*, sendo o avanço total igualmente máximo e igual a *2s=18*. O próximo cd_3 localizar-se-á na base *22+18=40*.

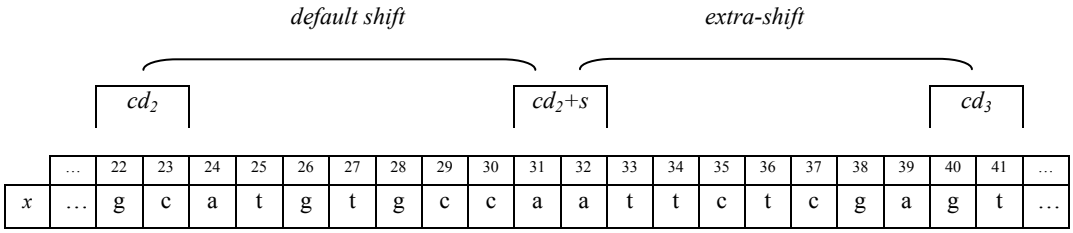


Figura 4.7 - Avanço da janela de pesquisa no final da 2ª iteração.

Na terceira iteração (ver Figura 4.8), a janela de pesquisa inclui um dupleto central cd_3="*gt*". O dupleto "*gt*" tem múltiplas ocorrências no padrão, o que poderá significar a necessidade de testar vários alinhamentos, porém a regra de compatibilidade obvia essa necessidade como se constata. Assim, a regra de compatibilidade para o dupleto envolvente $idcd_3$="*aa*" determina que, para o binómio (cd_3, $idcd_3$), todos os alinhamentos são incompatíveis e com esta única verificação descartamos qualquer verificação subsequente. A importância da regra da compatibilidade no GRASPm é determinante e decisiva no seu desempenho. Em rigor, através de um esforço de pré-processamento podemos, em tempo de pesquisa, realizar verificações em paralelo, no caso presente efectuaram-se 4 verificações analisando efectivamente 2 caracteres, o que optimiza a eficiência. Esta regra permite-nos com uma fracção do esforço de processamento dos algoritmos concorrentes obter os mesmos resultados, o que redunda em maior eficiência.

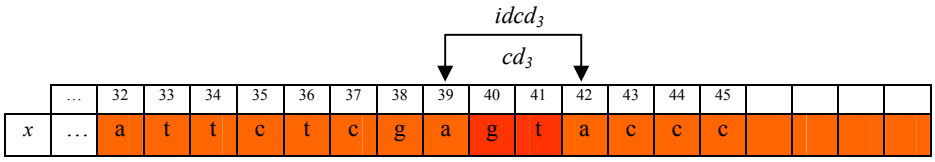


Figura 4.8 – Estado do algoritmo no decurso da 3ª iteração.

Findas as verificações, efectua-se o cálculo do avanço para a janela de pesquisa seguinte que, neste caso já não será analisada pois cd_4 apresentará um valor maior que *n*, o que

significa que toda a sequência foi pesquisada. Terminadas as iterações conclui-se o algoritmo.

Para formalizar e concluir os detalhes de implementação relativos à fase de pesquisa, fornece-se uma proposta de implementação para a fase de processamento do GRASPm (ver Figura 4.9), programada em linguagem C, e onde fica exposta a sua simplicidade.

```
void GRASPm_Search(char *seq, long n, char *pat, unsigned int m)
   {
   long cd;
   int s,id,idcd,ia,cal,ias,j,suf;

    s=m-1;
    cd=s-1;
    while (cd<n-1)
     {
       id=ituples[ibases[seq[cd]]][ibases[seq[cd+1]]];       // Verificação do
       if (duplets_in_pattern[id][1])                        // dupleto central
          {
            idcd=ituples[ibases[seq[cd-1]]][ibases[seq[cd+2]]];
            ia=1;
            while ((cal=compatibilities[id][idcd][ia])>0)   //  Ciclo de testes aos
              {                                             // alinhamentos compatíveis
               suf=m-cal-2;
               ias=cd+suf+2;
               j=0;
               while ((j<suf) && (seq[ias-j]==pat[s-j])) j++;
               if (j>=suf)
                 {
                  j+=4;
                  while ((j<m) && (seq[ias-j]==pat[s-j])) j++;
                  if (j>=m) printf("\nPattern at base %d.", ias-s);
                 }
               ia ++;
              }
          }
       cd+=s+xshift[ituples[ibases[seq[cd+s]]][ibases[seq[cd+s+1]]]];// Avanço cumulativo
     }
   }
```

Figura 4.9 – Proposta de implementação, em linguagem C, para a fase de pesquisa do GRASPm.

4.1.3.1. Casos particulares - garantir eficiência para padrões com *m≤2*

Para os casos particulares de padrões muito curtos, concretamente com *m=1* ou *m=2*, os algoritmos mais sofisticados não se demonstram adequados. De facto, para pesquisar caracteres simples, o algoritmo básico de força bruta que compara todos os caracteres, demonstra-se imbatível. Para os casos de *m=2* também é possível criar uma solução alternativa para aumentar a eficiência, aproveitando as vantagens decorrentes da dimensão reduzida do alfabeto genómico. Assim, desenvolveu-se uma solução baseada em processos simples que, apesar de não efectuar uma filtragem optimizada, efectua breves comparações para decidir o avanço a aplicar. Assente no facto simples de que um dupleto pode ser constituído por duas bases iguais ou diferentes, é possível prever o

comportamento do avanço caso o padrão seja de um tipo ou de outro. Iniciando a análise sempre pelo último caracter da janela simples (de tamanho *m*), se se procura um padrão constituído por duas bases iguais então se a verificação do último caracter da janela falhar significa que a janela pode avançar dois caracteres. Se por outro lado se procura um padrão constituído por duas bases diferentes e a falha ocorrer na segunda verificação então a janela pode igualmente avançar dois caracteres. Um lógica de pesquisa tão simples proporciona uma eficiência, em termos de tempo de execução, muito superior a qualquer um dos algoritmos em análise na comparação de desempenho efectuada na secção seguinte. Destarte, o GRASPm incorpora as metodologias referidas nesta subsecção para pesquisar padrões com $m \leq 2$.

4.1.4. Resultados experimentais e comparações

Para aferir o desempenho do novo algoritmo GRASPm, escolheram-se os mais competitivos de entre os algoritmos mais representativos na pesquisa de padrões exactos de utilização geral e de utilização orientada para literatura genómica. Assim, escolheu-se o BMH versão 2-gramas como representante dos algoritmos clássicos. Dos algoritmos recentes, escolheram-se apenas os mais eficientes e orientados para pequenos alfabetos, o SBNDM e o WML.

Quer o BNDM, quer o sucessor SBNDM são tidos como referência no domínio das pesquisas em ADN. As implementações utilizadas são genuínas, no caso do SBNDM a versão utilizada foi gentilmente cedida pelos seus autores e adaptada para ler 2-gramas sobrepostos e efectuar avanços baseados em 2-gramas, no caso do BMH versão 2-gramas foi adaptada de [192], na qual se integrou toda a lógica dos 2-gramas quer na pesquisa quer na regra de avanço.

O WML é um algoritmo muito recente (2007), que aproveita as vantagens da utilização de n-gramas de dimensão superior a 2 para tornar mais selectivas as pesquisas em dados derivados de alfabetos curtos. De forma a incluí-lo em condições de igualdade com os demais algoritmos em contenda foi necessário produzir uma adaptação para que o WML utilizasse apenas 2-gramas. Assim, denominar-se-á essa versão de WML2. As versões originais foram gentilmente cedidas pelo seu autor T. Lecroq.

Todas as implementações foram programadas em linguagem C e compiladas com o *gcc* (versão 3.4.2) parameterizado com optimização máxima –O3.

Como plataforma dos testes utilizou-se um sistema informático com base num processador Intel Pentium IV 3.4GHz, 8KB L1 + 512KB L2 de cache e 1GB de DDR-RAM, sobre Windows XP Professional SP2 OS. Os tempos de execução foram colectados com recurso a uma função incluída num biblioteca do sistema operativo, referida como *timeGetTime()*, que fornece medições instantâneas do tempo em milissegundos.

As sequências usadas foram: (*i*) o genoma da bactéria *E.Coli*, um ser procariota, cujo genoma completo ronda as 4,6 Megabases; e (*ii*) o Cromossoma 1 Humano, de um eucariota, cuja extensão se aproxima das 250 Mb. As escolhas são justificadas pela variedade a todos os níveis, sequências de eucariotas e procariotas, dimensão média e grande. Quanto aos padrões, foram criados aleatoriamente 100 padrões por cada uma das categorias de comprimentos de padrão testados, a saber, *m=2, 4, 8, 16, 32, 64, 128*. Ao todo são 700 padrões, que foram guardados num ficheiro, que serão utilizados para pesquisar as sequências antes anunciadas por cada um dos algoritmos em contenda. Para cada categoria será recolhido o tempo médio de pesquisa em milissegundos.

Os resultados verificados para o genoma da *E.Coli*, constam no gráfico presente na Figura 4.10, já na Figura 4.11 podem observar-se os resultados para o cromossoma 1 da espécie *Homo Sapiens*.

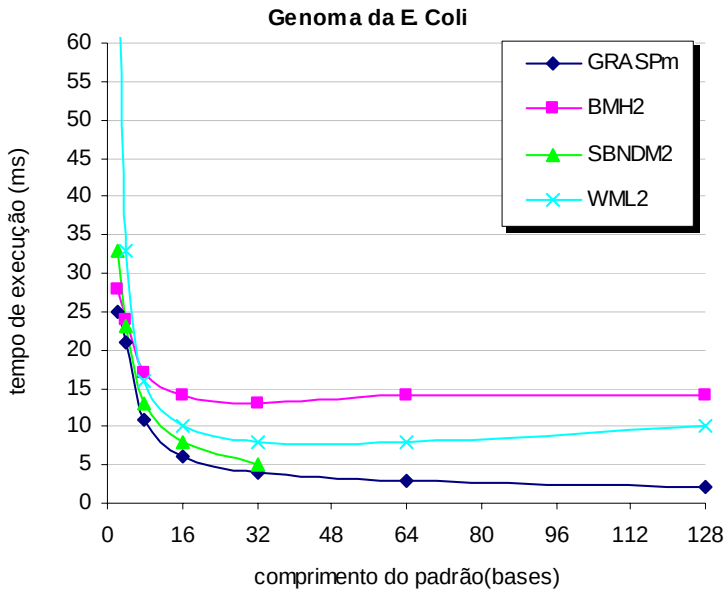


Figura 4.10 – Resultados de desempenho pesquisando padrões no genoma da *E. Coli*.

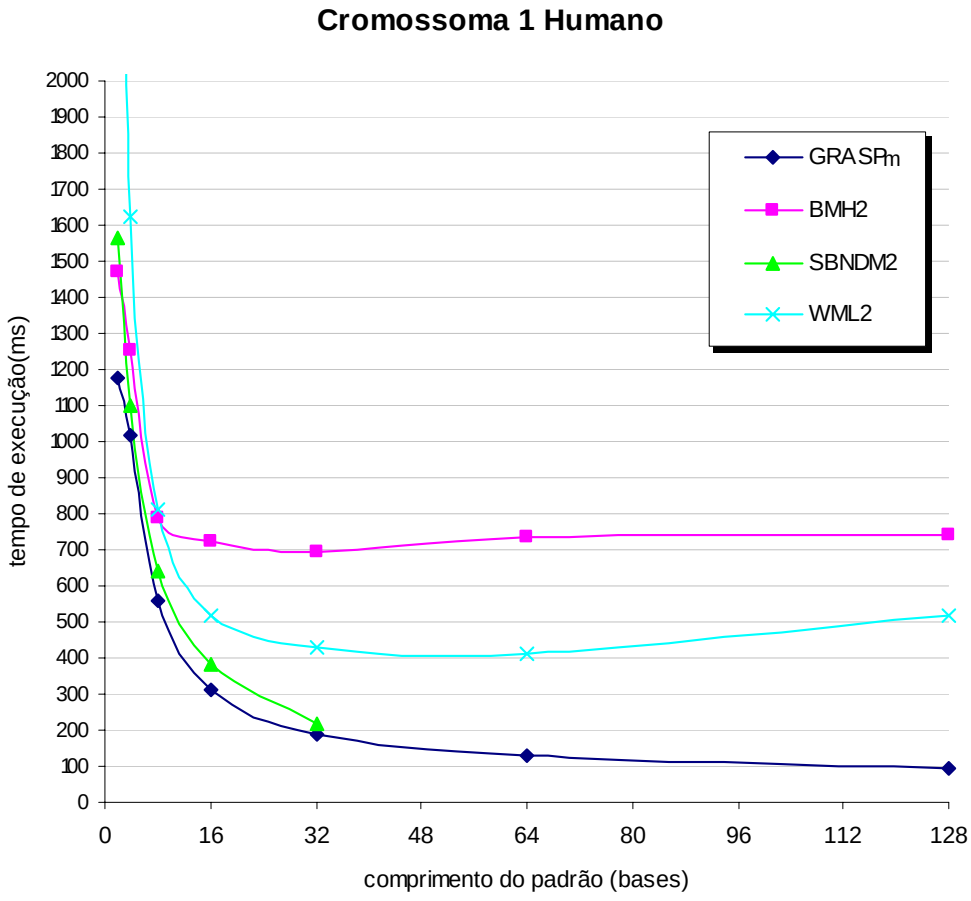


Figura 4.11 – Resultados de desempenho pesquisando padrões no cromossoma 1 humano.

Numa primeira análise é evidente a superioridade do GRASPm em todos os comprimentos de padrão considerados, sendo mais dilatada à medida que *m* aumenta. O algoritmo SBNDM2 é um concorrente competitivo, mas o GRASPm é superior em média em 30%, uma vantagem significativa. Já o algoritmo BMH2 reflecte uma clara inadequação à informação genómica, algo atenuada, é certo, pela utilização de 2-gramas mas, ainda assim com um tempo de execução muito superior à concorrência. O algoritmo BMH com 2-gramas padece dos mesmos problemas de fundo da versão 1-grama, ou seja, não está preparado para tirar partido do comprimento do padrão, sendo que o seu desempenho tende a estabilizar ou mesmo deteriorar-se para padrões de grande dimensão.

O algoritmo WML2 fica muito aquém, em termos de desempenho, das suas versões originais baseadas em n-gramas maiores. Em padrões de dimensão reduzida fica mesmo classificado em último lugar considerando os algoritmos em análise. Para obter um desempenho comparável ao GRASPm seria necessário recorrer ao WML3,com recurso a 3-gramas. Reforçando a justificação de não ter enveredado pela comparação de algoritmos baseados em diferentes dimensões de n-gramas, aponte-se que o algoritmo de Boyer-Moore [97] foi adaptado para pesquisar 4-gramas em [107], sendo que, com esse recurso, conseguiu superar o SBNDM na pesquisa genómica. Não será, no entanto científico afirmar a superioridade do algoritmo BM sobre o SBNDM pois se o SBNDM for igualmente adaptado para lidar com 4-gramas a diferença extingue-se e o SBNDM reafirma-se como superior ao BM em pesquisas genómicas. Pretende-se valorizar a metodologia essencial que o algoritmo utiliza na pesquisa e não a forma como pontualmente se contornam os problemas. Tal como o BHM2, o WML2 não consegue sustentar o aumento de desempenho à medida que o tamanho do padrão aumenta, sendo que para padrões maiores que 32 bases o desempenho estabiliza ou até se reduz ligeiramente.

Por seu turno o GRASPm e o SBNDM2 aceleram o seu desempenho em consonância com o incremento do padrão. No caso do SBNDM2 não foi possível testar para padrões superiores a $\omega=32$, que corresponde ao comprimento da palavra na arquitectura utilizada, isto porque o algoritmo original facultado pelos seus autores não incorpora essa possibilidade. Ainda assim, fica claro que mesmo para as melhores condições do SBNDM, o GRASPm emerge como claro dominador em desempenho. Ainda em abono

do GRASPm, acresce o facto de que não possui quaisquer limitações de implementação no tocante ao tamanho máximo do padrão a pesquisar.

Analisando os resultados noutra perspectiva, através dos gráficos cumulativos da Figura 4.12 e Figura 4.13, é possível constatar que no somatório dos tempos despendidos para, em média, pesquisar todas as categorias de padrões utilizadas, o GRASPm reafirma a sua superioridade por número destacados. Adicionalmente, estes gráficos permitem aferir os valores exactos, em milissegundos, que cada algoritmo emprega, em média, para pesquisar integralmente nas sequências utilizadas nos testes os diferentes tipos de padrões usados.

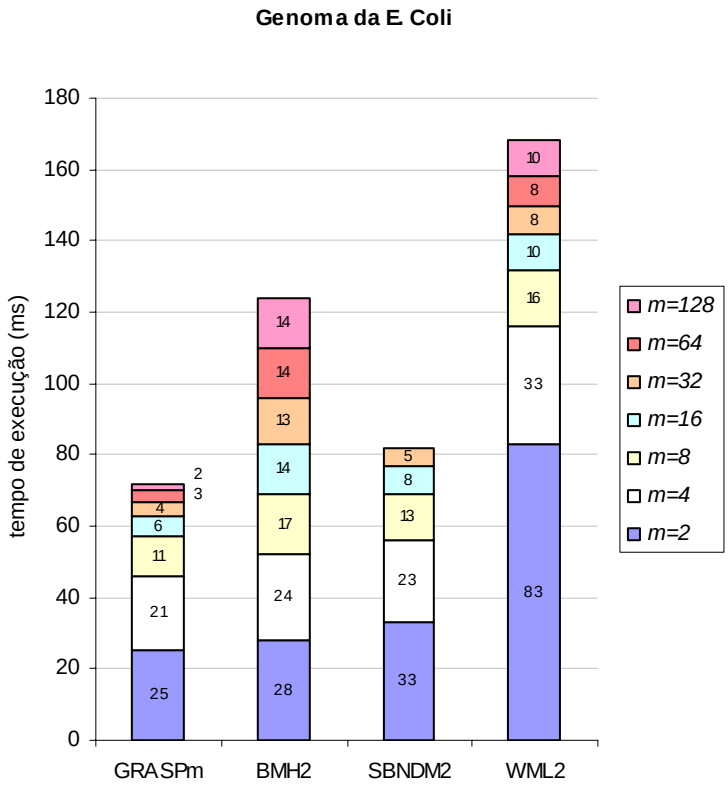


Figura 4.12 – Resultados de desempenho comparado no somatório dos tempos médios das pesquisas efectuadas para o genoma da *E. Coli*.

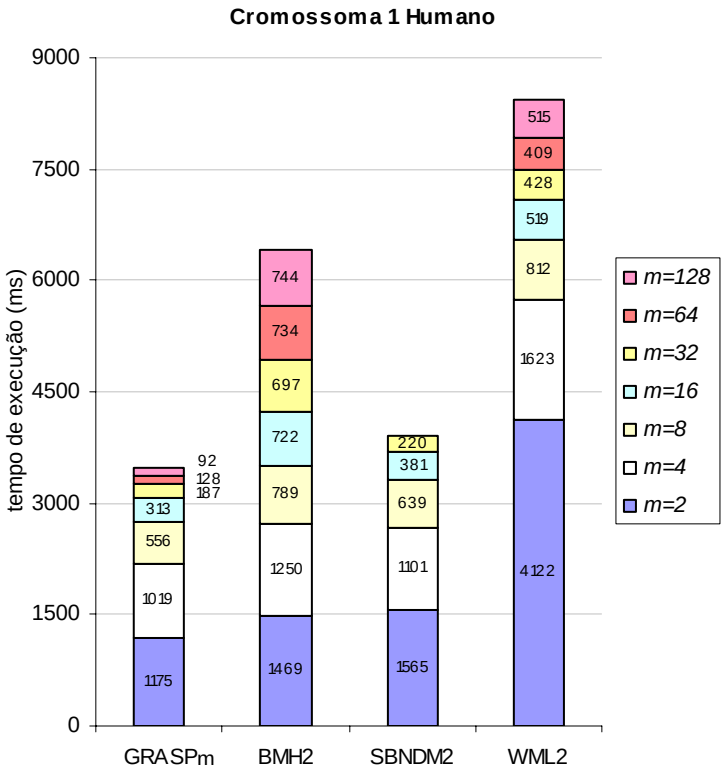


Figura 4.13 – Resultados de desempenho comparado no somatório dos tempos médios das pesquisas efectuadas para o cromossoma 1 humano.

Em complemento aos resultados de desempenho apresentados, incluem-se os resultados do número de comparações de caracteres (*ncc*) efectuadas pelos algoritmos com melhor desempenho, em média por cada categoria de padrões. De referir que o *ncc* é apenas um indicador, não há uma relação linear entre esse indicador e o desempenho global do algoritmo, isto porque a eficiência que inclui todas as componentes é a que se mede no tempo em que os resultados são computados e fornecidos. A Tabela 4.7 resume o *ncc* e o índice *ncc/n* verificados em testes com as mesmas configurações antes usadas para avaliar o desempenho dos algoritmos relativamente aos tempos de execução, desta feita usando apenas o genoma da *E. Coli*, com *n=4 639 221*. Perante os resultados, conclui-se que também no indicador *ncc* o GRASPm se superioriza. É de realçar que o SBNDM2 é igualmente um algoritmo extremamente eficiente, que resulta duma depuração da ideia original, já muito avançada e optimizada, e que o GRASPm está apenas na sua versão inicial.

	GRASPm		SBNDM2	
m	***ncc***	***ncc/n***	***ncc***	***ncc/n***
2	4317166	0,931	3297608	0,711
4	1981411	0,427	2187355	0,471
8	1276973	0,275	1290804	0,278
16	963560	0,208	968451	0,209
32	804039	0,173	809451	0,174
64	614791	0,133		
128	463904	0,100		

Tabela 4.7 – Número de comparações de caracteres versus comprimento do padrão.

4.1.5. Análise de complexidade

A análise de complexidade do algoritmo GRASPm pode ser vista nas duas fases do algoritmo. Na fase de pré-processamento a complexidade temporal e espacial é repartida pelas seguintes necessidades computacionais:

- Preparar as estruturas de dados;

Complexidade temporal associada: $O(m)+O(\sigma)$

Complexidade espacial associada: $O(m)+O(\sigma)$

- Analisar os dupletos que compõem o padrão;

Complexidade temporal associada: $O(2(m-1))$

Complexidade espacial associada: $O(\sigma^2 m)$

- Calcular a tabela de compatibilidades;

Complexidade temporal associada: $O(m^2)$

Complexidade espacial associada: $O(\sigma^4(m-1))$

- Calcular a tabela de avanços suplementares.

Complexidade temporal associada: $O(2m)$

Complexidade espacial associada: $O(\sigma^2)$

Na fase de processamento, a análise de complexidade que importa considerar é a temporal. Começando por analisar o melhor caso, que corresponderá a pesquisar toda a sequência sem encontrar qualquer dupleto central como constituinte do padrão e

aplicando sempre o avanço máximo *2(m-1)*. Considerando que para verificar o dupleto central é necessária uma iteração e que para calcular o avanço suplementar é necessária uma operação idêntica, pode considerar-se o melhor caso como *O (n/(m-1))*.

No oposto, o pior caso ocorre quando o algoritmo precisa de testar todos os *m-1* alinhamentos possíveis em cada janela, verificar por cada um os *m-4* caracteres restantes na totalidade (refira-se que o *cd* e o *idcd* são verificados em paralelo para todos os alinhamentos), e os avanços suplementares ficarão na versão mínima reduzidos a *m-1*. Em suma trata-se de uma sequência com repetições do mesmo símbolo e o padrão é uma subsequência dessa sequência. Com estes considerandos, pode inferir-se que a complexidade temporal no pior caso se cifra em *O ((n/(m-1))(4+(m-4)(m-1)))*, simplificando temos genericamente *O(nm)* no pior caso.

No caso médio, por análise experimental, verificamos que se trata de um algoritmo sublinear, com complexidade média inferior a *O(n)*, isto dado o comportamento em curva logarítmica descendente que é evidenciado nos gráficos de desempenho, bem como através da análise de *ncc*. A determinação teórica e formal do caso médio envolve enorme complexidade e constituirá certamente um interessante problema em aberto que ficará como desenvolvimento futuro.

4.1.6. Discussão e conclusões

Considerando neste contexto, desempenho como tempo de execução ou rapidez de execução, o algoritmo GRASPm demonstrou um desempenho comparado claramente superior ao dos seus concorrentes mais acreditados. A sua lógica é inovadora, incorporando uma série de ideias novas e outras conhecidas numa combinação que demonstrou grande eficiência na pesquisa de padrões exactos em sequências de ADN. O seu desempenho, qualquer que seja o tamanho do padrão é, no mínimo, 20% superior face ao melhor concorrente, e quando comparado com algoritmos congéneres baseados em heurísticas a superioridade ainda é mais notória atingindo, em média, factor 2. Em padrões de dimensão maior, com *m>32*, o desempenho do GRASPm é ainda mais convincente, chegando, nos casos extremos, a ser 5 vezes mais rápido que a concorrência. Cumulativamente, apresenta uma curva de desempenho de melhoria

continuada à medida que o tamanho do padrão pesquisado aumenta, algo que não é sempre verificado nos concorrentes e que demonstra a consistência do algoritmo. A avaliação com base no número de comparações de caracteres, através do indicador *ncc/n*, ratifica a constatação da superioridade do GRASPm.

A maior contribuição do GRASPm é a nova heurística de compatibilidade que permite aumentar a selectividade com que se analisam os alinhamentos, permitindo a inibição à priori de alinhamentos incompatíveis que de outro modo apenas serviriam para atrasar o algoritmo, retirando-lhe qualquer hipótese de ser competitivo. Assim, a regra de compatibilidade é decisiva e nuclear no GRASPm, sendo responsável pelo mérito do desempenho do novo algoritmo. É igualmente importante a estratégia de examinação que recorre a janelas de pesquisa de dimensão acrescida e testa como primeiro filtro o dupleto central. Quanto à heurística de avanço suplementar, esta tem apenas um papel acessório. Experiências efectuadas ao desempenho do algoritmo retirando-lhe o aporte desta regra, provaram que há apenas uma diferença de 5%, em benefício nos padrões com $m<50$ e em prejuízo nos padrões com $m>50$. Consequentemente, conduz apenas a uma ligeira melhoria nos padrões curtos a moderados.

Como ponto menos relevante mas igualmente importante, ficou patente que o GRASPm é mais constante nos tempos de execução para as várias instâncias de uma categoria de padrões, apresentando-se como menos sensível às variações de composição do padrão. Assim, o GRASPm é superior nos vários comprimentos de padrão e igualmente nas várias composições desses padrões, demonstrando consistência e robustez.

De referir igualmente que a estrutura de dados do GRASPm é mais onerosa que a da maioria dos algoritmos concorrentes, porém em nada impeditiva dos resultados de liderança apresentados. Em média o GRASPm necessita de uma estrutura de dados de aproximadamente 100KB, um valor perfeitamente comportável em qualquer sistema, incluindo os de computação móvel.

Na secção seguinte descreve-se um novo algoritmo, baseado no GRASPm mas, vocacionado para alfabetos maiores como é o caso do alfabeto de 20 aminoácidos das proteínas, esta variante é igualmente competitiva em pesquisa de padrões em linguagem natural.

4.2. DC: Algoritmo de pesquisa de padrões exactos, altamente eficiente e flexível, para aplicação na análise de proteínas e linguagem natural

O algoritmo GRASPm, descrito e analisado na secção anterior, está vocacionado para a informação genómica, porém a informação biológica inclui também a informação das cadeias péptidicas que formam as proteínas. Como visto no Capítulo 2, as proteínas resultam de uma sucessão de aminoácidos que, por sua vez resultaram de codões que os codificaram. Um codão corresponde a 3 bases consecutivas, e com 3 bases podem formar-se 64 diferentes combinações. Devido à existência da redundância introduzida pelos codões sinónimos apenas 20 diferentes aminoácidos são expressados pelo ADN. Assim, o alfabeto das proteínas tem dimensão $\sigma=20$, correspondendo a $\sum=\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$.

De forma a converter o modelo conceptual do GRASPm num algoritmo mais flexível, que permita pesquisar padrões exactos em qualquer tipo de informação, com especial ênfase em informação derivada de alfabetos maiores como é o caso das proteínas ou da linguagem natural, criou-se esta nova variante que referenciaremos doravante de DC (Deusdado, Carvalho).

O algoritmo DC é baseado em caracteres individualmente considerados, em 1-gramas portanto. Se pensarmos em proteínas é baseado em aminoácidos, se considerarmos a linguagem natural é baseado em caracteres: letras, números, símbolos de pontuação, etc.

A estratégia de pesquisa é similar à utilizada no GRASPm, a diferença decorre do facto de se utilizarem 1-gramas/caracteres como referência. Assim, o caracter central da janela de pesquisa marca o eixo usado para todos os alinhamentos, sendo que o tamanho da janela de pesquisa pode assim ser ligeiramente aumentado para $2m-1$ caracteres. Vejam-se os casos extremos (ver Tabela 4.8) para um padrão p com $m=10$ e os seus alinhamentos extremos numa janela de tamanho $2m-1=19$.

cc

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	2	3	4	5	6	7	8	9	10									
									1	2	3	4	5	6	7	8	9	10

Tabela 4.8 – Casos extremos de alinhamento de um padrão dentro da janela de pesquisa no algoritmo DC.

A estratégia de pesquisa do algoritmo DC difere do GRASPm no facto de que, ao contrário do GRASPm que estabelece uma janela de pesquisa sempre e quando o dupleto central faz parte do padrão, no DC executa-se um ciclo de avanços antes de criar a janela de pesquisa. Esse ciclo de avanços termina quando a regra de avanços extra, baseada na heurística do mau caracter do algoritmo BMH [105], encontrar no texto um caracter igual a *p[m]*, ou seja igual ao último caracter do padrão, o que equivale a dizer quando o avanço proporcionado por dita regra for nulo. A justificação passa por uma nova perspectiva de compreensão da tabela de avanços suplementares, que representa basicamente a distância da última ocorrência de um caracter no padrão para o último caracter do padrão. Assim, pode utilizar-se um ciclo de avanços independente enquanto não for encontrado um caracter igual a *p[m]* simplesmente percorrendo a sequência *x,* analisando inicialmente o caracter central da primeira janela em $cc_{1.1}=x[m]$, depois o caracter em $cc_{1.2}=m+f(x[cc_{1.1}])$, e assim sucessivamente, sendo *f*, a função de avanço suplementar. O definitivo cc_1 é determinado quando a função de avanço devolver um avanço nulo. Tendo por exemplo o padrão *p="AFFKRQYYER"* e a resultante tabela de avanços representada na Tabela 4.9, podemos afirmar que o algoritmo DC, na pesquisa do padrão *p*, só estabelecerá a primeira janela quando através dos sucessivos saltos proporcionados pela tabela de avanços encontrar em *x* o caracter "*R*".

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
9	10	10	1	7	10	10	10	6	10	10	10	10	4	0	10	10	10	10	2

Tabela 4.9 – Tabela de avanços do padrão "AFFKRQYYER" baseada na heurística do mau caracter.

Encontrada a primeira janela de pesquisa, sendo sempre certo que $x[cc_n]=p[m]$, haverão tantos alinhamentos a testar quantas as ocorrência do caracter *p[m]* no padrão. No caso em análise há apenas duas ocorrências de "R" no padrão *p*, mas no caso limite, que

corresponde ao padrão uniforme, poderia haver até *m* alinhamentos. Esses possíveis alinhamentos são classificados, tal como no GRASPm, em compatíveis ou incompatíveis, aplicando a regra da compatibilidade.

No algoritmo DC a regra de compatibilidade atende apenas ao caracter precedente do caracter central, portanto $x[cc_n\text{-}1]$. Assim, para todos os alinhamentos possíveis, que resultam das ocorrências do caracter *p[m]* no padrão, é analisado o caracter antecedente dessas ocorrências para definir que caracteres são compatíveis com o alinhamento. A lógica é a mesma usada no GRASPm, apenas reduzida ao caracter precedente das ocorrências do caracter em *p[m]* no padrão.

As diferenciações introduzidas na estratégia de pesquisa visam adequar o DC a alfabetos de maiores dimensões onde a probabilidade de alinhamentos é mais diminuta, ainda assim, a regra de compatibilidade continua a ser decisiva no desempenho do DC tal como é no GRASPm.

O algoritmo DC assenta em duas fases, a fase de pré-processamento e a fase de processamento ou pesquisa. A fase de pré-processamento é orientada ao padrão, mais concretamente ao estudo da composição do padrão bem como das condições de compatibilidade para cada alinhamento passível de resultar numa descoberta do padrão. Na fase seguinte, que corresponde à pesquisa de réplicas do padrão percorrendo o texto através do avanço iterativo das janelas de pesquisa, o conhecimento adquirido no pré-processamento é fundamental. Sobretudo, no sentido de estabelecer mecanismos simples e eficazes para aumentar a selectividade com que se lida com os *m* alinhamentos que podem existir dentro de uma janela de pesquisa.

Antecedendo e sucedendo a cada estabelecimento da janela de pesquisa ocorre um ciclo de avanços pela sequência até nova ocorrência do caracter em *p[m]*. De referir que o DC incorpora também uma componente de avanço fixa exercida após o término das verificações em cada janela estabelecida, essa componente fixa corresponde a *m* caracteres. Consequentemente, o primeiro avanço na procura da nova janela incorpora a componente fixa igual a *m*, à qual se segue o, já aludido, ciclo de avanços suplementares. Como súmula desta descrição mais informal segue-se a apresentação do fluxograma do DC, patente na Figura 4.14.

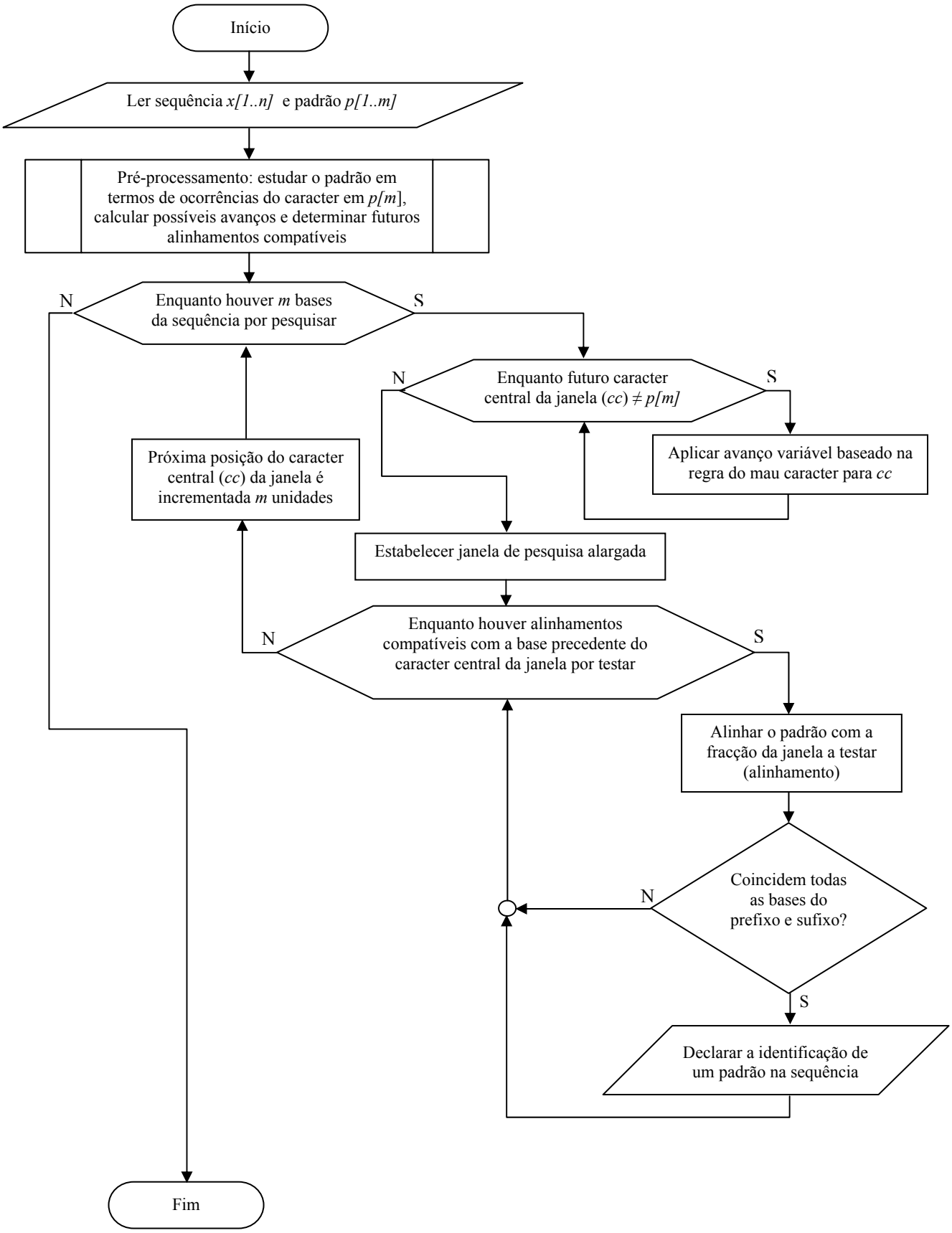


Figura 4.14 – O fluxograma do DC.

Numa abordagem lata, pode-se dizer que o DC é um algoritmo baseado em heurísticas, um híbrido entre o BMH e o GRASPm, orientado a 1-gramas, apresentando resultados que demonstram flexibilidade e eficiência em qualquer tipo de alfabeto. De facto, consegue ser mais eficiente que o BMH em qualquer tipo de alfabeto e mais eficiente que o GRASPm nos alfabetos de maiores dimensões como é o caso da informação biológica das sequências de proteínas.

Seguindo a mesma metodologia de explicação do algoritmo usada para o GRASPm, importa agora analisar em detalhe as diferentes fases do novo algoritmo.

4.2.1. Fase de pré-processamento

No pré-processamento do padrão estão incluídas as seguintes computações:
- a análise das ocorrências dos caracteres *p[m]* no padrão;
- o cálculo da tabela de compatibilidades dos alinhamentos das ocorrências de *p[m]* no padrão;
- o cálculo da tabela de avanços suplementares.

4.2.1.1. Análise das ocorrências dos caracteres *p[m]* no padrão

A tabela de estudo das ocorrências dos caracteres *p[m]* no padrão é muito simples. Trata-se de um vector que armazena o número total de ocorrências dos caracteres *p[m]* na primeira célula, guardando nas restantes a posição em que aparecem. Retomando o padrão *p="AFFKRQYYER"*, onde o caracter em *p[m]* ocorre na 5ª e 10ª posições, apresenta-se o resultado do estudo na Tabela 4.10.

1	2	3	4	…
2	5	10	0	…

Tabela 4.10 – Tabela de estudo das ocorrências de *p[m]* no padrão.

4.2.1.2. Cálculo da tabela de compatibilidades

A tabela de compatibilidades incide apenas sobre os alinhamentos das ocorrências de *p[m]* em *p*, pois daí resultarão os alinhamentos a, eventualmente, testar na futura fase de pesquisa. A Tabela 4.11 mostra esses alinhamentos e permite-nos observar que, atendendo apenas ao caracter precedente do futuro caracter central (*cc-1)*, "K" e "E" são as únicas compatibilidades a registar, qualquer outro caracter é incompatível.

					cc-1	*cc*				
					A	F	F	K	R	Q	Y	Y	E	R
A	F	F	K	R	Q	Y	Y	E	R					

Tabela 4.11 – Alinhamentos do padrão pelas ocorrências do caracter em *p[m]*.

A Tabela 4.12 encerra as compatibilidades observadas, na coluna correspondente ao caracter compatível guardam-se os alinhamentos compatíveis. Caso haja alinhamentos múltiplos para um mesmo *cc-1*, a ordem de registo corresponde à ordem inversa de aparição no padrão. Esta inversão justifica-se pelo facto de que na altura de testar os alinhamentos estes são testados pela ordem seguida por esta tabela. Note-se que quanto mais à esquerda se situar o alinhamento do padrão, menor será a posição onde ocorre na sequência, desta forma garante-se que se descobrem as instâncias do padrão na ordem de aparição correcta.

p[m]="R"

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
0	0	0	10	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Tabela 4.12 – Tabela de compatibilidades.

A tabela de compatibilidades, por via da utilização de 1-gramas e pelo facto de se utilizarem apenas os alinhamentos de *p[m]* fica muito compacta. Mais compacta que no GRASPm apesar de se usarem alfabetos maiores.

4.2.1.3. Tabela de avanços suplementares

A tabela de avanços suplementares já foi abordada e representada na Tabela 4.9. A mesma é utilizada para percorrer celeradamente as regiões da sequência a pesquisar que não apresentam ocorrências de *p[m]*. Esse percurso acontece continuadamente a partir do caracter cc_n+m, já que *m* é o avanço fixo após cada iteração. Os avanços suplementares são fornecidos inicialmente por $f(x[cc_n+m])$, sendo os subsequentes obtidos pelos incrementos >0 provenientes da função *f* para o novo caracter de destino, a função *f (c)* devolve de avanço suplementar para o caracter *c*.

4.2.2. Fase de pesquisa ou processamento

A fase de processamento principia com um ciclo de avanços que se inicia no caracter *x[m]*, assim, *x[m]* será o primeiro candidato a caracter central cc_1, que se referencia como $cc_{1.1}$. Verifica-se se $f(x[cc_{1.1}])>0$. Em caso afirmativo, a $cc_{1.1}$ incrementa-se $f(x[cc_{1.1}])$ para obter $cc_{1.2}$. Repete-se o processo até que $f(x[cc_{1.n}])=0$. Atingida esta condição obtém-se cc_1. A primeira janela de pesquisa é estabelecida com o seu caracter central a corresponder a $cc_1=p[m]$. O passo seguinte é verificar se há alinhamentos compatíveis a testar, para tal recorre-se à tabela de compatibilidade criada na fase de pré-processamento e caso haja alinhamentos a testar iniciam-se as comparações caracter a caracter para o primeiro alinhamento compatível. As verificações interrompem-se se suceder uma falha, ou então resultam numa ocorrência do padrão. As verificações não incorrem em redundância pois os caracteres *cc* e *cc-1* já foram anteriormente testados, assim testam-se apenas os caracteres sobrantes presentes no prefixo e/ou sufixo, sempre da esquerda para a direita.

Para clarificar todos os passos da fase de pesquisa do algoritmo DC, fornece-se, na Figura 4.15, uma proposta de implementação em linguagem C dessa fase.

```
void DC_Search(char *texto, long n, char *padrao, unsigned int m)
{
 int b,cc,na,precedente,ia,iap,j,prefixo;
 char c;

 b=padrao[m-1]; //Último caracter no padrão
 cc=m-1; // Primeiro caracter central cc
 while (((c=texto[cc])!=b) && (cc<=n)) cc+=xshift[c];  // Primeiro ciclo de avanços
 while (cc<=n)
   {
     precedente=texto[cc-1];
     ia=1;
     while ((na=compatibilidade[precedente][ia])>0)
         {
            prefixo=na-2; // Tamanho do prefixo
            iap=cc-prefixo-1; // Posição para alinhamento do padrão
            j=0;
            while ((j<prefixo) && (texto[iap+j]==padrao[j])) j++; // Teste do prefixo
            if (j==prefixo)
              {
               j+=2; //cc e cc-1 já foram testados
               while ((j<m) && (texto[iap+j]==padrao[j])) j++;   // Teste do sufixo
               if (j>=m) printf ("\nOcorrencia na posicao %d.", iap);
              }
            ia ++;  // Segue para o próximo alinhamento compatível
         }
     cc+=m;  // Avanço fixo
     while (((c=texto[cc])!=b) && (cc<=n)) cc+=xshift[c]; // Ciclo de avanços extra
   }
}
```

Figura 4.15 – Proposta de implementação da fase de pesquisa do algoritmo DC.

Tal como no GRASPm, no DC consideram-se casos particulares os padrões com $m \leq 2$ e usam-se as mesmas metodologias básicas para pesquisar esses padrões muito reduzidos. Tais metodologias já estão descritas na subsecção 4.1.3.1.

4.2.3. Análise de complexidade

A análise de complexidade do algoritmo DC pode ser vista nas duas fases do algoritmo. Na fase de pré-processamento a complexidade temporal e espacial é repartida pelas seguintes necessidades computacionais:

- Analisar as ocorrências de *p[m]* que compõem o padrão;

  Complexidade temporal associada: $O(2m)$

  Complexidade espacial associada: $O(m+1)$

- Calcular a tabela de compatibilidades;

  Complexidade temporal associada: $O(m)$

  Complexidade espacial associada: $O(\sigma m)$

- Calcular a tabela de avanços suplementares.

    Complexidade temporal associada: $O(\sigma+m)$

    Complexidade espacial associada: $O(\sigma)$

Na fase de processamento a análise de complexidade que importa considerar é a temporal. Começando por analisar o melhor caso, que corresponderá a pesquisar toda a sequência aplicando apenas o primeiro ciclo de avanços, portanto sem encontrar qualquer ocorrência de *p[m]* e, sempre com avanços máximos de *m*. Assim, podemos considerar o melhor caso como $O(n/m)$.

No outro extremo, o pior caso ocorre quando o algoritmo terá de estabelecer janelas a cada *m* caracteres, usando apenas o avanço fixo e nunca recorrendo a qualquer avanço suplementar. Concomitantemente, por cada janela teria de testar *m* alinhamentos (o máximo) e por cada alinhamento verificar *m-2* caracteres (os caracteres *cc* e *cc-1* são verificado apenas uma vez em paralelo para todos os alinhamentos). Em suma trata-se de uma sequência com repetições do mesmo símbolo e o padrão é uma subsequência dessa sequência. Com estes considerandos, pode inferir-se que a complexidade temporal no pior caso se cifra em $O((n/m)(2+m(m-2)))$. Simplificando, temos genericamente como pior caso, complexidade temporal $O(nm)$.

No caso médio, por análise experimental, verificamos que se trata de um algoritmo sublinear, com complexidade média inferior a $O(n)$, isto dado o comportamento em curva logarítmica descendente que é evidenciado nos gráficos de desempenho. A determinação/prova teórica do caso médio é matéria complexa e constituirá certamente um interessante problema em aberto que ficará como desenvolvimento futuro.

4.2.4. Resultados experimentais e comparações

Para testar o desempenho do algoritmo DC escolheram-se os algoritmos de referência em pesquisa de padrões exactos para alfabetos com $\sigma \geq 20$. Isto porque se pretendeu optimizar o DC para pesquisas em proteínas e linguagem natural. Assim, foi decido incluir nas comparações, pelo seu desempenho, o algoritmo BMH [105], por ser o mais rápido dos clássicos, o FJS [193] de 2005, que foi publicado como o mais rápido para

linguagem natural e os algoritmo SBNDM [107] e WML [110], já antes utilizado na comparação com o GRASPm (ver secção 4.1), por serem considerados como dos mais rápidos algoritmos para alfabetos moderados mas permanecendo competitivos em todos os alfabetos. Do algoritmo WML usou-se a versão 2-gramas considerando-a a versão mínima do WML, pois sendo um algoritmo baseado em *hashing* de *q*-gramas descendente do Shift-Or [113], perderia a sua essência se reduzido a um algoritmo orientado ao caracter como os demais. Quer o BMH, o SBNDM e o WML já foram minimamente descritos nesta dissertação (ver secção 3.1.1), quanto ao FJS convém efectuar uma descrição sumária do seu funcionamento.

O FJS é um algoritmo baseado em heurísticas, que reúne numa solução híbrida, ideias dos algoritmos de KMP [104], BM [97] e Sunday [194] na tentativa de aproveitar os pontos fortes de cada um deles. O resultado foi um algoritmo muito competitivo, dominante na data de apresentação, segundo os autores, para alfabetos com $\sigma \geq 8$.

As implementações usadas são genuínas, codificadas em linguagem C e compiladas, usando máxima optimização –O3, recorrendo ao compilador *gcc* - versão 3.4.2. A implementação do algoritmo de BMH foi adaptada de [192], a implementação dos algoritmos SBNDM e WML foi gentilmente fornecida pelos seus autores e a implementação do algoritmo FJS é providenciada pelos autores em [193].

Como plataforma dos testes utilizou-se um sistema computacional com base num processador Intel Pentium IV 3.4GHz, 8KB L1 + 512KB L2 de cache e 1GB de DDR-RAM, sobre Windows XP Professional SP2 OS. Os tempos foram colectados com recurso a uma função incluída num biblioteca do sistema operativo, referida como *timeGetTime()*, que fornece medições instantâneas do tempo em milissegundos.

As sequências usadas foram uma colectânea de proteomas obtidos da base de dados pública Integr8[3], que inclui os proteomas das espécies *Homo Sapiens, C. Elegans, A. Thaliana e Mouse Musculus*, cujo conjunto perfaz uma sequência com cerca de 50MB. Já para os testes em linguagem natural usou-se uma compilação de 37 livros electrónicos, contendo textos de diversos autores da literatura europeia obtidos a partir

[3] www.ebi.ac.uk/integr8/

da base de dados do Projecto Gutenberg[4], num total de cerca de 50MB de informação. Quanto aos padrões, para o caso das pesquisas em sequências de aminoácidos foram criados aleatoriamente 100 padrões por cada uma das categorias de comprimentos de padrão testados, a saber, *m=2, 4, 8, 16, 32, 64, 128*. Para o caso da linguagem natural foram usadas palavras ou frases, ou excertos das mesmas retiradas ao acaso do texto. Os algoritmos em contenda efectuaram as pesquisas para todos os padrões considerados, e para cada categoria foi recolhido o tempo médio de execução (T.E.) da pesquisa em milissegundos. Os resultados estão patentes nas Tabela 4.13 e Tabela 4.14.

m	DC		BMH		FJS		SBNDM		WML2	
	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank
2	**173**	**1**	271	4	174	2	177	3	672	5
4	**111**	**1**	149	4	120	2	132	3	235	5
8	**67**	**1**	86	3	77	2	96	4	111	5
16	**44**	**1**	56	3	52	2	61	5	60	4
32	32	2	42	5	40	4	**31**	**1**	37	3
64	**26**	**1**	33	4	32	3	$>\omega$	?	27	2
128	**22**	**1**	31	4	30	3	$>\omega$	?	**22**	**1**

Tabela 4.13 – Resultados de desempenho comparado do algoritmo DC com os concorrentes para diversas categorias de padrões pesquisando sequências de aminoácidos ($\sigma=20$).

m	DC		BMH		FJS		SBNDM		WML2	
	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank	T.E. (ms)	Rank
2	160	3	237	4	149	2	**144**	**1**	586	5
4	**94**	**1**	127	4	95	2	101	3	197	5
8	**58**	**1**	75	3	65	2	84	4	96	5
16	**44**	**1**	55	3	52	2	64	4	52	2
32	**31**	**1**	39	4	39	4	37	3	32	2
64	25	2	32	3	29	2	$>\omega$	?	**24**	**1**
128	**17**	**1**	21	4	19	3	$>\omega$	?	**17**	**1**

Tabela 4.14 - Resultados de desempenho comparado do algoritmo DC com os concorrentes para diversas categorias de padrões pesquisando linguagem natural ($\sigma=256$).

[4] www.gutenberg.org

4.2.5. Discussão e conclusões

Pela apreciação das tabelas de resultados fica claro o domínio do algoritmo DC na pesquisa de proteínas e também nas pesquisas em linguagem natural, a sua vantagem é, em média, de cerca de 15% para o melhor concorrente, o WML2, que, se sublinha tratar-se de um algoritmo baseado em 2-gramas. O WML2 destaca-se nos padrões de maior dimensão mas, com uma flexibilidade reduzida pois deprecia-se na pesquisa padrões curtos. O SBNDM revela-se mais competitivo nos alfabetos de menor dimensão, em alfabetos maiores consegue equiparar-se ao BMH e perdendo claramente para o WML2. O FJS demonstra alguma consistência nos padrões mais curtos apresentando-se, no entanto pouco flexível quando os padrões aumentam de dimensão.

Tal como aconteceu no GRASPm, foi possível conceber e implementar um algoritmo que superasse o estado da arte na matéria de pesquisa de padrões exactos, neste caso em alfabetos maiores. O algoritmo DC posiciona-se como algoritmo dominante para alfabetos em análise com $\sigma = 20$ (amino-ácidos) e $\sigma = 256$ (ASCII). O novo algoritmo é um modelo de eficiência e flexibilidade, adaptando-se com sucesso a qualquer alfabeto, dimensão ou composição de padrão. O algoritmo BMH é um algoritmo altamente eficiente em pesquisas de padrões exactos em linguagem natural. Ocasionalmente surgiram algoritmos (tal como o FJS) que melhoraram marginalmente o desempenho patenteado pelo algoritmo BMH, porém nenhum algoritmo baseado em heurísticas conseguiu destacar-se significativamente da referência do BMH como agora acontece, com o novo algoritmo aqui apresentado, o algoritmo DC. O algoritmo SBNDM é de outra categoria, baseado numa abordagem sustentada pelo *bit parallelism*, potenciada pelas rápidas operações de *bitwise*. Contudo, apresenta limitações no tamanho do padrão a pesquisar, estando limitado ao tamanho do número de bits da arquitectura do computador em que é executado, normalmente 32 ou 64 bits, que corresponderão a outros tantos caracteres. É possível contornar esta limitação segmentando o padrão e pesquisar cada segmento por separado, sendo que esta metodologia introduz *overhead* suplementar ao algoritmo pelo que a versão original preferiu ignorar os padrões maiores que o comprimento da palavra suportado pela arquitectura. O SBNDM é mais competitivo em alfabetos curtos, para $\sigma \geq 20$ apenas se assemelha ao DC em termos de desempenho quando o tamanho do padrão se situa em torno dos 32 caracteres, o que corresponde à maximização do aproveitamento do potencial do *bit parallelism*.

4.3. SimSearch: Uma nova variante da programação dinâmica para a descoberta óptima e sub-óptima de similaridades

Vistos os algoritmos desenvolvidos para pesquisa de padrões exactos, que se revelam importantes em bioinformática, dá-se agora lugar ao algoritmo de pesquisa de padrões aproximados desenvolvido, o SimSearch. A pesquisa de padrões aproximados é uma temática central e das mais importantes em bioinformática, que se reveste de fundamental importância no âmbito desta dissertação pois, o SimSearch será o responsável pela captação das redundâncias a integrar no dicionário de padrões, que constitui um dos recursos principais da compressão, cuja metodologia será desenvolvida no capítulo seguinte, o Capítulo 5. O SimSearch não é, no entanto, um algoritmo de verificação de correspondências numa sequência tendo um padrão de referência (*pattern-matching*), trata-se sim de um algoritmo de descoberta de similaridade na sequência, i.e., descobrindo ocorrências de pelo menos, dois padrões iguais ou semelhantes (*pattern discovery*). O SimSearch é conceptualmente um algoritmo óptimo ou, em alternativa sub-óptimo em função da parameterização, de pesquisa de similaridades inter ou intra-genómicas cuja metodologia assenta nos princípios da programação dinâmica.

A programação dinâmica é uma técnica amplamente usada em investigação operacional que permite optimizar a solução de um problema complexo, usando para tal uma análise recursiva das soluções dos sub-problemas para encontrar a melhor solução global. Em bioinformática é recorrentemente usada na área dos alinhamentos de sequências, a nível local e global. Os fundamentos da programação dinâmica e a sua utilidade em problemas bioinformáticos já foram expostos nesta dissertação na secção 3.2.2.1.

O algoritmo desenvolvido foi concebido com o intuito da identificação integral de padrões exactos e aproximados numa sequência, cuja redundância seja útil na obtenção de compressão por via da codificação compacta do dicionário de padrões. O SimSearch está preparado para fornecer uma solução de pesquisa exaustiva, da natureza das soluções óptimas providenciadas pela programação dinâmica, consequentemente trata-se de um algoritmo necessariamente mais lento que as soluções puramente heurísticas e com recurso a filtragem, sendo no entanto, assumida a necessidade de maior

processamento para a obtenção de um maior número de padrões. No entanto, poderá usar uma heurística de filtragem aquando da análise primária de verificação de existência de padrões, acelerando o processamento sem comprometer a pesquisa integral, isto porque há que referir que padrões demasiado curtos não têm interesse para o dicionário pois o seu custo de codificação implica que o ganho compressivo teórico se oblitere na prática. Assim, optou-se por uma solução de compromisso, ponderando sensibilidade e rapidez de execução.

O método de identificação de regiões de similaridade usado pelo SimSearch assenta na utilização de séries de distâncias entre bases do mesmo tipo. Basicamente, guarda-se a posição da primeira ocorrência da base *b* e os termos seguintes da série correspondem ao número de bases diversas que se interpõem até à próxima ocorrência de *b*. A ideia de séries de distâncias pode ser aplicada a dupletos e codões, o que latamente corresponderia a simular a utilização de *n*-gramas de tamanho 2 e 3, respectivamente. As séries de distâncias já foram abordadas em [195] para análise estatística de correlações em tripletos nos cromossomas bacterianos.

Para a análise de similaridades foi desenvolvida uma nova técnica, com recurso a princípios da programação dinâmica, que permite preencher a matriz de similaridade rapidamente devido a uma relação de recorrência simples e orientada para a descoberta de padrões que não incorpora esquema de pontuação na fase do preenchimento, e por uma análise subsequente igualmente rápida onde o esquema de pontuação é implícito e como tal não tem interferência negativa na fase de *traceback* ou de identificação dos padrões recorrentes. Esta nova metodologia permite identificar com rapidez e precisão padrões exactos e aproximados (com qualquer tipo de distanciamento) presentes na sequência analisada, quer na ordem normal ou na ordem reversa. Permite igualmente identificar outros tipos de repetições, como sejam as repetições adjacentes, as repetições de elementos transponíveis, e até mesmo as repetições comportamentais. As repetições comportamentais correspondem a repetições independentes das bases que as constituem, por exemplo, um segmento com a seguinte composição: quatro bases iguais, uma outra base intercalada, depois mais três bases como as iniciais, seguindo-se outra base diferente intercalada e termina com mais duas repetições da base inicial.

4.3.1. O preenchimento das sub-matrizes de análise de similaridades

Antes do preenchimento da matriz de similaridades foi desencadeado, aquando da leitura da sequência a analisar, um processo de construção de listas ligadas para guardar as distâncias de ocorrência de cada base. No concreto foram criadas quatro listas, uma por cada tipo de base existente, que guarda nos seus nós informação sobre a distância da próxima ocorrência em número de bases, com excepção do primeiro nó que indica a que distância do início da sequência se encontra a primeira ocorrência dessa base. Estas quatro listas ligadas são o recurso usado para o ulterior preenchimento da matriz de similaridades.

A abordagem seguida para a descoberta de padrões de forma optimizada usa tabelas parciais para registar as similaridades presentes em segmentos da sequência (de dimensão *m*), face ao resto da sequência a analisar/comprimir. O algoritmo de preenchimento de cada matriz é simples e pode descrever-se pelos seguintes passos:

1. A tabela, uma matriz quadrada de dimensão *m*m*, é inicializada com todas a células iguais a zero.
2. Para cada base analisada, que está representada por uma coluna da tabela, são assinaladas com 1s, as células dessa coluna, nas linhas que correspondam às distâncias a que se encontram as ocorrências seguintes dessa mesma base. Em rigor a pesquisa exaustiva apenas necessita de ser feita a primeira vez que se pesquisa cada base diferente (a,c,g,t). Nas seguintes ocorrências, a lista da ocorrências a jusante já é conhecida, o que pode poupar tempo de processamento.

Em relação aos algoritmos clássicos da programação dinâmica, a simplicidade do algoritmo propicia uma elevada rapidez de execução, o que o torna utilizável em análise de sequências longas.

Considerando a sequência *x*=”*tatccgcattatgcgata*” de tamanho *n*=18, vejamos como proceder para calcular o número de 1s a preencher na matriz de análise, bem como a complexidade temporal associada. O número total de 1s a preencher na matriz pode ser calculado pelo seguinte teorema.

Teorema 4.1

Sendo *n?=1+2+3+4+5+...+n*, com $n \in \mathrm{N}$, descrito como somatorial de *n*, tomando como válida para o seu cálculo a fórmula $n? = \frac{n(n+1)}{2}$, e considerando os somatórios das ocorrências dos símbolos/bases na sequência a analisar como *n1* (para as ocorrências de "A"), *n2* (para as ocorrências de *"C"*), *n3* (para as ocorrências de *"G"*), e *n4* (para as ocorrências de *"T"*), a fórmula de cálculo dos 1s a preencher seria:

Número de 1s=(n1-1)?+ (n2-1)?+ (n3-1)?+ (n4-1)?

Para o exemplo do segmento *x*="*tatccgcattatgcgata*", com *n=n1+n2+n3+n4=18* onde *n1=5*, *n2=4*, *n3=3* e *n4=6*, o número de células assinaladas seria:

Número de 1s=(4)?+ (3)?+ (2)?+ (5)?=10+6+3+15=34

Analisando a complexidade temporal associada a esta tarefa computacional, no pior caso, que ocorre quando todas as bases da sequência a analisar são iguais, temos para uma sequência de tamanho *n*, um complexidade temporal de *O((n-1)?)*. No caso médio, o mais realista, teremos *O(4((n/4)-1)?)*. Se considerar-mos analisar uma sequência de 1000 bases igualmente repartidas, e uma matriz de análise com um milhão de células (1000*1000) para preencher, teremos, em média, *4(249?)=124500* células a preencher com 1s. Considerando ainda, que para preparar a matriz para a análise de uma nova sequência teremos de a inicializar (recolocar todas as células a 0), a forma mais expedita de o fazer é reassinalar as células que antes colocamos a 1, agora a 0. Desta forma a complexidade temporal para um processo contínuo de análise, como acontece na realidade, por análise parcial das sub-sequências que compõem a sequência integral, deverá ser em média *O(8((n/4)-1)?)*

A matriz de análise irá conter informação útil relativa a padrões (denotada por sequências/concentrações de 1s) no seu triângulo superior. O triângulo inferior pode ser ignorado aquando da análise da presença de padrões. Vejamos o exemplo seguinte (Tabela 4.15), onde é analisado o segmento *x*="*tatccgcattatgcgata*" de tamanho *n*=18.

	T	A	T	C	C	G	C	A	T	T	A	T	G	C	G	A	T	A
1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
3	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
6	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
9	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 4.15 – Matriz de análise de similaridades.

Os padrões existentes revelam-se pelas sucessões de 1s, sem interrupções para os padrões exactos, e com interrupções para os padrões com mutações. A localização das repetições é obtida mediante a observação do número da linha onde se identifica o padrão, esse número indica a distância entre um padrão e a sua repetição. Mais adiante serão descritas em pormenor as metodologias para a análise e descoberta de padrões a partir da tabela de similaridades preenchida. Como exemplo, veja-se o padrão assinalado na Tabela 4.15, acima, onde o padrão "*tatccg*" se repete mais adiante (9 bases depois) e contém uma mutação na sua 4ª base.

Para o caso de sequências longas, cuja dimensão e respectiva necessidade de memória não permita a análise numa única tabela, o método subdivide a tabela em sub-tabelas de dimensão computável. Os padrões descobertos são mais tarde analisados para apurar as suas verdadeiras extensões através de eventuais contiguidades, emergindo os padrões máximos e globais. Reiterando a dispensa de processamento do triângulo inferior da

matriz de análise de similaridades, no caso do exemplo anterior é possível ter a análise fraccionada em sub-matrizes 3*3, tal como apresentada na Tabela 4.16.

	T	A	T	C	C	G	C	A	T	T	A	T	G	C	G	A	T	A
1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	
2	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1		
3	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0			
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
5	0	0	0	0	0	0	0	0	0	0	1	1	0					
6	0	1	1	0	0	0	0	0	0	0	0	0						
7	0	0	1	0	0	1	1	0	0	1	1							
8	1	0	0	0	0	0	0	1	1	0								
9	1	1	1	0	1	1	0	0	0									
10	0	0	0	1	0	0	0	1										
11	1	0	0	0	0	0	0											
12	0	0	0	0	0	0												
13	0	0	0	0	0													
14	0	1	1	0														
15	0	0	0															
16	1	1																
17	0																	
18																		

Tabela 4.16 – Subdivisão da matriz de análise de similaridades.

O número de matrizes de similaridade a preencher e analisar pelo SimSearch pode ser previsto em face da dimensão da sequência a analisar (n) e da dimensão das sub-tabelas (t) a usar. Recorrendo à fórmula do somatorial, presente no Teorema 4.1, o número de sub-tabelas corresponderá a $(n/t)?$. Comparativamente com os algoritmos de programação dinâmica, cuja complexidade temporal corresponde a $O(mn)$, sendo m e n, os comprimentos das sequências a comparar, o algoritmo desenvolvido apenas percorre metade da matriz, e de uma forma geral a complexidade temporal poderia ser $O((mn)/2)$. Esta redução corresponde à isenção de pesquisa de padrões no triangulo inferior da matriz global de similaridades. Acrescenta-se igualmente que as tabelas do SimSearch são binárias, o que evita operações aritméticas, relacionadas com esquemas de pontuação, no seu preenchimento.

Terminada a primeira fase do SimSearch para a obtenção da matriz de similaridades, o passo seguinte é definir um esquema de pontuação (*scoring scheme*) para valorizar cada um dos padrões (in)exactos descobertos dentro da sequência analisada, com a finalidade de os seleccionar como consideráveis, ou no caso especifico da utilização do SimSearch para a compressão, verificar se o padrão descoberto tem potencial para constituir uma entrada no dicionário ou, pelo contrário deve ser descartado.

4.3.2. Esquema de pontuação para análise de padrões detectados

A análise da matriz de similaridades tem por objectivo identificar e extrair padrões com utilidade para a compressão, esta utilidade deverá ser mensurada usando as métricas de distância entre palavras (*strings*), também denominada por distância de edição (esta temática foi abordada na subsecção 3.2.1.1). Uma *string* é mais ou menos semelhante a outra na medida em que a distância de edição entre ambas (recorde-se que a distância de edição é simétrica) é mais ou menos reduzida. A distância de edição é baseada em operações elementares que permitem transformar uma *string* noutra por inserções, delecções ou substituições. Os algoritmos mais recentes de análise de sequências usam igualmente o intervalo não correspondente (*gap*) como ferramenta de aproximar *strings*. O *gap* de tamanho *n* pode ser visto como uma inserção repetida *n* vezes.

Por tudo isto torna-se necessário estabelecer um esquema de pontuação que permita avaliar, dentro de certos limites, o que se entende como padrão. Deste modo premeiam-se e penalizam-se as operações a realizar no seguintes termos.

Correspondência (*Match*)=1 ponto;
Substituição=-2 pontos;
Inserção=-2 pontos;
Delecção=-1 ponto;
Abertura de *gap*=-5 pontos;

Note-se que, o esquema de pontuação apresentado é orientativo, ficando a cargo do utilizador ajustar as pontuações para a especificidade da análise que pretende efectuar.

De referir ainda que a pontuação é iniciada a 1 com um *match* descoberto, portanto uma célula da matriz de similaridades assinalada a 1, é abandonada se a pontuação sucessivamente recalculada para as bases subsequentes se degradar até -10, para ser retomada e reiniciada na próxima correspondência (*match*). Com estas regras é possível acelerar sobremaneira a análise da matriz, pois são conhecidas as posições dos 1s, e sempre que a análise de uma correspondência falhar pode-se saltar imediatamente para a seguinte. Esta metodologia corresponde a sensibilidade 100% pois, todas as correspondências de apenas uma base são analisadas, o que implica uma análise exaustiva e morosa. Como opção da aplicação desenvolvida, incorporou-se a possibilidade de estabelecer um parâmetro de sensibilidade, através do tamanho da *seed* (semente). Este parâmetro, identificado por *W*, deve situar-se no intervalo de 2 a 32, sendo que quanto maior mais rápido mas igualmente menos sensível, pois uma *seed* de tamanho 32 implica que o algoritmo só se detenha para analisar casos onde haja uma correspondência exacta de 32 bases contíguas. O valor 32 decorre do comprimento da palavra para a corrente arquitectura de 32 bits, permitindo assim operações de *bitwise* mais eficientes. Saliente-se que o termo *seed* no contexto do SimSearch tem uma interpretação diferente da usada vulgarmente na literatura bioinformática, isto porque as *seeds* do SimSearch correspondem a padrões já detectados não necessitando de qualquer desenho especial como acontece por exemplo no *PatternHunter* [118, 119], o estado da arte nesta matéria.

Para o trabalho presente, estabeleceu-se, por defeito, um mínimo de 20 pontos para considerar um padrão útil. Por exemplo, 25 correspondências entremeadas por: uma substituição (-2), uma inserção (-2) e uma delecção (-1). A partir do momento em que se obtêm 20 pontos, o padrão pode ser alargado sempre e quando a pontuação do alargamento o justifique, aqui interpretado como sendo sempre de saldo positivo, i.e., no alargamento por cada dois pontos conquistados só se pode ceder um ponto. Quando esta regra não se verifique então o padrão terá terminado e caso seja retomado mais adiante será considerado como um novo padrão. É notória a penalização acentuada dos *gaps*, o que se justifica pelo custo elevado que acarretam em termos de codificação da sequência comprimida em fase ulterior. Para codificar um *gap*, temos de referir a sua localização, extensão e qual a subsequência que o preenche.

O SimSearch é como já ficou patente um algoritmo exaustivo, que pesquisa todas as possibilidades de padrão sempre que encontra a indicação de similaridade mínima, ou seja uma base. Para tornar a aplicação mais flexível dotou-se o SimSearch da opção de parameterizar o tamanho da *seed*, tornando-a a aplicação mais rápida mas menos sensível à medida que se aumenta o tamanho da *seed*.

4.3.3. Análise da matriz de similaridades para a descoberta de padrões

Na matriz de similaridades ficam completamente assinalados todos os padrões existentes, desde os exactos normais, os exactos reversos, os exactos reversos complementares ou palindromas, os aproximados com substituições, delecções, inserções ou *gaps*. Nas subsecções seguintes analisam-se todos esses casos em detalhe.

A detecção dos padrões é efectuada com recurso às operações de *bitwise*, que lidam directamente com os bits em paralelo, para uma dimensão de palavra $\leq\omega$, a permitida pela arquitectura do computador, usualmente 32 ou 64 bits. Visto que as tabelas de similaridade do SimSearch contêm apenas valores binários, é possível, sem adendas, analisar essas sequências de bits de forma rápida recorrendo a operações de *bitwise*. Com o conhecimento adquirido por via do trabalho de investigação e nos testes em dados binários efectuados na avaliação comparativa do algoritmo DC (secção 4.2), optou-se por usar uma adaptação do algoritmo SBNDM (descrito na secção 3.1.1) por ser o que revelou mais eficiente nas pesquisas efectuadas em informação binária. Sendo que o tamanho da *seed* (W) é parameterizável, pode usar-se um valor $W\leq\omega$ para obrigar o algoritmo a apenas identificar W bits consecutivos de valor 1, acelerando assim o desempenho em detrimento da sensibilidade. Funções de deslocamento de bits (*shift*) e a operação lógica AND são suficientes para efectuar as verificações.

Sempre que se encontra uma *seed* é necessário verificar a verdadeira extensão do padrão, analisando as imediações da *seed* para detectar continuidades exactas ou aproximadas. Detectada ou não uma similaridade, a janela deslizante do SBNDM continua a sua progressão pelas linhas da tabela de similaridades, sendo que apenas o triângulo superior da tabela necessita verificação. Para acelerar mais o processo, e evitar detecção de padrões sobrepostos, sempre que se encontra um padrão, uma boa porção

da tabela de similaridades fica dispensada de verificações ulteriores como se descreve a seguir. Para finalizar, é de referir que o tamanho da *seed* (*W*) permite igualmente reduzir o espaço de pesquisa de padrões por omissão das (*W-1*) primeiras linhas e das (*W-1*) últimas linhas da tabela de similaridades, pois nunca poderão conter *seeds* de tamanho *W*, excepto no caso das primeiras e quando se trata de padrões sobrepostos, que são dispensáveis.

4.3.3.1. Padrões exactos

Como referido, os padrões ficam assinalados com sucessões de 1s na matriz de similaridade do SimSearch. Os padrões mais evidentes e fáceis de detectar são os padrões exactos, já que se revelam como uma sequência ininterrupta de 1's na horizontal, ou seja nas linhas da matriz.

Uma das perturbações com que este tipo de algoritmos tem de lidar são os padrões sobrepostos. Os padrões sobrepostos são comuns em sequências repetitivas de granularidade fina. Por exemplo o padrão "*aaa*" repete-se 3 vezes em "*aaaaa*", ou o padrão "*ctc*" repete-se 3 vezes em "*ctctctc*". No caso do BLAST [100] este problema é responsável por uma degradação significativa da eficiência pois usa *seeds* exactas, já no caso do *PatternHunter* [118, 119], o facto de usar *seeds* espaçadas previne sobremaneira a sobredetecção dos padrões. Na secção 3.2 desta dissertação caracterizaram-se os algoritmos mais relevantes na temática da pesquisa de similaridade em sequências biológicas, nomeadamente no que concerne a metodologias óptimas e metodologias baseadas em heurísticas, sub-óptimas portanto. Do ponto de vista da genómica funcional, os padrões sobrepostos são praticamente irrelevantes, sendo responsáveis pelo consumo redundante de recursos de processamento. Os padrões sobrepostos ficam obviamente registados na matriz de similaridades do SimSearch, porém a sua destrinça ocorre em tempo de análise da dita matriz.

Na Tabela 4.17, encontra-se representada a matriz de similaridades calculada pelo SimSearch para a sequência "*atcatcatc*", nela figurando apenas os 1s. Analisando a matriz podem inferir-se dois padrões:

- o primeiro corresponde a "*atcatc*", que se repete duas vezes, na posição 1 e posição 4 respectivamente, porém há alguma sobreposição nestas duas ocorrências;
- o segundo corresponde ao padrão "*atc*", que se repete na posição 1 e posição 7 respectivamente, neste caso não há qualquer sobreposição mas ambas ocorrências já estão reflectidas no primeiro padrão detectado.

	A	T	C	A	T	C	A	T	C
1									
2									
3	1	1	1	1	1	1			
4									
5									
6	1	1	1						
7									
8									

Tabela 4.17 – Matriz de similaridades do SimSearch para a sequência "*atcatcatc*".

Para complementar a exemplificação veja-se o caso da Tabela 4.18, que mostra a matriz de similaridades para a sequência "*aaaataaaa*".

	A	A	A	A	T	A	A	A	A
1	1	1	1			1	1	1	
2	1	1		1		1	1		
3	1		1	1		1			
4		1	1	1					
5	1	1	1	1					
6	1	1	1						
7	1	1							
8	1								

Tabela 4.18 - Matriz de similaridades do SimSearch para a sequência "*aaaataaaa*".

Este caso apresenta grande sobreposição de padrões, o mais relevante será o padrão "*aaaa*" repetido duas vezes (assinalado na tabela), mas o que é facto é que o padrão "*aa*" se repete 6 vezes, ou o padrão "*aaa*" se repete 4 vezes. Isto para comentar apenas os padrões exactos.

O SimSearch resolve parcialmente o problema dos padrões sobrepostos. Quando o SimSearch encontra um padrão exacto, a primeira operação que efectua é a despistagem para averiguar se esse padrão é apenas uma fracção de outro de maiores dimensões. Assim, um padrão detectado anula todos os que se encontram no sub-triângulo superior mínimo em que esse padrão se inclui. Para o caso do padrão assinalado na Tabela 4.18, este irá anular todos os que se encontram abrangidos pelo triângulo mínimo que o inclui, também assinalado. O mesmo se aplica ao exemplo da Tabela 4.17, pelo que o único padrão repostado seria nesse caso o "*atcatc*" verificado na posição 1 e 4, que como se viu inclui alguma sobreposição, o que justifica a afirmação de que o SimSearch resolve parcialmente as repetições sobrepostas. Uma solução mais detalhada não se justifica a este nível, podendo resolver-se casos específicos de sobreposições aquando da análise dos padrões detectados para construção da listagem definitiva.

4.3.3.2. Padrões reversos

Os padrões reversos não são muito representativos nas sequências genómicas, porém dado o curto alfabeto e a grande dimensão das sequências, a probabilidade de ocorrerem não deve ser desprezada, sobretudo quando temos em mente aplicar o SimSearch a uma ferramenta de compressão, que inclui a compressão por dicionário, e onde os padrões são a fonte principal de compressão. Um padrão reverso ou inverso é basicamente um padrão simétrico. Na sequência "*attgcgtta*", acontece um padrão reverso para "*attg*". Vejamos o que resulta de aplicar a matriz de similaridade para esta sequência.

Podemos detectar um padrão simétrico analisando as diagonais de inclinação 67,5º, caso sejam detectadas sucessões de 1s nessas diagonais então corresponderão a padrões reversos ou simétricos como o que está assinalado na Tabela 4.19.

	A	T	T	G	C	G	T	T	A
1		1					1		
2				1					
3									
4			1						
5		1	1						
6		1							
7									
8	1								

Tabela 4.19 - Matriz de similaridades do SimSearch para a sequência "*attgcgtta*".

4.3.3.3. Padrões reversos complementares ou palindromas

Os padrões reversos complementares também podem ser detectados pelo SimSearch, para tal torna-se necessário comparar a sequência normal com a complementar. Uma forma de o fazer é concatenar à sequência normal a complementar, depois podem detectar-se quaisquer padrões, os padrões reversos complementares são detectados na sequência complementar da mesma forma que se faz para os padrões reversos. Em suma, este processo equivale a comparar duas sequências independentes.

4.3.3.4. Padrões aproximados com substituições

Nas sequências genómicas os padrões aproximados são naturalmente mais abundantes, a explicação assenta nos mecanismos biológicos de manutenção do ADN compensados pelos mecanismos evolutivos [196]. O estudo das repetições aproximadas reveste-se de interesse crucial na genómica e são comuns aplicações dedicadas para a pesquisa deste tipo de similaridades [197, 198]. No SimSearch este tipo de padrões também se consegue identificar nas matrizes de similaridades. Nesse contexto, os padrões aproximados com substituições são sucessões de 1s com falhas controladas. Vejamos o exemplo da sequência "*agactaaac*", onde os padrões "*agac*" e "*aaac*" diferem apenas numa base. Na Tabela 4.20 está patente a respectiva matriz de similaridades do SimSearch, assinalando-se o padrão detectado com uma substituição/falha na 2ª base.

	A	G	A	C	T	A	A	A	C
1						1	1		
2	1					1			
3			1						
4			1						
5	1	0	1	1					
6	1								
7	1								
8									

Tabela 4.20 - Matriz de similaridades do SimSearch para a sequência "*agactaaac*".

4.3.3.5. Padrões aproximados com inserções/delecções

Os padrões aproximados com delecções são o inverso dos padrões aproximados com inserções, em bioinformática é comum designar estes casos como *indels*. De facto, para igualar estes padrões é necessário efectuar delecções e inserções nos padrões respectivos.

Analisando o exemplo da sequência "*aaggaatgga*", o padrão "*aagga*" ocorre com uma inserção, ou doutro ponto de vista, o padrão "*aatgga*" ocorre com uma delecção. Analisando a matriz de similaridade fornecida na Tabela 4.21, pode verificar-se a descontinuidade de uma linha que afecta a sucessão de 1s.

	A	A	G	G	A	A	T	G	G	A
1	1		1		1			1		
2										
3		1								
4	1	1		1						
5	1		1	1	1					
6			1							
7										
8		1								
9	1									

Tabela 4.21 - Matriz de similaridades do SimSearch para a sequência "*aaggaatgga*".

4.3.3.6. Padrões aproximados com *gaps*

Os padrões aproximados com *gaps* são um caso particular dos padrões aproximados com inserções. Assim, podemos considerar o exemplo anterior reflectido na Tabela 4.21 como um padrão com *gap* 1, que provocou uma descontinuidade de uma linha. Se o *gap* for de dimensão 3 então a descontinuidade corresponde a três linhas. No processo de verificação de *gaps*, o SimSearch pesquisa nas laterais do padrão detectado se, dentro do limite de discrepâncias introduzido pelo utilizador (*threshold*), há *gaps* que devam ser considerados.

4.3.4. Resultados experimentais e discussão

O SimSearch é um algoritmo exaustivo de descoberta de padrões exactos e aproximados baseado na análise matricial de séries de distâncias entre bases iguais. Utiliza os princípios da programação dinâmica uma vez que subdivide o problema, de dimensão intratável pelos meios computacionais presentes, e resolve por recurso a sub-tabelas de análise de vários segmentos, os sub-problemas que constituem o problema global. É igualmente capaz de combinar e articular todos os resultados parciais para obter a solução global. Um estudo de medição de desempenho de algoritmos de pesquisa de homologias e alinhamentos em sequências biológicas está publicado em [199], inclusive analisando versões dos algoritmos para utilização em computação paralela.

Dado tratar-se de um algoritmo exaustivo, o SimSearch tem que ser integrado na categoria dos algoritmos que fornecem uma solução óptima do problema, onde constam os algoritmos baseados em programação dinâmica para análise local de sequências genómica, como seja o algoritmo de Smith-Waterman [116] (consultar a secção 3.2.2.1). Numa segunda análise, faz sentido comparar as suas vantagens e desvantagens com os algoritmos baseados em heurísticas, que fornecem soluções aproximadas num tempo de execução mais curto e consumindo menos recursos computacionais. Como visto na secção 3.2.4.2, neste tipo de algoritmos, o mais recorrentemente usado por bioinformáticos em todo o mundo é o BLAST [100, 136], enquanto que o estado da arte é representado pelo PatternHunter [118, 119].

Para além do SimSearch, a comparação de desempenho efectuou-se envolvendo nos testes os seguintes algoritmos: (*i*) o de Smith-Waterman, com base em programação dinâmica e orientado para alinhamentos locais, incluído na implementação SSearch [200], da qual se usou a versão 35; (*ii*) o algoritmo BLAST, na variante BLASTn e na versão 2.2.18; e por último (*iii*) o PatternHunter, versão 2 – com recurso a *multiseeds*. O SSearch está contido pacote FASTA e pode ser obtido no respectivo sítio Internet fasta.bioch.virginia.edu. O BLAST pode ser obtido no sítio Internet do NCBI-National Center for Biotechnology Information em www.ncbi.nlm.nih.gov. O PatterHunter pode ser obtido, na versão académica, no sítio da empresa que o comercializa, designadamente em www.bioinformaticssolutions.com.

As métricas que se utilizam para comparar este tipo de algoritmos são basicamente a sensibilidade, o tempo de execução e a memória exigida. Quanto a memória exigida, o SimSearch opera por defeito com tabelas quadradas de dimensão 10.000 células. As matrizes podem ser de tipo *boolean*, ou bit, pelo que o seu tamanho em memória é pouco desgastante para o desempenho dos computadores actuais. O problema é que a maioria das linguagens não permitem tipos de dados orientados ao bit, portanto o mínimo é um byte. Assim, a tabela que corresponde à matriz de similaridades em análise, terá cerca de 100 MB, o que é um valor razoável para que a aplicação possa ser executada na maioria dos sistemas actuais.

Quanto a sensibilidade, o SimSearch é por natureza um algoritmo óptimo, logo preparado para obter 100% da similaridade presente. Contudo, sendo o SimSearch parameterizável, está preparado para actuar com *seeds* maiores, de dimensão compreendida entre 2 e 32, tornando-se desse modo menos sensível mas em compensação mais rápido na análise das sequências. Nos testes efectuados utilizaram-se *seeds* de tamanho 3 para pesquisa óptima e *seeds* de tamanho 11 para pesquisas mais rápidas abdicando de sensibilidade 100%. Nos algoritmos concorrentes, baseados em heurísticas, usaram-se os parâmetros por defeito, em ambos os casos usando *seeds* de tamanho 11, sendo no caso do PatternHunter *seeds* espaçadas com 11 caracteres exactos. No caso específico do PatternHunter foram efectuados testes usando 2 e 8 *seeds* em cooperação para avaliar a rapidez do algoritmo em diferentes cenários de exigência de sensibilidade.

Os algoritmos em contenda pesquisaram duas sequências representativas. A primeira corresponde a um gene humano (*humghcsa*), contendo cerca de 65 KBases e conhecida pelo elevado grau de redundância; a segunda refere-se ao genoma da *E. Coli*, que apresenta baixa entropia, numa extensão de 4,6 MBases.

O SimSearch foi implementado em linguagem C, o código foi compilado usando o *gcc* (versão 3.4.2) activando a opção de optimização máxima –O3.

Como plataforma dos testes de desempenho, utilizou-se um sistema com base num processador Intel Pentium IV 3.4GHz, 8KB L1 + 512KB L2 de cache e 1GB de DDR-RAM, sobre Windows XP Professional SP2 OS.

Quanto a tempo de execução para completar a análise de uma sequência pelo SimSearch, os testes realizados permitem-nos calcular o desempenho, para o sistema informático usado, em cerca de 1 segundo por cada tabela de similaridade com 10.000*10.000 células. Em termos comparativos o desempenho do SimSearch é muito superior ao algoritmo de Smith-Waterman, que apresenta um desempenho muitas vezes mais lento. Se compararmos com o BLAST ou o PatternHunter estes apresentam um desempenho mais rápido, porém com uma sensibilidade média a rondar os 85%. Para conseguir um desempenho mais próximo dos algoritmos baseados em heurísticas será necessário abdicar de maior sensibilidade e tal resolução poderá comprometer a descoberta dos padrões necessários para constituir um dicionário de dados mais eficiente. Uma súmula dos resultados esta patente na Tabela 4.22.

	Tempos de Execução					
	SimSearch (*seed 3*)	SimSearch (*seed 11*)	SSearch	BLASTn	PHunter (*2 seeds*)	PHunter (*8 seeds*)
humghcsa	24 segs	14 segs	7,5 horas	4 segs	2 segs	6 segs
E. coli	1,22 dias	17,67 horas	5 anos*	52 segs	14 segs	45 segs
Sensibilidade	~100%	~98%	100%	~80%	~85%	~90%

*estimativa

Tabela 4.22 – Desempenho vs sensibilidade.

Alguns algoritmos de compressão de informação biológica usam algoritmos genéricos de pesquisa de padrões e similaridades, tal como o DNACompress que usa o PatternHunter. Estas aplicações, por serem mais orientadas à investigação biológica,

estão programadas para descartar alguns padrões insignificantes em termos de genómica funcional como são os padrões constituídos por apenas uma base. No campo da compressão da informação estes padrões são ainda mais preciosos pois contêm redundância a dois níveis e, como tal, são essenciais para a efectiva obtenção de compressão.

O aumento da sensibilidade acarreta necessariamente um aumento do tempo de prospecção de padrões. O SimSearch atinge elevada sensibilidade encontrando todos os padrões, sendo apenas limitado pelo tamanho da *seed*, cumprindo assim as exigências de um algoritmo que foi concebido tendo em mente uma utilização como prospector de padrões para inclusão no dicionário a usar como metodologia parcial de compressão. Melhorar o desempenho dos actuais algoritmos de compressão de ADN passa por ter como recurso mais e/ou melhores redundâncias. Assim, antevê-se que essa necessidade implique um aumento no tempo de execução do processo de compressão, algo que logicamente derivará da inclusão do SimSearch na metodologia de compressão, concretamente na apreensão de regularidades do ADN.

Como adiantamento dos trabalhos futuros para aperfeiçoamento do SimSearch pode referir-se a utilização de *seeds* espaçadas, tal como acontece no PatternHunter. É crível que dessa forma se possa acelerar a descoberta de padrões pelo incremento do tamanho global da *seed*, mantendo ou até incrementando a sensibilidade.

Neste capítulo descreveram-se três algoritmos inovadores para a pesquisa e descoberta de padrões em sequências biológicas. Cada um deles, sem excepção, representa uma contribuição para o avanço científico nos domínios abordados, quer pela metodologia seguida quer pelos resultados obtidos. Todos os algoritmos foram sujeitos ao escrutínio de revisores científicos que ajudaram a depurá-los e os aprovaram para publicação em revistas ou em conferências internacionais como consta na secção 1.5. Com base nestes algoritmos fundamentais construiu-se o DNALight, a aplicação que integra as metodologias cooperativas (inicialmente por dicionário de padrões, seguido de predição com base probabilística) para a consecução da compressão do ADN. O DNALight é o tema central do próximo capítulo.

Capítulo 5. Aplicação desenvolvida para compressão de sequências genómicas: Implementação e resultados

A aplicação de compressão de informação genómica desenvolvida, denominada de DNALight, é multidisciplinar. Recorre à integração de várias metodologias de sucesso comprovado para lograr aproveitar mais e melhor todas as fontes de redundância presentes nas sequências de ADN. Quer a metodologia de compressão por recurso a dicionário de redundâncias, quer aquela que recorre à análise e predição probabilística de símbolos baseada em modelos de linguagem, são abordagens cientificamente bem sustentadas por décadas de investigação. O DNALight é inovador, numa primeira fase, pela forma como consegue aumentar as redundâncias presentes no dicionário – através de pesquisa multinível e óptima – e, numa segunda fase, pela introdução de modelos de linguagem compactos, especialmente concebidos e desenvolvidos para aproveitar o potencial de compressão presente nas redundâncias de extensão mais reduzida através de predição probabilística. Também na forma como são codificadas as predições o DNALight é inovador, guardando – por codificação aritmética – as transições necessárias para corrigir, caso necessário, as predições.

Este capítulo corresponde à descrição das metodologias desenvolvidas e integradas para a concretização da estimação de entropia e compressão de sequências de ADN. Sendo que o SimSearch, o algoritmo desenvolvido para a recolha de redundâncias para a

formação do dicionário, já foi detalhadamente descrito na secção 4.3, dar-se-á maior relevo à segunda fase do DNALight que consiste no aproveitamento das redundâncias menos extensas presentes na porção remanescente da sequência a comprimir. Os modelos de linguagem são a base da predição probabilística que se pretende aplicar a essa porção remanescente. A teoria dos modelos de linguagem está descrita nesta dissertação na secção 3.3.1.

Começa-se a abordagem ao DNALight pela sua arquitectura, expondo na Figura 5.1 um modelo conceptual de alto nível das metodologias que o integram. No esquema verifica-se a segmentação da informação a comprimir efectuada pela acção do SimSearch, com o qual se destaca a porção mais repetitiva, em termos de padrões extensos replicados, para formar o dicionário. Por outra via segue a porção remanescente, que será comprimida por acção de um modelo de predição probabilística finalizado com a codificação aritmética (descrita na secção 3.5.3). Pela unificação das porções comprimidas resultantes das duas vias forma-se o ficheiro compacto final. Um modelo conceptual mais detalhado encontra-se na secção seguinte, onde se aborda o modelo de codificação seguido. Em alternativa, sugere-se revisitar a Figura 1.1, que contém a projecção do modelo conceptual global para a aplicação a desenvolver.

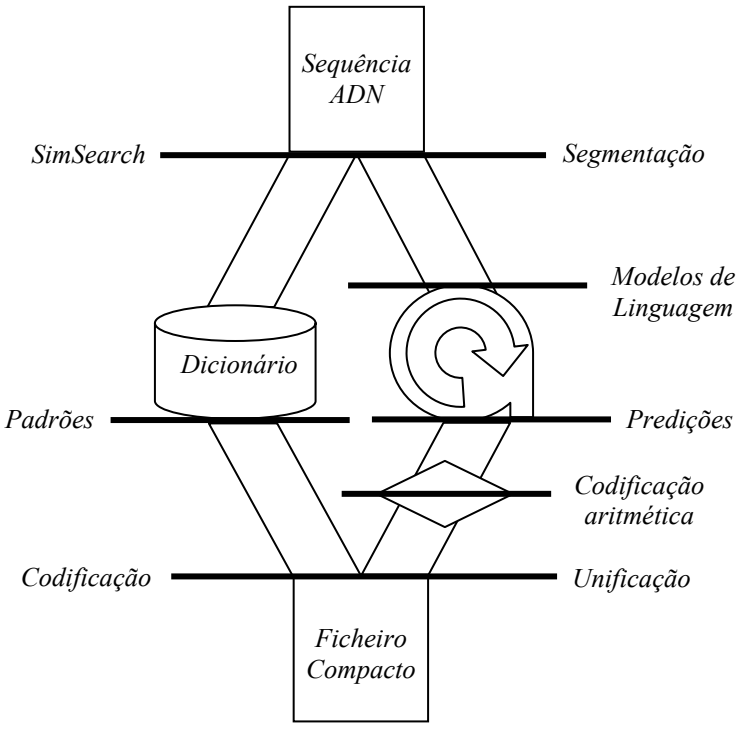


Figura 5.1 – Arquitectura do DNALight.

5.1. O modelo conceptual dos processos de codificação

A sequência das fases da codificação que levam ao ficheiro comprimido é algo complexa, em parte pela subsidiariedade do processo, que visa a maximização do aproveitamento da entropia intrínseca a cada sequência a comprimir pelo contributo e complementaridade de vários sistemas de codificação, cada qual com competências específicas para o aproveitamento de vários níveis de entropia. Em suma, trata-se de um modelo de codificação híbrido, onde as várias metodologias empregues funcionam em entreajuda, intervindo sequencialmente quando o potencial da predecessora se esgota com o intuito de explorar todos os recursos, que advêm da entropia presente, para os fins da compressão.

Os modelos de linguagem desenvolvidos foram especificamente pensados para a composição prevista das sequências que se lhe submetem, ou seja, para maximizar o aproveitamento de padrões mais curtos ou de recorrências estatísticas. Combinaram-se modelos globais e locais adaptativos, incluindo os resultantes da cadeia reversa complementar para gerar um maior número de "sugestões" por parte de todas as entidades relevantes para a predição, que expressarão o seu voto na base que prevêem como o símbolo seguinte. No cômputo geral o modelo de predição produzirá uma probabilidade percentual para cada base (a,c,t,g). Refira-se que o DNALight não prediz as primeiras trinta bases da sequência, estas são fornecidas para evitar que a predição tenha um início sem qualquer histórico. A estas trinta bases deu-se o nome de pistas. As pistas e os modelos globais são parte integrante do ficheiro compacto final.

Finalmente, antes de se proceder à codificação da *bitstream*, as predições serão comparadas com a sequência original para serem corrigidas. No processo de correcção guarda-se o número de transições necessárias para corrigir a predição, o valor zero significará que a predição estava correcta, três transições significa que a base predita como menos provável foi a que efectivamente ocorreu. Finaliza-se o processo de compressão codificando as transições/correcções recorrendo à codificação aritmética. Os detalhes da metodologia serão apresentados nas secções seguintes deste capítulo, no imediato fica o modelo conceptual global que suporta os processos de codificação do DNALight.

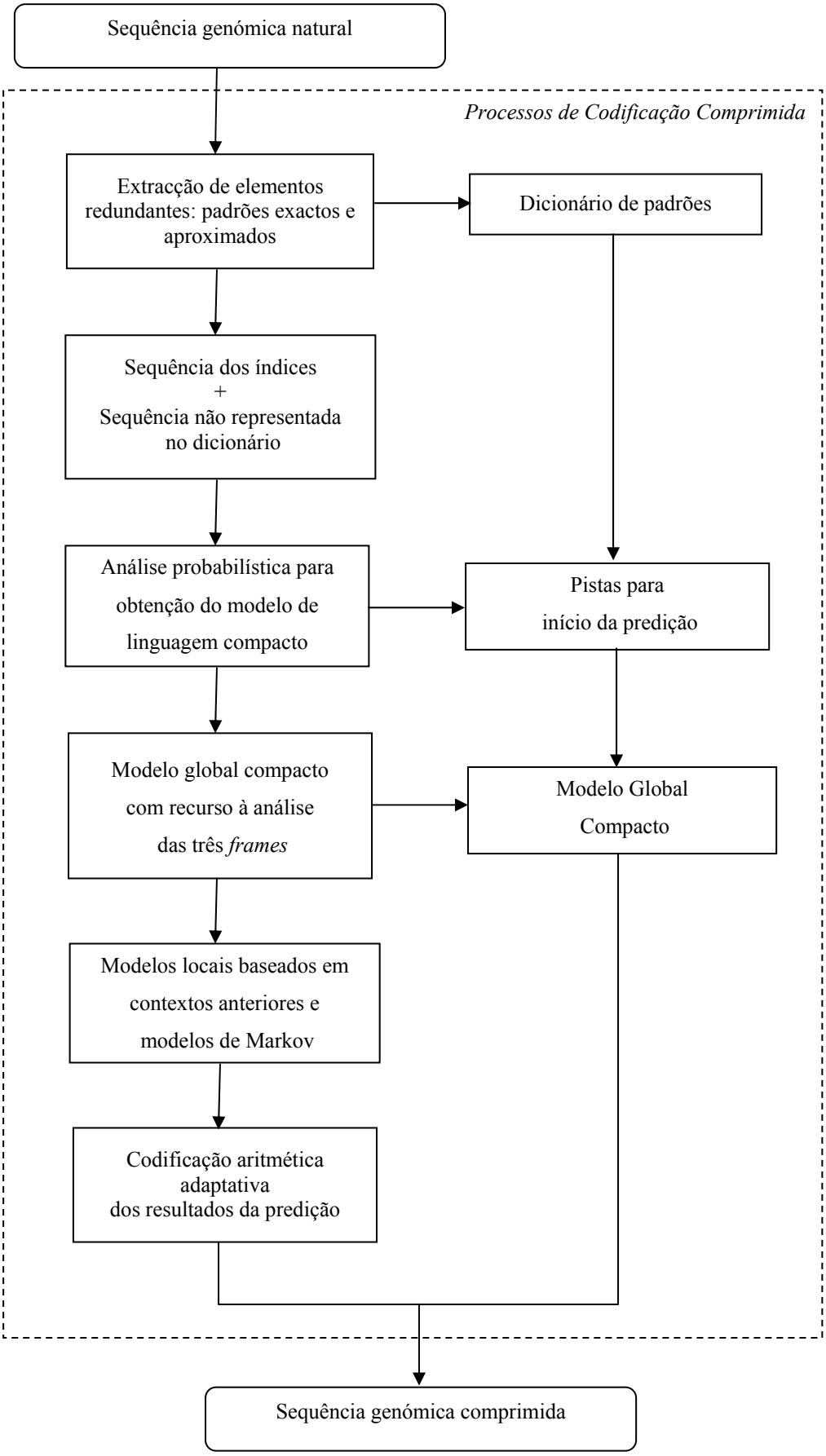


Figura 5.2 – Modelo conceptual para os processos de codificação.

5.2. A primeira fase: Criação do dicionário

A compressão de informação assenta sobretudo numa representação da informação mais eficiente ao nível da codificação das suas recorrências. A compressão de sequências genómicas, na modalidade de compressão por dicionário – a mais representativa neste âmbito – colecta na forma mais compacta possível, uma listagem ou índice de padrões que ocorram com frequência significativa na sequência a comprimir, por forma a poupar espaço ou extensão na representação compacta da mesma informação, mas mantendo a sua integridade e integralidade. A colecção de padrões pode então, qualificar-se como fundamental, quer ao nível da sua representatividade informacional quer ao nível da sua própria compactação. Por representatividade informacional pretende-se significar um nível de prospecção de padrões optimizado do ponto de vista simplesmente informacional, sem pretensão de identificar primariamente funcionalidade genómica.

Para maximizar a detecção de padrões, a pesquisa deverá ser exaustiva e multinível. Assim, na prospecção de padrões exactos e aproximados de uma sequência genómica podem resultar de uma sequência de m bases, um conjunto de padrões cuja extensão somada seja superior a m. A justificação passa pelos seguintes argumentos cumulativos: (*i*) – poderão ocorrer sobreposições nos padrões detectados; (*ii*) – analisam-se também padrões na ordem reversa; (*iii*) – a sequência é analisada em quatro dimensões, a saber, a natural, a complementar natural (A-T ; C-G), a complementar artificial (A-C ; T-G) e a complementar artificial (A-G ; T-C). A técnica usada baseia-se nos princípios da programação dinâmica, porém numa abordagem inovadora, cujos detalhes constituíram os conteúdos da secção 4.3.

Após a recolecção dos padrões pela metodologia exposta, acontece a análise e depuração dos padrões brutos. Esta depuração tem por objectivo eleger os padrões mais abrangentes, com maior proficuidade portanto, que permitirão representar sem redundâncias ou sobreposições as porções repetitivas da sequência a comprimir. Caso alguma redundância se apresente como aceitável face aos ganhos previstos, ou caso um padrão de curta dimensão apresente repetições suficientes, podem ser igualmente integrados no dicionário, havendo para esta selecção critérios de seriação bem definidos a apresentar mais adiante em secção própria.

A composição do dicionário é crítica para o sucesso da compressão. Por esse motivo, os algoritmos de compressão mais bem sucedidos tiram partido da análise da sequência face a padrões exactos e aproximados, obtidos por métodos heurísticos e optimizados. Referindo exemplos concretos, o DNACompress [12] recorre ao software PatternHunter [118, 119] devido à elevada sensibilidade das *spaced seeds* (cerca de 98%), outros algoritmos recorrem a uma solução óptima baseada em programação dinâmica, mais sensível mas igualmente mais morosa e dificilmente praticável em compressão *real-time* em genomas de seres complexos. Os algoritmos que recorrem a programação dinâmica procuram ultrapassar esta limitação introduzindo *seeds* exactas que servem de referencial para estabelecer uma zona pontual onde é aplicada a programação dinâmica, como é o caso do DNAPack [13]. Na aplicação desenvolvida, a prospecção de padrões é mais completa, porque praticada em várias dimensões, e realizada com uma sensibilidade optimizada, baseada num algoritmo inovador, o SimSearch (ver secção 4.3). Trata-se de uma abordagem através de uma nova variante de programação dinâmica mais eficiente, que mantém uma sensibilidade óptima sem incorrer em depauperação de recursos ou custos proibitivos em termos de capacidade e tempo de processamento.

5.3. Prospecção de padrões

A prospecção de padrões é, como já adiantado, exercida em várias dimensões da informação a comprimir. Obviamente, esta intensificação na prospecção de padrões tem prós e contras, se por um lado se detectam mais padrões e se reúne maior potencial de compressão, por outro consome-se mais tempo de processamento. Se os melhores algoritmos concorrentes pesquisam a sequência natural e a complementar, o DNALight adiciona mais dois níveis, incluindo as complementares artificiais. Pretende-se desta forma maximizar a detecção e extracção de padrões, o que equivale a maximizar os recursos de compressão que, se espera, resultem em melhores taxas de compressão. A prospecção de padrões identifica recorrências exactas e aproximadas, em ordem normal ou inversa, que será reiterada para cada forma alternativa de representação considerada. As formas de representação consideradas são quatro, nomeadamente, a sequência natural e as complementares possíveis, ou seja, a sequência complementar natural (A-T ; C-G), a complementar artificial (A-C ; T-G) e a complementar artificial (A-G ; T-C).

Considerar as cadeias complementares artificiais é inovador, por experimentação verifica-se que a sua valia é pouco significativa em termos absolutos mas a sua contribuição pode ser decisiva para atingir taxas de compressão de referência. Para uma melhor compreensão, considerando a sequência *x*="TATCCGCATTATGCGATA", atente-se no exemplo seguinte, disposto na Figura 5.3.

```
(N)Sequência Natural:                          TATCCGCATTATGCGATA
(C1)Complementar Natural:                      ATAGGCGTAATACGCTAT
(C2)Complementar Artificial (A-C ; T-G):       GCGAATACGGCGTATCGC
(C3)Complementar Artificial (A-G ; T-C):       CGCTTATGCCGCATAGCG
```

Figura 5.3 – As quatros representações consideradas para prospecção de padrões em sequências genómicas.

Todos os padrões recolhidos, bem como a lista completa das suas ocorrências, são depois sujeitos a uma análise secundária no sentido de os ordenar pelo binómio de utilidade baseado na sua recorrência e dimensão. Basicamente trata-se de mensurar o número de bases que cada padrão representa para a sequência em análise. Por exemplo, um padrão com 32 bases, que se repete 3 vezes, significa 96 bases envolvidas. Através destes cálculos é possível estabelecer a ordenação ou em casos pontuais, retirar um padrão do dicionário por falta de representatividade. A representatividade é negativamente afectada pelas operações de edição necessárias aos padrões aproximados. Por omissão, o DNALight utiliza o valor 30 como limiar de representatividade, o que equivale, por exemplo, a um padrão constituído por 15 bases com duas ocorrências exactas na sequência. Cada operação de edição requerida significa um consumo adicional de, em média, 8 bits, pelo que seria necessário um padrão com 19 bases para compensar uma operação de edição associada. Esta análise envolve numa primeira fase, durante a prospecção do SimSearch, o recurso ao esquema de pontuação envolvido e ao *threshold* definido, ambos parâmetros do SimSearch que realizam uma primeira filtragem dos padrões a considerar. Outra parameterização autorizada ao utilizador do DNALight será configurar o limiar de representatividade, tornando dessa forma, a aplicação mais flexível. Aumentar o limiar de representatividade significa tornar o dicionário mais selectivo e encaminhar mais segmentos para a segunda fase, a da compressão probabilística.

5.3.1. Codificação óptima dos padrões

Na Figura 5.3 pode ver-se um exemplo, à custa de padrões exactos, do que seria escrever uma sequência com base nos seus padrões multinível. O padrão TAT repete-se cinco vezes, na ordem normal, na sequência a comprimir, contudo nem todas as repetições acontecem ao mesmo nível, duas acontecem na ordem natural, e as restantes ocorrem nas formas complementares. Neste exemplo muito simplista, o dicionário conteria apenas o padrão TAT. A codificação da sequência seria feita à custa deste padrão e necessariamente da codificação dos segmentos não cobertos pelo dicionário. Os segmentos não cobertos pelo dicionário seriam reunidos e passariam à segunda fase do DNALight, que utiliza modelos de linguagem e codificação aritmética.

A primeira fase de codificação compreende a representação compacta do mapa dos padrões na sequência. Pela metodologia seguida é forçoso que os segmentos não repetitivos, portanto aqueles que não estão incluídos no dicionário, sejam reunidos numa nova sequência que será o *input* para a segunda fase do algoritmo. O mapa começa sempre com um intervalo variável de bases não representadas no dicionário, esse número é codificado usando um código de comprimento variável e auto-delimitado de Fibonacci, depois de um segmento não repetitivo vem sempre um repetitivo, excepto se for o fim da sequência. Deste modo a codificação do mapa segue as seguintes regras, usando 1 bit para sinalizar o tipo de segmento seguinte:
- o primeiro segmento é sempre não repetitivo;
- após um segmento não repetitivo vem sempre um repetitivo (padrão);
- após um segmento repetitivo pode acontecer um segmento não repetitivo (bit 1) ou um segmento repetitivo (bit 0).

Se os segmentos não repetitivos são apenas códigos de Fibonacci que quantificam o número de bases a intercalar, os padrões são bastante mais complexos de codificar. Cada padrão tem de referenciar o seu código, o nível onde se encontra, se a ordem é reversa ou normal, as edições necessárias e no final o bit (0) ou (1) que indica que tipo de segmento se lhe segue.

As regras para codificar um padrão são então as seguintes, e são aplicadas pela sequência em que serão apresentadas:
- código do padrão, que corresponde à sua ordem no dicionário (código Fibonacci);
- nível de representação (2 bits) - repetição natural N=00, e as complementares C1=01; C2=10; e C3=11;
- ordem da sequência (1 bit) – ordem reversa/normal – 0/1
- operações de edição (a desenvolver) (2bits) - como os *gaps* não são considerados individualmente, são codificados como inserções de *n* bases, podem utilizar-se dois bits para as codificar (00-Substituição; 01-Delecção; 10-Inserção) pelo que nos resta o código 11 para finalizar a codificação de um padrão.

Um exemplo concreto de codificação de um padrão será apresentada mais adiante na subsecção 5.3.1.2. Como facilmente se estima, a representação de um padrão pode significar um número elevado de bits. Este valor depende de vários factores mas, no mínimo são necessários cerca de 12 bits. Por esta estimativa fica justificada a necessidade de procurar padrões na sequência com a pontuação suficiente para compensar a despesa de codificação.

As operações de edição, que tornam utilizáveis os padrões aproximados, representam um custo significativo na codificação dos padrões. Merecem um destaque especial pois nelas deve incidir grande contenção para não desperdiçar o potencial de compressão que encerram os padrões.

5.3.1.1. Codificação das operações de edição

Substituição: Após o código que identifica a operação (00), seguem-se a posição da base a substituir (código de Fibonacci), em relação à última edição se houver, e o código da base substituta (1 ou 2 bits).

Delecção: Após o código que identifica a operação (01), segue-se a posição da base a eliminar (código de Fibonacci) em relação à última edição se houver.

Inserção: Após o código que identifica a operação (10), seguem-se a posição da base a inserir (código de Fibonacci), em relação à última edição se houver, e o código da nova base (1 ou 2 bits).

Fim do padrão: código que identifica a operação (11).

Para aproveitar cada bit, aquando da indicação da base a substituir ou a inserir optou-se por indicar no cabeçalho do ficheiro comprimido quais as bases mais frequentemente substituídas e inseridas, atribuindo a essas bases o código 0, e codificam-se por 10 e 11 as duas restantes que não são nem a substituída nem a mais frequente. Para exemplificar esta metodologia veja-se, reflectido na Tabela 5.1, o caso em que a base mais inserida é a "C", a segunda mais inserida é a "A", a terceira mais inserida é a "T" e a quarta mais inserida é a "G", bem como a codificação resultante nas várias situações possíveis.

Bases mais inseridas, ordem descendente	Codificação, caso a substituída seja "A"	Codificação, caso a substituída seja "C"	Codificação, caso a substituída seja "T"	Codificação, caso a substituída seja "G"
C	0		0	0
A		0	10	10
T	10	10		11
G	11	11	11	

Tabela 5.1 – Exemplo de codificação das bases inseridas a utilizar em operações de edição.

Refira-se também, que as delecções são casos particulares em que a dimensão da sequência aumenta ou diminui, pelo que não há necessidade de recalcular a dimensão do padrão e de afectar as posições onde se efectuam as edições seguintes.

5.3.1.2. Exemplo de codificação de um padrão

Para concluir a descrição do processo de codificação de padrões exemplifica-se o procedimento da codificação do padrão *p*="*atttttatttcggggagc*", com dimensão 18, sendo o 3º padrão do dicionário e, nesta instância utilizado na forma complementar C1 (01), na ordem reversa (0), com duas operações de edição: uma substituição (00) da 4ª base por "*a*" (10) e a delecção (01) da 12ª base. Para finalizar, indica-se que o segmento

seguinte é não repetitivo (1). A codificação deste padrão encontra-se a seguir, na Tabela 5.2.

Códigos Fibonacci para:
3-0011 – 3º padrão do dicionário;
4-1011 – 4ª base;
8-000011 – distância em bases da 4ª base para 12ª base;

Índice Padrão	Nível	Ordem	Substituição	Posição da base	Base Substituta	Delecção	Posição relativa da base	Fim do padrão	Tipo de Segm.
0011	01	0	00	1011	10	01	000011	11	1

Tabela 5.2 – Representação codificada de um segmento repetitivo ou padrão.

O padrão representado corresponde efectivamente a uma representação compacta pois para representar um padrão de 18 bases (36 bits) usaram-se apenas 26 bits. Considerando que para codificar o padrão no dicionário se consomem 36 bits, o padrão torna-se rentável, em média, após 3 utilizações. Refira-se que pelo menos uma utilização corresponde à versão do padrão que não necessita de edições, sendo que esta gastará apenas 10 bits. Apresentando os cálculos temos 36+10+2*26=98, 98/3=32.67 logo, quanto menor for o número de edições e maior o número de utilizações maior a rentabilidade, ou seja, o ganho de compressão. Este exemplo corresponde ao tamanho médio/baixo de um padrão, na verdade o dicionário recolhe, não raras vezes, padrões com centenas de bases. Quanto ao número de replicações, para o corpus de teste utilizado (ver secção 5.7.1), estas raramente ultrapassam as 10 replicações.

5.3.2. Representação compacta do dicionário

O dicionário é uma porção importante da informação a codificar. De facto, entre os padrões seleccionados pode haver recorrências, pelo que a codificação dos padrões do dicionário também pode ser optimizada. Esta optimização é, por opção, limitada. Pretende-se manter o dicionário pouco codificado de modo a permitir o acesso às recorrências mais significativas de uma sequência directamente a partir da versão comprimida. Portanto, sem necessidade de processar previamente a descompressão.

Acredita-se que esta funcionalidade pode ser importante inclusive como facilitadora da comparação de dicionários entre sequências analisadas.

No apartado de compactação do dicionário, usa-se o algoritmo GRASPm, também desenvolvido neste trabalho e descrito na primeira secção do Capítulo 4. O algoritmo GRASPm é um novo algoritmo ultra-rápido de pesquisa de padrões exactos. Sendo o primeiro objectivo encontrar ocorrências exactas dos padrões mais curtos nos padrões mais compridos, é nessa função que se emprega o GRASPm. O segundo objectivo é substituir, no próprio dicionário, a representação das ocorrências de alguns padrões dentro de outros por um índice do padrão contido, evitando assim redundâncias. No resto, o dicionário é constituído pela sequenciação ordenada dos padrões pela sua representatividade.

Refira-se que a compactação do dicionário é realizada apenas nos casos em que a relação custo/benefício seja favorável à compressão. O método de codificação é simples, para cada padrão referem-se, no seu início o número de sub-padrões (código de Fibonacci) – 0 (zero) para nenhum –, seguem-se as localizações e os códigos dos sub-padrões a inserir, no final codifica-se o resto do padrão (excluídas a redundâncias) na forma linear de 2 bits por base.

5.3.3. Parameterização do *SimSearch* para a primeira fase do algoritmo de compressão

As regras desta selecção foram ajustadas e optimizadas por experimentação, de forma a garantir que a selecção dos padrões corresponda a ganhos efectivos na compressão. Portanto, padrões muito curtos não serão recolhidos no dicionário por não representarem potencial de compressão pela metodologia do dicionário, e no caso dos padrões que necessitem de edição, o custo em bits das operações necessárias para lhe conferir utilidade terá de ser também vantajoso em termos de ganho de compressão.

Se bem que o SimSearch possa detectar 100% dos padrões presentes na sequência, para a finalidade da compressão, o algoritmo foi adaptado para ser mais rápido uma vez que os padrões mais curtos, com $m<7$, não são rentáveis e como tal não é fundamental que

sejam detectados. A função de ganho de compressão nesta sub-metodologia do dicionário prevê que, se o número de operações de edição necessárias para utilizar um padrão for impeditivo da acumulação de determinada pontuação pelo padrão, então este é rejeitado e transitará para a segunda fase, a predição probabilística (ver Figura 5.2). A função de ganho de compressão (*C*) depende da pontuação mínima que deve acumular (*b*), e do tamanho máximo do padrão (*m*) que atingiu essa pontuação. É necessário verificar *C* para o padrão *p*, com C(*b,m*), para justificar a integração de *p* no dicionário. No caso do compressor desenvolvido, o DNALight, *b=7* é a pontuação mínima exigida para um comprimento do padrão máximo de $m=b+|\sqrt{b}|-1$, que correspondem a *C(7,8)*. As possibilidade para obter os mínimos podem resultar de um padrão exacto de 7 bases, ou de um padrão aproximado de 8 bases com 1 erro. Se *b=100* já são admitidas 9 operações mínimas de edição, no máximo. Destas considerações resulta que o algoritmo SimSearch pode ser configurado com comprimento de *seed* (*W*) igual a 7, já que qualquer padrão rentável terá de apresentar um subsegmento de pelo menos 7 bases consecutivas sem erros.

O relatório produzido pelo SimSearch é a base do dicionário. O dicionário resulta da depuração da listagem das recorrências através da análise de representatividade, já abordada na secção 5.3, sendo que todos os padrões cuja representatividade se situe abaixo do limiar definido são excluídos do dicionário.

5.4. Segunda fase: Criação do modelo de linguagem e predição probabilística

Pretende-se obter uma predição baseada em modelos de linguagem que produzam, com base estatística e probabilística, os símbolos que integram a sequência. Da capacidade de acerto destes modelos depende a eficácia da compressão. Os modelos empregues procuram apreender as recorrências da sequência e, na presunção que aconteçam sucessões repetitivas de bases ou codões espera-se empregar códigos mais compactos quando o modelo acertar ou acertar à segunda tentativa. Os desvios das predições são finalmente codificados recorrendo à codificação aritmética (abordada na secção 3.5.3).

O esquema da Figura 5.4 revela o modelo conceptual seguido na segunda fase do DNALight.

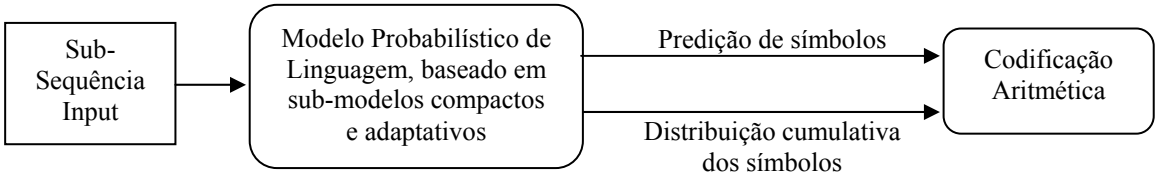


Figura 5.4 – Modelo conceptual da compressão probabilística seguido na segunda fase do DNALight.

Os modelos de linguagem (descritos na secção 3.3.1) são baseados na história das co-ocorrências de n-gramas, sendo desse conhecimento que emanam as predições probabilísticas [153, 154]. A forma de lidar com a probabilística é factorizando-a, a probabilidade de um n-grama é então dependente do produtório das probabilidades dos n-gramas antecedentes. A Equação (5.1) aplica-se na determinação dessas probabilidades que organizadas perfazem o modelo de linguagem. A representação desse historial necessita de operações de alisamento para tornar o modelo mais homogéneo e com melhores resultados globais.

$$P\left(x_1 x_2 x_3 \ldots x_N\right) = \prod_{n=1}^{N} P\left(x_n | x_1 x_2 x_3 \ldots x_{n-1}\right) \tag{5.1}$$

Para o objectivo da compressão de informação a modelação da linguagem visa reduzir o número de bits por símbolo necessários para a sua codificação recorrendo à eficaz predição probabilística. Na Equação (5.2) formaliza-se a expressão através da qual se pode calcular o número de bits por símbolo requeridos, o êxito da compressão depende da minimização desta equação.

$$\langle l \rangle = \sum_{x_i, \{\mathbf{x}\}} P\left(x_i | \mathbf{x}\right) \log_2 \frac{1}{Q\left(x_i | \mathbf{x}\right)} \quad \text{bits por símbolo.} \tag{5.2}$$

Os modelos de linguagem avaliam-se pela medida de perplexidade que apresentam. Intuitivamente, perplexidade refere as situações em que o modelo oferece garantias de

codificação usando menos bits que a codificação linear, no ADN a referência linear são 2 bits por base. A perplexidade pode definir-se com recurso à Equação (5.2) como $2^{(l)}$. A teoria dos modelos de linguagem, incluindo a noção de perplexidade de um modelo podem ser consultadas na secção 3.3.1.

Os modelos probabilísticos de linguagem utilizados nas diferentes metodologias de compressão de ADN que recorrem à predição probabilística, visam sobretudo tirar partido na natureza repetitiva do ADN, modelando o seu comportamento para que as predições incidam sobretudo nas regiões repetitivas. No DNALight, por via de se terem apartado numa primeira fase as ocorrências mais relevantes (com maior pontuação) das subsequências repetitivas, o modelo de linguagem criado não está orientado para os grandes padrões, mas antes para regularidades de menor dimensão e com ênfase nas probabilidades estudadas adaptativamente para os contextos que antecedem a predição (ver esquema da Figura 5.5). Em [185] descartaram-se os padrões com dimensão inferior a 7 bases por serem considerados improdutivos na compressão por dicionário, assim incluir esses padrões no dicionário corresponderia a desaproveitar o seu potencial.

Os modelos de linguagem usados no DNALight diferem da noção clássica de aplicação em modelação de linguagem. Normalmente existe um corpus de treino que produz o modelo, no caso do ADN esse corpus de treino poderia ser formado por um conjunto de sequências representativas de diversos genomas de entre vírus, bactérias, outros seres mais complexos e ainda incorporando sequências dos seus genes isolados. Os modelos resultantes são normalmente volumes significativos de informação que guardam um historial muito completo da caracterização das recorrências estatísticas verificadas na sucessão de N-gramas considerados. No caso do DNALight os modelos de linguagem são sempre calculados e, no caso dos modelos locais, actualizados em tempo de execução da análise que se faz à sequência a comprimir. Desta forma torna-se o compressor independente, sem a necessidade de distribuir juntamente com o DNALight o modelo de linguagem necessário. Os modelos globais do DNALight denominam-se de compactos pois são de facto muito reduzidos. Existe a necessidade de contenção na dimensão dos modelos globais, encontrando o equilíbrio entre eficiência e compacidade, uma vez que os modelos globais juntam-se ao ficheiro compacto como se fossem uma espécie de "segundo dicionário". Por experimentação concluiu-se que a dimensão dos

modelos globais que melhor servem os propósitos do DNALight são modelos de ordem 10 para sucessões de codões (3-gramas).

Na Figura 5.5 apresenta-se o modelo conceptual seguido para operar a segunda fase de compressão do DNALight onde pontificam os modelos de linguagem. A predição probabilística é gerada pela contribuição em concorrência de vários modelos, entre globais (que têm por base a totalidade da sequência) e locais (que se baseiam num histórico limitado mais ou menos recente das bases que antecedem a predição).

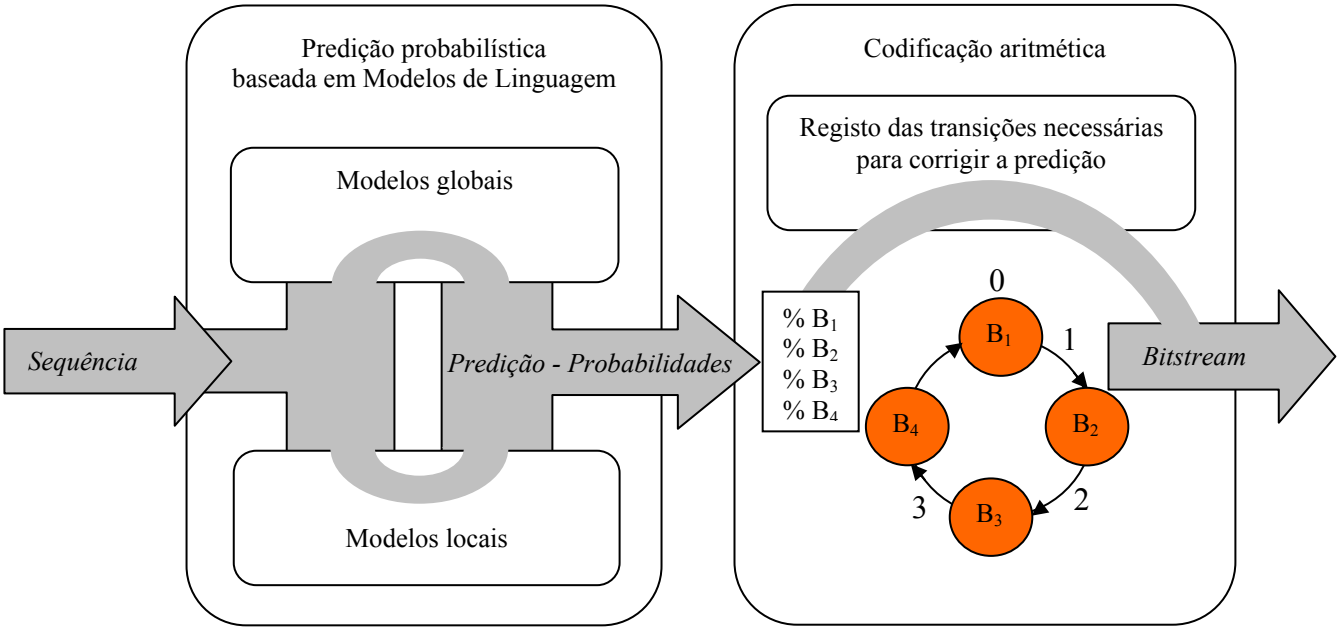


Figura 5.5 – Esquema de compressão usado na segunda fase do DNALight.

A codificação final é outra das inovações do DNALight que, recorrendo à codificação aritmética, guardará números que correspondem às transições necessárias para corrigir a predição e não o símbolo propriamente dito. Mais adiante, na secção 5.5, serão fornecidos os detalhes destes procedimentos.

Sabendo-se que a informação que é submetida a esta metodologia de compressão corresponde à de menos valia para o dicionário, sabe-se igualmente que essa informação não contém subsequências repetitivas, exactas ou aproximadas, acima de determinado *threshold*, aquele utilizado como fasquia para a incorporação no dicionário. Os padrões que podem ser encontrados são sobretudo comportamentais que redundam em pequenos padrões que não são interessantes para a compressão por dicionário.

Conhece-se que tipicamente, a codificação aritmética pode atingir taxas de compressão para as sequências de ADN entre 1,9 e 1,98 bits por base, entre 1 e 5% portanto [171, 186]. Se toda a informação não representada no dicionário fosse entregue a esta metodologia corresponderia, em termos aproximados, a desistir do aproveitamento da redundância dos remanescentes elementos repetitivos de menor valia. Assim, antes de aplicar a codificação aritmética, que entendemos deve ser o último recurso, optou-se por criar um modelo probabilístico de linguagem compacto, orientado para a modelação de padrões curtos, baseados em vários sub-modelos de contexto diferenciado como sejam: vários modelos locais, modelo global, modelo de palindromas. Todos os modelos serão concorrentes, aproveitando-se aquele que conseguir prever o próximo símbolo baseado na ordem mais elevada. Caso todos os modelos tenham suporte mínimo na predição então considerar-se-á a emissão do modelo global.

5.4.1. Modelos locais

Os modelos locais utilizados correspondem ao conhecimento organizado de um conjunto de bases anteriores àquela que se pretende prever, de cuja composição se apuraram ilações probabilísticas, e podem variar em dimensão e em distanciamento. Por exemplo podem ser constituídos pelas 1000 bases anteriores à predita com um distanciamento de 500 bases, ou seja se considerarmos que a base a prever será a 3450ª, então o modelo local que corresponde ao descrito estará compreendido entre as bases 1950ª e 2950ª. A introdução do distanciamento é de particular importância para a especificidade das sequências genómicas. Acontece que, um determinado padrão genómico vulgarmente apresenta as suas réplicas várias centenas ou mesmo milhares de bases adiante. Possuir um contexto com as 500 bases anteriores à predita nem sempre a torna mais previsível, se por outro lado alargarmos o contexto para 1500 bases estamos a diluir as probabilidades interessantes para além de sobrecarregarmos inutilmente a carga de processamento. Assim, usam-se vários modelos locais, de dimensão adequada e com distanciamentos diferentes para alargar a cobertura sem diluir as probabilidades de interesse. Estes modelos são adaptativos pois são actualizados durante a progressão da predição.

De um ponto de vista canónico, as sequências de ADN não são verdadeiramente Markovianas. De facto, o símbolo seguinte nem sempre depende de um contexto anterior, isto porque é vulgar existirem repetições em palindromas, que correspondem à representação reversa complementar de um determinado padrão passado. Os modelos locais são, por esta razão, desdobrados nos seus simétricos para dispor de mais recursos aquando da necessidade de predizer um novo símbolo com algum grau de certeza. Contudo, os modelos de Markov já provaram ser uma ferramenta preciosa na predição de estruturas do ADN pelo que podem funcionar como um complemento aos modelos de linguagem.

5.4.2. Modelo global

O modelo global é, como o próprio nome indica, destinado a fornecer uma visão global em termos probabilísticos sobre a sequência a predizer. Este modelo permite distinguir os padrões mais frequentes na globalidade da sequência, sendo o único que é necessário codificar para integrar a codificação final da sequência, garantindo assim a disponibilidade desde o início da predição. Quanto aos modelos locais estes são exclusivamente realizados à custa de símbolos passados e não necessitam de estar integrados já que são calculados em tempo de execução e adaptativamente actualizados.

Nas sequências genómicas é usual considerarem-se três quadros abertos de leitura (*ORF – Open Reading Frame*), o que significa que se lêem codões sobrepostos, i.e., cada base da sequência é a inicial de um codão. Por questões de simplicidade, é usual referir-se um ORF simplesmente por *frame*.

O modelo global é baseado em codões e está subdividido pelas três *frames* de leitura. A motivação para esta metodologia é suportada suposição de que, a primeira filtragem operada pela captação das regularidades mais representativas aquando da formação do dicionário, terá implicitamente criado uma segmentação entre regiões codificantes e não codificantes. Entre genes e "*junk DNA*" ou entre exons e introns. Devido a esta segmentação – cujo resultado será semelhante ao fenómeno de *splicing* do ADN (ver secção 2.2.2) – teremos reunido as regiões com menor entropia (que incluem normalmente as regiões não codificantes) no dicionário, relegando a sequência

remanescente, que corresponderá à aglomeração de regiões com maior entropia (que incluem normalmente as regiões codificantes), para a codificação por modelo probabilístico de predição. Então, se lidamos nesta fase com sequências com alguma probabilidade de serem regiões codificantes, faz sentido analisar as várias *frames* pelos codões que apresentam. Corroborando esta estratégia estão publicados alguns trabalhos firmados e recentes [172, 201] que sublinham a periodicidade dos codões nas regiões codificantes. Neste sentido, estabeleceu-se por cada *frame* um sub-modelo.

O modelo global usado utiliza ordem 10, ou seja, por cada *frame* são analisadas as frequências dos codões e sequenciados segundo as probabilidades de ocorrerem após a ordem anterior. Sendo 64 possíveis codões, o modelo global guarda 64*10 codões por cada *frame*. No total são 3*64*10=1920 codões, ou 1920*3=5760 bases, ou ainda 11520 bits. Se se considerar que por cada um dos sub-modelos, a ordem 0 não necessita de ser codificada pois corresponde aos valores iniciais (de 1 a 64), então podem-se deduzir 64*3*3*2=1152 bits e cifrar o custo total do modelo global em 10368 bits. A utilidade do modelo global não se reduz apenas à normal predição mas também ao auxílio na escolha dos modelos locais adjuvantes.

O modelo global normal é convertido no modelo global reverso complementar no intuito de capturar eventuais palindromas dos padrões mais frequentes.

Quer os modelos locais quer os globais são modelos probabilísticos de linguagem compactos, baseados em tabelas que proporcionam simplicidade e rapidez de processamento. A estrutura desses modelos será abordada nas secções seguintes.

5.4.3. Estrutura do modelo global

Por cada *frame*, 1, 2 e 3, guarda-se uma sub-tabela como a apresentada na Tabela 5.3. Cada *frame* corresponde, como se mostra a seguir, na Figura 5.6, a uma das três possibilidade para ler sequencialmente codões na sequência de ADN.

Codões da frame 1

Frame 1

AAA	CGT	TTA	TAC	GCG	CAT	GCG	CAT	GAG	CAT	GCG	ATG	CAG	ATG	GCG

Frame 2

AAC	GTT	TAT	ACG	CGC	ATG	CGC	ATG	AGC	ATG	CGA	TGC	AGA	TGG

Frame 3

ACG	TTT	ATA	CGC	GCA	TGC	GCA	TGA	GCA	TGC	GAT	GCA	GAT	GGC

Figura 5.6 – As três *frames* possíveis para a leitura de codões nas sequências de ADN.

A sub-tabela guarda, por cada *frame*, as sequências mais prováveis construídas com os codões que apresentam maior probabilidade de sucederem aos anteriores. Em suma, para o codão 1 (AAA) assinala-se o codão sucessor mais provável, a seguir, considerando os dois anteriores guarda-se o sucessor mais provável e assim sucessivamente até ordem 10. O modelo conseguido reflecte os padrões mais recorrentes presentes na sequência a modelar, a conjugação das três *frames* permite complementar o modelo e reforçar a sua capacidade predictiva. De facto, no caso de haver alguma diferença simples de edição entre padrões (padrões aproximados), a presença das três *frames* permite ultrapassar facilmente essa divergência.

Frame 1		Ordem										
		0	1	2	3	4	5	6	7	8	9	10
Código	Codão	Codão com maior probabilidade de ocorrer como codão sucessor dos anteriores										
1	AAA	1	4	4	16	63	29	32	1	1	2	34
2	AAC	2	12	12	57	23	44	43	5	4	1	31
…	…	…	…	…	…	…	…	…	…	…	…	…
63	GGT	63	52	2	64	4	11	13	24	6	12	59
64	GGG	64	11	13	24	6	12	59	17	55	32	4

Tabela 5.3 – Sub-tabela do modelo global para a *frame* 1.

Do modelo global para a *frame* 1 apresentado na Tabela 5.3 podemos constatar que o codão que com maior frequência sucede ao 1 é o 4. O codão sucessor mais provável para os codões 1+4 é o 4, o sucessor mais provável para os codões 1+4+4 é o 16, e assim sucessivamente. Em caso de não ser possível discernir o mais provável, dada a inexistência ou empate, utilizam-se análises globais às três *frames*. Se ainda assim subsistir a indecisão assinala-se o primeiro dos mais frequentes, na ordem em que estão registados na tabela. Caso mesmo assim não exista qualquer sucessor, e antes de recorrer à predição de ordem 1, então inverte-se o sentido de leitura da tabela do modelo, procurando coincidências no maior número de ordens possível mas não necessariamente iniciadas na ordem 1. Para exemplificar este último conceito veja-se o caso seguinte, no codão 64 temos a partir da 5ª ordem a sucessão provável, 12-59-17-55→32. Pois bem, se o codão 12 (ou outro) não possuir melhor probabilidade (em termos de número de ordens que sustentam a predição) para a predição de um próximo símbolo antecedido pela sucessão de codões 12-59-17-55, então serão utilizadas as ordens do codão 64 apresentadas.

Os casos normais de predição correspondem à utilização das ordens 1 e seguintes, o ideal seria ordem 9 (a 10 corresponde à última predição). Caso uma predição com a máxima sustentação não seja possível, recua-se na ordem até obter uma predição fundamentada. Este processo é formalmente designado alisamento por *backoff* e foi desenvolvido na secção 3.3.1.1 desta dissertação. Obviamente, as predições baseadas em ordens maiores são mais valorizadas e como tal precedem às produzidas por ordens menores.

As predições baseadas no modelo global correspondem apenas a um símbolo, apesar deste modelo estar organizado para analisar codões e efectivamente poder predizer codões. É sabido que os codões sinónimos variam sobretudo na última base, assim é mais seguro predizer apenas uma base de cada vez.

5.4.4. Estrutura dos modelos locais

Os modelos de linguagem locais são de ordem 20, utilizando um contexto de 1000 bases, portanto baseados em bases individuais e não em codões como o global. Por

questões de desempenho em termos de tempo de execução, o distanciamento é recalculado a cada 500 bases. Para cada modelo, à semelhança do realizado para o modelo global, é criada a versão reversa complementar. A estrutura destes modelos é similar à apresentada para o modelo global. Recorda-se que os padrões mais extensos já foram excisados pela análise primária com recurso ao SimSearch, pelo que já constam no dicionário e vão ser representados pelo índice correspondente, os modelos de predição usados visam tirar partido das repetições mais curtas, pelo que a ordem 20 é suficiente. Pela experiência adquirida, os modelos locais têm uma eficácia esporádica e muito variável, i.e., funcionam bem durante algumas predições mas podem ser totalmente inúteis durante centenas de predições sucessivas. O ideal seria dotar esses modelos de adaptabilidade no sentido de alterar o distanciamento e a dimensão perante as características da sequência a comprimir versus o continuado insucesso da predição, algo empiricamente intuído mas não desenvolvido, pelo que tais evoluções são necessariamente remetidas para a secção dos desenvolvimentos futuros.

5.4.5. Realização das predições

A realização e apuramento das predições corresponde a um sistema de votações semelhante ao sistema democrático. Contudo, alguns modelos possuem votos de maior qualidade que outros. Da contabilização dessas votações resulta o modelo probabilístico a passar ao codificador aritmético.

No protótipo testado o sistema de predição inclui 6 modelos de linguagem concorrentes:
1- modelo global normal;
2- modelo global reverso complementar;
3- modelo local com distanciamento longo (3000);
4- modelo local com distanciamento longo reverso complementar;
5- modelo local com distanciamento curto (500); e
6- modelo local com distanciamento longo curto complementar.

Os modelos globais são de ordem 10, mas baseados em codões. Os modelos locais são de ordem 20 mas baseados em bases individuais. Os modelos globais possuem conhecimento de toda a sequência, por isso têm sempre prioridade na predição caso

apresentem palpites de ordem 3 no mínimo. Nestes termos cada um dos 6 modelos supra listados terá por cada predição um palpite. O modelo mais confiante na sua predição, atendendo à ordem que utilizou para a realizar, efectua um número de votos na base da sua preferência para próximo símbolo da sequência predita proporcional à ordem usada. Como referido, os modelos globais têm prioridade pelo que os seus votos são multiplicados por 3, os votos dos modelos locais correspondem à ordem usada simplesmente. Para não haver probabilidade 0 para nenhum símbolo, que traria problemas na codificação aritmética, caso um símbolo não receba nenhum voto então é-lhe atribuído o valor 1.

Para exemplificar o esquema de predição apresentado veja-se o seguinte caso:
1- modelo global normal, prevê a base A, baseado em ordem 5;
2- modelo global reverso complementar, prevê a base C baseado em ordem 1;
3- modelo local com distanciamento longo, prevê a base T, baseado em ordem 3;
4- modelo local com distanciamento longo reverso complementar, prevê a base A, baseado em ordem 1;
5- modelo local com distanciamento curto, prevê a base C, baseado em ordem 2; e
6- modelo local com distanciamento longo curto complementar, prevê a base T, baseado em ordem 1.

Candidatos a Símbolo Seguinte	Votos por Modelo						Totais	Probabilidade da predição
	1	2	3	4	5	6		
A	15			1			16	62%
C		3			2		5	19%
T			3			1	4	15%
G							1	4%

Tabela 5.4 – Esquema de apuramento da predição usado no DNALight.

A probabilidade da predição, resultante da Tabela 5.4, é seguidamente transmitida ao codificador aritmético como modelo, através do qual serão codificados os símbolos 0 (caso a predição esteja acertada), 1, 2, ou 3, caso sejam necessárias 1,2, ou 3 transições respectivamente ao símbolo predito como se descreve na secção seguinte.

5.5. Terceira fase: Codificação aritmética das predições

O modelo de predição gera um novo símbolo para adicionar à cadeia. Independentemente da probabilidade de a predição estar ou não correcta, o sistema de codificação deverá sempre comportar códigos para corrigir a predição caso esta divirja da sequência original. Manter a sequência correcta é fundamental para actualizar os modelos de predição locais.

No trabalho desenvolvido criou-se um sistema de codificação que tira o máximo proveito da capacidade de predição do modelo e também da capacidade de predição aproximada, algo que dentro do nosso conhecimento é inovador. De facto, confiando na capacidade do nosso sistema de predição poder-se-ía pensar que mais tarde aquando da compressão da *bitstream* fornecida ao codificador aritmético, na descrição das probabilidades dos símbolos da fonte, o intervalo mais provável, logo maior, seria o que corresponde às predições do modelo. Com esta nova filosofia, o modelo até poderia errar boa parte dos símbolos e ainda assim conseguir boa compressão, isto obviamente se esses erros fossem maioritariamente mínimos.

Para sistematizar este conceito de predição aproximada há a necessidade de introduzir a noção de modelo adaptativo para a codificação aritmética. É sabido que a codificação aritmética pode ser realizada com base em modelos estáticos ou dinâmicos. Os modelos estáticos assumem que a sequência a comprimir é homogénea em termos de composição e como tal as probabilidades não apresentam variação significativa considerando diversos segmentos para a sequência total. Por outro lado, os modelos adaptativos procuram aproveitar as diferenças acentuadas de distribuição das probabilidades nos diferentes segmentos. Por exemplo, se determinada porção da sequência apresentar uma predominância GC muito acentuada, pode ser vantajoso fornecer dois modelos mais as localizações onde alternar a sua aplicação na compressão. Mais intensivo seria adaptar constantemente o modelo a cada símbolo adicionado. Esta última situação acarretaria a necessidade de um processamento intensivo para recalcular o modelo em função dos caracteres passados.

Retomando o conceito de predição aproximada e do seu aproveitamento, consideremos que o modelo probabilístico realiza uma determinada predição para um símbolo baseado nas seguintes probabilidades de ocorrência:

A: 34%;
C: 30%;
T: 20%;
G: 16%.

Neste caso, o modelo irá prever a base A como símbolo seguinte. Para o caso de a predição estar errada, e se o modelo dispusesse de uma segunda tentativa então prediria a base C, e assim por diante. É crível que as probabilidades apresentadas pelo modelo sejam minimamente fiáveis e sustentadas pelo que, se num caso hipotético se quisesse codificar o próximo símbolo com base nestas probabilidades, então ter-se-ía de optar por compressão aritmética acompanhada de um modelo dinâmico, pois a cada novo caracter as probabilidades serão distintas. Este caso, porém, implica um ónus acrescido em relação ao modelo dinâmico convencional, que reutiliza os símbolos passados para actualizar o modelo, e como tal não necessita de informação explicitamente fornecida de forma suplementar para esse efeito, ou seja a sobrecarga resultante é funcional e não devida a informação adicional. Note-se que esta informação adicional representa um custo que pode por em causa o desiderato principal que é a compressão. Para obviar esta situação e ainda assim tirar algum partido das segundas escolhas, a estratégia de codificação seguida utiliza os pressupostos descritos a seguir.

Considerando um esquema como o representado na Figura 5.7, e considerando como estado principal o correspondente à predição do modelo, sempre que a predição do modelo resultar acertada então adiciona-se o símbolo zero "0" à bitstream. Caso a predição esteja errada então codificam-se as transições necessárias para corrigir a predição. Supondo que o símbolo correcto fosse o "T", o símbolo que dava entrada na *bitstream* seria o dois "2", já que são necessárias duas transições até chegar ao 3º símbolo, sendo os símbolos ordenados por probabilidade descendente associada.

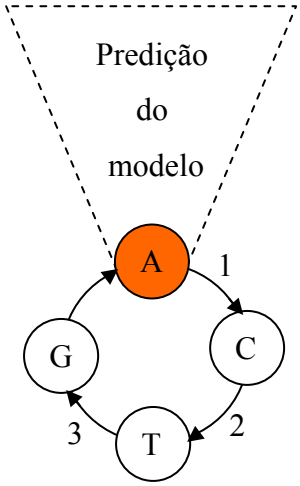


Figura 5.7 – Representação dos acertos ou desacertos predictivos para adicionar à *bitstream*.

Como se descreveu, os símbolos que constituem a *bitstream* são símbolos de indicação do aproveitamento das predições do modelo e não correspondem directamente a símbolos. Pode parecer estranho mas, são esses símbolos que constituem o grosso do ficheiro compactado, e em rigor os símbolos reais só serão recuperados na fase de descodificação.

Aquando da descodificação, e justificado pelo facto da mecânica ser idêntica na descompressão e os modelos usados estarem igualmente disponíveis, será possível redescobrir a ordenação dos símbolos usada para uma determinada predição. Em suma e por paradoxal que possa parecer, na realidade o ficheiro codificado não contém a sequência correcta mas apenas indicações de como atingir essa sequência correcta. Numa antevisão do que poderá ser o conteúdo do ficheiro comprimido resultante da sub-sequência extra-dicionário, no sentido de obter uma compressão substantiva, estima-se que poderá conter uma distribuição percentual dos símbolos correctivos (0, 1, 2 e 3) com a seguinte conformação média: 0-35%; 1-30%; 2-25% e 3-10%. Ou seja, o modelo predictivo não só terá de acertar cerca de um terço das predições como terá de errar apenas por uma transição praticamente a mesma percentagem de predições. Algum optimismo advém do facto de serem apenas quatro símbolos que constituem o alfabeto genómico e que numa predição ao acaso há 25% de probabilidades de acertar, e outras tantas de acertar com uma transição. Assim, o que o modelo necessita fazer é desequilibrar estas probabilidades para um pouco mais de acertos. Nas sequências mais repetitivas isso é, obviamente, mais fácil de atingir.

Adicionalmente é possível efectuar para estes valores uma simulação aproximada da compressão obtida por codificação aritmética recorrendo à fórmula da Equação (5.3), a qual nos permite saber o número de bits necessários para codificar um símbolo sabendo a sua probabilidade de ocorrência na sequência.

$$p = \frac{probabilidade\ do\ símbolo}{probabilidade\ de\ todos\ os\ símbolos}$$
$$-\log(p) = bits / símbolo \tag{5.3}$$

Na Tabela 5.5 estão reunidos os cálculos para as taxas de compressão a esperar perante a distribuição percentual dos palpites produzidos pelo modelo predictivo. Na estratégia seguida não é forçoso que os acertos sejam a maioria, no caso hipotético de se falharem 90% das predições também seria positivo pois a distribuição percentual resultaria semelhante e a codificação aritmética é baseada no desequilíbrio das probabilidades. O facto é que é tão difícil errar sempre da mesma forma como acertar. Acresce que todo o trabalho foi realizado para dotar o modelo que qualidade predictiva para acertar, assim sendo, o desequilíbrio que se pretende atingir é por via do aumento dos acertos ou, em alternativa, dos falhos com desvio mínimo. Sempre que esses dois grupos somem mais que 50% ter-se-á obtido compressão, no exemplo apresentado esses dois grupos somam 65%.

Símbolo	Percentagem	Bits/símbolo
0 – Acerto	35%	–log(0,35)=1,515 bits
1 – Desvio 1	30%	–log(0,3)=1,737 bits
2 – Desvio 2	25%	–log(0,25)=2 bits
3 – Desvio 3	10%	–log(0,1)=3,322 bits

Tabela 5.5 – Previsão das taxas de compressão obtidas por codificação aritmética.

A previsão total corresponderia a:

*1,515*0,35+1,737*0,3+2*0,25+3,322*0,1=1,88 bits por base*

Sabe-se que a codificação aritmética convencional permite, para a informação genómica natural, taxas médias de compressão entre os 1,90 e 1,98 bits por base de média. O valor alcançado pela incorporação do modelo probabilístico de linguagem é significativamente mais baixo. O factor diferenciador decisivo foi o aproveitamento da capacidade probabilística do modelo e também a capacidade de aproximação.

A taxa de compressão média de 1,88 não constitui, por si só, um desempenho suficiente para ombrear com os melhores algoritmos de compressão de ADN mas, importa referir, em defesa do trabalho desenvolvido, que a porção mais repetitiva já tinha sido excisada da sequência antes desta ser submetida ao modelo probabilístico. No final a compressão é melhorada pela contribuição do dicionário.

5.6. Processo de descodificação

O processo de descodificação é em tudo similar ao processo de codificação. Trata-se em termos gerais de um processo mimético à codificação, para o qual estão reunidos os mesmos recursos (modelos de linguagem e estratégia) pelo que se repetem os resultados, simplesmente na descodificação não se gera uma *bitstream* mas antes se vão adicionando as bases desvendadas a uma *string* temporária de saída que há-de ser usada para intercalar com os padrões do dicionário num ficheiro de saída que no final corresponderá à sequência original. Para iniciar a predição são fornecidas algumas bases iniciais (tipicamente as 30 bases iniciais), que no DNALight se denominam de pistas. Estas pistas já foram usadas também para adjuvar o início da predição, impedindo dessa forma que os modelos locais não possuíssem qualquer história ou probabilidades nulas. Na descodificação geram-se novas predições que deverão ser corrigidas com as transições guardadas na informação comprimida pela codificação aritmética, os símbolos originais recuperados são assim recuperados para serem intercalados no mapa dos segmentos repetitivos e não repetitivos aquando da descompressão do ficheiro compacto.

Obviamente, na descompressão não se realiza a função de prospecção de similaridades do SimSearch, que constitui o maior dispêndio de tempo de processamento da compressão, pelo que, para as sequências testadas (ver secção 5.7.1), o tempo de

execução da descompressão é mínimo podendo esta operação ser qualificada como imediata.

5.7. Resultados experimentais e comparações

Como já foi dito, os compressores generalistas não são bem sucedidos na especificidade do ADN, ao invés de comprimirem as sequências provocam a sua expansão. A compressão de ADN é um desafio difícil que só pode ser vencido usando, em complemento com as melhores metodologias, conhecimento das propriedades das sequências genómicas.

Os testes realizados ao DNALight comprovaram a reversibilidade e a compressão efectiva, sendo comprovada a sua eficácia em todas as sequências testadas. É igualmente com assertividade que se comprova que o DNALight supera, ainda que liminarmente, o estado da arte nos domínios da compressão. De facto, nas últimas décadas assistiu-se a um gradual melhoramento das taxas de compressão médias desde os 2 bits por base até aos actuais 1,68 bits por base. Pode parecer pouco mas representa cerca de 20 anos de esforço de investigação.

O DNALight foi implementado em linguagem C, e compilado usando o *gcc* (versão 3.4.2) com optimização máxima –O3. Os testes foram realizados numa plataforma comum, trata-se de um sistema informático com um processador Intel Pentium IV – 3,4GHz, 8KB L1 + 512 KB L2 de cache, com 1GB RAM-DDR e disco rígido de 250 GB.

Como termo de comparação foram escolhidos os algoritmos considerados como o estado da arte em compressão de ADN (julgados pelo binómio tempo de execução/taxa de compressão), incluindo igualmente os mais clássicos para conseguir uma visão da evolução conseguida. Nos eleitos figuram as abordagens mais bem sucedidas, desde a compressão baseada em dicionário à compressão estatística e probabilística. Por ordem de antiguidade, usaram-se os seguintes algoritmos concorrentes: BioCompress [185], GenCompress [171], DNACompress [12], DNAPack [13], CDNA [45], GeMNL [176] e Xpert Models [14]. Todos os algoritmos usados foram anteriormente descritos na

secção 3.6 desta dissertação. Os valores dos seus desempenhos são aqueles que foram publicados pelos seus autores.

5.7.1. Corpus de testes

O corpus de testes corresponde a um conjunto de sequências que integram genomas e genes de diversas espécies com níveis de entropia bem diferenciados. Este corpus é recorrentemente usado nas publicações que versam sobre novas propostas de compressão de informação genómica e, por esse motivo é possível dispor de dados comparativos de todas as aplicações de compressão de ADN de referência para esse corpus, das pioneiras às mais recentes que representam o estado da arte. Dessa forma, pode observar-se a lentíssima evolução que foi conseguida nesta área de investigação.

As onze sequências que compõem o corpus estão disponíveis no GenBank[5] do *National Center for Biotechnology Information* (NCBI), com tamanhos compreendidos entre as 38,8Kb e as 229,4kb. A sua discriminação é a seguinte.

Dois genomas mitocondriais:

MPOMTCG – 186608 pares de bases
PANMCGATPA – 100314 pares de bases

Dois cloroplastos:

CHNTXX – 155844 pares de bases
CHMPXX – 121024 pares de bases

Cinco sequências do genoma humano:

HUMGHCSA – 66495 pares de bases
HUMHBB – 73323 pares de bases
HUMHDABCD – 58864 pares de bases
HUMDYSTROP – 38770 pares de bases
HUMHPRTB – 56737 pares de bases

[5] www.ncbi.nlm.nih.gov

Dois genomas de vírus:

VACCG – 191737 pares de bases

HEHCMVCG – 229354 pares de bases

5.7.2. Resultados de compressão

Os resultados obtidos, medidos pela dimensão em bits do ficheiro compacto dividida pelo número de bases da sequência, dos quais resultaram os bits/bases necessários, estão representados na Tabela 5.6. As diferenças de desempenho entre os algoritmos concorrentes são escassas, pelo que os arredondamentos são apenas efectuados na quarta casa decimal. Como indicador apresenta-se igualmente a média do desempenho, em termos de taxa de compressão, para cada algoritmo no cômputo geral das compressões efectuadas às sequências do corpus de teste.

Sequência	BioC	GenC	DNAC	DNAP	CDNA	GeMNL	XM	DNALight
CHMPXX	1.6848	1.6730	1.6716	1.6602	-	1.6617	1.6577	**1.6415**
CHNTXX	1.6172	1.6146	1.6127	1.6103	1.65	1.6101	1.6068	**1.5971**
HEHCMVCG	1.8480	1.8470	1.8492	1.8346	-	1.8420	1.8426	**1.8317**
HUMDYSTROP	1.9262	1.9231	1.9116	1.9088	1.93	1.9085	1.9031	**1.8905**
HUMGHCSA	1.3074	1.0969	1.0272	1.0390	**0.95**	1.0089	0.9828	0.9724
HUMHBB	1.8800	1.8204	1.7897	1.7771	1.77	-	1.7513	**1.7416**
HUMHDAB	1.8770	1.8192	1.7951	1.7394	1.67	1.7059	1.6671	**1.6571**
HUMHPRTB	1.9066	1.8466	1.8165	1.7886	**1.72**	1.7639	1.7361	1.7278
MPOMTCG	1.9378	1.9058	1.8920	1.8932	1.87	1.8822	1.8768	**1.8646**
MTPACG	1.8752	1.8624	1.8556	1.8535	1.85	**1.8440**	1.8447	1.8442
VACCG	1.7614	1.7614	1.7580	1.7583	1.81	1.7644	1.7649	**1.7542**
Média	1.7837	1.7428	1.7254	1.7148	-	-	1.6940	**1.6839**

Tabela 5.6 – Resultados obtidos pelo DNALight na compressão das sequências do corpus, expressos em bits por base, e comparados com o desempenho dos algoritmos concorrentes.

De referir que para os algoritmos CDNA e GeMNL, algumas sequências não foram testadas pelos seus autores, nem as aplicações estão disponíveis para efectuar esses testes em falta. Consequentemente, para esses algoritmos a média final é omissa.

Analisando os resultados, verifica-se que as propostas mais recentes conseguem apenas avanços marginais, e o DNALight não é excepção. O DNALight apresenta resultados consistentes, as suas metodologias combinadas capturam melhor a entropia pelo que, a justificação para os resultados pouco destacados poderá residir em detalhes da codificação. O avanço marginal alcançado não deve ser, contudo, destituído de mérito científico dada a dificuldade da missão. Em boa verdade, o ADN constitui porventura a arena onde os compressores mais se aproximaram dos reais valores de entropia da informação com que lidam. O DNALight explora novas metodologias de pesquisa de padrões exactos e aproximados para lograr um dicionário mais válido, integrando igualmente a predição probabilística e a codificação aritmética. Ao fim e ao cabo, uma súmula das metodologias mais bem sucedidas na compressão do ADN. O preço a pagar por uma melhoria reduzida na compressão é um esforço computacional maior, que com os computadores actuais não constitui problema mas, deve ser considerado como igualmente importante. Em termos práticos, a compressão de sequências com GBases não é possível em tempo real, nem sequer em alguns minutos. Comprimir significa compreender. Apreender as recorrências de genomas complexos é um processo moroso porque muito exigente computacionalmente. Dada a dispensa desta pesquisa na descompressão, esta torna-se muito mais célere. A descompressão não é, contudo, linear no DNALight, pois os modelos probabilísticos necessitam de repetir as predições para a obtenção da sequência original.

O algoritmo DNALight segue uma abordagem híbrida, sendo basicamente constituído por três módulos, que actuam sequencialmente aproveitando-se das redundâncias encontradas, em primeiro lugar pelo SimSearch (secção 4.3) que captura os padrões rentáveis que serão depositados no dicionário para se reutilizarem na codificação por índices das regularidades representadas. A fase seguinte corresponde à predição probabilística baseada num modelo de linguagem (secção 5.4), também ele híbrido, com sub-modelos locais e globais compactos, que aproveita a sub-sequência que o dicionário não representa. Na terceira fase desenvolve-se a codificação aritmética adaptativa das predições, ou melhor, das transições necessárias para efectuar as correcções caso necessário (secção 5.5).

Com este resumo das computações necessárias à produção do ficheiro comprimido pode entender-se melhor o tempo de processamento necessário. Por ser um algoritmo híbrido

que emprega mais computação que os concorrentes também apresenta dos maiores tempos de processamento. Para as sequências presentes no corpus utilizado, os tempos de execução da compressão cifram-se em cerca de 40 segundos para as sequências menores e cerca de 2 minutos para as maiores. A pesquisa multinível efectuada pelo SimSearch é a maior responsável pelo alargamento dos tempos de execução. Sem a pesquisa multinível o tempo melhoraria significativamente, já que seriam pesquisados apenas dois níveis da sequência como acontece nos algoritmos concorrentes – o normal e o reverso complementar –, reduzindo para metade o tempo de pesquisa de similaridades. No entanto, a pequena contribuição da pesquisa multinível é fundamental e indispensável na obtenção dos resultados apresentados. Simulações efectuadas sem essa contribuição resultaram na perda de, em média, 1% de taxa de compressão.

Dizer que a compressão do ADN já atingiu os limites enunciados por Shannon é com certeza uma falácia, contudo os últimos 10 anos de investigação parecem não ter conseguido um avanço muito significativo. O DNALight representa um avanço de 0,6% em média. Pode parecer pouco mas veja-se o ritmo de evolução nos últimos 20 anos, de facto a evolução é feita centésima a centésima, e nalguns casos mesmo milésima a milésima, e o DNALight contribui com a sua centésima. De facto, a compressão do ADN é um caso paradigmático da compressão da informação e, não será de estranhar que tendo sido a bioinformática inicialmente importadora das metodologias da informática clássica, num futuro próximo venha a ser exportadora de metodologias para aplicação em muitos outros domínios da informática, onde por exemplo a exploração dos padrões aproximados e *data mining* vão assumindo um papel cada vez mais proeminente.

Neste capítulo deu-se a conhecer o DNALight, uma aplicação híbrida que emprega metodologias cooperativas para a compressão de informação genómica. A ênfase colocada na identificação e capitalização de similaridades e recorrências permite-lhe atingir resultados de referência em termos de taxa de compressão, porém tal intensificação da compreensão da sequência a comprimir implica maior tempo de execução. O DNALight não se arroga como uma aplicação acabada, trata-se de um protótipo elaborado para testar e validar as metodologias desenvolvidas neste trabalho de investigação. Ao ser uma aplicação modular, certamente incorpora módulos de maior e menor valia, uns mais desenvolvidos que outros, uns mais profícuos que outros.

Considera-se que ao nível da estimação da entropia foi realizado um trabalho mais completo mas, ao nível da codificação muito mais haveria para fazer. A optimização da codificação escolhendo as melhores variantes dos códigos de Fibonacci existentes ou estabelecer as melhores heurísticas de codificação é um trabalho que deveria merecer maior atenção, mas que foi considerado como mais acessório face aos objectivos estabelecidos. Uma análise crítica mais global, bem como o elenco das conclusões finais são os principais conteúdos do próximo e último capítulo.

Capítulo 6. Discussão, conclusões e perspectivas

Neste capítulo final importa sistematizar topicamente todo o trabalho produzido e descrito nesta dissertação, destacando a sua integração nas vias de investigação da bioinformática actual, a sua actualidade e oportunidade. Importa igualmente relevar as contribuições e resultados substantivos que emanam da investigação e experimentação realizada.

O famoso cientista Russel Doolittle divulgou como conclusão da sua investigação: "Nós somos as nossas proteínas." Trabalhar com dados biológicos respeitantes a sequências de ADN que contêm genes que se expressam e originam as proteínas é, sem dúvida, mais empolgante cientificamente do que trabalhar com outro tipo de informação. A informação genómica é objecto de estudo de uma comunidade cada vez maior de cientistas em todo o mundo que paulatinamente vão acumulando conhecimento das suas propriedades. Conhecer essas propriedades aproxima-nos do conhecimento necessário para o tratamento ou prevenção de muitas doenças do foro genético e de outras causadas por invasões víricas. A inteligência biológica acumulada e depurada em milhões de anos de evolução, registada e transmitida no ADN às novas gerações não pode ser vista como um mero objecto de estudo. Humildemente, perante o genoma humano, os biliões de bases que o constituem apresentam-se como um móbil científico que marcará todo este século e que poderá ser decisivo na própria evolução Humana. Esta é a emoção

científica da bioinformática. Sendo mais pragmáticos, nesta fase de instalação da bioinformática é necessário desenvolver as metodologias de análise da informação genómica em todos os quadrantes: matemática, estatística, física, linguística, informática e biológica são as perspectivas mais previsíveis e nas quais mais se aposta. O desenvolvimento destas metodologias de base potenciará o aparecimento de novas ferramentas capazes de extrair e capitalizar o conhecimento mais útil incluído nos genomas que vão sendo sequenciados. Este trabalho situa-se no patamar da investigação fundamental em bioinformática pois, considera-se que a essência do trabalho realizado corresponde ao desenvolvimento de metodologias de análise de sequências genómicas. A integração das metodologias desenvolvidas providenciou sustentação a uma aplicação de compressão de informação genómica que se apresenta nesta dissertação como o corolário do trabalho desenvolvido. A qualidade do corolário depende necessariamente da qualidade dos teoremas que o sustentam e foi nessas qualidades de base que maior esforço de investigação se investiu.

Para um informático com a formação típica do anos 90, desenvolver investigação em bioinformática só é possível após muita preparação, muitas horas de estudo, de formação e aprendizagem na área da biologia molecular. Querer obter resultados em bioinformática com as ideias clássicas da informática e sem conhecer as especificidades da informação biológica é perder o tempo, eufemisticamente falando. Tomando como exemplo a compressão de ADN, os algoritmos de compressão generalistas, que temos como referências para as linguagens naturais, imagens e outros tipos de dados mais vulgares, falham na compressão do ADN, provocando inclusivamente a expansão para além dos 2 bits por base que a teoria da informação nos indica como caso de entropia máxima.

Ser pioneiro nalguma coisa implica correr riscos, e por ponderados que sejam não deixam de ser riscos. A investigação em bioinformática em Portugal é ainda pouco relevante, tornando-se um terreno difícil para orientandos e orientadores. A compressão da informação genómica é um desafio. Nos últimos anos os avanços conseguidos nesta matéria, em valor absoluto parecem insignificantes. Equipas de investigadores usam tempo e recursos em abundância para conseguir um ganho compressivo de apenas umas centenas de bytes em relação aos antecessores. Aceitar este desafio era aceitar estas regras, ainda que muito galvanizados pela riqueza do tema e com o optimismo típico de

um português e transmontano, os factos eram irrefutáveis e os resultados seriam necessariamente modestos, vistos em termos estritamente numéricos. Ainda assim, será justo considerar os objectivos atingidos. E em abono da verdade, o objectivo principal não se resume à compressão conseguida, mas deve ser considerada a corrida de fundo necessária para atingir essa meta. A quantidade de ciência produzida, decorrente do desenvolvimento das componentes que aportaram a sua decisiva contribuição para lograr a compressão, é significativa. Desenvolveram-se diversos algoritmos inovadores na pesquisa de padrões exactos e aproximados que superam ou se equiparam ao estado da arte. Desenvolveu-se uma aplicação híbrida, que integra de forma inovadora as melhores contribuições na área da compressão de informação genómica, que usando novas metodologias supera ligeiramente o estado da arte, proporcionando um passo em frente neste domínio.

Tendo consciência das limitações a que estamos sujeitos, também importa incorporar neste último capítulo uma análise crítica do trabalho realizado. Sendo um trabalho altamente multidisciplinar, para o seu sucesso concorrem conhecimentos aprofundados de muitas ciências, como sejam a informática, com destaque para a teoria da informação e algoritmos de análise de dados, a estatística, a biologia molecular, a matemática, etc., pelo que dada a sua complexidade e vastidão das áreas de conhecimento a abranger se torna difícil antever resultados óptimos. A bioinformática é, nestes seus primórdios, um ramo da ciência que se desenvolve lentamente, precisamente porque para produzir ciência nesta área, a formação e qualificação necessária tem de ser muito ampla. O trabalho em equipas multidisciplinares tem dados bons frutos, mas será no futuro, quando a iniciativa privada exija das universidades técnicos altamente qualificados nesta área que equipas de bioinformáticos seniores, com a consultoria necessária, atingirão resultados impulsionadores da bioinformática para estágios dificilmente antevistos e proporcionem avanços determinantes, sobretudo nas áreas de suporte à biomedicina.

Se existe a esperança de se ter contribuído modestamente para o desenvolvimento da investigação bioinformática no campo da análise de informação genómica, não é menos certo que, neste estágio, se acumulam inquietações e novas ideias de como prosseguir os desenvolvimentos para mais eficientemente superar os problemas atacados. Uma súmula dessas ideias, incluída em secção própria, encerrará este capítulo.

6.1. Avaliação do trabalho desenvolvido

No momento de encerramento da dissertação justifica-se uma macro-avaliação como avaliação final do trabalho desenvolvido. Nos dois capítulos anteriores concedeu-se vasta atenção aos detalhes técnicos de desenvolvimento e implementação das novas metodologias apresentadas. O conjunto e o concurso dessas metodologias são fundamentais na persecução dos objectivos inicialmente estabelecidos. O somatório das qualidades e limitações de cada módulo totalizam o valor do trabalho desenvolvido. No final o saldo afigura-se como claramente positivo. Importa agora sustentar, no global, esta última asserção, ponderando sobre objectivos e contribuições alcançadas.

O elemento aglutinador de todo o trabalho desenvolvido – a aplicação de compressão de informação genómica desenvolvida que se denominou de DNALight –, foi inspirado no modelo conceptual apresentado no primeiro capítulo. Modelo e direccionamento do trabalho reajustaram-se mutuamente ao longo dos impulsos de desenvolvimento ocorridos mas, no essencial o modelo original revelou-se acertado e um instrumento conceptual essencial.

A teoria da informação que serve de suporte ao trabalho produzido pode identificar-se como uma combinação das visões de Shannon [41] e Kolmogorov [87], sendo a compressão por dicionário mais próxima da teoria da informação de Shannon e a compressão por predição probabilística mais próxima da teoria de informação de Kolmogorov.

6.1.1. Avaliação do modelo conceptual seguido

A estrutura modular do DNALight é flexível, e pode ser decomposta com relativa facilidade. Os módulos que a integram são, mormente, baseados em metodologias inovadoras que se revelaram eficientes por análise experimental comparativa. As metodologias desenvolvidas podem ser transferidas para outras aplicações bioinformáticas com relativa facilidade pois têm, geralmente, função independente. Obviamente, depois da qualidade científica, este poderá ser, mais que qualquer outro, o

factor fundamental para o sucesso deste trabalho e para a sua aceitação pela comunidade científica.

O esquema da Figura 6.1 mostra a estrutura modular do DNALight, onde a informação correspondente a uma sequência genómica representa um bloco heterogéneo de redundâncias. A função de cada um dos algoritmos desenvolvidos e integrados nos módulos do DNALight é recolher redundâncias desse bloco, actuando cada um na porção que lhe está destinada e para a qual tem competências. O objectivo final é extrair e codificar de forma compacta a máxima quantidade de elementos redundantes, qualquer que seja a sua natureza.

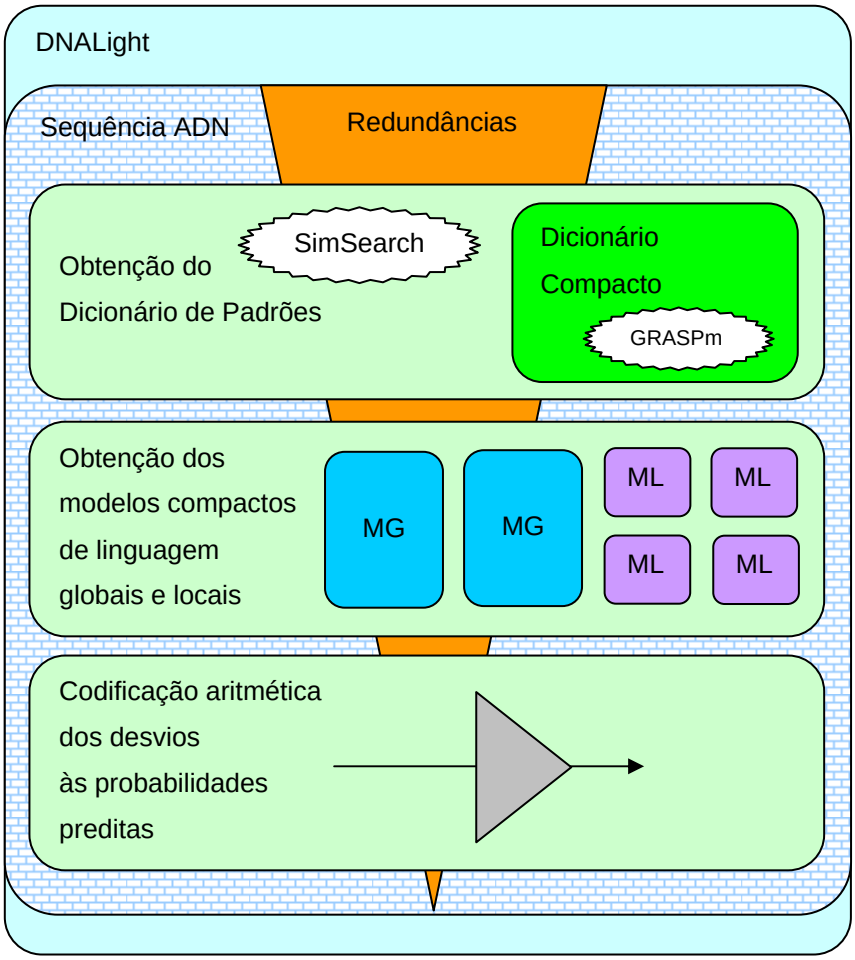


Figura 6.1 – Esquema modular da estrutura do DNALight.

Os resultados obtidos permitem concluir que o modelo conceptual seguido é válido. As metodologias são válidas quando individualmente consideradas e revelam-se igualmente eficientes quando integradas em cooperação. A entropia presente nas

sequências genómica é eficazmente apreendida, as metodologias empregues apresentam boa captação das redundâncias e a codificação demonstra-se suficientemente eficiente, culminando em resultados de taxas de compressão abonatórios para o modelo seguido e para as metodologias que o integram.

6.1.2. Avaliação dos objectivos atingidos

No geral, considera-se que os objectivos inicialmente constituídos foram atingidos. Que melhor constatação dessa realidade que a publicação dos resultados obtidos em revistas científicas internacionais, arbitradas e de qualidade reconhecida na área de conhecimento em questão, ou em reuniões científicas internacionais.

Inicialmente dividiram-se os objectivos em dois grupos representativos, a saber, *i*) produzir investigação científica que redundasse em avanço científico nas metodologias de análise de informação genómica em relação aos elementos repetitivos que a constituem e; *ii*) integrar o conjunto de metodologias desenvolvidas para a captação de entropia e codificação compacta visando constituir uma alternativa metodológica eficiente para a compressão de informação genómica.

A consecução dos objectivos é ratificada pelos resultados parciais obtidos. Particularizando, o algoritmo GRASPm, por via da heurística de filtragem inovadora que lhe confere o decisivo avanço metodológico – que se denominou de regra da compatibilidade – posiciona-se como um algoritmo de referência nas pesquisas de padrões exactos em informação genómica, superando as melhores expectativas iniciais. Derivado da mesma lógica central desenvolveu-se o DC, um algoritmo de *pattern-matching* mais generalista e flexível que se demonstrou altamente eficiente em pesquisas em informação baseada em alfabetos com $\sigma \geq 20$, situando-o como uma referência para identificação de padrões exactos em sequências proteómicas.

O SimSearch cumpre com notável eficiência pesquisas exaustivas de similaridades em sequências genómicas. Com um correcto balanceamento entre sensibilidade e rapidez de execução, o SimSearch é um algoritmo baseado nos princípios da programação dinâmica, que assenta a sua análise de similaridades em séries de distâncias entre

ocorrências da mesma base, incluindo heurísticas inovadoras para a descoberta de padrões exactos ou aproximados, em ordem normal ou reversa. O SimSearch caracteriza-se por apresentar um desempenho semelhante aos algoritmos baseados em heurísticas mas que reúne as vantagens dos algoritmos óptimos baseados em programação dinâmica. O desenvolvimento do SimSearch teve por objectivo o cumprimento da função de prospecção de redundâncias, que serão analisadas e validadas por critérios de representatividade para integrarem a composição do dicionário de padrões, a utilizar como metodologia inicial para lograr a compressão da informação genómica.

Depois de excisados os padrões de rentabilidade compressiva verificada, a compressão probabilística encarrega-se de completar a análise de recorrências estatísticas presentes na sequência remanescente a comprimir. Concebeu-se um modelo de predição baseado em modelos compactos de linguagem que combina modelos locais e globais. Este modelo de predição é inovador porque adaptado às previsíveis características das sub-sequências que terá de analisar. As boas predições geradas pelo modelo permitem converter os acertos predictivos em suporte probabilístico para a consecução da compressão, utilizando como finalizadora do processo a codificação aritmética. Os resultados obtidos permitem afirmar que os objectivos foram atingidos quanto à taxa de compressão alcançada, já quanto ao tempo de execução a utilização cumulativa de metodologias de análise aprofundada de recorrências e redundâncias implica computação intensiva, alargando os tempos de execução para valores acima da média das aplicações concorrentes.

6.1.3. Avaliação das principais contribuições

As contribuições, definidas nesta altura, são necessariamente uma declaração de expectativas. Dependem da divulgação do trabalho e da aceitação e continuação do mesmo, se da primeira premissa já existe algo de concreto na sua consecução, da segunda ainda é prematuro qualquer balanço. Assim, espera-se contribuir nas seguintes áreas de investigação bioinformática por via de:

- prover algoritmos mais eficientes para *pattern-matching* exacto em informação biológica, nomeadamente genómica e proteómica;
- contribuir para o desenvolvimento e optimização de ferramentas de análise de informação genómica em termos dos elementos repetitivos que a integram;
- disponibilizar à comunidade de investigadores em genómica funcional e biologia molecular uma compilação completa e em tempo útil das recorrências presentes numa sequência genómica;
- prover melhores soluções para a estimação de entropia em sequências de ADN;
- prover metodologias eficientes e inovadoras para a compressão de informação genómica, facilitando o seu armazenamento e comunicação; e
- num âmbito mais alargado, dinamizar a investigação em bioinformática em Portugal.

6.1.4. Análise crítica do trabalho desenvolvido

Apesar de existirem múltiplas metodologias de compressão de informação desvendadas pela investigação científica, cada qual com a sua especificidade, forças e fraquezas, simplicidade ou complexidade, no fundo todas elas estão limitadas pela entropia da informação que intentam comprimir. Supondo que existem 30% de redundâncias numa sequência, não é pelo facto de usarem em cascata as várias metodologias que esses 30% vão aumentar. Poderíamos de facto usar todos os métodos e apenas tirar partido de 4/5 do potencial. De facto existem métodos que antes de qualquer operação de compressão procuram aumentar as redundâncias, como acontece com os métodos que usam a transformada de Burrows-Wheeler [173] mas ao alargar, por um lado, o potencial de compressão também, por outro, incrementam a informação a transmitir, pelo que o saldo final não se altera significativamente. O segredo do DNALight poderá estar nas metodologias inovadoras adaptadas às peculiaridades da informação genómica, por exemplo na pesquisa multinível efectuada pelo SimSearch para recolher redundâncias. As vantagens decorrentes poderão ser pouco significativas mas a diferença entre os bons e os melhores é quase sempre diminuta, mas existe. Se dos 30% de redundância, todos as metodologias capturam 25%, então é nos restantes 5% que se fazem as diferenças e se define o ranking. Outra questão é: com que eficiência se atingem os objectivos.

Em relação à eficiência também o DNALight se destaca. A sequência das metodologias empregues é pensada para maximizar a eficiência na captação de redundâncias. A eficiência temporal é negativamente afectada pela intensidade na prospecção de redundâncias. As duas metodologias empregues estão parameterizadas de forma consciente para não depauperar a eficiência temporal mas, o acréscimo em análise implica a dilatação do tempo de execução. É sabido que a compressão por dicionário é rápida – nem tanto se incluídos os padrões aproximados – e como tal deve ser utilizada primeiro, não apenas por essa razão mas também porque é aquela que melhor partido tira dos grandes padrões. Os modelos de linguagem probabilísticos usados são versões muito compactas, desenvolvidos num compromisso entre eficácia e eficiência, (isto porque a sequência que analisam já não tem grandes padrões, mas espera-se capturar as regularidades estatísticas locais e globais), que se calculam e se aplicam com rapidez na predição do próximo símbolo. E por fim, a codificação aritmética é apenas aplicada à subsequência que resultou da excisão dos maiores padrões e das suas ocorrências na primeira fase, sendo assim reduzida ao mínimo necessário, isto porque se trata de um método computacionalmente mais intenso. A descompressão, isentada da prospecção de regularidades para obtenção do dicionário, é o percurso inverso e em tudo idêntica à compressão e, portanto, realizando-se com a mesma eficiência. Apesar de todos estes cuidados, e recordando que o SimSearch é um algoritmo exaustivo para capturar optimamente os padrões existentes, o tempo de execução do DNALight não é de referência, porém é apenas ligeiramente superior à média dos concorrentes.

A escolha da linguagem de programação para implementação dos algoritmos desenvolvidos recaiu na linguagem C, o factor preponderante foi o desempenho, isto porque o processamento deste tipo de aplicações é muito intensivo e é necessária toda a ajuda que se possa reunir para aumentar o desempenho. Como contraponto, a linguagem C apresenta várias dificuldades associadas, as quais pontualmente dificultaram os desenvolvimentos, com sejam a gestão da memória e a reduzida transparência da linguagem na manipulação de *strings*.

O dicionário de padrões não foi exaustivamente explorado em termos de compressão. A sua estrutura foi deliberadamente mantida independente do resto da codificação. A justificação é que mantendo-o minimamente codificado e independente seria possível aceder aos elementos redundantes sem necessidade de descompressão do ficheiro.

6.2. Conclusões

A era pós-genómica foi recentemente iniciada. Uma panóplia de genomas completos, incluindo o humano, estão agora disponíveis para a fase subsequente à sequenciação, a análise funcional e extracção de conhecimento. Se a sequenciação do genoma humano tardou mais de uma década, a análise e compreensão podem demorar um século. E sem embargo, a acumulação de informação genómica relativa a outros seres continua, das centenas passar-se-á rapidamente aos milhares de espécies sequenciadas e das Gigabases passar-se-á às Terabases de informação biológica disponível. Numa nova fase, prevê-se à sequenciação dos indivíduos, e o volume da bioinformação será massivo. Considerando que existem mais de 5 biliões de pessoas, e que cada uma encerra cerca de 3,2 Gigabases de informação no seu genoma, a quantidade é, apenas atendendo à espécie *Homo Sapiens*, dizimadora da nossa capacidade de armazenamento de informação digital. Se se contabilizarem as demais espécies e os seus indivíduos, simplesmente tal volume de informação torna-se incomensurável.

Cumulativamente com o crescimento exponencial do volume de bioinformação, a comunicação deste tipo de dados reflecte-se com igual proporcionalidade no volume de transacções da Internet. Apenas por estas razões, estaria suficientemente justificada a necessidade de investigar e desenvolver metodologias mais eficientes na compressão da bioinformação, de forma a optimizar o seu armazenamento e comunicação. Paralelamente, a investigação em análise informacional do ADN, que está na base da descoberta das recorrências, elementos redundantes ou porções repetitivas, permite que um repositório de padrões com ou sem caracterização funcional fique disponível para os biólogos que se encarregarão de estudos ulteriores que podem culminar em descobertas importantes ao nível da genómica, proteómica, metabolómica ou filogenética.

Desenvolveu-se uma ferramenta modular de compressão de sequências de ADN, baseada na cooperação das melhores metodologias para a especificidade dos dados biológicos, que apresenta eficácia e eficiência numa abordagem a muitos níveis inovadora. A ferramenta foi denominada de DNALight e afigura-se como nova parte do estado da arte na matéria de compressão de informação genómica.

Este trabalho aporta contribuições ao nível da compressão de sequências de nucleótidos com recurso a dicionário de elementos redundantes complementada com predição probabilística baseada em modelos de linguagem compactos, com o seu núcleo a corresponder à pesquisa de padrões exactos e aproximados em sequências genómicas, empregando métodos inovadores e implicando nessa pesquisa vários níveis da representação do ADN, denominada de pesquisa multinível. Assim, a pesquisa de similaridade intra-genómica, com recurso ao algoritmo desenvolvido e denominado de SimSearch, faz-se a quatro níveis que incluem, a sequência normal, com a sua representação 5' – 3', a complementar natural (3' – 5'), e as duas outras complementares artificiais possíveis.

A pesquisa de padrões em sequências de ADN obtém sucesso variável. Os padrões, aqui interpretados como segmentos de ADN repetidos com ou sem exactidão, são mais prolíficos nos eucariotas e ainda mais nas suas regiões não codificantes. Como a situação ideal, para o objectivo primordial da compressão, seria ter 100% de cobertura das sequências a comprimir por referência a padrões, de forma a maximizar a descoberta de padrões optou-se pela pesquisa multinível. Os padrões detectados nos níveis adicionais são pouco representativos em termos absolutos e nada se sabe em relação à sua significância funcional mas, no contexto da compressão revelam-se úteis possibilitando dispor de mais recursos para atingir melhores taxas de compressão. Concomitantemente, desenvolveu-se o SimSearch, um algoritmo de pesquisa de padrões exactos e aproximados que recorre a uma nova variante de programação dinâmica, mais simplificada e consequentemente mais rápida e menos exigente em termos de processamento e memória. Efectuando uma selecção optimizada dos padrões recolhidos, o dicionário de padrões é, ele próprio sujeito a compressão subsequente.

Para aproveitar todas as possibilidades de compressão, a compressão com recurso a dicionário, aqui tida como primária, é complementada com compressão obtida por análise probabilística e predição baseada em modelos de linguagem compactos, esta técnica é portanto, aplicada à subsequência formada por aglutinação das regiões não representadas no dicionário. Esta codificação secundária por aproveitamento da predição probabilística é posteriormente codificada recorrendo à codificação aritmética.

A codificação compacta é efectuada, mormente, com recurso a entradas do dicionário, indexadas por um código auto-delimitado de comprimento variável, e.g., Fibonacci, e intercaladas com a extensão dos segmentos não repetitivos da sequência. A sub-sequência dispensada pelo SimSearch será objecto de análise pelo modelo probabilístico de linguagem que, primeiramente constrói modelos locais e globais que lhe permitam uma base alargada de conhecimento estatístico para efectuar predições sustentadas. Mais tarde, as predições do modelo são apreciadas em termos de precisão aproximada, os eventuais desvios correspondem às transições necessárias para corrigir a predição, zero transições corresponde a um acerto. São esses desvios, expressos pelas suas probabilidades, que são submetidos à codificação aritmética.

O dicionário é, ele próprio, sujeito a compressão mas não aritmética de forma a garantir a plena acessibilidade aos padrões. Desta forma, é possível aceder directamente os padrões que compõem a sequência sem necessidade de descompressão. A descompressão é extremamente simples e rápida no que concerne ao dicionário e mais morosa no que toca a nova predição para a descompressão.

O DNALight é portanto, o resultado da combinação de várias metodologias de compressão para o aproveitamento máximo das redundâncias identificadas e modeladas por metodologias inovadoras. Comparativamente com os algoritmos homólogos, e apesar da extrema dificuldade em comprimir informação biológica, o algoritmo proposto é mais eficiente em termos de taxa de compressão que os melhores concorrentes. Em termos de rapidez na compressão, é claramente mais rápido que os algoritmos que se socorrem da programação dinâmica convencional, mas mais lento que os algoritmos que usam longas *seeds* ou *spaced seeds*, contudo há que interpor como atenuante o facto do algoritmo apresentado efectuar pesquisa óptima em quatro níveis de representação do ADN para garantir maior abundância de padrões descobertos, para além da laboriosa predição probabilística.

Para atingir os objectivos de fundo desenvolveram-se vários algoritmos complementares que se revelam essenciais quando integrados modularmente no DNALight, e quando analisados independentemente se manifestam como detentores de um desempenho superior ao estado da arte. Esta referência visa claramente os algoritmos GRASPm, vocacionado determinação de réplicas exactas de um padrão em sequências de ADN, e

o SimSearch, um algoritmo baseado em séries de distâncias e programação dinâmica para a descoberta de padrões aproximados, similaridades ou alinhamentos locais. O algoritmo DC é igualmente notável para uso generalista, com ênfase para as pesquisas de padrões em proteínas.

As melhores conclusões são a constatação dos objectivos atingidos, e felizmente parafraseiam as contribuições, podendo resumir-se nos seguintes pontos:
- Pesquisa de padrões exactos em sequências genómicas mais eficiente, em média com ganhos de 20% a 200%, dependentemente do tamanho dos padrões;
- Pesquisa de padrões exactos em sequências de aminoácidos mais eficiente, melhorada em 15% em média;
- Como efeito colateral das contribuições anteriores, resultaram novas abordagens em pesquisa de padrões exactos em linguagens naturais;
- Descoberta de similaridade intra-genómica e inter-genómica melhorada, introduzindo uma nova metodologia com sensibilidade optimizada, comparável com a programação dinâmica, e de elevado desempenho, comparável com as solução baseadas em heurísticas.
- Análise informacional de sequências genómicas multinível para maximização de aproveitamento de elementos redundantes na compressão da informação;
- Compressão de informação genómica melhorada, possibilitando o seu armazenamento e comunicação de forma mais eficiente;
- Recolecção exaustiva de elementos repetitivos do ADN, alguns com significância biológica conhecida, e outros assim disponibilizados à comunidade científica da especialidade para estudo e análise funcional.

6.3. Desenvolvimentos futuros

Alardear de ter concluído este trabalho seria expor a fraca compreensão do mesmo. Seguramente surgirão novos trabalhos que tornarão este e outros obsoletos, mas terão seguramente como bases os trabalhos anteriores, como assim aconteceu com o trabalho que aqui se apresenta. Crê-se que o conjunto de algoritmos e ferramentas aqui apresentado contribuirá inovadora e aditivamente para o avanço da ciência no domínio da análise e compressão de informação genómica. Em breve este trabalho será

disponibilizado à comunidade científica e a todos os utilizadores que necessitem deste tipo de aplicações. Será nessa fase que surgirão certamente sugestões e evidências das melhorias a serem introduzidas. Ainda assim, aventam-se algumas considerações em termos de desenvolvimentos futuros.

- A parte da análise de sequências pode ser melhor explorada pelo SimSearch, a maior parte do trabalho está feito, assim basta continuar o seu desenvolvimento para análises mais variadas das sequências genómicas, por exemplo para a descoberta de *motifs*, predição de genes ou mesmo a sua adaptação para aplicações filogenéticas;

- O DNALight poderia proporcionar a possibilidade ao utilizador de escolher que metodologias, das integradas, utilizar, ou ainda melhor, ser inteligente e adaptativo revelando auto-suficiência consciente nessa escolha. Dicionário, predição e codificação aritmética é a sequência de operação do DNALight actual;

- O distanciamento dinâmico dos modelos de linguagem locais seria uma área interessante de investigação para optimizar essa variável, de facto constatou-se a esporádica utilidade dos modelos locais pelo que, aperfeiçoando a sua adaptabilidade se poderia conseguir melhorar o seu desempenho predictivo;

- De forma a facilitar a utilização do DNALight seria interessante desenvolver uma interface gráfica que confira maior usabilidade à aplicação, em complemento seria igualmente útil desenvolver um site de apoio aos futuros utilizadores do trabalho desenvolvido;

- Por último, testar a reutilização das metodologias desenvolvidas noutras áreas aplicacionais da bioinformática.

PostScript.

*No último parágrafo quero citar, tal como no início o fiz, Fernando Pessoa: "**Tudo vale a pena quando a alma não é pequena.**" Se o trabalho valeu a pena a alma ficou certamente maior. No sentido anagógico, uma alma maior reduz a distância para o mistério absoluto ou divino. E essa é uma fonte inesgotável de inspiração.*

Bibliografia

[1] J. D. Watson and F. H. C. Crick, "A structure for desoxyribose nucleic acid," *Nature*, vol. 171 (4356), pp. 737-738, 1953.

[2] J. Cohen, "Computer Science and Bioinformatics," *Communications of the ACM*, vol. 48, nº3(3), 2005.

[3] B. Lewin, *Genes*, VIII ed. New York: Oxford University Press Inc., 2003.

[4] W. White and M. Hendy, "Compressing DNA sequence databases with coil," *BMC Bioinformatics*, 2008.

[5] B. Podobnika, J. Shaoc, N. V. Dokholyand, V. Zlatice, H. E. Stanleyc, and I. Grossef, "Similarity and dissimilarity in correlations of genomic DNA," *Physica*, vol. 373, pp. 497-502, 2007.

[6] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis, 2nd Edition*. NY: Cold Springer Harbor Press, 2004.

[7] I. Mukhopadhyay, A. Som, and S. Sahoo, "Word organization in coding DNA: A mathematical model," *Theory in Biosciences*, vol. 125, pp. 1-17, 2006.

[8] A. Fukushima, M. Kinouchi, Y. Kudo, S. Kanaya, H. Mori, and T. Ikemura, "Statistical Analysis of Genomic Information: Various Periodicities in DNA Sequence," *Genome Informatics*, vol. 12, pp. 435–436, 2001.

[9] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. IT-24, pp. 530-536, 1978.

[10] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. IT-23, pp. 337-343, 1977.

[11] D. R. Powell, D. L. Dowe, L. Allison, and T. I. Dix, "Discovering Simple DNA Sequences by Compression," *Department of Computer Science, Monash University, Clayton, Australia*, 1998.

[12] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," *Bioinformatics*, vol. 18(12), pp. 1696-1698, 2002.

[13] B. Behzadi and F. L. Fessant, "DNA compression challenge revisited," *Lecture Notes in Computer Science*, 2005.

[14] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A Simple Statistical Algorithm for Biological Sequence Compression," presented at IEEE Data Compression Conference (DCC), 2007.

[15] X. Cheng, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its application in genome comparison," presented at GIW'99, 10th workshop on Genome Informatics, pp. 51-56, Tokio, Japan, 1999.

[16] V. D. Gusev, L. A. Nemytikova, and N. A. Chuzhanova, "On the complexity measures of genetic sequences," *Bioinformatics*, vol. 15(12), pp. 994-999, 1999.

[17] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, "An information-based sequence distance and its application to whole mitochondrial genome phylogeny," *Bioinformatics*, vol. 17(2), pp. 149-154, 2001.

[18] L. Allison, L. Stern, T. Edgoose, and T. I. Dix, "Sequence complexity for biological sequence analysis," *Computers and Chemistry*, vol. 24(1), pp. 43-55, 2000.

[19] E. Rivals, J. Delahaye, M. Dauchet, and O. Delgrange, "A first step toward chromosome analysis by compression algorithms," presented at First International Symposium on Intelligence in Neural and Biological Systems (INBS'95), pp. 233, 1995.

[20] E. Rivals, O. Delgrange, J. P. Delahaye, M. Dauchet, M. O. Delorme, A. Henaut, and E. Ollivier, "Detection of signficant patterns by compression algorithms: the case of approximate tandem repeats in DNA sequences," *CABIOS*, vol. 13, nº2, pp. 131-136, 1997.

[21] N. Goodman, "Biological data becomes computer literate: new advances in bioinformatics," *Curr. Op. Biotech.*, vol. 13, pp. 66-71, 2002.

[22] J. Cohen, "Bioinformatics - An Introdution for Computer Scientists," *ACM Computer Surveys*, vol. 36, nº2, pp. 122-158, 2004.

[23] H. P. Yockey, *Information Theory an Molecular Biology*. Cambridge - UK: Cambridge University Press, 1992.

[24] A. Danchin and S. Noria, "Genome strutures, operating systems and the image of the machine," 2004.

[25] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*. NY: Cold Springer Harbor Press, 2001.

[26] C. A. Orengo, D. T. Jones, and J. M. Thornton, *Bioinformatics: Genes, Proteins and Computers*: BIOS Scientific Publishers, Oxford, UK, 2003.

[27] D. Krane and M. Raymer, *Fundamental Concepts of Bioinformatics*. Boston: Addison Wesley, 2003.

[28] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, and e. al., "The Sequence of the Human Genome," *Science*, vol. 291, 2001.

[29] A. F. A. Smit, "The origin of the interspersed repeats in the human genome," *Current Opinions in Genetics and Development*, vol. 6, pp. 743-748, 1996.

[30] C. Patience, D. A. Wilkinson, and R. A. Weiss, "Our retroviral heritage," *Trends in Genetics*, vol. 13, pp. 116-120, 1997.

[31] M. Riddley, *Genome: The autobiography of a species in 23 chapters*. New York: Perennial, 2000.

[32] S. Acharya, "Some Aspects of Physicochemical Properties of DNA and RNA," vol. PhD Thesis: Upsala University, 2006, pp. 76.

[33] D. L. Hartl and E. W. Jones, *Genetics, Principles and Analysis*: Jones and Bartlett Publishers, 1998.

[34] G. F. Joyce, "The antiquity of RNA-based evolution," *Nature*, vol. 418, pp. 214-221, 2002.

[35] J. A. Kolkman and W. P. Stemmer, "Directed Evolution of proteins by exon shuffling," *Nat. Biotechnology*, vol. 19, pp. 423-428, 2001.

[36] L. Patthy, "Modular assembly of genes and evolution of new functions," *Genetica*, vol. 118, pp. 217-231, 2003.

[37] S. Bortoluzzi, F. d'Alessi, C. Romualdi, and G. A. Danieli, "Differential expression of genes coding for ribosomal proteins in different human tissues," *Bioinformatics*, vol. 17(6), pp. 1152-1157, 2001.

[38] A. J. F. Griffiths, W. M. Gelbart, J. H. Miller, and R. C. Lewontin, *Modern Genetic Analysis*. New York: W. H. Freeman and Company, 1998.

[39] J. V. Braun and H.-G. Muller, "Statistical Methods for DNA Sequence Segmentation," *Statistical Science*, vol. 13 (2), pp. 142-162, 1998.

[40] J. Hagenauer and J. Mueller, "Genomic Analysis using Methods from Information Theory," presented at ITW'04, San Antonio-Texas, 2004.

[41] C. E. Shannon and W. Weaver, "The Mathematical Theory of Communication," *Univeristy of Illinois Press*, 1949.

[42] C. Peng, S. V. Buldyrev, A. L. Goldberger, S. Havlin, F. Sciortino, M. Simons, and H. E. Stanley, "Long-range correlations in nucleotide sequences," *Nature*, vol. 356, pp. 168-170, 1992.

[43] R. F. Voss, "Evolution of long-range fractal correlations and 1/f noise in DNA base sequences," *Phisics Review Letters*, vol. 67, pp. 3805-3808, 1992.
[44] P. Bernaola-Galván, P. Carpena, R. Román-Roldán, and J. L. Oliver, "Study of statistical correlations in DNA sequences," *Gene*, vol. 300, pp. 105-115, 2002.
[45] D. Loewenstern and P. N. Yanilos, "Significantly Lower Entropy Estimates for Natural DNA Sequences," *Journal of Computational Biology*, vol. 6(1), 1997.
[46] M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner, and J. Ziv, "On the entropy of DNA: algorithms and measurements based on memory and rapid convergence," presented at Sixth annual ACM-SIAM symposium on Discrete algorithms, San Francisco, California, 1995.
[47] G. Achaz, E. P. C. Rocha, P. Netter, and E. Coissac, "Origin and fate of repeats in bacteria," *Nucleic Acids Research*, vol. 30(13), pp. 2987-2994, 2002.
[48] G. Toth, Z. Gaspari, and J. Jurka, "Microsatellites in different eukariotik genomes," *Genome Research*, vol. 10, pp. 967-981, 2000.
[49] W. B. Langdon and W. Banzhaf, "Repeated Sequences in Linear GP Genomes," presented at GECCO'04, Seattle, 2004.
[50] E. Rivals, M. Dauchet, J.-P. Delahaye, and O. Delgrange, "Fast Discerning Repeats in DNA Sequences with a Compression Algorithm," *Genome Informatics*, vol. 8, pp. 215-266, 1997.
[51] M. F. Singer, "SINEs and LINEs: highly repeated short and long interspersed sequences in mammalian genomes.," *Cell*, vol. 28(3), pp. 433-437, 1982.
[52] P. L. Deininger and M. A. Batzer, "Mammalian retroelements," *Genome Research*, vol. 12(10), pp. 1455-1465, 2002.
[53] A. N. Carnell and J. I. Goodman, "The Long (LINEs) and the Short (SINEs) of It: Altered Methylation as a Precursor to Toxicity," *Toxicol. Sci.*, vol. 75(2), pp. 229-235, 2003.
[54] P. A. Yates, R. W. Burman, P. Mummaneni, S. Krussel, and M. S. Turker, "Tandem B1 elements located in a mouse methylation center provide a target for de Novo DNA methyation," *J. Biol. Chem.*, vol. 274, pp. 36357-36361, 1999.
[55] R. R. Sinden, V. N. Potaman, E. A. Oussatcheva, C. E. Pearson, Y. L. Lyubchenko, and L. S. Shlyakhtenko, "Triplet repeat DNA structures and human genetic disease: dynamic mutations from dynamic DNA," *Journal of Biosciences*, vol. 27(1-S1), pp. 53-65, 2002.
[56] A. M. Hauth and D. A. Joseph, "Beyond tandem repeats: complex pattern structures and distant regions of similarity," *Bioinformatics*, vol. 18(Suplement 1:S31-7), 2002.
[57] A. Krishnan and F. Tang, "Exhaustive whole-genome tandem repeats search," *Bioinformatics*, vol. 20(16), pp. 2702-2710, 2004.
[58] D. Sokol, G. Benson, and J. Tojeira, "Tandem repeats over the edit distance," *Bioinformatics*, vol. 23(2), pp. 30-35, 2007.
[59] D. Sokol and J. Tojeira, "Filtering Tandem Repeats in DNA Sequences," presented at BIOCOMP 2006, pp. 161-167, 2006.
[60] D. R. F. Leach, "Long DNA palindromes, cruciform structures, genetic instability and secondary structure repair," *BIoEssays*, vol. 16(12), pp. 893-900, 2005.
[61] J. A. Shapiro, "A 21st century view of evolution: genome system architecture, repetitive DNA, and natural genetic engineering," *Gene*, vol. 345(1), pp. 91-100, 2005.
[62] M. S. Vieira, "Statistics of DNA sequences: a low-frequency analysis," *Phys. Rev. E*, vol. 60(5), pp. 5932–5937, 1999.

[63] F. Piazza and P. Liò, "Statistical analysis of simple repeats in the human genome," *Physica A*, vol. 347, pp. 472-488, 2005.
[64] S. Karlin and V. Brendel, "Patchiness and correlations in DNA sequences," *Science*, vol. 259 (5095), pp. 677-680, 1993.
[65] A. Provata and T. Oikonomou, "Power law exponents characterizing human DNA," *Phys. Rev. E*, vol. 75, pp. 6, 2007.
[66] D. Kugiumtzis and A. Provata, "Statistical analysis of gene and intergenic DNA sequences," *PHISICA A*, vol. 342(3-4), pp. 623-638, 2004.
[67] S. Karlin and G. Ghandour, "Comparative Statistics for DNA and Protein Sequences: Multiple Sequence Analysis," *Proc. Natl. Acad. Sci. USA*, vol. 82, pp. 6186-6190, 1985.
[68] J. R. Lobry, "The Black Hole Of Symmetric Molecular Evolution." Lyon, France: University of Lyon, 2000.
[69] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*: Prentice Hall, 1990.
[70] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. NY: Wiley & Sons, 1991.
[71] E. N. Trifonov, "Making sense of the human genome," *Structure & Methods Adenine Press*, vol. 1, pp. 69-77, 1990.
[72] H. Wan, L. Li, S. Federhen, and J. C. Wootton, "Discovering simple regions in biological sequences associated with scoring schemes," *Journal of Computational Biology*, vol. 10, pp. 171-185, 2003.
[73] J. M. Hancock, "Genome size and the accumulation of simple sequence repeats: implications of new data from genome sequencing projects.," *Genetica*, vol. 115, pp. 93-103, 2002.
[74] Y. L. Orlov and V. N. Potapov, "Complexity: an internet resource for analysis of DNA sequence complexity," *Nucleic Acids Research*, vol. 32, pp. 628-633, 2004.
[75] L. L. Gatlin, *Information theory and the living systems*. New York: Columbia University Press, 1972.
[76] H. Herzel, A. Schmitt, and W. Ebeling, "Finite sample effects in sequence analysis," *Chaos, Solitons & Fractals*, vol. 4(1), pp. 97-113, 1994.
[77] M. A. Jimenez-Montano, W. Ebeling, T. Pohl, and P. E. Rapp, "Entropy and complexity of finite sequences as fluctuating quantities," *Biosystems*, vol. 64(1-3), pp. 23–32, 2002.
[78] J. K. Lanctot, M. Li, E. Yang, and, "Estimating DNA sequence entropy," presented at 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'00), pp. 409-418, San Francisco, CA, 2000.
[79] V. Chechetkin and V. Lobzin, "Levels of ordering in coding and noncoding regions of DNA sequences," *Physics Letters A*, vol. 22, pp. 354-360, 1996.
[80] Y. Orlov, V. Filippov, V. Potapov, and N. Kolchanov, "Complexity software tools for analysis of information measures of genetic texts," presented at Workshop on Genomic Signal Processing and Statistics (GENSIPS), Raleigh, North Carolina, USA, 2002.
[81] O. Weiss, M. A. Jimenez-Montano, and H. Herzel, "Information content of protein sequences," *J. Theor. Biol.*, vol. 206(3), pp. 379-386, 2000.
[82] A. Hariri, B. Weber, and J. Olmsted, "On the validity of Shannon information calculations for molecular biological sequences," *J. Theor. Biol.*, vol. 147(2), pp. 235–54, 1990.

[83] D. Loewenstern and P. N. Yanilos, "Significantly Lower Entropy Estimates for Natural DNA Sequences," *Computational Biology*, vol. 6, nº1, 1997.
[84] H. Herzel, "Complexity of Symbol Sequences," *Systems Analysis Modelling Simulation*, vol. 5(5), pp. 435-444, 1988.
[85] A. O. Schmitt and H. Herzel, "Estimating the entropy of DNA sequences," *J. Theor. Biol.*, vol. 188, pp. 369-377, 1997.
[86] J. K. Lanctot, M. Li, E. Yang, and, "Estimating DNA sequence entropy," presented at Symposium on Discrete Algorithms, 2000.
[87] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications, 2nd Ed.* New York: Springer, 1997.
[88] V. K. Balasubrahmanyan and S. Naranan, "Information Theory and Algorithmic Complexity: Applications to Language Discourses and DNA Sequences as Complex Systems Part II: Complexity of DNa Sequences, Analogy with Linguistic Discourses," *Journal of Quantitative Linguistics*, vol. 7(2), pp. 153-183, 2000.
[89] H.-K. Pao and J. Case, "Computing Entropy for Ortholog Detection," presented at Proceedings of World Academy of Science, Engineering and Technology, vol. 1, pp. 89-92, 2005.
[90] J. Liu and D. Li, "Conditional LZ Complexity of DNA Sequences Analysis and its Application in Phylogenetic Tree Reconstruction," presented at International Conference on BioMedical Engineering and Informatics, pp. 111-116, 2008.
[91] M. Dehnert, R. Plaumann, W. E. Helm, and M.-T. Hütt, "Genome Phylogeny Based on Short-Range Correlations in DNA Sequences," *Journal of Computational Biology*, vol. 12(5), pp. 545-553, 2005.
[92] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, vol. 26(2), pp. 147-160, 1950.
[93] J. Kieffer and E. Yang, "Grammar Based Codes: A New Class of Universal Lossless Source Codes," *IEEE Transactions on Information Theory*, vol. 46(3), pp. 737-754, 2000.
[94] J. Kieifer and E. Yang, "Ergodic Behavior of Graph Entropy," *ERA Amer. Math. Society*, vol. 3(1), pp. II-16, 1997.
[95] H. Herzel and I. Grosse, "Correlations in DNA sequences:The role of protein coding segments," *Phys. Rev.*, vol. E 55, pp. 800-810, 1997.
[96] D. Holste, I. Grosse, S. Beirer, P. Schieg, and H. Herzel, "Repeats and correlations in human DNA sequences," *Phys. Rev.*, vol. 67(1), pp. 061913.1-061913.7, 2003.
[97] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, pp. 762-772, 1977.
[98] G. Navarro and M. Raffinot, "A Bit-Parallel Approach to Suffix Automata: Fast Extended String Matching," *Lecture Notes in Computer Science*, vol. 1448, pp. 14-31, 1998.
[99] S. A. D. Deusdado and P. M. M. Carvalho, "GRASPm: an efficient algorithm for exact pattern-matching in genomic sequences, to appear," *International Journal of Bioinformatics Research and Applications*, 2008.
[100] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. Mol. Biol.*, vol. 215, pp. 403-410, 1990.
[101] G. Navarro, "A Guided Tour to Approximate String Matching," *ACM Computer Surveys*, vol. 33(1), pp. 31-88, 2001.

[102] I. Lee, A. Apostolico, C. S. Iliopoulos, and K. Park, "Finding approximate occurrences of a pattern that contains gaps," presented at 14th Australasian Workshop on Combinatorial Algorithms, pp. 89-100, 2003.
[103] S. Kurtz, "Approximate string searching under weighted edit distance," presented at 3rd South American Workshop on String Processing - WSP'96, pp. 156-170, 1996.
[104] D. Knuth, J. Morris, and V. Pratt, "Fast Pattern Matching in Strings," *SIAM Journal on Computing*, pp. 323-350, 1977.
[105] R. Horspool, "Practical fast searching in strings," *Software-Practice and Experience*, vol. 10(6), pp. 501-506, 1980.
[106] D. Sunday, "A Very Fast Substring Search Algorithm," *CACM*, vol. 33(8), pp. 133-142, 1983.
[107] H. Peltola and J. Tarhio, "Alternative algorithms for bit-parallel string matching," *Lecture Notes in Computer Science, Proc. SPIRE '03*, vol. 2857, pp. 80-94, 2003.
[108] R. Baeza-Yates, "Improved string matching," *Software-Practice and Experience*, vol. 19(3), pp. 257-271, 1989.
[109] K. Fredriksson, "Faster String Matching with Super-Alphabets," presented at 9th International Symposium on String Processing and Information Retrieval, pp. 44-57, 2002.
[110] T. Lecroq, "Fast exact string matching algorithms," *Information Processing Letters*, vol. 102(6), pp. 229-235, 2007.
[111] K. Fredriksson, "Shift-or string matching with super-alphabets," *Information Processing Letters*, vol. 87(4), pp. 201-204, 2003.
[112] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching," *Technical Report TR-94-7, Dept. Computer Science, University of Arizona, Tucson, AZ*, 1994.
[113] R. A. Baeza-Yates and G. H. Gonnet, "A new approach to text searching," *Commun. ACM*, vol. 35(10), pp. 74-82, 1992.
[114] T. Yan, D. Yoo, T. Z. Berardini, L. A. Mueller, D. C. Weems, S. Weng, M. Cherry, and S. Y. Rhee, "PatMatch: a program for finding patterns in peptide and nucleotide sequences," *Nucleic Acids Research*, vol. 33, pp. 262-266, 2005.
[115] S. Kumar and A. Filipski, "Multiple sequence alignment: In pursuit of homologous DNA positions," *Genome Research*, vol. 17, pp. 127-135, 2007.
[116] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, pp. 195-197, 1981.
[117] L. Noé and G. Kucherov, "Improved hit criteria for DNA local alignment," *BMC Bioinformatics*, vol. 5(149), 2004.
[118] B. Ma, J. Tromp, and M. Li, "Pattern Hunter: fast and more sensitive homology search," *Bioinformatics*, vol. 18, pp. 440-445, 2002.
[119] M. Li, B. Ma, D. Kisman, and J. Tromp, "PatternHunter II: Highly Sensitive and Fast Homology Search," *J Bioinform Comput Biol*, vol. 2(3), pp. 417-439, 2004.
[120] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou, "LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA," *Genome Research*, vol. 13(4), pp. 721-731, 2003.
[121] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707-710, 1966.
[122] R. Bellman, *Dynamic Programming*: Princeton University Press, 1957.
[123] E. Denardo, *Dynamic Programming – Models and Applications*: P. Hall, 1982.

[124] S. Dreyfus and A. Law, *The Art and Theory of Dynamic Programming*: A.C. Press, 1977.
[125] J. B. Kruskal, *An overview of squence comparison. In Time warps, string edits and macromolecules: the theory and practice of sequence comparison*: Addison Wesley, 1983.
[126] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, pp. 443-453, 1970.
[127] P. H. Sellers, "On the theory and computation of evolutionary distances," *J. Appl. Math*, vol. 26, pp. 787-793, 1974.
[128] S. E. R. Durbin, A. Krogh, and G. Mitchison, "Biological Sequence Analysis," 1998.
[129] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*: Cambridge University Press, 1999.
[130] Y. Ephraim and N. Merhav, "Hidden Markov Processes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1518-1569, 2002.
[131] L. Patcher, M. Alexandersson, and S. Cawley, "Applications of generalized pair hidden Markov models to alignment and gene finding problems," presented at Proceedings of the fifth annual international conference on Computational Biology, pp. 241 - 248, Montreal, Quebec, Canada, 2001.
[132] V. D. Fonzo, F. Aluffi-Pentini, and V. Parisi, "Hidden Markov Models in Bioinformatics," *Current Bioinformatics*, vol. 2(1), pp. 49-61, 2007.
[133] W. J. Wilbur and D. J. Lipman, "Rapid similarity searches of nucleic acid and protein data banks," *Proc. Natl. Acad. Sci.*, vol. 80, pp. 726-730, 1983.
[134] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 2444-2448, 1988.
[135] W. R. Pearson, "Flexible sequence similarity searching with the FASTA3 program package," *Methods Mol. Biol.*, vol. 132, pp. 185-219, 2000.
[136] S. F. Altschul, T. L. Madden, A. A. Schaer, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucl. Acids Res.*, vol. 25(17), pp. 3389-3402, 1997.
[137] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning DNA sequences," *J. Comput. Biol.*, vol. 7((1-2)), pp. 203-214, 2000.
[138] J. Buhler, U. Keich, and Y. Sun, "Designing seeds for similarity search in genomic DNA," presented at RECOMB'03, pp. 67-75, 2003.
[139] M. Li, B. Ma, and L. Zhang, "Superiority and Complexity of the Spaced Seeds," presented at ACM-SIAM Symposium on Discrete Algorithms, pp. 444-453, Miami, FL, 2006.
[140] D. G. Brown, "Optimizing Multiple Seeds for Protein Homology Search," *Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 2(1), pp. 29-38, 2005.
[141] K. P. Choi, F. Zeng, and L. Zhang, "Good Spaced Seeds For Homology Search," presented at Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04), pp. 379-390, 2004.
[142] J. Xu, D. Brown, M. Li, and B. Ma, "Optimizing Multiple Spaced Seeds for Homology Search," *Journal of Computational Biology*, vol. 13(7), pp. 1355-1368, 2006.

[143] Y. Sun and J. Buhler, "Designing Multiple Simultaneous Seeds for DNA Similarity Search," *Journal of Computational Biology*, vol. 12(6), pp. 847-861, 2005.
[144] L. Hirschman, J. Park, J. Tsujii, L. Wong, and C. H. Wu, "Accomplishments and challenges in literature data mining for biology," *Bioinformatics*, vol. 18(12), pp. 1553–1561, 2002.
[145] M. Andrade and P. Bork, "Automated extraction of information in molecular biology," *FEBS Letters*, vol. 476, pp. 12-17, 2000.
[146] C. Blaschke, L. Hirschman, and A. Valencia, "Information extraction in molecular biology," *Briefings in BioInformatics*, vol. 3, pp. 1-12, 2002.
[147] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann Publishers, 2005.
[148] F. Browne, H. Wang, H. Zheng, and F. Azuaje, "An assessment of machine and statistical learning approaches to inferring networks of protein-protein interactions," *Journal of Integrative Bioinformatics*, vol. 3(2), 2006.
[149] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten, "Data mining in bioinformatics using Weka," *Bioinformatics*, vol. 20, pp. 2479-2481, 2004.
[150] N. Chomsky, *The Logical Structure of Linguistic Theory*: MIT Press, Cambridge, MA, 1995.
[151] B. D. Dyer, M. D. Leblanc, S. Benz, P. Cahalan, B. Donorfi, P. Sagui, A. Villa, and G. William, "A DNA motif lexicon: cataloguing and annotating sequences," *In silico Bio*, vol. 4, 2004.
[152] S. P. Li, K.-L. Ng, and M. C. Chung, "Quantitative linguistic study of DNA sequences," *PHISICA A*, vol. 321, pp. 189-192, 2003.
[153] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-35(3), 1987.
[154] J. T. Goodman, "A Bit of Progress in Language Modeling, Extended Version. Technical Report," Microsoft Research 2001.
[155] R. Rosenfeld, "Two decades of Statistical Language Modeling: Where Do We Go From Here?" presented at Proceedings of the IEEE, vol. 88(8), pp. 1270-1278, 2000.
[156] J. L. Hutchens, "Natural Language Grammatical Inference," in *An Honours Dissertation in Information Technology*, vol. PhD: University of Western, Australia, 1995.
[157] P. F. Brown, V. J. D. Pieta, P. V. DeSouza, and J. C. Lai, "Class-based n-gram Models of Natural Language," *Computational Linguistics*, vol. 18(4), pp. 467 - 479, 1992.
[158] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*: Cambridge University Press, 1999.
[159] O. White, T. Dunning, G. Sutton, M. Adams, J. C. Venter, and C. Fields, "A quality control algorithm for DNA sequencing projects," *Nucleic Acids Research*, vol. 21(16), pp. 3829 -3838, 1993.
[160] M. Osborne, "Predicting DNA Sequences using a Backoff Language Model," *Unpublished ms*, 2003.
[161] I. Witten and T. Bell, "The zero frequency problem: Estimating the probabilities of novel events in adaptative text compression," *IEEE Transactions on Information Theory*, vol. 37(4), pp. 1085-1094, 1991.

[162] I. J. Good, "The population frequencies of species and the estimation of population parameters," *Biometrika*, vol. 40(3-4), 1953.
[163] P. Clarkson and R. Rosenfeld, "Statistical Language Modeling using the CMU-Cambridge toolkit," presented at Eurospeech '97, 1997.
[164] E. Rivals, M. Dauchet, J. Delahaye, and O. Delgrange, "Compression and genetic sequence analysis," *Biochimie*, vol. 78, pp. 315–322, 1996.
[165] L. Gatlin, *Information Theory and the Living Systems*: Columbia University Press, 1972.
[166] L. Allison, D. Powell, and T. I. Dix, "Compression and Approximate Matching," *Computer Journal*, vol. 1(42), pp. 1-10, 1999.
[167] K. Sayood, *Lossless Compression Handbook*: Academic Press, Elsevier Science USA, 2003.
[168] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The Context Tree Weighting Method: Basic Properties," *IEEE Trans. Inform. Theory*, vol. 41(3), pp. 653-664, 1995.
[169] T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," *Genome Informatics*, vol. 11, pp. 43-52, 2000.
[170] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," presented at Proceedings of the I.R.E., vol. 40(9), pp. 1098-1101, 1952.
[171] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences and its applications in genome comparison," presented at Annual Conference on Research in Computational Molecular Biology, pp. 107, Tokyo, Japan, 2000.
[172] P. J. S. G. Ferreira, A. J. R. Neves, V. Afreixo, and A. J. Pinho, "Exploring Three-Base Periodicity for DNA Compression and Modeling," presented at ICASSP 2006, vol. 5, Toulouse, France, 2006.
[173] D. Adjeroh, Y. Zhang, A. Mukherjee, M. Powell, and T. Bell, "DNA Sequence Compression Using the Burrows-Wheeler Transform," presented at IEEE Computer Society Bioinformatics Conference (CSB'02), 2002.
[174] A. Apostolico and S. Leonardi, "Compression of biological sequences by greedy off-line textual substitution," presented at Data Compression Conference, pp. 143-152, 2000.
[175] I. Tabus, G. Korodi, and J. Rissanen, "DNA Sequence Compression Using the Normalized Maximum Likelihood Model for Discrete Regression," presented at Data Compression Conference (DCC'03), 2003.
[176] G. Korodi and I. Tabus, "An efficient normalized maximum likelihood algorithm for DNA sequence compression," *ACM Trasactions on Information Systems*, vol. 23, pp. 3-34, 2005.
[177] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Commun. ACM*, vol. 30(6), pp. 520-540, 1987.
[178] P. Skibiński, S. Grabowski, and S. Deorowicz, "Revisiting dictionary-based compression," *Software-Practice and Experience*, vol. 35(15), pp. 1455–1476, 2005.
[179] J. G. Cleary and I. H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Transactions on Communications of the ACM*, vol. 32(4), pp. 396-402, 1984.
[180] J. Cleary, W. Teahan, and I. Witten, "Unbounded length contexts for PPM," presented at DCC-95, 1995.
[181] N. Abramson, *Information Theory and Coding*. New York: McGraw-Hill, 1963.
[182] J. J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Res. Develop.*, vol. 20, pp. 198-203, 1976.

[183] R. Pasco, "Source Coding Algorithms for Fast Data Compression," vol. Ph. D. Dissertation: Stanford University, 1976.
[184] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems*, vol. 16(3), pp. 256-294, 1998.
[185] S. Grumbach and F. Tahi, "Compression of DNA Sequences," presented at Data Compression Conference, DCC'93, pp. 340-350, Snowbird, UT, USA, 1993.
[186] G. Manzini and M. Rastero, "A simple and fast DNA compressor," *Software-Pratice and Experience*, vol. 34, pp. 1397-1411, 2004.
[187] B. Behzadi and F. L. Fessant, "DNA Compression Challenge Revisited," presented at Symposium on Combinatorial Pattern Matching (CPM'2005), Korea, 2005.
[188] T. I. Dix, D. R. Powell, L. Allison, J. Bernal, S. Jaeger, and L. Stern, "Comparative analysis of long DNA sequences by per element information content using different contexts," *BMC Bioinformatics*, vol. 8 S(2), 2007.
[189] J. Y. Kim and J. Shawe-Taylor, "Fast String Matching using an n-gram Algorithm," *Software-Practice and Experience*, vol. 24, pp. 79-88, 1994.
[190] S. Kim, "A New String Matching Algorithm Using Partitioning and Hashing Efficiently," *The ACM Journal of Experimental Algorithmics*, vol. 4(2), 1999.
[191] G. Navarro and M. Raffinot, "Fast and Flexible String Matching by Combining Bit-Parallelism and Suffix Automata," *ACM Journal of Experimental Algorithms*, vol. 5(4), 2000.
[192] T. Lecroq, "Experimental results on string matching algorithms," *Software Practice & Experience*, vol. 25(7), pp. 727-765, 1995.
[193] F. Franek, C. G. Jennings, and W. F. Smith, "A simple fast hybrid pattern-matching algorithm," *Lecture Notes in Computer Science, CPM2005*, vol. 3537, pp. 288-297, 2005.
[194] D. Sunday, "A Very Fast Substring Search Algorithm," *Commun. of the ACM*, vol. 33(8), pp. 133-142, 1990.
[195] M. V. José, T. Govezensky, and J. R. Bobadilla, "Statistical properties of DNA sequences revisited: the role of inverse bilateral symmetry in bacterial chromosomes," *Physica A: Statistical Mechanics and its Applications*, vol. 351(2-4), pp. 477-498, 2005.
[196] T. Schmidt and J. S. Heslop-Harrison, "Genomes genes and junk: the large-scale organization of plant chromosomes," *Trends Plant Sci.*, vol. 3, pp. 195-199, 1998.
[197] R. Kolpakov, G. Bana, and G. Kucherov, "mreps: efficient and flexible detection of tandem repeats in DNA," *Nucleic Acids Res.*, vol. 31, pp. 3672-3678, 2003.
[198] A. Lefebvre, T. Lecroq, H. Dauchel, and J. Alexandre, "FORRepeats: detects repeats on entire chromosomes and between genomes," *Bioinformatics*, vol. 19, pp. 319-326, 2002.
[199] F. Sanchez, E. Salami, A. Ramirez, and M. Valero, "Performance Analysis of Sequence Alignment Applications," presented at IEEE International Symposium on Workload Characterization, pp. 51-60, 2006.
[200] W. R. Pearson, "Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms," *Genomics*, vol. 11, pp. 635-650, 1991.
[201] R. Mantegna, S. Buldyrev, A. Goldberger, S. Havlin, C. Peng, M. Simons, and H. Stanley, "Systematic analysis of coding and noncoding DNA sequences using methods of statistical linguistics," *Phys. Rev. E*, vol. 52(3), pp. 2939-2950, 1995.