Universidade Federal Fluminense
Instituto de computação

Keity Yamamoto

Arredondamento Randômico e o Problema da Seqüência mais Próxima

DEZEMBRO DE 2004

Universidade Federal Fluminense
Instituto de computação

Arredondamento Randômico e o Problema da Seqüência mais Próxima

Keity Yamamoto

Dissertação apresentada ao Curso de Pós-Graduação em Computação Aplicada e Automação da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Computação Aplicada e Automação.

Orientadores: Carlos Alberto de Jesus Martinhon
Helena Cristina da Gama Leitão

Arredondamento Randômico e o Problema da Seqüência mais Próxima

Keity Yamamoto

Dissertação apresentada ao Curso de Pós-Graduação em Computação Aplicada e Automação da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Computação Aplicada e Automação.

BANCA EXAMINADORA

_____

Prof. Dr. Carlos Alberto de Jesus Martinhon (Orientador)

_____

Prof. Dr. Carlile Campos Lavor

_____

Prof. Dr. Fábio Protti

Aos meus pais, esposa e filha.

Agradecimentos

Caminho difícil, tortuoso, longo, mas consegui chegar até seu final. Com muita vontade, esforço, dedicação e colaboração de muitos, pude realizar este sonho, esta grande conquista. Gostaria de deixar registrado aqui, meus sinceros e profundos agradecimentos a todos envolvidos direta ou indiretamente neste trabalho.

Inicio meus agradecimentos primeiramente a Deus, por ter me dado esta oportunidade de cursar o mestrado e estar sempre ao meu lado em todos os momentos de minha vida.

Na UFF, gostaria de agradecer aos meus orientadores Dr. Carlos Alberto Martinhon e Drª. Helena Cristina da Gama Leitão, que estiveram sempre presentes e com quem aprendi muito em relação à área de otimização e biologia molecular. Também cito aqui, os professores que tem acompanhado minha evolução desde a graduação, e pude novamente estar em sala de aula com eles no mestrado. A galera que me acompanha desde a graduação, Dalboni, Dayse e Dudu. E a galera que conheci no mestrado.

Agradeço aos membros da banca examinadora, professor Dr. Carlile Campos Lavor e professor Dr. Fábio Protti, pelas suas disposições e boa vontade em julgar este trabalho. Tenho muita satisfação em tê-los presente a minha defesa.

Na Acol, empresa onde trabalho desde que me formei na graduação da UFF (1999), gostaria de agradecer aos diretores Sérgio Pinna e Eriton Santana por me apoiarem em todos os momentos desde minha contratação, e principalmente no período que estive realizando este trabalho no mestrado. À todos os companheiros e companheiras da Acol que estiveram me acompanhando e apoiando.

À minha família boliviana, Jorge (pai), Beatriz (mãe), Jorge, Gabriel e Suzana (irmãos), que sempre estiveram presentes em minha vida, desde os meus dois anos de idade. À todos meus amigos que torceram por mim, e à tia Jô (*in memorian*).

À família Yamamoto. Meus pais Itaru Yamamoto e Ana Tomico Yamamoto que souberam construir nossa maravilhosa família. Tenho muito orgulho deles, e tudo que sou

devo a eles. Por isso, tudo que faço, faço para eles com muito amor. Completando a família, as minhas irmãs Lue e Tie, seu marido Cezar e o mais novo integrante da família David. Em especial á minha esposa, Tatiana, pelo amor, companheirismo, apoio e compreensão, e minha filha Ana Zélia, esta sim, sempre presente nos meus estudos, rabiscando todos os meus materiais de estudo.

Aos meus avós paternos Sho Yamamoto (*in memorian*) e Kazue Yamamoto e maternos Tiyoiti Hosaku e Yoshiko Hosaku (*in memorian*).

Obrigado, este trabalho é nosso!

Resumo

Neste trabalho, apresentamos inicialmente alguns conceitos básicos e definições presentes na biologia molecular, visando uma maior compreensão de alguns dos problemas combinatórios mais freqüentes descritos na literatura.

Daremos uma atenção especial ao problema da seqüência mais próxima (PSMP), que consiste na determinação de uma seqüência de caracteres que mais se aproxima, segundo uma dada métrica, de um conjunto pré-definido S de seqüências. Assim, dado um conjunto $S = \{s_1,s_2,...,s_m\}$ de seqüências (todas de tamanho n) sobre um alfabeto Σ, deseja-se determinar uma seqüência s_H de tamanho n que minimize a maior distância entre s_H e s_i, para todo $i \in \{1...m\}$.

Estudamos a prova de NP-Completude do PSMP e fizemos uma análise detalhada dos principais algoritmos aproximativos, determinísticos e randômicos, existentes na literatura. Na verdade, a grande maioria dos procedimentos existentes para o problema são baseados em métodos probabilísticos. Desta forma, fazemos uma descrição mais detalhada das técnicas de arredondamento randômico e *derandomização*, em particular o método das probabilidades condicionais, mostrando os avanços mais recentes obtidos até o momento.

Desenvolvemos finalmente, uma estratégia de *derandomização* baseada no método de estimadores pessimistas introduzida por Raghavan em 1988.

Palavras-chave: Problema da seqüência mais próxima, biologia computacional, *derandomização*, arredondamento randômico.

Abstract

In this work, we first present some basic concepts and definitions normally used in molecular biology. They are used, in order to describe some of most important combinatorial problems posed by the biologists.

We will give a special attention to the Closest String Problem (CSP), which consists in the determination of a sequence s_H of characters (belong to some alphabet Σ) that is closer to a given set $S = \{s_1, s_2, \ldots, s_m\}$ of strings of Σ each of length n. The objective in this case is to find a sequence s_H such that the maximum distance (according to a given metric) between s_H and s_i for $i=1 \ldots m$ is minimized.

We study the proof of completeness of the CSP and concentrate our attention in the deterministic and randomized approximation algorithms listed in the literature. In fact, the major part of these techniques are based on probabilistic methods. For this reason, we present the most recent results, and give a detailed description of these strategies such as the randomized rounding procedure and the derandomization techniques, in particular, the method of conditional probability.

Finally, we develop a derandomization strategy using pessimistic estimators as proposed by Raghavan in 1988.

Keywords: closest string problem, computational biology, derandomization, randomized round.

Sumário

Lista de Figuras

Introdução

Desde a descoberta da estrutura do DNA em 1953 por Watson e Crick [Watson & Crick, 1953a], [Watson & Crick, 1953b], grandes avanços têm sido presenciados na biologia molecular, que tem como principal objetivo o estudo dos ácidos nucléicos e proteínas. A principal dificuldade desta área é a grande quantidade de dados envolvida. Conseqüentemente, muitos dos problemas práticos relevantes são altamente combinatórios (muitos deles NP-Difíceis), sendo portanto, fundamental o desenvolvimento de técnicas eficientes para sua solução.

Daremos uma atenção especial ao Problema da Seqüência Mais Próxima-PSMP (C*losest String Problem*), apresentando uma coletânea dos principais algoritmos aproximativos (determinísticos e randômicos) existentes na literatura. Formalmente falando, seja $S = \{s_1,s_2,...,s_m\}$ um conjunto de seqüências (todas de tamanho n) sobre um alfabeto Σ. O objetivo será encontrar uma seqüência s_H de tamanho n que minimize a maior distância d entre s_H e s_i, $\forall\ i$=1...m, ou seja, desejamos minimizar d, onde $d_H(s_H, s_i) \leq d$, para todo i=1...m.

Este trabalho está organizado da seguinte forma. O capítulo 1 apresenta uma descrição sucinta dos principais conceitos e definições da biologia molecular necessários para uma melhor compreensão de alguns dos problemas combinatórios mais freqüentes encontrados na literatura. No capítulo 2, apresentamos uma pequena amostra destes problemas, destacando dentre outros, o problema de comparação de seqüências, sequenciamento e mapeamento de DNA, e rearranjo de genomas. No capítulo 3, definimos o problema da seqüência mais próxima (PSMP), estudamos sua prova de NP-Completude e descrevemos as técnicas de arredondamento randômico e *derandomização*, em particular estudamos o método das probabilidades condicionais e o método dos estimadores pessimistas. No capítulo 4,

apresentamos um algoritmo 2-aproximado bastante natural para o problema e o algoritmo aproximativo de Ben-Dor *et al*. [Ben-Dor *et al*., 1997], um algoritmo com razão de performance próximo do valor ótimo para *d* suficientemente grande. Desenvolvemos ainda, a estratégia de *derandomização* baseada no método dos estimadores pessimistas sugerida em [Ben-Dor *et al*., 1997]. No capítulo 5, apresentamos o algoritmo $4/3(1+\varepsilon)$-aproximado de Lanctot *et al*. [Lanctot *et al*., 1999]. No capítulo 6, apresentamos o esquema de aproximação polinomial desenvolvido por Li *et al*. [Li *et al*., 2002] e, finalmente, apresentamos as conclusões e sugestões para trabalhos futuros.

Capítulo 1

Introdução à Biologia Molecular

1.1 Introdução

Desde a descoberta da estrutura do DNA em 1953 por Watson e Crick [Watson & Crick, 1953a], [Watson & Crick, 1953b], grandes avanços têm sido presenciados na biologia molecular, cujo principal objetivo é o estudo dos ácidos nucléicos e proteínas. Seu problema mais conhecido é a obtenção do sequenciamento completo do genoma dos organismos. O genoma contém a informação necessária para a existência dos seres vivos, controlando processos vitais, dentre eles a síntese protéica.

A principal dificuldade no estudo de um genoma é a quantidade de dados envolvida. A informação básica que se deseja extrair do DNA é a seqüência de pares de base, processo conhecido como sequenciamento. Considere por exemplo, o genoma humano (o organismo humano possui 23 pares de cromossomos). Um cromossomo humano tem aproximadamente 10^8 pares de base (unidade de medida do DNA). Por outro lado, a maior seqüência que um laboratório pode sequenciar é de aproximadamente 700 pares de base. Desta forma, observe-se que existe uma diferença de aproximadamente 10^5 entre a capacidade atual dos laboratórios e o tamanho de um cromossomo, aumentando ainda mais, quando comparada ao genoma completo. Esta diferença é o cerne da maioria dos problemas encontrados na biologia molecular (em sua maioria problemas combinatórios). Portanto, a biologia molecular sozinha não seria capaz de lidar com os desafios do sequenciamento genético, na verdade, tais propósitos somente foram possíveis graças aos avanços conseguidos na ciência da computação. Esta colaboração tornou-se tão importante que deu origem a uma nova área, conhecida como biologia computacional [Waterman, 1995], [Gusfield, 1997], [Pevzner, 2000], [Meidanis & Setúbal, 1997], que se dedica ao estudo de problemas de biologia

molecular (DNA e proteínas), e como a computação pode auxiliar na solução destes problemas.

Na verdade, em relação ao papel da computação na biologia molecular alguns autores [Lancia, 2004] dividem os problemas de biologia molecular em duas áreas principais: a primeira seria a bioinformática, que estuda os problemas de armazenamento, organização e distribuição da grande quantidade dos dados genômicos hoje existentes, e a segunda, a biologia computacional, que estuda os problemas de interpretação e análise dos dados genômicos. Seguindo essa definição, serão tratados aqui os problemas voltados para biologia computacional.

A computação possibilita também que os genomas seqüenciados, seqüência de proteínas, dentre outras informações, sejam armazenados em grandes bancos de dados (*GenBank* (http://www.ncbi.nm.nih.gov/) mantido pela *National Center for Biotechnology Information* (NCBI), EMBL (http://www.embl-heidelberg.de/), PIR (http://www.gbd.org/ e http://www.mips.biochem.mpg.de/) , PDB (http://www.pdb.bnl.gov/), etc), em sua maioria disponíveis na internet para pesquisa, possibilitando o acesso de pesquisadores das mais diversas áreas de atuação, auxiliando portanto, no diagnóstico e tratamento de doenças, projetos de novas drogas farmacêuticas, prevenção de pragas, etc.

A seguir serão apresentadas algumas definições importantes da biologia molecular [Brown, 1999], [Cantor & Smith, 2000] para melhor compreensão dos próximos capítulos deste trabalho.

1.2 Ácidos nucléicos (DNA e RNA)

Ácidos nucléicos são macromoléculas que armazenam as informações relativas ao desenvolvimento e divisão das células, as quais formam os organismos vivos. Os ácidos nucléicos também são responsáveis pela manutenção dos organismos em toda sua vida. Na natureza existem dois tipos de ácidos nucléicos: DNA (ácido desoxirribonucléico) e RNA (ácido ribonucléico).

A estrutura primária do ácido nucléico pode ser vista como uma cadeia linear composta de simples unidades químicas chamadas bases nitrogenadas, são elas: Adenina (A), Citosina (C), Guanina (G), Timina (T) e Uracila (U). No DNA são encontradas as bases nitrogenadas A, G, C e T; e no RNA são encontradas as bases A, C, G e U.

1.2.1 DNA

O DNA é formado por uma dupla fita de cadeias de nucleotídeos que formam uma estrutura helicoidal [Brown, 1999], [Cantor & Smith, 2000]. Cada nucleotídeo é formado por uma molécula de açúcar, um fosfato e uma base nitrogenada. Nas moléculas de açúcar são encontrados cinco átomos de carbono, que são enumerados de 1' a 5' (vide Figura 1.1). Estes carbonos são importantes para formação de uma fita do DNA.

Em uma fita de nucleotídeos, estes são ligados pelo carbono 3' de um nucleotídeo e o carbono 5' do nucleotídeo vizinho. Por essa razão uma fita de nucleotídeos segue uma orientação, convencionalmente definida no direcionamento do carbono 5' para o carbono 3' entre nucleotídeos vizinhos (vide Figura 1.1). Note também, que as duas fitas tem orientações opostas.

Ligado ao carbono 1' está a base nitrogenada (A, C, G ou T), que é responsável pela ligação entre as duas fitas, formando assim, a dupla fita do DNA. Cada base nitrogenada de uma fita está ligada à outra base da fita complementar. Nota-se que a base A está sempre ligada à base T, e a base C está sempre ligada à base G. Estes pares de bases são complementares, e são utilizados como unidade de medida de comprimento de uma molécula de DNA, denotado por bp (pares de bases).

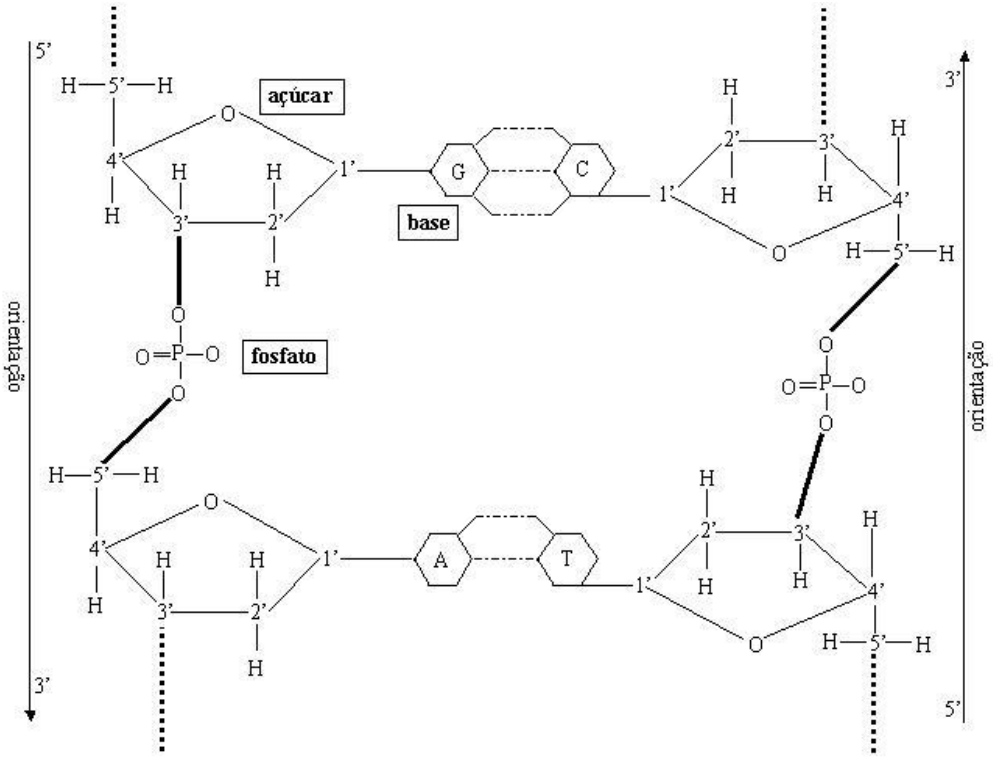


Figura 1.1: DNA

Convencionalmente, o DNA é representado como uma seqüência de letras (A, C, G, T), onde cada letra representa uma base. A Figura 1.2 apresenta um exemplo de uma dupla fita de uma seqüência de DNA, colocando uma fita em cima da outra. Observe que a figura mostra a paridade entre as bases (A com T e C com G). Nota-se, na figura, que o fim 3' de uma fita corresponde ao fim 5' da outra fita. Esta propriedade é mais conhecida como antiparalelo. A conseqüência fundamental desta estrutura é o fato de que, dada uma fita, pode-se inferir a outra fita correspondente (fita complementar). A operação que possibilita a obtenção da fita correspondente é chamada complementar reversa. Por exemplo, dada uma fita s = AGACGT, pode-se obter sua complementar reversa da seguinte forma: primeiramente reverte-se s obtendo s' = TGCAGA, posteriormente substitui-se cada base com sua base complementar, obtendo $\bar{s}$ = ACGTCT.

5’ ... TACTGAA ... 3’

3’ ... ATGACTT ... 5’

Figura 1.2: Uma dupla fita de seqüência de DNA
representada por uma seqüência de letras

1.2.2 RNA

A molécula de RNA tem apenas uma fita, e é bem curta se comparada à fita de DNA. Existem três tipos de moléculas de RNA: RNA mensageiro (mRNA), RNA transportador (tRNA) e RNA ribossomal (RNA). O RNA também tem função importante na síntese de proteínas.

1.3 Gene, cromossomo e genoma

O genoma encontra-se no interior das células dos organismos, armazenado em estruturas chamadas cromossomos. Nos cromossomos localizam-se os genes que são responsáveis pela síntese protéica. Os genes podem ser considerados como segmentos contíguos e discretos de moléculas de DNA, onde são armazenadas as informações genéticas. A informação contida em cada gene é responsável pela construção de uma proteína que tem uma função específica num organismo vivo.

Os organismos vivos podem ser divididos em dois grupos:

- Eucariotos: organismos cujas células possuem núcleo, onde se encontram os cromossomos. Incluem-se neste grupo os animais, plantas, fungos e protozoários.
- Procariotos: organismos cujas células não possuem núcleo (bactérias)

Uma outra diferença entre estas duas divisões está relacionada ao número de cromossomos presentes em cada célula. Uma célula de procarioto em geral possui apenas um cromossomo, enquanto uma célula eucariótica, tem um número fixo de cromossomos que varia de espécie para espécie (por exemplo, a célula humana possui 23 pares de cromossomos).

Nos procariotos todo o trecho de um gene codifica a proteína. Entretanto, nos eucariotos, um gene é composto de subtrechos conhecidos como *introns* e *exons*. No processo de construção de uma proteína, apenas os *exons* são utilizados para codificá-la. No segmento de *exons* de um gene, cada tripla de nucleotídeos forma um *codon* que codificará um aminoácido que formará uma proteína. Apesar de existirem 64 combinações entre triplas de nucleotídeos (4^3 combinações), apenas 20 aminoácidos são codificados (vide Figura 1.3). Isso ocorre, devido a um aminoácido poder ser representado por mais de uma tripla ou uma tripla não representar nenhum aminoácido (vide Figura 1.3, a tripla ATC não codifica nenhum aminoácido). Assim, a seqüência gerada de aminoácidos corresponde a uma proteína.

2ª posição da tripla

1ª posição da tripla	A		G		T		C		3ª posição da tripla
A	Fenilalanina	(AAA)	Serina	(AGA)	Tirosina	(ATA)	Cisteína	(ACA)	A
	Fenilalanina	(AAG)	Serina	(AGG)	Tirosina	(ATG)	Cisteína	(ACG)	G
	Leucina	(AAT)	Serina	(AGT)	*Sem sentido*	(ATT)	*Sem sentido*	(ACT)	T
	Leucina	(AAC)	Serina	(AGC)	*Sem sentido*	(ATC)	Triptofano	(ACC)	C
G	Leucina	(GAA)	Prolina	(GGA)	Histidina	(GTA)	Arginina	(GCA)	A
	Leucina	(GAG)	Prolina	(GGG)	Histidina	(GTG)	Arginina	(GCG)	G
	Leucina	(GAT)	Prolina	(GGT)	Glutamina	(GTT)	Arginina	(GCT)	T
	Leucina	(GAC)	Prolina	(GGC)	Glutamina	(GTC)	Arginina	(GCC)	C
T	Isoleucina	(TAA)	Treonina	(TGA)	Aspargina	(TTA)	Serina	(TCA)	A
	Isoleucina	(TAG)	Treonina	(TGG)	Aspargina	(TTG)	Serina	(TCG)	G
	Isoleucina	(TAT)	Treonina	(TGT)	Lisina	(TTT)	Arginina	(TCT)	T
	Metionina	(TAC)	Treonina	(TGC)	Lisina	(TTC)	Arginina	(TCC)	C
C	Valina	(CAA)	Alanina	(CGA)	Aspartato	(CTA)	Glicina	(CCA)	A
	Valina	(CAG)	Alanina	(CGG)	Aspartato	(CTG)	Glicina	(CCG)	G
	Valina	(CAT)	Alanina	(CGT)	Glutamato	(CTT)	Glicina	(CCT)	T
	Valina	(CAC)	Alanina	(CGC)	Glutamato	(CTC)	Glicina	(CCC)	C

Figura 1.3: Tabela do Código Genético

1.4 Proteínas

As proteínas são macromoléculas orgânicas que desempenham muitas funções num organismo vivo. As proteínas mais importantes são as enzimas que catalizam (aceleram) os processos químicos indispensáveis à vida dos organismos.

1.5 Mutação

A mutação é definida como uma alteração na seqüência de nucleotídeos no DNA, causada por uma falha no processo de replicação. Essa falha no processo de replicação pode ser causada por exposição à radiação ultravioleta, ou por outras condições ambientais. Existem dois diferentes níveis de mutação. Um ao nível de gene, e outro ao nível de cromossomo. Neste último, segmentos do DNA podem ser trocados no mesmo cromossomo ou entre cromossomos.

Na mutação ao nível de gene, podem ocorrer:

- Substituição: A alteração de um nucleotídeo na seqüência de DNA.
- Inserção: A inserção de um ou mais nucleotídeos na seqüência de DNA.
- Remoção: A remoção de um ou mais nucleotídeos na seqüência de DNA.

Este tipo de mutação pode ser dividido de acordo com sua influência na proteína resultante:

- *Missense*: A mutação altera o *codon*. Este *codon* alterado codifica um aminoácido diferente.
- *Silent*: A mutação não altera o *codon*. Essa mutação não acarreta nenhuma alteração na codificação do aminoácido.

No entanto, deve ser mencionado que apesar da mutação alterar um aminoácido, não necessariamente afetará a função da proteína. Isso porque a similaridade química entre diferentes aminoácidos pode resultar em um pequeno ou nenhum impacto na estrutura final da proteína, preservando assim sua função.

As mutações têm uma importância relevante na biologia molecular. Dentre outras coisas, criam novas espécies e adaptam as espécies já existentes para as alterações das condições ambientais.

1.6 Mapeamento e Sequenciamento de DNA

Um intenso esforço tem sido feito por vários cientistas em todo mundo para sequenciar o DNA dos diferentes organismos vivos existentes na natureza, processo conhecido como Projeto Genoma. Cada organismo tem seu Projeto Genoma, que pode ser desenvolvido por várias comunidades científicas e estar espalhada em diferentes localizações no mundo. Este trabalho tem revolucionado a pesquisa médica e biológica e de outras áreas afins.

Uma importante parte deste processo consiste no mapeamento físico. O mapeamento físico mostra localizações e estima distâncias entre marcações ao longo do genoma, cromossomo, ou mesmo uma grande cadeia do genoma. Estas marcações podem ser genes ou simplesmente subseqüências arbitrárias do DNA.

O mapeamento do genoma é uma ferramenta essencial para encontrar novos genes. A construção do mapeamento físico de um DNA pode ser feita atualmente por diferentes técnicas; utilizando enzimas de restrição, hibridização, dentre outras. Em particular trabalha-se com pedaços do DNA bem menores que um cromossomo, mas bem maiores à uma seqüência que pode ser seqüenciada diretamente pelos laboratórios.

Uma outra parte do Projeto Genoma é o sequenciamento, que deseja encontrar a seqüência completa de nucleotídeos do DNA (uma visão mais detalhada do DNA).

Apesar dos avanços tecnológicos na manipulação de moléculas de DNA, existe ainda um limitante nos procedimentos laboratoriais para leitura de uma seqüência de DNA, restringindo-se à leitura de uma seqüência de aproximadamente 700 bp. Desta forma, para que os processos de mapeamento e sequenciamento sejam realizados, a seqüência de DNA é clivada (cortada) em grandes cadeias, estas cadeias são fragmentadas em cadeias ainda menores, até que alcancem o tamanho adequado para leitura nos laboratórios (com tamanhos de aproximadamente 700 bp). Após o estudo destas pequenas cadeias, inicia-se o processo inverso de remontagem das cadeias, até alcançar a cadeia original (DNA completo). É neste processo de remontagem que se encontram os problemas computacionais mais importantes.

Nota-se que existem duas visões para o trabalho de leitura e estudo das seqüências de DNA. O mapeamento físico, onde se tem uma visão mais macro da cadeia de DNA,

trabalhando com grandes blocos de DNA; e o sequenciamento com uma visão micro, onde se têm pequenos fragmentos de DNA que podem ser lidos pelos laboratórios.

Para que o mapeamento e sequenciamento possam ser realizados, muitas outras técnicas de laboratório são necessárias, tais como: fragmentação, clonagem, medição de fragmentos, etc, que serão discutidas a seguir.

1.7 Fragmentação

Atualmente, existem duas técnicas básicas para clivar uma molécula de DNA: o método que utiliza enzimas de restrição e o método do canhão (*shotgun*).

No método que utiliza enzimas de restrição, as enzimas clivam o DNA em segmentos específicos chamados de sítios de restrição. Os sítios de restrição têm sua seqüência conhecida e possuem tamanhos curtos (4, 6 ou 8 bp). Cada enzima de restrição tem seu sítio de restrição específico. Quando uma enzima de restrição é aplicada em uma molécula de DNA, esta é clivada em todas as ocorrências do sítio de restrição específico à enzima de restrição aplicada. Existem centenas de enzimas que reconhecem, respectivamente, diversos sítios de restrição.

Um dos problemas encontrados na utilização desta técnica é que podem ser gerados fragmentos excessivamente grandes. Por outro lado, podem ser gerados fragmentos muito pequenos, que não são úteis nos processos que necessitam utilizar a técnica de fragmentação.

No método do canhão, uma solução contendo uma seqüência de DNA é submetida a elevadas taxas de vibração, fazendo com que as moléculas sejam clivadas aleatoriamente em diferentes pontos. O processo de vibração pode ser acionado com ar ou com ultra-som, respectivamente chamados de nebulização e sonificação.

Após a clivagem dos fragmentos, os tamanhos dos fragmentos podem ser calculados através da técnica conhecida como eletroforese em gel. Neste processo, os fragmentos de DNA são colocados num gel, onde as moléculas do gel, quando resfriadas, formam uma matriz de poros. Quanto mais concentrada a solução do gel, mais a matriz se concentra, e os poros se tornam menores. Desta forma, aplicando-se uma corrente elétrica nesta solução, as

moléculas de DNA carregadas negativamente se movem para o pólo positivo do campo elétrico. Como as moléculas de DNA se movem pelos poros formados no gel, as moléculas mais longas encontram maior resistência, movendo-se mais vagarosamente. Conseqüentemente, os fragmentos menores movem-se mais rapidamente. Desta forma, após um certo tempo, pode-se separar as moléculas de DNA pelos seus tamanhos.

1.8 Clonagem

Para fazer experimentos (estudos) com DNA, é necessária uma quantidade razoável do mesmo material, a fim de obter uma maior confiança nos resultados dos experimentos. Por isso, a necessidade de se clonar várias vezes as seqüências de DNA.

Uma das maneiras de clonar um DNA é usar uma célula hospedeira (vírus e bactéria). Nessa abordagem, uma seqüência de DNA fonte (chamaremos neste trabalho de inserto), que se deseja clonar, é introduzida no DNA da célula hospedeira, que ao multiplicar-se, replica seu material genético para seus descendentes junto com a seqüência de DNA inserida. Assim, após um determinado período de incubação, uma quantidade celular exponencial é produzida, com uma cópia da seqüência de DNA fonte. As cópias da seqüência de DNA são retiradas das células pelo processo de purificação.

A molécula de DNA que recebe o inserto é chamada de vetor. O DNA resultante da combinação entre o vetor e o inserto é chamado de DNA recombinante. Seguem abaixo, alguns dos principais vetores usados na clonagem de DNA.

- Plasmídeos: Plasmídeos são moléculas de DNA circulares autônomas encontradas em bactérias e leveduras. Seu comprimento em geral varia entre 1 e 10 kb (1000 pares de base). Os vetores de plasmídeos têm em geral 3kb de comprimento. Desta forma, isso implica na limitação dos tamanhos dos insertos.
- Bacteriófagos: Bacteriófagos são vírus que infectam bactérias. Os vírus possuem uma estrutura simples, composta em geral de uma molécula de DNA ou RNA envolvida por uma cápsula formada de proteínas. Ao infectar uma célula, o vírus pode exibir um comportamento ativo, do qual são produzidas cópias do DNA viral. Ou podem

assumir um comportamento passivo, ficando incubado na célula até que eventualmente assumam seu estado ativo. Os bacteriófagos, também conhecidos como fagos, tem seu comportamento e estruturas moleculares bem conhecidos. Um exemplo desse tipo de vetor é o fago λ. Esse fago possui uma cabeça em forma hexagonal que envolve o DNA genômico viral de aproximadamente 50 kb. Para ser usado como vetor de clonagem, o DNA do fago λ foi modificado, sendo removido o gene que faz com que o vírus tenha o comportamento passivo (outros segmentos também foram removidos), desta forma o vírus terá um comportamento sempre ativo. Este vetor permite um inserto de no máximo 25 kb.

- Cosmídeos: O vetor fabricado com elementos do fago λ e do plasmídeo é chamado de cosmídeo. Este vetor permite inserto de 35 a 45 kb.
- Outros vetores: Existem outros vetores que visam aceitar insertos mais longos, como: vetores de YACs, do bacteriófago P1, de BACs e de PACs.

Capítulo 2

Problemas Combinatórios em Biologia Molecular

2.1 Introdução

Muitos dos problemas relevantes na biologia molecular tem natureza altamente combinatória e envolvem uma grande quantidade de dados e informação, sendo portanto, de fundamental importância o desenvolvimento de técnicas e estratégias eficientes para sua resolução. Dentre as áreas da ciência da computação, a otimização combinatória é uma parceira importante para auxiliar a biologia molecular na solução de seus problemas.

Neste capítulo, serão apresentados alguns dos principais problemas de biologia molecular que estão relacionados à área de otimização combinatória. Nas Seções 2.2 e 2.3, são apresentados problemas de comparação de seqüências, de duas seqüências e múltiplas seqüências respectivamente. Nas Seções 2.4 a 2.6 são apresentadas algumas das técnicas de mapeamento de DNA que têm como função o sequenciamento de grandes seqüências. Sobre este tema serão citadas nas Seções 2.4 e 2.5 as técnicas de mapeamento utilizando enzimas de restrição e na Seção 2.6 apresentamos o problema de mapeamento por hibridização. Na Seção 2.7, abordamos uma das técnicas de remontagem de fragmentos, que trata do problema de sequenciamento de DNA e finalmente, na Seção 2.8 discute-se sobre o problema de rearranjo genômico.

2.2 Comparação e Alinhamento de Seqüências

A comparação de seqüências é a operação primitiva mais importante dentro da biologia molecular computacional [Waterman, 1995], [Gusfield, 1997], [Pevzner, 2000], [Meidanis &

Setúbal, 1997]. A grosso modo, esta operação consiste em encontrar que partes das seqüências são parecidas e que partes são diferentes.

Existem vários problemas onde esta técnica pode ser aplicada. Segue abaixo, alguns exemplos apresentados em Meidanis e Setúbal [Meidanis & Setúbal, 1997]:

1) Sejam duas seqüências do mesmo alfabeto, ambas com o mesmo tamanho (dezenas de milhares de caracteres). Sabe-se que as seqüências são similares, com poucas diferenças isoladas. A média freqüente dessas diferenças é baixa (uma em cada 100). Deseja-se encontrar os locais onde estas diferenças ocorrem. Este problema aparece quando o mesmo gene é seqüenciado por diferentes laboratórios e deseja-se comparar os resultados.
2) Sejam duas seqüências do mesmo alfabeto, com poucas centenas de caracteres cada. Deseja-se saber se existe um prefixo de uma cadeia o qual é similar a um sufixo de outra cadeia. Este problema e o próximo (3), aparecem no contexto do sequenciamento de fragmentos, em programas que ajudam no sequenciamento de larga escala de fragmentos de DNA.
3) Considere novamente o mesmo problema descrito em (2), desta vez, várias centenas de seqüências devem ser comparadas. Além disso, sabe-se que a grande maioria dos pares de seqüência não se relaciona, isto é, não tem um grau requerido de similaridade.
4) Sejam duas seqüências do mesmo alfabeto com poucas centenas de caracteres cada. Deseja-se saber se existem duas *sub-cadeias*, uma em cada seqüência, que são similares. Este problema e o próximo (5), aparecem no contexto de busca por locais similares usando grandes base de dados de bio-seqüências.
5) Dado o mesmo problema (4), mas ao invés de duas seqüências, uma seqüência deve ser comparada com milhares de outras.

Para comparação de seqüências, duas etapas básicas são fundamentais:

1) Alinhamento: faz o emparelhamento das seqüências, de modo que fique clara sua correspondência. No alinhamento são adicionados espaços em localizações arbitrárias nas seqüências para que elas fiquem com o mesmo tamanho (vide Exemplo 2.1).

Dadas duas seqüências $s=s_1...s_m$ e $t=t_1...t_n$, com símbolos pertencentes ao mesmo alfabeto Σ, onde $m, n \geq 0$. Um alinhamento de s e t representa um mapeamento de s e t nas seqüências s' e t', respectivamente, cujos símbolos pertencem ao alfabeto $\Sigma' = \Sigma \cup \{\text{'-'}\}$, onde o símbolo '-' é chamado de espaço, tal que:

- $|s'| = |t'| = l$, onde $|x|$ representa o tamanho da seqüência x;
- A remoção dos espaços em s' e t' levam às seqüências originais s e t, respectivamente;
- Não é permitido um espaço ('-') em uma mesma posição de s' e t'.

2) Similaridade: função que mede o quanto as seqüências são similares. Uma das formas mais utilizadas para calcular a similaridade entre as seqüências de DNA ou RNA, é atribuindo valores na comparação de cada posição em todo comprimento das seqüências. No caso das proteínas são utilizadas as matrizes PAM [George et al., 1990], [Dayhoff et al., 1978]. A atribuição de valores para seqüências de DNA ou RNA pode ser dada da seguinte forma (vide Exemplo 2.1):

- Se os caracteres forem iguais soma-se +1;
- Se os caracteres forem diferentes soma-se –1;
- Se tiver espaço soma-se –2.

O cálculo da similaridade é dado por:

$$sim = \sum_{i=1}^{l} \sigma\left(s'_i, t'_i\right), \tag{2.1}$$

onde $\sigma : \Sigma' x \Sigma' \rightarrow \Re$ é uma função simétrica tal que $\sigma(a,b)$ denota o valor da comparação, como definido anteriormente, entre o símbolo a e o símbolo b, $\forall$ a e $b \in \Sigma'$.

Definição 2.1: Dadas duas seqüências *s* e *t*, encontrar o alinhamento entre elas, de forma a maximizar a função de similaridade *sim* (2.1).

Exemplo 2.1: Na comparação de seqüências buscamos o alinhamento de similaridade máxima, ou seja, que torne as seqüências as mais parecidas possíveis. Veja a seguir, o alinhamento com similaridade máxima entre as seqüências *s* = GACGGATTAG e *t* = GATCGGAATAG.

s'	G	A	--	C	G	G	A	T	T	A	G	} Alinhamento
t'	G	A	T	C	G	G	A	A	T	A	G	
	+1	+1	-2	+1	+1	+1	+1	-1	+1	+1	+1	Similaridade = 6

2.2.1 Aplicações da comparação de seqüências

O primeiro sucesso na comparação de seqüências foi a descoberta da ligação entre os genes que causam câncer e um gene envolvido no crescimento e desenvolvimento celular [Doolittle *et al*., 1983], [Waterfield *et al*., 1983]. *Oncogenes* são genes do vírus que causam um tipo de câncer nas células infectadas. O Oncogene *v-sys* do *simian sarcoma vírus* causa um descontrole no crescimento da célula e leva os macacos ao câncer. Muito parecido com *growth factor* PDGF, uma proteína que estimula o crescimento da célula. Quando estes genes foram comparados, significantes similaridades foram encontradas. Esta descoberta confirmou uma conjectura de que o câncer pode ser causado por um gene normal de crescimento, onde o tempo de crescimento é alterado para um tempo errôneo.

Outra aplicação para comparação de seqüências é o estudo da evolução de espécies. A evolução dos organismos conserva padrões de seqüências por várias gerações. Quando uma nova seqüência tem uma grande similaridade com alguma seqüência já cadastrada na base de dados de seqüências, existe uma grande chance das seqüências serem homólogas, ou seja, oriundas de um ancestral comum. Assim, novas e úteis hipóteses biológicas são formadas com a comparação de seqüências.

2.2.2 Discussão

A programação dinâmica resolve o problema de comparação de duas seqüências em tempo polinomial. Needleman e Wunsch [Needleman & Wunsch, 1970] foram os pioneiros na utilização da programação dinâmica nos problemas de comparação de seqüências. Eles apresentaram um algoritmo que maximizava o número de coincidências subtraindo o número de inserções e remoções, conhecido como critério de similaridade máxima. Associando apenas pesos não negativos para substituições, inserções e remoções no alinhamento, Sellers [Sellers, 1974] apresentou um algoritmo de programação dinâmica, onde o alinhamento ótimo corresponde a menor distância (soma de todos os pesos) do alinhamento, conhecido como critério de distância mínima. Uma referência clássica na comparação de seqüências é o livro de Sankoff e Kruskal [Sankoff & Kruskal, 1983]. Uma boa revisão sobre alinhamento ótimo, similaridade, distâncias e algoritmos relacionados ao assunto podem ser encontrados em [Waterman, 1989]. Vários métodos utilizando a programação dinâmica podem ser encontrados em [Pearson & Miller, 1992]. Outra referência muito utilizada é o livro de Heijne [Heijne, 1987] para estudo da comparação de seqüências.

2.3 Alinhamento de Múltiplas Seqüências

No alinhamento múltiplo, comparam-se mais de duas seqüências. Utilizado em diferentes aplicações na biologia molecular, como: encontrar informações sobre função e estrutura das moléculas, estimar distâncias evolucionárias entre as espécies, dentre outras aplicações. Muitas vezes a comparação de duas seqüências não mostra padrões biológicos importantes que somente são mostradas na comparação de múltiplas seqüências.

Existem várias formas de se calcular a similaridade entre as seqüências no alinhamento múltiplo. Uma delas seria calcular para cada coluna (de todas as seqüências) um valor de similaridade, seguindo algum critério, e no final teríamos uma somatória destes valores, onde o alinhamento ótimo seria definido como o alinhamento que minimize esta somatória final. A soma de pares (*Sum-of-Pairs*) é um exemplo (desta forma de cálculo de similaridade)

bastante comentado na literatura, onde para um alinhamento múltiplo $A = (a_{ih})$, o alinhamento A_{ij} entre duas seqüências a_i e a_j pode ser dada por:

$$s(A_{ij}) = \sum_{h=1}^{m} d(a_{ih}, a_{jh}),$$

onde d representa a distância entre símbolos do alfabeto $\Sigma' = \Sigma \cup \{\text{'-'}\}$ e m o tamanho final das seqüências (após o alinhamento). A soma de pares para o alinhamento A será dada por $\sum_{i,j} s(A_{ij})$.

Definição 2.2: O alinhamento múltiplo das seqüências s_1, s_2, ..., s_n será representado pelo conjunto de seqüências s'_1, s'_2, ..., s'_n (com espaços), que obtenham a maior similaridade do alinhamento múltiplo, dado que:

- $|s'_1| = |s'_2| = ... = |s'_n|$, onde $|s|$ é o tamanho de s;
- Removendo os espaços de s'_1, s'_2, ..., s'_n obtém-se respectivamente s_1, s_2, ..., s_n.

2.3.1 Discussão

A programação dinâmica resolve o problema de alinhamento múltiplo para k seqüências de tamanho n, em tempo de $O((2n)^k)$. Diferentes algoritmos de programação dinâmica foram apresentados [Sankoff, 1975], [Sankoff, 1985], [Waterman *et al*., 1976]. Entretanto, algoritmos exatos para o problema de alinhamento múltiplo para valores de k muito grande (muitas seqüências) são inviáveis [Wang & Jiang, 1994]. Wang e Jiang mostraram que o problema é NP-difícil quando utilizado a medida de soma-de-pares (*SP measure*) e o alinhamento em árvore, para resolução do problema. Kececioglu [Kececioglu, 1993] formalizou a noção de que o alinhamento múltiplo pode ser considerado como um conjunto de alinhamento de duas seqüências, conhecido como problema do peso máximo (*maximum weight trace problem*). Ele provou que o problema é NP-difícil e apresentou um algoritmo *branch-and-bound*. Feng e Doolittle [Feng & Doolittle, 1987] usam um par de seqüências

com a melhor similaridade e a partir destas duas seqüências gera-se uma nova, preservando o princípio de "*once a gap*, *always a gap*" (isso implica que no processo de alinhamento, sempre que se adiciona um *gap* na seqüência resultante também deve ser adicionado um *gap* nas seqüências já alinhadas). Como resultado, o alinhamento múltiplo de k seqüências é reduzido para o alinhamento de k-1 seqüências, e assim por diante. Muitos outros algoritmos de alinhamento múltiplos usando estratégia similar foram apresentados na literatura [Barton & Sternberg, 1987], [Taylor, 1987], [Bains, 1986], [Higgins *et al*., 1996].

Apesar do algoritmo de Feng e Doolittle [Feng & Doolittle, 1987] trabalhar bem para seqüências próximas (similares), este método não garantia performance. O primeiro algoritmo aproximativo para o problema de alinhamento múltiplo a garantir performance (com razão de aproximação 2-2/k) foi proposto por Gusfield [Gusfield, 1993].

2.4 Problema de Digestão Dupla (*Double Digest Problem (DDP)*)

Duas enzimas de restrição são aplicadas nas seqüências de DNA clivando-as em pequenos fragmentos. Os tamanhos destes fragmentos são medidos utilizando a técnica de eletroforese em gel. Conhecendo os tamanhos dos fragmentos gerados pela aplicação individual das duas enzimas *enz1* e *enz2*, e a aplicação das duas enzimas juntas, sobre estas três seqüências iguais (clonadas), o objetivo será determinar a seqüência original.

Aplicando as enzimas *enz1* e *enz2* separadamente e, *enz1* e *enz2* juntas, em três seqüências iguais clonadas, obtém-se os respectivos conjuntos:

$$A=\{\, a_i,\ 1\le i\le n \,\},\ B=\{\, b_i,\ 1\le i\le m \,\} \text{ e } C=\{\, c_i,\ 1\le i\le l \,\},$$

onde cada elemento dos conjuntos acima representa o tamanho de cada fragmento criado. O total de elementos de cada conjunto A, B e C é representado respectivamente por n, m e l.

Seja L o tamanho da seqüência original (antes da ação das enzimas de restrição) e supondo que os tamanhos dos fragmentos são medidos sem erros (caso ideal do problema), pode-se garantir que:

$$\sum_{1\le i\le n} a_i = \sum_{1\le i\le m} b_i = \sum_{1\le i\le l} c_i = L$$

Seja σ uma permutação do conjunto A e μ uma permutação do conjunto B, pode-se obter a partir destas permutações o conjunto das localizações de corte (esta definição pode ser melhor entendida no Exemplo 2.2).

$$S = \left\{ s : s = \sum_{1\le j\le r} a_\sigma(j) \quad ou \quad s = \sum_{1\le j\le t} b_\mu(j) \right\}, \quad \text{onde } 1 \le r \le n \text{ e } 1 \le t \le m, \text{ sem}$$

repetições.

Reordenando o conjunto S, obtém-se:

$$S = \{ s_j,\ 1 \le j \le l \}, \text{ onde } s_i \le s_j,\ i \le j.$$

Isto implica que $C(\sigma,\mu)$ pode ser definido da seguinte forma:

$$C(\sigma, \mu) = \{ c_j(\sigma, \mu) = s_j - s_{j-1} \}, \text{ para todo } 1 \le j \le l.$$

Definição 2.3: Dados 3 conjuntos A, B e C de inteiros, encontrar as permutações μ e σ, do conjunto A e B respectivamente, de forma que $C = C(\mu, \sigma)$.

Segue abaixo um exemplo ilustrativo para melhor compreensão (Exemplo 2.2).

Exemplo 2.2: Sejam 3 seqüências iguais (clonadas). Aplicando a enzima *enz1* numa seqüência de DNA, esta enzima cliva a seqüência em pequenos fragmentos que são medidos pelo processo eletroforese em gel resultando no conjunto de inteiros A = {3,8,6,10}. Aplicando a enzima *enz2* na segunda seqüência clonada, obtém-se o conjunto B = {4,5,11,7} e aplicando as enzimas *enz1* e *enz2* juntas na terceira e última seqüência clonada, obtém-se o conjunto C = {3,1,5,2,6,3,7} (vide Figura 2.1).

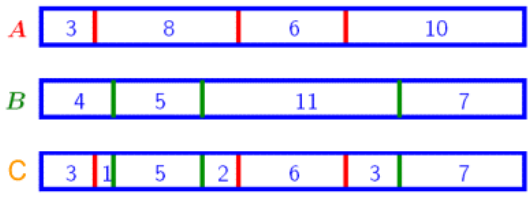


Figura 2.1: Exemplo do DDP

O objetivo é permutar o conjunto A e o conjunto B, de forma a sobrepor um conjunto no outro em diferentes ordens dos fragmentos, observando que os conjuntos estão desordenados, e a partir desta sobreposição os novos tamanhos resultem no conjunto C.

$$S = \sum_{1 \le j \le r} a_\sigma(j) \quad + \quad \sum_{1 \le j \le t} b_\mu(j)$$

$$\sum_{1 \le j \le r} a_\sigma(j) = \{3,11,17,27\}$$

$$\sum_{1 \le j \le t} b_\mu(j) = \{4,9,20,27\}$$

$$S = \{3,4,9,11,17,20,27\}$$

$$C = \{3,1,5,2,6,3,7\}$$

Desta forma, o conjunto C obtido representa uma solução para o problema.

2.4.1 Variações do Problema

Neste problema pode-se encontrar alguns erros de experimentos de laboratório que não serão modelados aqui, são eles:

- Clivagem parcial. A enzima pode falhar e não clivar a cadeia no local devido. Assim, existirão grandes fragmentos na instância do problema.
- Tamanho do fragmento. O processo de medida dos fragmentos (eletroforese em gel) não consegue medir os tamanhos dos fragmentos com exatidão, admitindo erros típicos entre 2% e 7%.

- Pequenos fragmentos podem ser perdidos.
- *Doublets*. Dois fragmentos com o mesmo tamanho podem se sobrepor, e um pode ser perdido.

2.4.2 Discussão

Muitos algoritmos têm sido propostos para este problema [Pevzner, 1992], [Pevzner, 1995], [Schmitt & Waterman, 1991], [Waterman & Griggs, 1986], dentre outros. Goldstein e Waterman [Goldstein & Waterman, 1987] provaram que este problema é NP-completo e mostraram que o número de soluções para o DDP cresce exponencialmente à medida que o número de cortes aumenta (isto é, à medida que aumenta o número de posições onde as enzimas clivam a seqüência de DNA). Em 1991, Schmitt e Waterman, mostraram que apesar do número de soluções crescer exponencialmente, como mostrado em [Goldstein e Waterman, 1987], a maioria das soluções são similares.

2.5 Problema de Digestão Parcial (*Partial Digest Problem (PDP)*)

Este problema é similar ao DDP. No entanto, utiliza-se apenas uma única enzima de restrição. Esta enzima de restrição é aplicada várias vezes em uma cadeia de DNA em períodos diferentes de tempo. O resultado dessa ação é um multi-conjunto $\binom{n}{2}$ de fragmentos (vide Figura 2.2), onde n representa o número de posições onde a enzima de restrição cliva a cadeia de DNA. Similarmente ao DDP, o propósito é encontrar a seqüência original a partir do multi-conjunto dado.

Definição 2.4: Seja a seqüência original $X = \{X_1, X_2, ..., X_n \mid X_1 < X_2 < ... < X_n\}$, onde cada X_i representa uma posição onde a enzima cliva a seqüência X, para $1 \le i \le n$. Dado o conjunto de inteiros $D = \{|X_i - X_j| \mid 1 \le i \le j \le n\}$ (resultado da aplicação da enzima de restrição na cadeia de DNA), reconstruir a série original $X_1, X_2, ..., X_n$.

Exemplo 2.3:

Entrada: D = {2, 5, 7, 7, 9, 9, 14, 14, 16, 23}

Saída: X = {0, 7, 9, 14, 23}

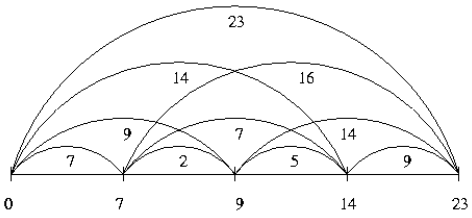


Figura 2.2: Exemplo do PDP

2.5.1 Variações do Problema

No PDP ocorrem os mesmos erros laboratoriais do problema anterior, desta forma, várias extensões do problema clássico citado anteriormente foram apresentados na literatura. Serão citadas a seguir, duas extensões na versão de otimização, buscando soluções para alguns desses erros laboratoriais [Cieliebak *et al*., 2003].

O problema de minimização do super-conjunto do PDP, modela a situação de omissão de alguns fragmentos, buscando o conjunto de pontos cujo número de fragmentos omitidos seja mínimo.

Definição 2.5: Dado um conjunto $D = \{d_1,d_2,...,d_k\}$ de k inteiros (onde alguns fragmentos foram perdidos), encontrar o menor conjunto $P = \{p_1,...,p_m\}$ de m inteiros, tal que $D \subseteq \{|p_i - p_j| \mid 1 \le i \le j \le m\}$.

O problema de maximização do sub-conjunto do PDP modela a situação de fragmentos adicionados erroneamente, buscando o conjunto de pontos cujo número de fragmentos adicionais seja mínimo, maximizando o conjunto *P* de pontos.

Definição 2.6: Dado um conjunto $D = \{d_1,d_2,...,d_k\}$ de *k* inteiros, encontrar o maior conjunto $P = \{p_1,...,p_m\}$ de *m* inteiros, tal que $D \supseteq \{|p_i - p_j| \mid 1 \le i \le j \le m\}$.

2.5.2 Discussão

Este problema também é conhecido como problema rodoviário (*turnpike problem*), onde são dadas as distâncias entre cada uma das cidades, que se encontram numa rodovia (não se conhece a ordem das cidades ao longo da rodovia), deseja-se encontrar a ordem das cidades ao longo da rodovia [Dakic, 2000]. O PDP pode ser resolvido em tempo pseudo-polinomial [Lemke & Werman, 1988], [Rosenblatt & Seymour, 1982]. Skiena *et al*. [Skiena *et al*., 1990] propuseram um algoritmo *backtracking* que tinha uma performance muito boa na prática, mas em alguns casos requeria tempo exponencial. Em 1994, Skiena e Sundaram, propuseram um algoritmo *branch-and-bound*. Entretanto, ainda não se conhece um algoritmo polinomial para este problema. Cieliebak *et al*. [Cieliebak *et al*., 2003] provaram que para o PDP considerando o caso real, onde erros são encontrados nas informações de entrada (no conjunto de fragmentos), a complexidade do problema se torna NP-difícil.

2.6 Mapeamento Físico Usando Hibridização

No mapeamento físico usando hibridização são utilizadas as informações de sobreposição entre os fragmentos. Os fragmentos são replicados usando a técnica de clonagem. Depois de copiados, os clones da seqüência de DNA original são clivados com diferentes técnicas (processo de clivagem). As informações dos clones são obtidas através de experimentos de hibridização. Nestes experimentos tenta-se verificar se pequenas seqüências, chamadas de sondas (*probes*), hibridizam o clone (isto é, se a sonda está presente no clone).

A impressão digital (*fingerprint*) de um clone é representada pelo conjunto de sondas que o hibridizam. Para dois clones que compartilham parte destas impressões digitais é muito provável que estes tenham regiões sobrepostas na cadeia de DNA original, conseqüentemente é muito provável que os fragmentos sejam adjacentes.

Dada uma biblioteca de clones *CL*, cujos clones correspondem a subintervalos de grandes fragmentos de um determinado DNA *S*, e um conjunto *P* de sondas. Cada sonda $p_j \in P$ é rotulada e testada contra os clones da biblioteca. Se um clone contém uma seqüência que corresponda a sonda testada, esta seqüência será hibridizada. O resultado destes experimentos é uma matriz $A_{clonesxsondas}$ que mostra a relação clone x sonda (vide Figura 2.3) onde $a_{ij} = 1$, se a sonda p_j hibridiza o clone c_i, e $a_{ij} = 0$ caso contrário, para $1 \leq i \leq |CL|$ e $1 \leq j \leq |P|$.

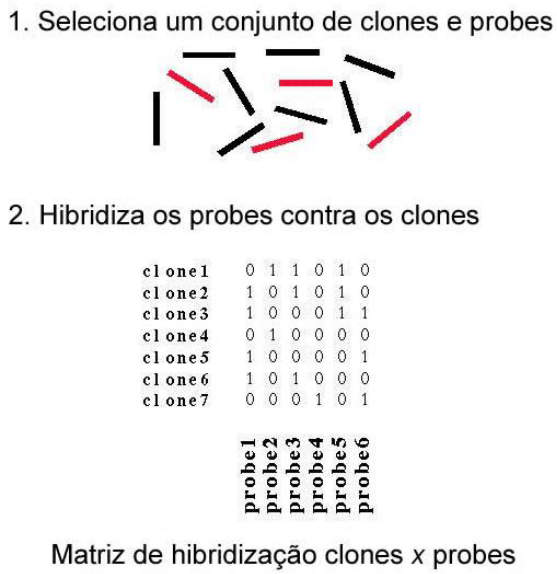


Figura 2.3: Detecção de sobreposição por hibridização

O problema consiste em encontrar a ordem das sondas *P* que correspondam a suas reais posições no DNA *S*, de forma que a ordem dos fragmentos do DNA seja encontrada, conseqüentemente, a seqüência de DNA original é encontrada. A seguir, é apresentado um modelo clássico para resolução do problema de mapeamento com hibridização sem

considerar erros de experimentos de laboratório e considerando também as seguintes premissas:

- Sondas são únicas no fragmento, em um único fragmento de DNA uma sonda aparece apenas uma vez.
- Não existem erros
- Todos "clones x sondas" são encontrados na matriz.

Definição 2.7: Dada uma matriz hibridizada, encontrar a permutação das colunas (sondas) tal que a nova matriz reordenada contenha em cada linha *i* no máximo um bloco consecutivo de uns (de sondas marcadas).

2.6.1 Variações do Problema

Considerando o problema citado anteriormente, conhecido como mapeamento com sondas únicas, pode ser resolvido em tempo polinomial. No entanto, o problema se torna NP-difícil [Golumbic *et al*., 1994], quando são considerados erros de experimentos laboratoriais, como:

- Falsos positivos, uma sonda que hibridizou onde não existe.
- Falsos negativos, uma sonda que falhou na hibridização onde deveria existir.
- Inconsistência causada por repetidas seqüências
- Clones com informações deletadas, incluídas ou alteradas.

Uma definição similar a anterior, pode ser modelada da seguinte forma na versão de otimização:

Definição 2.8: Dada uma matriz hibridizada, encontrar a permutação das colunas (sondas) tal que a nova matriz reordenada minimize, em cada linha *i*, o número de blocos consecutivos de uns (de sondas marcadas).

2.6.2 Discussão

Para a primeira definição (Definição 2.7), um algoritmo polinomial (*consecutive ones*) foi apresentado por Booth e Lueker [Booth & Lueker, 1976]. Este algoritmo roda em $O(n+m+r)$ e codifica de maneira compacta todas as soluções possíveis (onde r é o número total de 1s na matriz). Entretanto, melhorias foram apresentadas por [Hsu, 1992], [Meidanis & Munuera, 1996]. Outras referências para o mapeamento por hibridização com sondas únicas foram estudadas em [Greenberg & Instrail, 1994], [Greenberg & Instrail,1995], [Alizadeh *et al*., 1995]. A segunda referência contém uma excelente descrição de vários aspectos do problema, e vários algoritmos foram apresentados. O mapeamento por hibridização com sondas não únicas foi sugerido por Poustka *et al*. [Poustka *et al*., 1986]. Em [Mayraz & Shamir, 1999] foram apresentados algoritmos que trabalham bem com a presença de erros de hibridização. Muitas outras extensões do problema clássico são encontradas na literatura.

2.7 Problema da Menor Super-Cadeia Comum (*Shortest Common Superstring* (SCS))

A idéia do modelo da menor super-cadeia comum consiste em: dado um conjunto de cadeias, encontrar a menor cadeia possível de modo que todas as cadeias originais sejam sub-cadeias da cadeia solução.

Definição 2.9: Dado um conjunto $S = \{s_1,s_2,...,s_n\}$ de cadeias, encontrar a menor cadeia X tal que cada cadeia $s_i \in S$ seja uma sub-cadeia de X.

Veja a figura 2.4 a seguir ilustrando este método de sequenciamento de DNA.

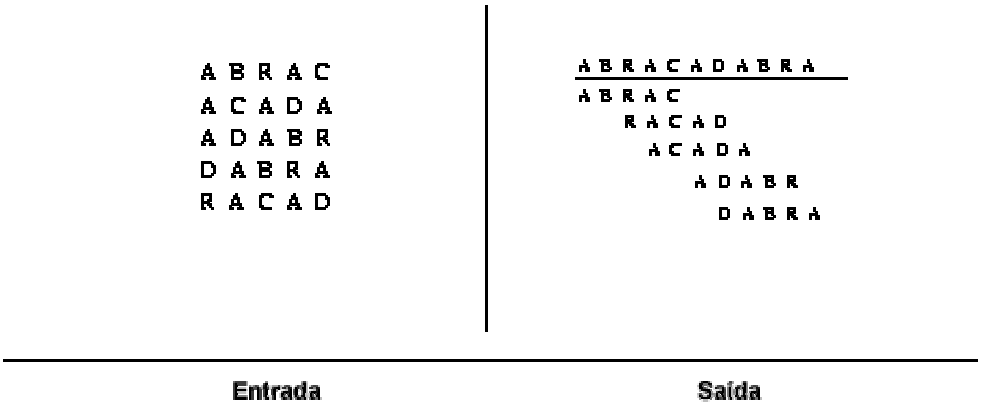


Figura 2.4: Entrada e saída para o SCS

2.7.1 Discussão

Este problema é NP-completo [Gallant *et al*., 1980]. Turner [Turner, 1989] apresentou vários algoritmos, incluindo o método guloso. O primeiro algoritmo de aproximação constante relacionado ao tamanho da menor superstring foi apresentado por Blum *et al*. [Blum *et al*., 1994], que desenvolveram um algoritmo 3-aproximado. Este trabalho se baseou na relação do SCS com os problemas do caixeiro viajante e o problema de cobertura cíclica (*cycle cover problem*) [Turner, 1989]. A partir de então, várias melhorias na razão de aproximação foram apresentadas. Em [Teng & Yao, 1993] foi apresentado uma razão de performance 2+8/9; Czumaj *et al*. [Czumaj *et al*., 1994] apresentaram uma razão de performance 2+5/6; Kosaraju *et al*. [Kosaraju *et al*., 1994] apresentaram uma razão de performance 2+50/63; e Armen e Stein apresentaram uma razão de performance 2+3/4 [Armen & Stein, 1995] e 2+2/3 [Armen & Stein, 1996]. A maioria destes algoritmos rodam no tempo $O(|S| + n^3)$, onde $|S|$ representa o tamanho da superstring e n representa o número de fragmentos.

2.8 Rearranjo de Genoma

Atualmente, uma grande quantidade de informações genéticas se encontram disponíveis para pesquisa. Desta forma, um desafio importante para biologia molecular é obter

conclusões biológicas relevantes. Uma das formas de se estudar estas informações é através da comparação de genomas, que tem como objetivo buscar semelhanças e diferenças entre os genomas dos organismos, desejando encontrar uma "distância" entre as espécies analisadas [Pevzner, 2000], [Meidanis & Setúbal, 1997].

Observa-se que da mesma forma que seqüências de um gene são alteradas (ao nível de bases) no decorrer da evolução molecular, isso também ocorre em um nível mais alto, onde grandes trechos de um cromossomo podem ser movidos ou copiados para outra localização, no mesmo cromossomo ou entre cromossomos. Esses movimentos são conhecidos como rearranjo genômico. O principal propósito no estudo do rearranjo de genomas é encontrar uma série de operações, "rearranjos", que transformem um genoma em outro. As duas principais operações estudadas no rearranjo de genomas são: a reversão e a transposição. A reversão inverte a ordem do gene (ou bloco), e a transposição corta e cola em outra posição um gene (ou bloco).

Um modelo possível para estudar a evolução dos genomas é representar os genomas como permutação de genes. O objetivo é encontrar a menor distância, ou seja, o menor número de operações, para transformar um genoma em outro.

Considere os genes representados como inteiros de 1 a n , e um genoma π, representado pela permutação de genes, onde $\pi:\{1,2,...,n\}\mapsto\{1,2,...,n\}$ pode ser descrito por $(\pi_1\pi_2...\pi_n)$, e π_i representa um gene (ou bloco), para $1 \le i \le n$. O gene π_i, para $1 \le i \le n$, pode possuir sinal positivo ou negativo para modelar a orientação do gene.

Considere a ordem e orientação dos genes em um cromossomo representada pela permutação $\pi=(\pi_1\pi_2...\pi_n)$, onde cada π_i é um inteiro com sinal, para $1 \le |\pi_i| \le n$ e $|\pi_i| \ne |\pi_j|$ onde $i \ne j$.

A função reversão $r(i,j)$, definida por dois inteiros i e j, onde $1 \le i \le j \le n$, reverte a ordem e o sinal de π_k, para $i \le k \le j$. Assim, tem-se:

$$r(i,j).\ (\pi_1 \cdots \pi_{i-1}\, \pi_i \cdots \pi_j\, \pi_{j+1} \cdots \pi_n) = (\pi_1 \cdots \pi_{i-1}\, \overline{\pi}_j \cdots \overline{\pi}_{i+1} \overline{\pi}_i\, \pi_{j+1} \cdots \pi_n)$$

onde $\overline{\pi}_k$ significa π_k com o sinal oposto.

A função transposição $t(i,j,k)$ é definida por três inteiros i, j e k, onde $1 \leq i < j \leq n+1$, e $k \notin [i,j]$. Esta função corta a porção entre as posições i e j-1(inclusive) e cola na posição k. Assim, se $i < j < k$, pode-se escrever:

$$t(i,j,k).(\pi_1 \cdots \pi_{i-1}\pi_i \cdots \pi_{j-1}\pi_j \cdots \pi_{k-1}\pi_k \cdots \pi_n) = (\pi_1 \cdots \pi_{i-1}\pi_j \cdots \pi_{k-1}\pi_i \cdots \pi_{j-1}\pi_k \cdots \pi_n)$$

Definição 2.10: Dadas duas permutações π e σ, deseja-se computar a menor série de operações (reversão e transposição) para transformar π em σ.

Assim, deseja-se encontrar $\rho_1, \rho_2, \ldots, \rho_u$, onde ρ_i é uma operação de reversão ou de transposição tal que $\rho_u.\rho_{u-1}...\rho_2.\rho_1.\pi = \sigma$ e u seja mínimo, onde u representa o número de operações (distância) de reversão e de transposição entre π e σ.

Segue abaixo, um exemplo para o cálculo da menor distâncias entre dois genomas conhecidos utilizando a operação de reversão.

Considere os genomas da alfafa e ervilha (vide Figura 2.5). Cada seta denota um bloco. Um bloco é uma seção do genoma que contém mais de um gene. A seta denota que o bloco tem orientação. A razão pelo qual cada bloco tem orientação está diretamente ligada a orientação da fita de DNA (5' → 3'). Dois blocos em diferentes genomas têm o mesmo número (label da seta) se eles são homólogos, isto é, contém os mesmos genes.

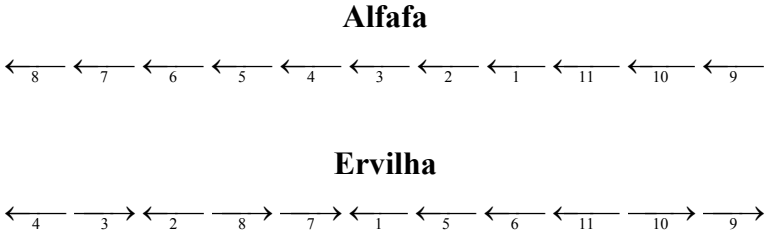


Figura 2.5: Genoma cloroplasta da alfafa (genoma inicial) e genoma cloroplasta da ervilha (genoma final)

A operação reversão inverte as setas e os blocos. O objetivo é encontrar o menor número de reversões para chegar ao genoma final (genoma da ervilha) a partir do genoma inicial (genoma da alfafa).

Uma solução possível considerando blocos orientados é dada pela Figura 2.6. A primeira linha representa o genoma da alfafa, e a última linha o genoma da ervilha. A partir da segunda linha, cada linha é obtida a partir da linha anterior aplicando-se a reversão dos blocos sublinhados.

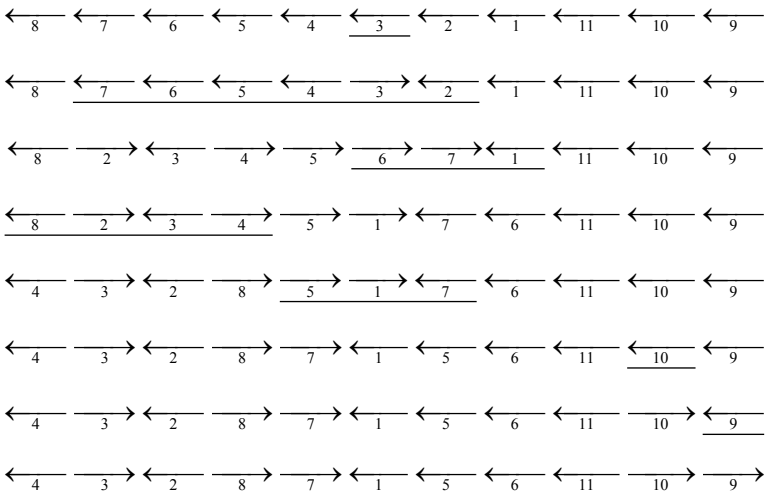


Figura 2.6: Solução para blocos orientados da figura 2.5

Em alguns casos não se tem informação suficiente da orientação dos blocos. Assim, o problema será tratado sem orientação (vide Figura 2.7).

8	7	6	5	4	3	2	1	11	10	9
8	7	1	2	3	4	5	6	11	10	9
4	3	2	1	7	8	5	6	11	10	9
4	3	2	8	7	1	5	6	11	10	9

Figura 2.7: Solução para blocos não-orientados da figura 2.6

2.8.1 Discussão

Em [Bafna & Pevzner, 1996], [Hannenhalli & Pevzner, 1999], [Kececioglu & Sankoff, 1995] foram apresentadas grandes contribuições para o assunto rearranjo de genomas. Outras contribuições importantes devem ser citadas; [Berman & Hannenhalli, 1996] que apresentaram a implementação mais eficiente para o algoritmo de reversão ordenada. Hannenhalli e Pevzner [Hannenhalli & Pevzner, 1995b] apresentaram uma solução polinomial para o problema de rearranjo de genoma envolvendo conjunto de cromossomos. Ferreti *et al*. [Ferreti *et al*., 1996] estudaram o problema da evolução inferida nas informações sobre que genes ocorrem em qual cromossomo.

Uma análise da evolução genômica foi apresentada por Sankoff [Sankoff, 1993]. Caprara *et al*. [Caprara *et al*., 1995] propuseram um algoritmo para ordenação de permutações não orientadas com bons resultados na prática. Caprara [Caprara, 1997] mostrou que para as permutações não orientadas este problema é NP-difícil. Para as permutações orientadas existem algoritmos polinomiais (citados anteriormente).

Capítulo 3

Arredondamento Randômico e o Problema da Seqüência Mais Próxima (PSMP)

3.1 Introdução

Em muitos problemas da biologia molecular deseja-se comparar e encontrar regiões comuns ou que sejam próximas a um conjunto de seqüências de DNA, RNA ou proteínas. Estes problemas são encontrados em várias aplicações como: busca de regiões conservadas em seqüências não alinhadas, identificação de drogas genéticas, formulação de sondas (*probes*) genéticas, entre outras [Hertz & Stormo, 1995], [Lawrence & Reilly, 1990], [Posfai *et al*., 1989], [Lucas *et al*., 1991], [Proutski & Holme, 1996], [Stormo, 1990], [Stormo & Hartzell, 1991], [Waterman *et al*., 1984], [Waterman & Griggs, 1986], [Waterman & Perlwitz, 1984]. No Problema da Seqüência Mais Próxima – PSMP (*Closest String Problem*) deseja-se determinar a seqüência que mais se aproxima, segundo alguma métrica, de um dado conjunto de seqüências.

O problema da seqüência mais próxima tem sido muito estudado. Frances e Litman [Frances & Litman, 1997] mostraram que o problema é NP-difícil. Berman *et al*. [Berman *et al*., 1997] apresentaram um algoritmo exato polinomial de uma versão parametrizada. Neste caso, a distância d entre a seqüência a ser determinada e o conjunto dado de seqüências é constante. Ben-Dor *et al*. [Ben-Dor *et al*., 1997] e Gasieniec *et al*. [Gasieniec *et al*., 1999] apresentaram um algoritmo aproximativo, com razão de performance próxima do valor ótimo para d suficientemente grande. Lanctot *et al*. [Lanctot *et al*., 1999] obtiveram um algoritmo $(4/3+\varepsilon)$-aproximado (para $\varepsilon > 0$). Li *et al*. [Li *et al*., 2002] apresentaram um esquema de aproximação polinomial para o problema. Finalmente, Pardalos *et al*. [Pardalos *et al*., 2004] propuseram métodos exatos baseados em *branch-and-bound* e três novas formulações de

programação linear inteira para o problema. Propuseram também uma heurística usada para limitar superiormente o valor da solução ótima do PSMP.

Neste capítulo serão apresentados primeiramente a definição do problema e os conceitos básicos necessários. Posteriormente, serão analisadas a prova de NP-completude e a utilização da técnica de arredondamento randômico (*Randomized Rounding*). Nesta técnica, basicamente, formula-se o problema original como um problema de programação linear inteira, encontra-se a solução relaxada deste modelo, que definirá probabilidades a serem utilizadas na etapa randômica para geração de uma solução aproximada.

Será apresentada também a técnica de *derandomização*, que consiste na construção de algoritmos determinísticos, a partir de probabilidades definidas pela relaxação linear. Uma atenção especial será dada ao método das probabilidades condicionais e o método dos estimadores pessimistas.

3.1.1 Definição do Problema e Conceitos Básicos

Várias medidas têm sido propostas para medir a distância (ou diferença) entre seqüências. Utilizaremos a distância de *Hamming*, denotada pela função $d_H(.)$. A distância de *Hamming* entre duas seqüências de mesmo tamanho é calculada simplesmente contando-se o número de posições correspondentes dos caracteres onde as duas seqüências diferem (uma justificativa mais técnica no uso freqüente da distância de *Hamming* para comparação de seqüências na biologia molecular, pode ser encontrada em [Lanctot *et al*., 1999]).

Considere um alfabeto finito Σ. Seja s uma seqüência finita do alfabeto Σ, onde $|s|$ denota o tamanho de s e $s[i]$ o i-ésimo caractere de s.

Considere um conjunto $S = \{s_1, s_2, ..., s_m\}$ de seqüências (todas de tamanho n), sobre um alfabeto Σ. No PSMP deseja-se encontrar uma seqüência s_H de tamanho n que minimize d onde, para cada $s_i \in S$, tenhamos $d_H(s_H, s_i) \leq d$, para i=1...m.

Formalmente, temos:

$$\min d$$
$$s.a \quad \begin{cases} d_H(s_H, s_i) \le d, & i = 1,...,m \\ s_H \in \Sigma^n \end{cases}$$

onde Σ^n representa o conjunto de todas as seqüências com n caracteres.

São apresentados alguns lemas auxiliares [Motwani & Raghavan, 1995], desigualdades de Chernoff-Hoeffding (Lemas 3.1, 3.2, 3.3 e 3.4), que serão utilizados posteriormente na análise da razão de aproximação dos algoritmos propostos na literatura.

Lema 3.1: Sejam X_1, X_2, ..., X_n, variáveis 0-1 aleatórias independentes, onde $X_i = 1$ com probabilidade p_i, $0 < p_i < 1$. Seja $X = \sum_{i=1}^{n} X_i$ e $\mu = E[X]$, então para qualquer $\delta > 0$,

(1) $$\Pr\left(X > (1+\delta)\mu\right) < \left[\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right]^{\mu},$$

(2) $$\Pr\left(X < (1-\delta)\mu\right) \le \exp\left(-\frac{1}{2}\mu\delta^2\right).$$

Do Lema 3.1, pode-se provar os seguintes resultados:

Lema 3.2: Sejam X_i, para $1 \le i \le n$, X e μ, como definidos no Lema 3.1. Então, para qualquer $0 < \varepsilon \le 1$,

(1) $$\Pr\left(X > \mu + \varepsilon n\right) < \exp\left(-\frac{1}{3}n\varepsilon^2\right),$$

(2) $$\Pr\left(X < \mu - \varepsilon n\right) \le \exp\left(-\frac{1}{2}n\varepsilon^2\right).$$

Lema 3.3: Sejam X_1, X_2, ..., X_n, variáveis aleatórias independentes, onde X_i assume valores no intervalo real $[a,b]$. Seja $X = \sum_{i=1}^{n} X_i$ e $\delta > 0$ então:

$$\Pr\left(X \geq (1+\delta)E[X]\right) \leq \exp\left(-\frac{E[X]\delta^2}{3(b-a)^2}\right)$$

Lema 3.4: Sejam X_1, X_2, ..., X_n, variáveis aleatórias independentes, onde X_i assume valores no intervalo real [a,b]. Seja $X = \sum_{i=1}^{n} X_i$ e $\delta > 0$ então:

$$\Pr\left(X - E[X] > \delta\right) \leq \exp\left(-\frac{2\delta^2}{(b-a)^2 n}\right)$$

Considere finalmente o seguinte lema auxiliar:

Lema 3.5: Sejam X_1, X_2, ..., X_n, variáveis aleatórias arbitrárias. Então, max ($E[X_1]$, ..., $E[X_n]$) $\leq E[\max(X_1, ..., X_n)]$. Equivalentemente, min ($E[X_1]$, ..., $E[X_n]$) $\leq E[\min(X_1, ..., X_n)]$.

3.2 Complexidade do PSMP

O PSMP é classificado como NP-Difícil. Em [Lanctot *et al*., 1999] foi mostrado inicialmente, que o problema 3-SAT (NP-Completo) é redutível ao Problema da Seqüência Mais Distante na versão decisão (PSMD-Decisão). Logo após, mostrou-se que o PSMD-Decisão é redutível ao PSMP-Decisão.

Para provar que um problema P_1 é redutível a um problema P_2 (representado por $P_1 \propto P_2$), basta transformar uma instância I_1 do problema P_1 em uma instância I_2 do problema P_2 em tempo polinomial, de tal maneira que, resolvendo-se P_2, resolve-se P_1 implicitamente.

Segue abaixo, a definição do problema auxiliar 3-SAT.

Definição 3.1: (Problema 3-SAT) Seja B uma expressão booleana qualquer na Forma Normal Conjuntiva - FNC, ou seja, B será formada por uma conjunção de cláusulas e uma disjunção de literais. Considere um conjunto $C = \{c_1,c_2,...,c_m\}$ de cláusulas onde $|c_i| = 3$, para

i = 1,...,m. Desejamos saber se existe ou não uma atribuição aos literais de B que a torne verdadeira.

Definição 3.2: (Problema PSMD-Decisão) Dado um conjunto $S = \{s_1,s_2,...,s_m\}$ de seqüências, todas de tamanho n, sobre um alfabeto Σ e d um inteiro positivo. Desejamos saber se existe uma seqüência $s_F \in \Sigma^n$ tal que $d_H(s_F, s_i) \geq d$, $\forall\ i \in \{1...m\}$.

A seguir, será mostrada a prova de NP-Completude do PSMD-Decisão. Esta prova será dividida em dois casos: para o alfabeto $|\Sigma| > 2$ e para o alfabeto binário ($|\Sigma| = 2$).

Proposição 3.1: O PSMD-Decisão é NP-Completo para o caso $|\Sigma| > 2$.

Prova: Primeiramente será analisado o caso particular $|\Sigma| = 3$, onde $\Sigma=\{0,1,*\}$. Logo após, será mostrado a generalização para qualquer alfabeto $|\Sigma| > 2$.

Seja dada uma instância $I_{m,n}$ do problema 3-SAT com m cláusulas $\{C_1, C_2, ..., C_m\}$ e n literais $\{l_1, l_2, ..., l_n\}$. O objetivo será transformar a instância $I_{m,n}$ numa instância do problema PSMD-Decisão (representada por $S_{m,n,d}$) de maneira que, ao resolver o PSMD-Decisão estaremos resolvendo o problema 3-SAT implicitamente.

Cada cláusula C_i (onde i=1..m) da instância $I_{m,n}$ estará associada a uma seqüência s_i de $S_{m,n,d}$, onde $|s_i| = n+2$. A seqüência s_i será codificada da seguinte forma, para j = 1..n+2, onde j representa a j-ésima posição da seqüência s_i:

$$s_i[j] = \begin{cases} 0, & se \quad l_j \in C_i \\ 1, & se \quad \bar{l}_j \in C_i \\ *, & se \quad l_j \ não \ aparece \ em \ C_i \\ *, & se \quad j = n+1, n+2 \end{cases}$$

Além das m seqüências codificadas, como definido acima, serão adicionadas mais 9 seqüências à instância $S_{m,n,d}$, que servirão para garantir que a solução gerada seja uma solução válida para os problemas 3-SAT e PSMD-Decisão.

No entanto, estas 9 seqüências adicionadas serão codificadas diferentemente. As n primeiras posições conterão sempre o caracter *, e as duas últimas posições serão diferentes para cada uma das 9 seqüências, visto que, são duas posições para combinação dos 3 caracteres {0,1,*} do alfabeto.

Agora, deseja-se mostrar que a instância $I_{m,n}$ é satisfatível se, e somente se, existir uma seqüência s_H de tamanho n+2 cuja $d_H(s_H, s_i) \geq d$, para $1 \leq i \leq m$+ 9 e $d = n$.

Primeiramente, considere a instância $I_{m,n}$ satisfatível para uma atribuição $s'_H \in \{0,1\}^n$. Define-se então, $s_H = s'_H00 = s'_H[1]s'_H[2]...s'_H[n]00$ como solução de $S_{m,n,n}$. Para as primeiras m seqüências de $S_{m,n,n}$, se s'_H satisfaz C_i, logo, os três pontos seguintes ocorrem.

- Existe no mínimo uma diferença entre as seqüências s_H e s_i, entre os caracteres 1 e 0, nas primeiras n posições.
- Existem n - 3 diferenças nas primeiras n posições, entre o caracter 0 ou 1 da seqüência s_H, com o caracter * da seqüência s_i.
- Finalmente, existem mais duas diferenças nas duas últimas posições entre as seqüências s_H e s_i.

Nas últimas 9 seqüências, como s_H não contém nenhum símbolo *, logo, $d_H(s_H,s_{m+j}) \geq n$, para todo $1 \leq j \leq 9$. Finalmente, conclui-se que $d_H(s_H, s_i) \geq n$, para $1 \leq i \leq m$+9.

Reciprocamente, se s_H contiver *, no mínimo, uma das 9 últimas seqüências invalidará o PSMD-Decisão. Esta é a razão pela qual $s'_H \in \{0,1\}^n$, induzindo uma atribuição para os literais l_j, para $j = 1..n$. Esta atribuição será satisfatória para o problema 3-SAT, desde que, no mínimo um literal de cada cláusula seja verdadeiro em função da diferença entre s'_H e as n primeiras posições de s_i, para $1 \leq i \leq m$. Desta forma, pode-se construir uma solução para o problema 3-SAT atribuindo-se verdadeiro a l_j, se $s'_H[j] = 1$, e falso caso contrário, para j=1..n.

Desta forma, pode-se garantir que a seqüência s'_H satisfaz o problema 3-SAT, se, e somente se, a seqüência s_H satisfizer o PSMD-Decisão.

Agora, considere a generalização para o alfabeto $|\Sigma| > 2$. Seja um alfabeto de tamanho p, onde $p > 2$, $(p\text{-}2)p^2$ seqüências extras deverão ser adicionadas ao conjunto de m seqüências, o qual serão grupados em p-2 grupos, onde cada grupo conterá p^2 seqüências. Em cada um dos (p-2) grupos, suas respectivas seqüências conterão nas primeiras n posições apenas um dos caracteres do alfabeto Σ-{0,1} ($|\Sigma\text{-}\{0,1\}| = p\text{-}2$), e as duas últimas posições a combinação (p^2) de 2 caracteres do alfabeto.

Considere o exemplo ilustrativo a seguir:

Exemplo 3.1: Considere a seguinte instância $I_{m,n}$ do problema 3-SAT,

$$I_{m,n} = \left\{ \left(l_1 \vee l_2 \vee l_3 \right) \wedge \left(l_1 \vee \bar{l}_2 \vee l_4 \right) \wedge \left(l_2 \vee \bar{l}_3 \vee l_4 \right) \wedge \left(\bar{l}_2 \vee \bar{l}_3 \vee \bar{l}_4 \right) \wedge \left(\bar{l}_1 \vee l_3 \vee \bar{l}_4 \right) \right\}.$$

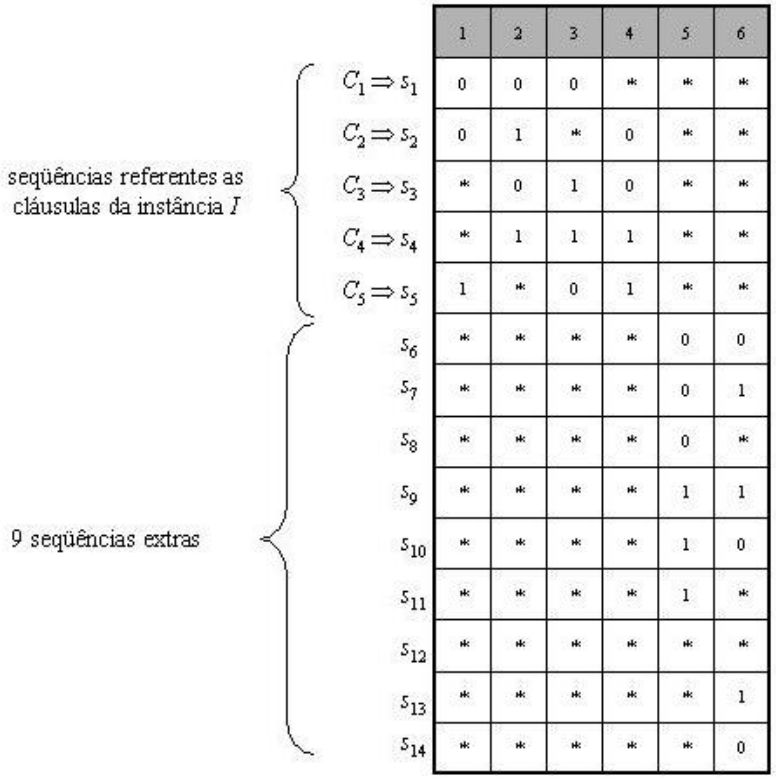
onde se destacam as seguintes cláusulas,

$$\begin{aligned} C_1 &= \left(l_1 \vee l_2 \vee l_3 \right) \\ C_2 &= \left(l_1 \vee \bar{l}_2 \vee l_4 \right) \\ C_3 &= \left(l_2 \vee \bar{l}_3 \vee l_4 \right) \\ C_4 &= \left(\bar{l}_2 \vee \bar{l}_3 \vee \bar{l}_4 \right) \\ C_5 &= \left(\bar{l}_1 \vee l_3 \vee \bar{l}_4 \right) \end{aligned}$$

Neste caso, m = 5 e n = 4. Pode-se construir a seguinte instância $S_{m,n,d}$ em tempo polinomial (vide Figura 3.1).

Pode-se observar, que dada uma seqüência $s_H = s'_H00 \in \{0,1\}^{n+2}$ que satisfaça o PSMD-Decisão, o problema 3-SAT também será satisfeito por s'_H. Exemplo: s_H = 100000.

Note, que as seqüências extras (s_6,...,s_{14}) servem para garantir que a solução $s_H[j] \in \{0,1\}$, onde j=1..$|s_H|$. Desta forma, se garante solução para os dois problemas citados. Caso contrário (se não adicionasse as seqüências extras), existiriam casos onde não se garantiria solução viável para os dois problemas. Veja contra-exemplo: s_H = 1*00**, é solução para o problema 3-SAT, entretanto, não é solução para o PSMD-Decisão.



	1	2	3	4	5	6
$C_1 \Rightarrow s_1$	0	0	0	*	*	*
$C_2 \Rightarrow s_2$	0	1	*	0	*	*
$C_3 \Rightarrow s_3$	*	0	1	0	*	*
$C_4 \Rightarrow s_4$	*	1	1	1	*	*
$C_5 \Rightarrow s_5$	1	*	0	1	*	*
s_6	*	*	*	*	0	0
s_7	*	*	*	*	0	1
s_8	*	*	*	*	0	*
s_9	*	*	*	*	1	1
s_{10}	*	*	*	*	1	0
s_{11}	*	*	*	*	1	*
s_{12}	*	*	*	*	*	*
s_{13}	*	*	*	*	*	1
s_{14}	*	*	*	*	*	0

Figura 3.1: Instância $S_{m,n,d}$.

Proposição 3.2: O PSMD-Decisão é NP-Completo para o caso do alfabeto binário $|\Sigma| = 2$.

Prova: Seja dada uma instância $I_{m,n}$ do problema 3-SAT com m cláusulas $\{C_1, C_2, ..., C_m\}$ e n literais $\{l_1, l_2, ..., l_n\}$. Para cada cláusula C_i, constrói-se uma seqüência $s_i = s_i[1]s_i[2]...s_i[n]$ de tamanho $2n \in \{00,01,11\}^n$, onde $s_i[j]$ segue o formato definido abaixo, para i=1..m e j=1..n:

$$s_i[j] = \begin{cases} 00, & se \quad l_j \in C_i \\ 11, & se \quad \bar{l}_j \in C_i \\ 01, & se \quad l_j \text{ não aparece em } C_i \end{cases}$$

Seja a função $p(x,i)$ representada pela seqüência $(10)^{i-1}x(10)^{n-i}$, e $P_n(x)$ o conjunto de n seqüências, onde $\{p(x,i) \mid 1 \le i \le n\}$. Da mesma forma, seja a função $q(x,i)$ representada pela

seqüência $(01)^{i-1}x(01)^{n-i}$, e $Q_n(x)$ o conjunto de n seqüências, onde $\{q(x,i) \mid 1 \leq i \leq n\}$. A instância $S_{m,n,d}$ do PSMD-Decisão será representada da seguinte forma:

$$S_{m,n,d} = P_n(00) \cup P_n(11) \cup P_n(01) \cup \{(10)^n\} \cup Q_n(00) \cup Q_n(11) \cup Q_n(10) \cup \{(01)^n\} \cup \{s_i \mid 1 \leq i \leq m\}$$

Claramente, pode ser observado que a instância $S_{m,n,d}$ é computada em tempo polinomial, e igualmente a Proposição 3.1, o propósito da inclusão de vários conjuntos de seqüências extras servem para forçar uma solução $s_H \in \{00,11\}^n$.

Agora, será mostrado que a instância $I_{m,n}$ é satisfeita se, e somente se, existir uma seqüência s_H de tamanho $2n$ que tenha no mínimo n-1 diferenças para cada seqüência do conjunto $S_{m,n,d}$.

Primeiramente, assume-se que existe uma solução s_H para o problema 3-SAT. Assim, uma seqüência $s_H = s_H[1]s_H[2]...s_H[n]$, onde $s_H \in \{00,11\}^n$ pode ser construída de forma que exista no mínimo n-1 diferenças entre s_H e cada seqüência $s_i \in S_{m,n,d}$. Segue abaixo a construção da seqüência s_H, para $j = 1..n$.

$$s_H[j] = \begin{cases} 11, & se \quad l_j \ é \ Verdadeiro \\ 00, & se \quad l_j \ é \ Falso \end{cases}$$

Desde que $s_H \in \{00,11\}^n$, a distância para todas as seqüências do conjunto $S_{m,n,d}$ será no mínimo n-1 ($d = n$-1). Caso contrário, s_H não satisfaz o PSMD-Decisão.

Portanto, seja $s_H \in \{00,11\}^n$ uma seqüência que satisfaz o PSMD-Decisão, também será uma solução do problema 3-SAT, visto que para uma diferença entre as subseqüências 00 e 11 ou 11 e 00, entre s_H e s_i, corresponde a um literal verdadeiro na cláusula C_i (similar a Proposição 3.1), para i=1..m.

Proposição 3.3: PSMD-Decisão $\propto$ PSMP-Decisão

Prova: Esta proposição mostra a redução do PSMD-Decisão para o PSMP-Decisão, garantindo desta forma, que o PSMP-Decisão é NP-Completo. Conseqüentemente, o PSMP é NP-Difícil.

Primeiramente, considera-se o caso onde $|\Sigma| = 2$. Se existe, para o PSMD-Decisão, uma seqüência s_F e uma distância d_F, onde $d_H(s_F, s_i) \geq d_F$, para todo $s_i \in S$, logo, existe no mínimo d_F diferenças entre s_F e s_i. Conseqüentemente serão encontradas no máximo $d_H = n - d_F$ diferenças (não-coincidências) entre s_H e s_i, onde s_H é a seqüência complementar de s_F. Assim, teremos para o PSMP-Decisão, $d_H(s_H, s_i) \leq d_H$ para cada $s_i \in S$.

O caso binário pode ser generalizado para o caso $|\Sigma| > 2$ da seguinte forma: dada uma instância S do PSMP-Decisão, sem perda de generalidade pode-se utilizar uma função não-binária para converter a instância de entrada, onde $|\Sigma| > 2$, numa instância binária. Posteriormente, resolve-se o problema no formato binário e no final converte a solução binária para o alfabeto original. Finalmente, provou-se que o PSMP é NP-Difícil.

3.3 Algoritmos Aproximativos: Determinísticos e Randômicos

Em um problema de otimização combinatória deseja-se minimizar (ou maximizar) uma função objetivo $f(.)$ sujeita a um conjunto discreto X de soluções viáveis. Seja $\prod$ um problema de otimização combinatória, e I uma instância qualquer de $\prod$ (representada por $I \in \prod$). Se A é um algoritmo aproximativo (determinístico ou randômico) para $\prod$, então $x_A(I)$ é o valor da função objetivo gerado por A, $\forall\ I \in \prod$. O valor da solução ótima associada será representado por $x^*(I)$.

Idealmente, nos algoritmos aproximativos, deseja-se obter uma solução que difira da solução ótima apenas por uma pequena constante. Medidas desse tipo serão denominadas *medidas de aproximação absoluta*. Algoritmos aproximativos que se encaixam nesse conceito para algum k positivo serão chamados *algoritmos de aproximação absoluta*. Mais formalmente, tem-se a seguinte definição:

Definição 3.3: (Algoritmos de aproximação absoluta)

Um algoritmo aproximativo A será de *aproximação absoluta* para um problema $\prod$ se, e somente se, para algum inteiro positivo k, tem-se $| x_A(I) - x^*(I) | \leq k$, $\forall\ I \in \prod$.

Note que a definição acima se aplica indistintamente para problemas de minimização e maximização. Eliminando-se o módulo da desigualdade, conclui-se diretamente que: $x_A(I) \leq x^*(I)+k$ para problemas de minimização, e $x_A(I) \geq x^*(I)-k$ para problemas de maximização (vide Figura 3.2(a)).

Claramente, conseguir um algoritmo de aproximação absoluta é o melhor que se espera obter para problemas NP-Árduos. Infelizmente, para uma grande quantidade de problemas, algoritmos de aproximação absoluta só existirão se P=NP! Em outras palavras, encontrar um algoritmo de aproximação absoluta para um problema de otimização Π (NP-Árduo) poderá ser tão difícil quanto encontrar um algoritmo de complexidade polinomial para o problema de decisão associado! Em função disso, criou-se uma outra medida de desempenho denominada *medida de performance relativa*.

Garey *et al*. [Garey *et al*., 1972] e posteriormente Johnson [Johnson, 1974] formalizaram o conceito de algoritmos aproximativos. Como discutido anteriormente, um algoritmo aproximativo deverá, necessariamente, ser polinomial no tamanho de qualquer instância para o problema. Considere então as seguintes definições:

Definição 3.4: (Algoritmo $f(n)$-aproximativo)

Um algoritmo A com solução $x_A(I)$ é $f(n)$-aproximado para um problema de minimização (de maximização) Π se, e somente se, qualquer que seja a instância I de tamanho n, a solução obtida é no máximo (no mínimo) $f(n)$ vezes o valor da solução ótima $x^*(I)$.

Observe através da definição acima que, se A é $f(n)$-aproximado, então $x_A(I) \leq f(n).x^*(I)$ para problemas de minimização, e $x_A(I) \geq f(n).x^*(I)$ para problemas de maximização. Normalmente, em problemas de maximização assume-se que $x^*(I) > 0$.

Definição 3.5: (Algoritmo δ-aproximativo)

Um algoritmo A $f(n)$-aproximado é δ-aproximado para um problema de minimização (de maximização) Π se, e somente se, $f(n) \leq \delta$ ($f(n) \geq \delta$) para algum $\delta > 0$.

Naturalmente, deve-se ter $\delta \geq 1$ em problemas de minimização, e $0 < \delta \leq 1$ em problemas de maximização. Observe ainda que, quanto mais δ se aproxima de 1, melhor a qualidade da solução obtida pela heurística. O parâmetro δ é também conhecido como razão de performance absoluta ou fator de aproximação do algoritmo A. Em problemas de maximização é também comum representar o fator de aproximação por $1/\delta$ (vide Figura 3.2(b)).

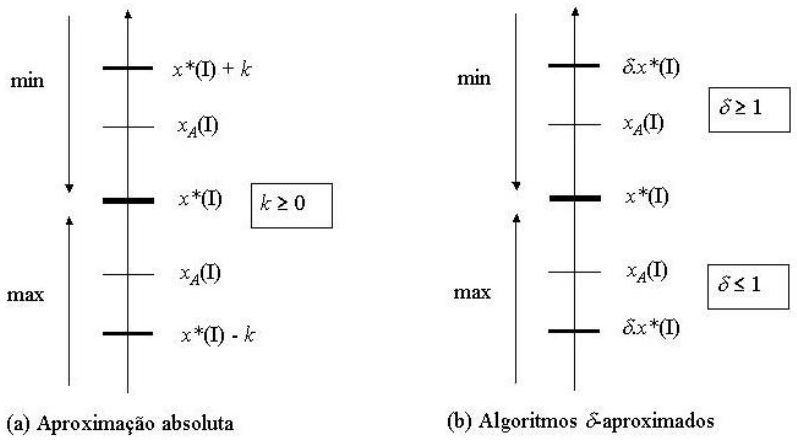


Figura 3.2: Razão de Performance em Problemas de minimização e maximização

De maneira geral, se um algoritmo aproximativo A é $(1+\varepsilon)$-aproximado para um problema Π, então:

$$\frac{|x_A(I) - x^*(I)|}{x^*(I)} \leq \varepsilon, \forall\ I \in \Pi \text{ e } \varepsilon > 0$$

Neste caso, basta fazer $\delta = 1 - \varepsilon$ em problemas de maximização, e $\delta = 1 + \varepsilon$ em problemas de minimização.

Definição 3.6: (Esquema de aproximação polinomial)

Uma família de algoritmos aproximativos para um problema Π, $\{A_\varepsilon\}_\varepsilon$ é chamada um esquema de aproximação polinomial se, e somente se, o algoritmo A_ε for $(1+\varepsilon)$-aproximado e seu tempo de processamento for polinomial no tamanho da entrada para ε fixo.

Em outras palavras, o algoritmo polinomial A $(1+\varepsilon)$-aproximado pode ser visto como uma família de algoritmos $\{A_\varepsilon | \varepsilon > 0\}$. Vale ressaltar ainda que, se A_ε é algoritmo $(1+\varepsilon)$-aproximado para um problema de maximização então $1/\delta = 1 + \varepsilon$ é fator de aproximação. Desta forma, o fator de aproximação dos problemas de maximização e minimização são sempre maiores que 1, e a Definição 3.6 se aplica a ambos os casos.

Para exemplificar esta definição, suponha que um algoritmo A_ε para um problema Π qualquer tenha complexidade igual a $n^{1/\varepsilon}$, onde $\varepsilon > 0$ e n é o tamanho do problema. Note que, embora $n^{1/\varepsilon}$ represente uma função polinomial em n, o tempo de processamento cresce bastante quando $\varepsilon \to 0$. Idealmente, deseja-se que o tempo de processamento cresça mais lentamente quando ε decresce. Esta situação pode ser mais bem formalizada através da definição de esquemas de aproximação totalmente polinomiais.

Definição 3.7: (Esquema de aproximação totalmente polinomial)

Uma família de algoritmos aproximativos para um problema Π, $\{A_\varepsilon\}_\varepsilon$ é chamada de esquema de aproximação totalmente polinomial se, e somente se, o algoritmo A_ε for $(1+\varepsilon)$-aproximado e seu tempo de processamento for polinomial no tamanho da entrada n e $1/\varepsilon$.

Como exemplo, se A_ε tem complexidade igual à $(1/\varepsilon)^2 n^4$ para um problema Π qualquer, então A_ε define um esquema de aproximação totalmente polinomial para Π.

Definição 3.8: (Razão de performance em uma instância I)

Seja A um algoritmo aproximativo para um problema de minimização Π. A razão de performance na instância I, representada por $R_A(I)$ é definida como:

$$R_A(I) = \frac{x_A(I)}{x^*(I)} \geq 1$$

Se $\prod$ é um problema de maximização então:

$$R_A(I) = \frac{x^*(I)}{x_A(I)} \geq 1$$

Note que, independentemente do problema $\prod$ ser de maximização ou minimização, a performance do algoritmo é melhor quando R_A(I) se aproxima de 1.

Considere agora a seguinte definição de algoritmo randômico δ-aproximado:

Definição 3.9: (Algoritmo randômico δ-aproximado)

Seja $\prod$ um problema de minimização. Um algoritmo randômico A de complexidade polinomial é δ-aproximativo para $\prod$ se, e somente se, $E(x_A(\mathrm{I})) \leq \delta.x^*(\mathrm{I})$, onde $\delta \geq 1$ e x_A(I) é uma variável aleatória. Se $\prod$ é um problema de maximização então A é δ-aproximativo para $\prod$ se, e somente se, $E(x_A(\mathrm{I})) \geq \delta.x^*(\mathrm{I})$, onde $\delta \leq 1$.

Em outras palavras, pode-se dizer que, se A é um algoritmo randômico δ-aproximativo para um problema de minimização então $\Pr[x_A(\mathrm{I}) \leq \delta.x^*(\mathrm{I})] \geq 1/2$ onde $\delta \geq 1$. Analogamente, $\Pr[x_A(\mathrm{I}) \geq \delta.x^*(\mathrm{I})] \geq 1/2$ para $\delta \leq 1$ em problemas de maximização.

Na determinação de uma solução randômica aproximada, pode-se por exemplo, atribuir valores aleatoriamente às variáveis associadas ao problema, satisfazendo obviamente, alguma distribuição estatística. Outra possibilidade interessante é a adoção de Programação Linear ou Programação Semidefinida na geração de probabilidades para o algoritmo randômico considerado. Nos dois modelos será possível utilizar algoritmos polinomiais baseados nos métodos de pontos interiores [Wright, 1997]. Essa técnica é mais conhecida na literatura como arredondamento randômico e será discutida a seguir.

3.3.1 Arredondamento Randômico (*Randomized Rounding*)

Na técnica de arredondamento randômico, introduzida inicialmente por Raghavan e Thompson [Raghavan & Thompson, 1987], formula-se primeiramente um modelo de programação linear inteira associado ao problema original. Em seguida resolve-se o problema relaxado obtendo-se limitantes inferiores para o valor ótimo do problema original. A solução relaxada gerada definirá probabilidades a serem utilizadas na etapa randômica, que corresponde à geração de valores inteiros em função das probabilidades definidas pelas relaxações.

Sem perda de generalidade, considere o seguinte modelo geral de programação linear inteira 0-1:

$$x^*(I) = \min c^T x \quad \begin{cases} Ax \geq b \\ x \in \{0,1\}^n \end{cases} \qquad \text{(PPL)}$$

onde I representa a instância formada pelas matrizes A, b e c sendo $A \in \Re^{mxn}$, $c \in \Re^{nx1}$ e $b \in \Re^{mx1}$. A Relaxação Linear (RL) do PPL acima é obtida substituindo-se a restrição $x \in \{0,1\}^n$, por $x \in [0,1]^n$.

Pode-se resumir a técnica de arredondamento randômico no procedimento genérico a seguir (Algoritmo 3.1).

No passo (1) o algoritmo resolve o modelo de relaxação linear, obtendo uma solução fracionária y. No passo (2) atribui-se inicialmente ∞ ao valor da solução heurística. Posteriormente, no passo (3), a solução fracionária y define probabilidades na geração de uma solução inteira $\bar{x}$ (arredondamento randômico). É fundamental que a solução gerada randomicamente no passo 3 seja viável com probabilidade constante maior que zero (mesmo que pequena). Desta forma, repetições adicionais do passo 3, sempre salvando a melhor solução viável a cada passo, reduzem arbitrariamente a probabilidade de falha. Neste caso, uma falha (ou *mau* evento) pode indicar uma solução inviável ou uma solução cuja razão de aproximação esteja fora de um limite pré-estabelecido (razão de performance).

O critério de parada no Algoritmo 3.1 será discutido mais adiante quando falamos do Método de Monte Carlo.

Início

1. Resolver o modelo de relaxação linear (RL) obtendo uma solução $y \in [0,1]^n$
2. $x_A(I) \leftarrow \infty$
3. **Repita**
   a. Gerar solução inteira $\bar{x} \in \{0,1\}^n$, da seguinte forma:

      $\Pr(\bar{x}_i = 1) = y_i$

      $\Pr(\bar{x}_i = 0) = 1 - y_i$, onde i=1..n

   b. **Se** (($\bar{x}$ é viável) e ($c^T \bar{x} < x_A(I)$)) **então**

      $x_A(I) \leftarrow c^T \bar{x}$

   Até (condição de parada)
4. Retorna $x_A(I)$

Fim

Algoritmo 3.1: Arredondamento Randômico

Pode-se utilizar as probabilidades obtidas no arredondamento randômico para geração de uma solução determinística (*derandomização*). Desta forma, basta provar que uma solução randômica gerada no passo 3 do Algoritmo 3.1 é solução viável com probabilidade não nula. Segue, uma síntese do algoritmo para geração de uma solução determinística.

Início

1. Resolver o modelo de Relaxação Linear (RL) obtendo uma solução $y \in [0,1]^n$
2. Utilizando y constrói-se uma solução determinística *derandomizada* $x \in \{0,1\}^n$.

Fim

Algoritmo 3.2: Técnica de Arredondamento Randômico para geração de uma solução determinística

No passo (1) o algoritmo resolve o modelo de relaxação linear, obtendo uma solução fracionária y. Posteriormente, no passo (2), pode-se usar a solução fracionária y para geração de uma solução determinística do problema (*derandomização*). É fundamental provar que a solução gerada randomicamente no passo 3 do Algoritmo 3.1, ocorra com probabilidade de sucesso maior que zero! Executa-se então a etapa de *derandomização* (passo 2 do algoritmo 3.2), que será discutida a seguir.

3.3.2 Derandomização

Em algumas situações é possível construir algoritmos determinísticos com o auxílio de técnicas probabilísticas. Em outras palavras, deseja-se construir um algoritmo determinístico sem que se sacrifique muito a qualidade da solução e/ou tempo de processamento obtidos no procedimento randômico. Infelizmente, não se conhece um mecanismo universal de conversão que seja aplicável a todas as situações.

Apesar de não existir um termo apropriado em português para esta técnica, talvez o mais conveniente seja chamá-la simplesmente de *derandomização* (semelhante ao termo inglês *derandomization*). Entre os métodos de *derandomização* mais conhecidos na literatura pode-se citar: o método de expectâncias condicionais, o método dos estimadores pessimistas, *k-wise independence* [Alon & Spencer, 1992] entre outros. Neste capítulo será enfatizado o método das probabilidades condicionais e o método dos estimadores pessimistas introduzido inicialmente por [Raghavan, 1988]. Neste trabalho, Raghavan mostra que, uma vez garantida uma solução viável com probabilidade de sucesso estritamente positiva (passo 3 do arredondamento randômico), uma solução viável $x \in \{0,1\}^n$ do PPL pode ser obtida deterministicamente.

No método das probabilidades condicionais, faz-se uma analogia com árvores de decisão, construindo deterministicamente um vetor x, correspondente ao caminho de descida da árvore de decisão.

Considere uma árvore binária completa T com n níveis. O j-ésimo nível de T (onde $j \in \{1..n\}$) irá representar a atribuição de valores 0-1 à variável aleatória x_j. Cada folha da

árvore irá corresponder a um *bom* ou *mau* evento. O evento será *bom*, se a solução x obtida for viável ou dentro da razão de aproximação e *mau*, caso contrário. Em outras palavras, um *mau* evento B representa uma solução inviável ou uma solução que não satisfaz à razão de performance pretendida. O objetivo será percorrer a árvore T da raiz até uma folha *boa* em tempo determinístico polinomial (vide Figura 3.3).

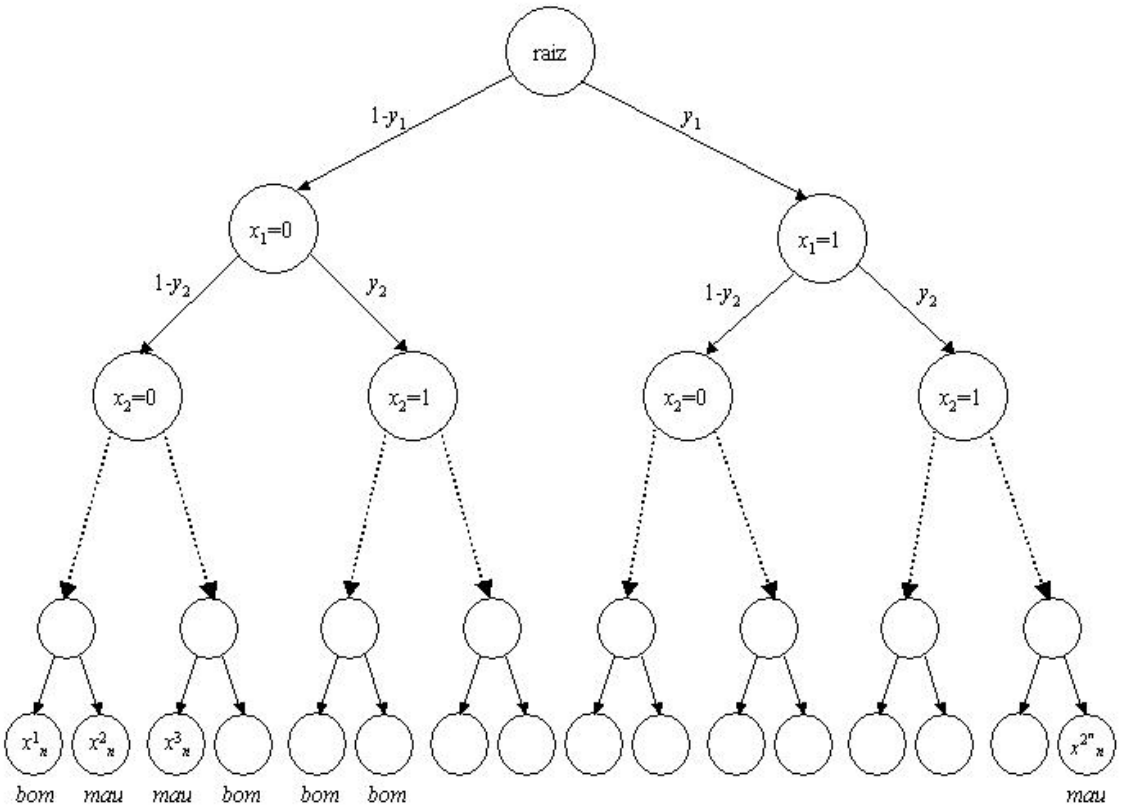


Figura 3.3: Árvore de decisão para busca de uma solução viável

A caminhada na árvore de decisão é realizada da seguinte forma. Se à variável aleatória x_1 é atribuído o valor 1, percorre-se da raiz para seu filho à direita. Se $x_1 = 0$, percorre-se para o filho esquerdo e assim sucessivamente até que se chegue a uma folha da árvore T.

Note que, cada folha de T, corresponde a uma entre 2^n (representada na Figura 3.3 por x^i, onde $i \in \{1.. 2^n\}$) seqüências possíveis de $x = (x_1 x_2 ... x_n)$.

No método probabilístico precisa-se mostrar inicialmente que Pr(B)<1 (probabilidade de um *mau* evento), ou equivalentemente, Pr($\overline{B}$)>0 (onde $\overline{B}$ representa um evento complementar ou *bom* evento). A questão natural que se coloca agora é: como percorrer da raiz até uma folha *boa* de T?

Seja P_1 = Pr(B) e seja $Y = (y_1, y_{2, ..., } y_n)$ a solução da Relaxação Linear RL discutida na seção precedente. Da expressão de probabilidade absoluta tem-se [Meyer, 1983]:

$$P_1 = \Pr(x_1=1).P_2(B|x_1=1) + \Pr(x_1=0).P_2(B|x_1=0)$$

Como Pr(x_1=1) = y_1 e Pr(x_1=0) = 1 - y_1, tem-se:

$$P_1 \geq y_1.\min\{ P_2(B|x_1=1) ; P_2(B|x_1=0) \} + (1 - y_1).\min\{ P_2(B|x_1=1) ; P_2(B|x_1=0) \}$$
$$= P_2(B|\hat{x}_1)$$
$$P_1 \geq P_2(B|\hat{x}_1),$$

onde $\hat{x}_1$ = 1 se, e somente se, $P_2(B|x_1=1) \leq P_2(B|x_1=0)$ e $\hat{x}_1$ = 0, caso contrário.

De maneiro geral, seja j um nível de T, para $j \in \{1..n\}$, e $P_j(B|\hat{x}_1, ..., \hat{x}_{j-1})$ a probabilidade condicional da ocorrência de um *mau* evento, dado que os valores $\hat{x}_1, ..., \hat{x}_{j-1}$ já tenham sido obtidos. Tem-se então que:

$$P_j\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1}\right) = y_j P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},1\right) + \left(1-y_j\right)P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},0\right)$$

$$P_j\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1}\right) \geq \min\left\{P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},1\right), P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},0\right)\right\} = P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_j\right)$$

Desta forma, a probabilidade de P_j será sempre maior ou igual a probabilidade de P_{j+1}. O algoritmo determinístico sugerido será apresentado a seguir (Algoritmo 3.3).

Observe finalmente que, se P_1 = Pr(B) < 1:

$$1 > P_1 \geq P_2\left(B\middle|\hat{x}_1\right) \geq ... \geq P_{n+1}\left(B\middle|\hat{x}_1...\hat{x}_n\right) = \Pr(\textit{folha})$$

Como Pr(*folha*) < 1, então se garante que a solução escolhida será uma *boa* folha, visto que, $P_{n+1}\left(B\middle|\hat{x}_1....\hat{x}_n\right)=\Pr(folha)=0$. Em outras palavras, a solução determinística obtida ocorre com probabilidade de erro zero. Como as probabilidades condicionais nem sempre são determinadas facilmente, outras técnicas de *derandomização* poderão ser utilizadas [Alon & Spencer, 1992].

Início

1. **Para** $j=1$ **até** n **faça**

$$\hat{x}_j=\begin{cases}1 & se\ P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},1\right)\le P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_{j-1},0\right)\\ 0 & caso\ contrário\end{cases}$$

   Fim Para

2. Retorna $\hat{x}$

Fim

Algoritmo 3.3: Algoritmo de *derandomização*, utilizando o método de probabilidades condicionais.

No método dos *Estimadores Pessimistas* introduzido por Raghavan [Raghavan, 1988], as probabilidades condicionais são limitadas superiormente através de uma função $U:[0,1]^n\rightarrow[0,1)$ (denominada estimador pessimista) satisfazendo às seguintes condições:

1) $U_1(y_1,...,y_n)<1$, onde y é uma solução da relaxação linear do PPL descrito na Seção 3.3.1.
2) $U_{j+1}\left(\hat{x}_1,...,\hat{x}_j,y_{j+1},...,y_n\right)\ge P_{j+1}\left(B\middle|\hat{x}_1,...,\hat{x}_j\right)$, onde $\hat{x}_k$ para k=1,...,j é uma atribuição dada às variáveis $x_k\in\{0,1\}$ e $j\in\{1,...,n\}$.
3) $U_j\left(\hat{x}_1,...,\hat{x}_{j-1},y_j,...,y_n\right)\ge\min\left\{U_{j+1}\left(\hat{x}_1,...,\hat{x}_{j-1},1,y_{j+1},...,y_n\right);U_{j+1}\left(\hat{x}_1,...,\hat{x}_{j-1},0,y_{j+1},...,y_n\right)\right\}$

O algoritmo de *derandomização* será idêntico àquele descrito acima, bastando substituir as probabilidades condicionais pelos estimadores pessimistas correspondentes. Obviamente, neste caso, a função $U : [0,1]^n \rightarrow [0,1)$ deverá ser computada facilmente.

Capítulo 4

Algoritmos aproximativos para o PSMP

4.1 Introdução

Neste capítulo e nos dois capítulos seguintes (Capítulos 5 e 6), serão apresentados os principais algoritmos aproximativos citados na literatura para o PSMP. Aqui, dois algoritmos serão apresentados. Na Seção 4.2 é apresentado um algoritmo 2-aproximado bastante natural para o problema. Na Seção 4.3, é apresentado o algoritmo aproximativo de Ben-Dor *et al*. [Ben-Dor *et al*., 1997], um algoritmo com razão de performance próximo do valor ótimo para d suficientemente grande, onde d representa a maior distância a ser minimizada (vide Seção 3.1.1). Finalmente, na ultima seção, desenvolvemos a estratégia de *derandomização* sugerida em [Ben-Dor *et al*., 1997]. Através do método dos estimadores pessimistas, mostramos como construir uma solução determinística através das probabilidades geradas na relaxação linear.

4.2 Algoritmo 2-aproximado

Como discutido anteriormente, considere um conjunto $S = \{s_1,s_2,...,s_m\}$ de seqüências, todas de tamanho n, sobre um alfabeto Σ. Seja $s_{opt} \in \Sigma^n$ uma solução ótima e d_{opt} a distância ótima para o PSMP. Logo, para cada $s_i \in S$, tem-se $d_H(s_{opt}, s_i) \leq d_{opt}$, para i=1...m (vide Seção 3.1.1).

Na primeira heurística apresentada, seleciona-se uma seqüência s_H qualquer do conjunto S como solução do problema. Desta forma, pode-se garantir que $d_H(s_H, s_i) \leq 2d_{opt}$, para i=1...m. Esse resultado pode ser facilmente provado como a seguir.

Pela desigualdade triangular tem-se, $d_H(s_H, s_i) \leq d_H(s_H, s_{opt}) + d_H(s_i, s_{opt})$. Como para qualquer $s_i \in S$, $d_H(s_i, s_{opt}) \leq d_{opt}$, pode-se garantir que, para qualquer seqüência $s_i \in S$,

$d_H(s_H, s_i) \leq 2d_{opt}$, sendo $s_H \in S$ uma seqüência escolhida arbitrariamente como solução do problema.

4.3 Algoritmo aproximado de Ben-dor *et al*.[1997]

4.3.1 Introdução

A estratégia adotada por Ben-Dor *et al*. [Ben-Dor *et al*., 1997] foi formular um modelo de programação linear inteira para o PSMP, resolver o modelo de relaxação linear, e utilizar as componentes desta solução relaxada como probabilidades no arredondamento randômico (*randomized rounding*).

Poderá ser observado mais adiante (Capítulos 5 e 6) que as diferenças entre os resultados obtidos na qualidade dos algoritmos randômicos aproximativos apresentados na literatura para o PSMP, são referentes a pequenas adaptações na técnica de arredondamento randômico, visto que, esta abordagem é empregada na maioria dos algoritmos aqui apresentados.

Seja $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_P\}$ um alfabeto finito de símbolos, S um conjunto de m seqüências $s_i \in \Sigma^n$, e s_{opt} a seqüência que se "deseja encontrar" (solução ótima). Define-se, para cada caracter $\sigma \in \Sigma$ e cada caracter $s_{opt}[j]$, para $j = 1..n$, uma variável binária $x_{j,\sigma}$, onde $x_{j,\sigma} = 1$ se, e somente se, $s_{opt}[j] = \sigma$, caso contrário $x_{j,\sigma} = 0$. Considere agora o seguinte modelo de programação linear inteira:

$$d_{opt} = \min d \qquad (4.1)$$

$$s.a \begin{cases} \sum_{\sigma \in \Sigma} x_{j,\sigma} = 1 & , j = 1..n \quad (4.2) \\ \sum_{j=1}^{n} \sum_{\sigma \in \Sigma} x_{j,\sigma}\, d_H(\sigma, s_i[j]) \leq d & , i = 1..m \quad (4.3) \\ x_{j,\sigma} \in \{0,1\} & , j = 1..n\,;\, \sigma \in \Sigma \quad (4.4) \end{cases}$$

No modelo apresentado, (4.1) representa a função objetivo que se deseja minimizar. As restrições (4.2) asseguram que somente um símbolo $\sigma \in \Sigma$ será selecionado em cada posição

da solução s_{opt}. As desigualdades (4.3) especificam que a distância entre cada seqüência de S e s_{opt}, deve ser menor ou igual a d_{opt}. As desigualdades (4.4) indicam que somente valores 0 ou 1 podem ser atribuídos às variáveis $x_{j,\sigma}$.

Segue abaixo a descrição detalhada do algoritmo aproximativo.

Entrada: $s_1, s_2, ..., s_m \in \sum^n$

Saída: seqüência $s_H \in \sum^n$ e a distância d_H.

Início

1. Resolver o modelo de relaxação linear, retornando os valores relaxados $\bar{x}_{j,\sigma}$ e $\bar{d}$
2. Construir a seqüência s_H a partir dos valores relaxados $\bar{x}_{j,\sigma}$

   Para $j = 1, ..., n$ **faça**

   $\Pr(s_H[j] = \sigma) = \bar{x}_{j,\sigma}$, onde $\sigma \in \Sigma$
3. Calcular distância $d_H = \max_{1 \le i \le m} \left\{ d_H(s_H, s_i) = \sum_{1 \le j \le n} d_H(s_H[j], s_i[j]) \right\}$
4. Retornar a seqüência s_H e a distância d_H.

Fim

Algoritmo 4.1: Algoritmo aproximativo de Ben-Dor *et al.*[1997]

No passo (1), resolve-se o modelo de relaxação linear obtendo-se valores fracionários para $\bar{x}_{j,\sigma}$, onde $1 \le j \le n$ e $\sigma \in \Sigma$. Com os valores de $\bar{x}_{j,\sigma}$ no passo (2), constrói-se a solução s_H através do processo de arredondamento randômico. Finalmente, no passo (3), calcula-se a maior distância d_H, comparando a solução s_H com todas as seqüências do conjunto S.

4.3.2 Análise de Aproximação

Como observado no algoritmo acima, sejam $\bar{x}_{j,\sigma}$ e $\bar{d}$ soluções obtidas na relaxação linear. Claramente, observa-se que $\bar{d} \le d_{opt}$. Os valores de $\bar{x}_{j,\sigma}$ são valores fracionários, não representando, necessariamente, uma solução viável para o problema. Note que estes valores

serão usados como probabilidades no arredondamento randômico. A solução heurística obtida, após o arredondamento randômico, será sempre viável já que o valor da solução d_H é calculado posteriormente à obtenção da seqüência s_H.

Suponha que s_H seja uma solução $(1+\delta)$-aproximada para algum $\delta > 0$. Da definição de algoritmo randômico aproximativo deve-se provar que $E(d_H) \leq (1+\delta)d_{opt}$ para algum $\delta > 0$, onde $d_H = \max \{d_H(s_H,s_i);\ i=1..m\}$ (vide Figura 4.1). Equivalentemente, pode-se provar que $\Pr(d_H > (1+\delta)d_{opt}) < 1$, ou seja, a probabilidade de falha é estritamente menor que 1. Desta forma, no método de Monte Carlo, discutido mais adiante, a probabilidade de falha se aproxima arbitrariamente de 0 após sucessivas repetições dos passos 2 e 3 do algoritmo definido anteriormente (vide Algoritmo 4.1), sempre salvando-se a melhor solução a cada iteração.

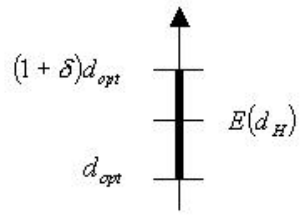


Figura 4.1: Razão de aproximação

O estudo da solução gerada pela heurística, será baseado na probabilidade de falha, representada pelo evento $B \equiv (d_H > (1+\delta)d_{opt})$. Como discutido anteriormente, é necessário provar que a probabilidade de falha será estritamente menor que 1. Dessa forma garante-se que a probabilidade de sucesso será sempre maior que 0.

Seja $X_i = d_H(s_i,s_H)$ (para $i=1,...,m$) variáveis aleatórias representado a distância entre as seqüências s_i e s_H. Considere então, o seguinte resultado preliminar:

Lema 4.1: $E[X_i] \leq d_{opt}$, $\forall\ i=1..m$.

Prova:

$$E[X_i] = E[d_H(s_H, s_i)] = E\left[\sum_{j=1}^{n} d_H(s_H[j], s_i[j])\right]$$

$$= \sum_{j=1}^{n} E\left[d_H\left(s_H[j], s_i[j]\right)\right] \text{ (pela linearidade da expectância)}$$

Da desigualdade (4.3) tem-se que:

$$= \sum_{j=1}^{n} \sum_{\sigma \in \Sigma} E\left[x_{j,\sigma}\right] d_H\left(\sigma, s_i[j]\right)$$

$$= \sum_{j=1}^{n} \sum_{\sigma \in \Sigma} \bar{x}_{j,\sigma} . d_H\left(\sigma, s_i[j]\right) \quad \leq \bar{d} \leq d_{opt} \qquad \blacksquare$$

Note agora que um *mau* evento ocorre (i.e., $B \equiv (d_H > (1+\delta)d_{opt})$) se, e somente se, $B_i \equiv (X_i > (1+\delta)d_{opt})$ ocorre para pelo menos um índice $i \in \{1..m\}$. Assim, dado $0 < \varepsilon < 1$, deve-se provar então que:

$$\Pr(B) = \Pr\left(\bigcup_{i=1}^{m} B_i\right) \leq \varepsilon < 1$$

Logo, como $E[X_i] \leq d_{opt}$ para i=1..m, e $\delta > 0$, espera-se que:

$$\Pr(B_i) = \Pr\left(X_i > (1+\delta)d_{opt}\right) \leq \Pr\left(X_i > (1+\delta)E(X_i)\right) \leq \frac{\varepsilon}{m}, \ \forall\ i=1..m.$$

Da desigualdade de Chernoff-Hoeffding (Lema 3.3), tem-se que:

$$\Pr\left(X_i \geq (1+\delta)E[X_i]\right) \leq \exp\left(-\frac{E[X_i]\delta^2}{3(b-a)^2}\right), \qquad \forall\ i=1..m.$$

onde $X_i = \sum_{j=1}^{n} X_{ij}$, e $X_{ij} = d_H(s_i[j], s_H[j])$, para j=1..n, são variáveis aleatórias independentes assumindo valores no intervalo [0, D], sendo $D = max_{\alpha,\beta \in \Sigma}\{d_H(\alpha, \beta)\}$ a maior distância entre dois caracteres do alfabeto Σ. Neste caso, $a = 0$ e $b = D$.

Finalmente, espera-se que:

$$\frac{\varepsilon}{m} \geq \exp\left(-\frac{E[X_i]\delta^2}{3D^2}\right), \ \forall\ i=1..m.$$

Ou ainda, como $E[X_i] \leq d_{opt}$:

$$\Pr\left(X_i > (1+\delta)d_{opt}\right) \leq \frac{1}{\exp\left(\frac{d_{opt}\delta^2}{3D^2}\right)} \leq \frac{1}{\exp\left(\frac{E[X_i]\delta^2}{3D^2}\right)} \leq \frac{\varepsilon}{m}, \ \forall\ i=1..m. \qquad (4.5)$$

Portanto,

$$\delta \geq D\sqrt{\frac{3\ln(m/\varepsilon)}{d_{opt}}}, \ \forall\ i=1..m.$$

Assim, obtém-se um algoritmo $\left(1+D\sqrt{\frac{3\ln(m/\varepsilon)}{d_{opt}}}\right)$-aproximado, com probabilidade de sucesso maior ou igual a (1-ε). Equivalentemente, tem-se:

$$\Pr\left(d_H \geq d_{opt} + D\sqrt{3d_{opt}\ln(m/\varepsilon)}\right) \leq \varepsilon, \text{ para } 0 < \varepsilon < 1.$$

Observe que a qualidade desta aproximação é dependente de d_{opt}. Em outras palavras, quanto maior o valor de d_{opt}, melhor a aproximação obtida.

Note também que a razão de aproximação do algoritmo de Ben-Dor *et al*. [Ben-Dor *et al*., 1997] está diretamente relacionada à probabilidade de falha ε, onde $0 < \varepsilon < 1$. Desta forma, para melhorar a aproximação, deve-se aumentar a probabilidade de falha (ε próximo de 1). Logo, surge a seguinte questão: melhora-se a aproximação aumentando-se a probabilidade de falha, ou piora-se a aproximação diminuindo-se a probabilidade de fracasso ε de uma iteração do algoritmo? Vejamos como tratar desta questão utilizando o método de Monte Carlo.

A técnica de Monte Carlo pode ser empregada da seguinte forma: fixa-se uma probabilidade de falha ε suficientemente próxima de 1, melhorando desta forma a

aproximação (1+δ) da solução randômica. Agora, o algoritmo pode ser parametrizado, recebendo como dado de entrada um erro $\gamma > 0$ desejado, arbitrariamente pequeno. Portanto, se $\Pr(B) \leq \varepsilon < 1$, então $\lceil |\log\gamma / \log\varepsilon| \rceil$ iterações, ou seja, um número constante de repetições do Algoritmo 4.1, serão necessárias para que o resultado seja (1+δ)-aproximado com probabilidade de sucesso 1-γ.

De fato, espera-se que após k repetições do algoritmo se tenha $\varepsilon^k \leq \gamma$, onde $\Pr(B) = \varepsilon$ e γ é arbitrariamente pequeno. Logo, $k\log\varepsilon \leq \log\gamma$. Como, $\log\varepsilon$ e $\log\gamma$ são negativos (pois $\varepsilon < 1$ e $\gamma < 1$) e k é um número inteiro, tem-se finalmente:

$$k(-\log\varepsilon) \geq (-\log\gamma) \Rightarrow k \geq \left\lceil \left| \frac{\log\gamma}{\log\varepsilon} \right| \right\rceil .$$

O algoritmo de Monte Carlo é sintetizado a seguir.

Entrada: $s_1, s_2, \ldots, s_m \in \sum^n$, um número real ε (próximo de 1) e um número real γ, onde $0<\gamma<1$ representa a probabilidade de falha B.

Saída: seqüência $s_H \in \sum^n$ e a distância d_H.

Início

1. Inicializar $d_H \leftarrow \infty$
2. Resolver o modelo de relaxação linear, retornando os valores relaxados $\bar{x}_{j,\sigma}$ e $\bar{d}$
3. **Para** $z = 1$ **até** $\lceil |\log\gamma / \log\varepsilon| \rceil$ **faça**
   a. Construir a seqüência s'_H aleatoriamente a partir dos valores relaxados $\bar{x}_{j,\sigma}$

      Para $j = 1, \ldots, n$ **faça**

      $\Pr(s'_H[j] = \sigma) = \bar{x}_{j,\sigma}$, $\sigma \in \sum$

   b. Calcular $d'_H = \max_{1 \leq i \leq m} \{ d_H(s'_H, s_i) \}$

   c. **Se** $d'_H < d_H$ **então** $d_H \leftarrow d'_H$ e $s_H \leftarrow s'_H$.
4. Retornar melhor solução s_H e d_H.

Fim

Algoritmo 4.2: Algoritmo aproximativo de Ben-Dor *et al*. [1997] - versão Monte Carlo

4.3.3 Outra Abordagem

Na análise vista anteriormente, não foi possível explicitar δ, já que este é função de d_{opt}. Utilizando uma versão diferente da desigualdade de Chernoff-Hoeffding (descrita no Lema 3.4) onde X é uma soma de variáveis aleatórias independentes, tem-se:

$$\Pr\left(X - E[X] > \delta\right) \leq \exp\left(-\frac{2\delta^2}{D^2 n}\right).$$

Logo, obtém-se um valor de δ limitado inferiormente por:

$$\delta \geq D\sqrt{\frac{n}{2}\ln\left(m/\varepsilon\right)}\ ,$$

e com probabilidade de fracasso dada por:

$$\Pr\left(d_H > d_{opt} + \delta\right) \leq \varepsilon\ ,\ \text{onde } 0 < \varepsilon < 1 \text{ é uma constante.}$$

De fato, para um algoritmo de aproximação $d_{opt} + \delta$ (vide Definição 3.3 de aproximação absoluta), deve-se provar que $E(d_H) \leq d_{opt} + \delta$, para algum $\delta > 0$, onde $d_H = \max\ \{d_H(s_H,s_i);\ i=1..m\}$. Equivalentemente, pode-se provar que $\Pr(d_H > d_{opt} + \delta) < 1$, ou seja, que a probabilidade de falha é estritamente menor que 1.

O estudo da solução gerada será baseada na probabilidade de falha, representada pelo *mau* evento $B \equiv (d_H > d_{opt} + \delta)$.

Note agora que um *mau* evento ocorre se, e somente se, $B_i \equiv (X_i > d_{opt} + \delta)$ ocorre para pelo menos um índice $i \in \{1..m\}$. Assim, dado $0 < \varepsilon < 1$, deve-se provar então que:

$$\Pr\left(B\right) = \Pr\left(\bigcup_{i=1}^{m} B_i\right) \leq \varepsilon < 1$$

Logo, como $E[X_i] \leq d_{opt}$, para $i=1..m$ e $\delta > 0$, espera-se que:

$$\Pr(B_i) = \Pr\left(X_i > d_{opt} + \delta \right) \leq \Pr\left(X_i > E(X_i) + \delta \right) \leq \frac{\varepsilon}{m}, \ \forall\ i=1..m.$$

Da desigualdade de Chernoff-Hoeffding (Lema 3.4), tem-se que:

$$\Pr\left(X_i - E[X_i] > \delta\right) \leq \exp\left(-\frac{2\delta^2}{(b-a)^2 n} \right), \ \forall\ i=1..m.$$

onde $X_i = \sum_{j=1}^{n} X_{ij}$, e $X_{ij} = d_H(s_i[j], s_H[j])$, para $j=1..n$, são variáveis aleatórias assumindo valores no intervalo $[0,D]$. Novamente neste caso, $a = 0$ e $b = D$.

Finalmente, espera-se que:

$$\frac{\varepsilon}{m} \geq \exp\left(-\frac{2\delta^2}{D^2 n} \right)$$

Portanto,

$$\delta \geq D\sqrt{\frac{n}{2}\ln(m/\varepsilon)}$$

Assim, obtém-se um algoritmo com probabilidade de sucesso maior ou igual a $(1-\varepsilon)$, onde $\Pr\left(d_H \geq d_{opt} + D\sqrt{\frac{n}{2}\ln(m/\varepsilon)} \right) \leq \varepsilon$, para $0 < \varepsilon < 1$.

4.4 Derandomização

Conforme discutido na Seção 3.3.2, para aplicação do método das probabilidades condicionais é fundamental que se garanta inicialmente $\Pr(B) < 1$, onde B representa um *mau* evento. Se $(1+\delta)$ é razão de aproximação, no PSMP, um *mau* evento ocorre sempre que $B \equiv (d_H > (1+\delta)d_{opt})$, onde d_H é variável aleatória representando o valor da função objetivo. Veremos mais adiante que as probabilidades condicionais não são obtidas diretamente, logo definiremos um estimador pessimista como mostrado na Seção 3.3.2.

Antes de desenvolvermos a *derandomização* sugerida em Ben-Dor *et al*. [Ben-Dor *et al*., 1997], considere a seguinte notação auxiliar. Representaremos por $x_j = x_{j\sigma_1}, x_{j\sigma_2}, ..., x_{j\sigma_p}$ onde $j \in \{1, ..., n\}$ e $|\Sigma| = p$, o vetor de variáveis associado à j-ésima posição de s_H (solução heurística). Se todas as componentes de x_j já forem conhecidas teremos então $\hat{x}_j = \hat{x}_{j\sigma_1}, \hat{x}_{j\sigma_2}, ..., \hat{x}_{j\sigma_p}$. Note neste caso, que cada vetor $\hat{x}_j$ irá representar uma seqüência de valores onde apenas uma coordenada é 1 e as demais são 0. Para aplicação da técnica de *derandomização* é fundamental que se calcule, na k-ésima iteração, a probabilidade $\Pr(B|A_k(\sigma'))$ onde $A_k(\sigma') \equiv (\hat{x}_1, \hat{x}_2, ..., \hat{x}_{k-1}, x_{k\sigma'} = 1 \; e \; x_{k\sigma} = 0 \;\; \forall \, \sigma' \neq \sigma)$ e $\sigma' \in \Sigma$.

Portanto, o caracter $\sigma' \in \Sigma$ escolhido na k-ésima iteração deverá ser tal que:

$$\Pr(B|\hat{x}_1, ..., \hat{x}_{k-1}) \geq \min_{\sigma' \in \Sigma} \{\Pr(B|A_k(\sigma'))\} = \Pr(B|\hat{x}_1, ..., \hat{x}_{k-1}, \hat{x}_k)$$

O processo deverá ser repetido até que tenhamos uma solução $\hat{x} = (\hat{x}_1, ..., \hat{x}_n)$ onde $\Pr(B|\hat{x}_1, ..., \hat{x}_n) = 0$.

Seja $\hat{d}_{i,k-1} = \sum_{j=1}^{k-1} \sum_{\sigma \in \Sigma} \hat{x}_{j\sigma} \, d_H(\sigma, s_i[j])$, para i=1,...,m. Se k = 1 então faremos $\hat{d}_{i,0}$ = 0. Na k-ésima iteração, calculamos o seguinte problema relaxado RL(k,σ') associado ao caracter $\sigma' \in \Sigma$.

$$\bar{d}_{k,\sigma'} = \min d \tag{4.6}$$

$$s.a \begin{cases} \sum_{\sigma \in \Sigma} x_{j\sigma} = 1 & , j = k+1..n \quad (4.7) \\ \sum_{j=k}^{n} \sum_{\sigma \in \Sigma} x_{j\sigma} \, d_H(\sigma, s_i[j]) \leq d - \hat{d}_{i,k-1} & , i = 1..m \quad (4.8) \\ x_{j\sigma} = \hat{x}_{j\sigma} & , j = 1..k-1 ; \sigma \in \Sigma \quad (4.9) \\ x_{k\sigma'} = 1 & , para \;\; a\,\lg um \; \sigma' \in \Sigma \quad (4.10) \\ x_{k\sigma} = 0 & , para \; \sigma' \neq \sigma \quad (4.11) \\ x_{j\sigma} \in [0,1] & , j = k+1..n ; \sigma \in \Sigma \quad (4.12) \end{cases}$$

Este modelo é bastante semelhante àquele apresentado em (4.1)-(4.4). Neste caso entretanto, deve ser observado que o valor $\hat{d}_{i,k-1}$ é uma constante obtida pelas atribuições dadas na restrição (4.9). As restrições (4.10) e (4.11) asseguram que somente um símbolo $\sigma' \in \Sigma$ será selecionado na *k*-ésima posição da solução heurística s_H. Note que, uma solução de RL(k, σ') será do tipo $(\hat{x}_1,...,\hat{x}_{k-1},\hat{x}_k,\bar{x}_{k+1},...,\bar{x}_n)$ onde $\hat{x}_k$ é tal que $\hat{x}_{k\sigma'}=1$ e $\hat{x}_{k\sigma}=0$, para todo $\sigma \neq \sigma'$. Observe que para cada componente $k \in \{1, ..., n\text{-}1\}$ resolvemos *p* problemas de programação linear, ou seja, faremos $x_{k\sigma'}=1$ para cada σ' em Σ.

Sem perda de generalidade, poderemos assumir que um *mau* evento *B* ocorre sempre que $B \equiv \left(d_H > (1+\delta)\bar{d}\right)$ onde $\bar{d} \leq d_{opt}$ é o valor da relaxação linear de (4.1)-(4.4). Seguindo-se o mesmo raciocínio desenvolvido em 4.3.2 obtemos um algoritmo (1+δ)-aproximado para:

$$\delta = D\sqrt{\frac{3\ln(m/\varepsilon)}{\bar{d}}}$$

Além disso, $\Pr(B) = \Pr\left(d_H > (1+\delta)\bar{d}\right) < 1$.

Seja $X_{i,k+1} = \sum_{j=k+1}^{n} d_H(s_H[j], s_i[j])$ (para cada *i*=1,...,*m*) uma variável aleatória representando uma solução heurística obtida através das relaxações $\bar{x}_{j\sigma}$ para *j* = *k*+1,...,*n* e $\sigma \in \Sigma$ (solução de (4.6)-(4.12)).

Considere agora o seguinte resultado auxiliar:

Lema 4.2: $E[X_{i,k+1}] \leq \bar{d}_{k,\sigma'} - \hat{d}_{i,k-1} - d_H(\sigma', s_i[k])$, $\forall$ *i*=1,...,*m*, $k \in \{1,...,n\}$ e $\sigma' \in \Sigma$.

Prova: Como $X_{i,k+1} = \sum_{j=k+1}^{n} d_H(s_H[j], s_i[j])$, temos que:

$$E[X_{i,k+1}] = E\left[\sum_{j=k+1}^{n} d_H(s_H[j], s_i[j])\right]$$

Das desigualdades (4.8)-(4.11) e da linearidade do valor esperado temos:

$$E\left[X_{i,k+1}\right] = \sum_{j=k+1}^{n} \sum_{\sigma\in\Sigma} \Pr\left(s_H[j]=\sigma\right) d_H\left(\sigma, s_i[j]\right)$$

$$= \sum_{j=k+1}^{n} \sum_{\sigma\in\Sigma} \bar{x}_{j,\sigma}\, d_H\left(\sigma, s_i[j]\right) \le \bar{d} - \hat{d}_{i,k+1} - d_H\left(\sigma', s_i[k]\right)$$

É fácil ver que $\bar{d} \le \bar{d}_{k,\sigma'}$. Portanto:

$$E\left[X_{i,k+1}\right] \le \bar{d}_{k,\sigma'} - \hat{d}_{i,k+1} - d_H\left(\sigma', s_i[k]\right) \qquad \blacksquare$$

Note que $B_i \equiv \left(X_i > (1+\delta)\bar{d}\right)\ \forall\ i=1,...,m$. Portanto:

$$\left(B_i \middle| A_k(\sigma')\right) = \left(X_{i,k+1} > (1+\delta)\bar{d}_{k,\sigma'} - \hat{d}_{i,k-1} - d_H\left(\sigma', s_i[k]\right)\right)$$

Temos então que:

$$\Pr\left(B_i \middle| A_k(\sigma')\right) = \Pr\left(X_{i,k+1} > (1+\delta)\bar{d}_{k,\sigma'} - \hat{d}_{i,k-1} - d_H\left(\sigma', s_i[k]\right)\right),\ \text{p/ } i=1,...,m \text{ e } k \in \{1,...,n\}.$$

Logo do Lema 4.2, temos para $\forall\ i=1,...,m$:

$$\Pr\left(B_i \middle| A_k(\sigma')\right) = \Pr\left(X_{i,k+1} > \bar{d}_{k\sigma'} - \hat{d}_{i,k-1} - d_H\left(\sigma', s_i[k]\right) + \delta\,\bar{d}_{k\sigma'}\right)$$
$$\le \Pr\left(X_{i,k+1} > E\left[X_{i,k+1}\right] + \delta\,\bar{d}_{k\sigma'}\right)$$

Para utilização da desigualdade de Chernoff-Hoeffding conforme descrito no Lema 3.2, consideraremos o caso binário onde $d_H(\alpha,\beta) \in \{0,1\}$, $\forall\ \alpha,\beta \in \Sigma$. Neste caso, teremos:

$$X_{i,k+1} = \sum_{j=k+1}^{n} d_H\left(s_H[j], s_i[j]\right) = \sum_{j=k+1}^{n} y_j \qquad (4.13)$$

e portanto $X_{i,k+1}$ pode ser visto como uma soma de variáveis aleatórias independentes $y_j \in \{0,1\}$. Assim $y_j = 1$ sempre que $s_H[j] \neq s_i[j]$ e, $y_j = 0$ caso contrário. Temos então que:

$$\Pr\left(B_i \middle| A_k(\sigma')\right) \le \exp\left(-\frac{1}{3}\bar{d}_{k\sigma'}\delta^2\right) \le \frac{\varepsilon}{m}, \text{ para } i=1,...,m.$$

Segue então que:

$$\Pr\left(B \middle| A_k(\sigma')\right) = \Pr\left(\bigcup_{i=1}^{m} B_i \middle| A_k(\sigma')\right) \le \sum_{i=1}^{m} \Pr\left(B_i \middle| A_k(\sigma')\right) \le \frac{m}{\exp\left(\frac{\bar{d}_{k,\sigma'}\delta^2}{3}\right)} < \varepsilon < 1$$

onde,

$$\delta = D\sqrt{\frac{3\ln(m/\varepsilon)}{\bar{d}}} \tag{4.14}$$

Logo, um estimador pessimista $U : [0,1]^{pn} \to [0,1)$ pode ser obtido diretamente para as probabilidades condicionais fazendo-se:

$$U\left(A_k(\sigma')\right) = U\left(\hat{x}_1,...,\hat{x}_{k-1}, x_{k\sigma'} = 1 \; e \; x_{k\sigma} = 0 \; para \; \sigma \neq \sigma'\right) = \frac{m}{\exp\left(\frac{\bar{d}_{k\sigma'}\delta^2}{3}\right)}, \; \forall k \in \{1,...,n\} \text{ e } \sigma' \in \Sigma.$$

Substituindo d_{opt} por $\bar{d}$, é fácil ver de (4.5) e (4.14) que: $U(\bar{x}_1,...,\bar{x}_n) = \frac{m}{\exp\left(\frac{\bar{d}\delta^2}{3}\right)} \le \varepsilon < 1$

e, portanto, a condição 1 da definição de estimador pessimista é satisfeita.

A condição 2 também pode ser verificada diretamente pois:

$$\Pr\left(B \middle| A_k(\sigma')\right) \le U\left(A_k(\sigma')\right), \; \forall \; k \in \{1,...,n\} \text{ e } \sigma' \in \Sigma.$$

Finalmente, devemos provar que:

$$U\left(\hat{x}_1,...,\hat{x}_{k-1},\bar{x}_k,...,\bar{x}_n\right) \ge \min_{\sigma' \in \Sigma}\left\{U\left(A_k(\sigma')\right)\right\}, \text{ onde } k \in \{1,...,n\}.$$

De fato, note que:

$$U(\hat{x}_1,...,\hat{x}_{k-1},\bar{x}_k,...,\bar{x}_n) = U(A_{k-1}(\sigma'),\ para\ a\lg um\ \sigma' \in \Sigma) = \frac{m}{\exp\left(\frac{\bar{d}_{k-1,\sigma'}\delta^2}{3}\right)}$$

Como $\bar{d}_{k-1,\sigma'} \leq \bar{d}_{k,\sigma}, \forall \sigma \in \Sigma$, temos em particular que $\bar{d}_{k-1,\sigma'} \leq \max\{\bar{d}_{k,\sigma}, \forall \sigma \in \Sigma\} = \bar{d}_{k,\sigma''}$. Segue então que:

$$U(\hat{x}_1,...,\hat{x}_{k-1},\bar{x}_k,...,\bar{x}_n) = \frac{m}{\exp\left(\frac{\bar{d}_{k-1,\sigma'}\delta^2}{3}\right)} \geq \min\{U(A_k(\sigma)), \forall \sigma \in \Sigma\} = \frac{m}{\exp\left(\frac{\bar{d}_{k,\sigma''}\delta^2}{3}\right)}$$

Observe portanto que a função $U : [0,1]^{pn} \rightarrow [0,1)$ define um estimador pessimista para nossas probabilidades condicionais. Logo, temos o seguinte algoritmo de *derandomização* (vide Seção 3.3.2) sintetizado a seguir.

Entrada: $s_1, s_2, ..., s_m \in \sum^n$

Saída: seqüência $s_H \in \sum^n$ e a distância d_H.

Início

1. **Para** $k = 1,..., n$ **faça**
   a. **Para cada** $\sigma' \in \Sigma$ **faça**
      i. $\hat{x}_{k\sigma'} = 1$
      ii. $\hat{x}_{k\sigma} = 0$, para $\sigma' \neq \sigma$
      iii. Resolver RL(k, σ') e retornar $\bar{d}_{k,\sigma'}$
   Fim Para
   b. Retorna σ'' tal que $d_{k,\sigma''} = \max_{\sigma \in \Sigma}\{\bar{d}_{k\sigma} : \sigma \in \Sigma\}$
   c. $s_H[k] \leftarrow \sigma''$

Fim Para

Fim

Algoritmo 4.3: *Derandomização* de Ben-Dor *et al*. [Ben-Dor *et al*., 1997]

Como cada problema de programação linear tem complexidade $O(n^3L)$ o algoritmo acima terá complexidade total igual a $O(n^4L|\Sigma|)$ onde L representa o número total de *bits* utilizado na entrada do problema de programação linear [Wright, 1997].

Observe agora no caso geral que, para $d_H(\alpha,\beta) \in [0,D]$ e $\alpha,\beta \in \Sigma$ onde $D \in Z^+$ (como definido em [Ben-Dor *et al*., 1997]), não poderemos considerar o Lema 3.2 utilizado para construção do estimador pessimista $U : [0,1]^{pn} \rightarrow [0,1)$, como descrito acima. Note que, nas hipóteses do Lema 3.2, deveremos considerar apenas variáveis 0-1, aleatórias e independentes entre si (vide (4.13)). O caso geral pode ser convertido em binário utilizando-se as variáveis $x_{j,\sigma} \in \{0,1\}$, para j=1,...,n e $\sigma \in \Sigma$. Entretanto, as variáveis $x_{j,\sigma}$ para j fixo não são independentes entre si, o que inviabiliza a aplicação do Lema 3.2.

Outra alternativa seria a utilização do Lema 3.3, onde temos:

$$\Pr(X_i \geq (1+\delta)E(X_i) \leq \exp\left(-\frac{E(X_i)\delta^2}{3D^2}\right)$$

onde $X_i = \sum_{j=1}^{n} X_{ij}$ e $X_{ij} = d_H(s_H[j], s_i[j])$, para j=1,...,n.

Embora tenhamos X_{ij} independentes entre si (onde $X_{ij} \in [0,D]$) não conseguimos eliminar $E[X_i]$ através de uma majoração conveniente, não obtendo portanto, um estimador pessimista.

Como trabalho futuro uma outra alternativa é pesquisar o caso geral utilizando-se outras desigualdades (majorações) distintas daquelas apresentadas por Chernoff-Hoeffding.

Capítulo 5

Um algoritmo 4/3(1+ε)-aproximado

5.1 Introdução

Neste capítulo estudamos o algoritmo aproximativo apresentado em [Lanctot *et al*., 1999]. Através de pequenas adaptações nas idéias apresentadas em [Ben-Dor *et al*., 1997] e ainda utilizando a técnica de arredondamento randômico, os autores exibem um fator de aproximação constante independente de n, m e d_{opt}. Basicamente na estratégia adotada por Lanctot, consideram-se apenas as k posições não coincidentes ($k \leq n$) entre as duas piores seqüências de S.

5.2 Algoritmo de Lanctot

Antes de apresentar a abordagem de Lanctot *et al*. [Lanctot *et al*., 1999] para o PSMP, considere a seguinte notação auxiliar.

Seja x uma seqüência qualquer em Σ^n. As subseqüências x' e x" representam respectivamente, os primeiros p caracteres e os últimos n-p caracteres de x. Por exemplo, $x = x'x"$, onde $x' = x[1]x[2]...x[p]$ e $x" = x[p+1]x[p+2]...x[n]$, isto é, x é a concatenação de x' e x".

Considere novamente, a seguinte formulação matemática (simplificada) para o PSMP original:

$$\min d$$
$$s.a \quad \begin{cases} d_H(s, s_i) \leq d, & \forall\, i = 1,..., m \\ s \in \Sigma^n \end{cases}$$

onde Σ^n representa o conjunto de todas as seqüências possíveis dentro do alfabeto Σ, $s_i \in S$ e $S \subseteq \Sigma^n$.

Com o auxílio da notação definida anteriormente, o problema original pode ser reescrito da seguinte forma:

$$\begin{aligned} &\min d \\ &s.a. \quad \{\ d_H\left(s', s'_i\right) + d_H\left(s'', s''_i\right) \leq d\ ,\ \ i = 1, ..., m \end{aligned}$$

onde s', $s'_i \in \Sigma^p$ e s'', $s''_i \in \Sigma^{n-p}$.

Na estratégia adotada por Lanctot *et al*. [Lanctot *et al*., 1999], inicialmente compara-se par a par todas as seqüências do conjunto S e utiliza-se as posições coincidentes das duas "piores" seqüências, s_1 e s_2 como parte da solução heurística a ser obtida. Considerando-se k posições não-coincidentes entre as seqüências s_1 e s_2, pode-se garantir um limite superior de k para o valor da distância ótima para o problema original. Diz-se que o par (s_1,s_2) define o pior par de seqüências de S, se $d_H(s_1,s_2) \geq d_H(s_i,s_j)$, $\forall\ i,j \in \{1,..,m\}$ e $i \neq j$.

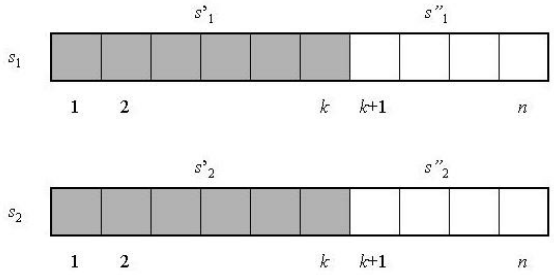


Figura 5.1: Determinação de $d_H(s_1, s_2)$.

Seja $k = d_H(s_1,s_2)$, sem perda de generalidade pode-se assumir que as k diferenças entre s_1 e s_2 ocorrem nas primeiras k posições, isto é, $s_1[j] \neq s_2[j]$ se $1 \leq j \leq k$ e $s_1[j] = s_2[j]$ se $k < j \leq n$ (vide Figura 5.1). Isto é válido porque os caracteres e seqüências do conjunto S podem ser permutados sem que o valor da solução ótima seja alterado. Seja π uma permutação em 1,...,n. Para todo comprimento n de uma seqüência t, pode-se usar t^{π} para denotar a seqüência

$t[\pi(1)]\ t[\pi(2)]...t[\pi(n)]$. Claramente, s_{opt} é uma solução ótima para as seqüências (s_1, ..., s_m) se, e somente se, s^{π} é uma solução ótima para as seqüências ($s_1^{\pi},...,s_m^{\pi}$) com o mesmo custo d.

Na estratégia apresentada por Lanctot *et al*. [Lanctot *et al*., 1999], introduz-se inicialmente o seguinte modelo de programação linear inteira auxiliar:

$$d_{1,opt} = \min d \qquad (5.1)$$

$$s.a \quad \begin{cases} \sum_{\sigma \in \Sigma} y_{j,\sigma} = 1 & , j = 1..k \quad (5.2) \\ \sum_{j=1}^{k} \sum_{\sigma \in \Sigma} y_{j,\sigma}\, d_H(\sigma, s_i[j]) \le d - d_H(s''_1, s''_i) & , i = 1..m \quad (5.3) \\ y_{j,\sigma} \in \{0,1\} & , j = 1..k\,;\, \sigma \in \Sigma \quad (5.4) \end{cases}$$

Seja $s_{1,opt} = s'_{1,opt}s''_1$ a seqüência que se deseja encontrar (solução ótima), onde $s'_{1,opt}$ é uma seqüência desconhecida de tamanho k. Define-se, para cada caracter $\sigma \in \Sigma$ e cada caracter $s'_{1,opt}[j]$, para $j = 1..k$, uma variável binária $y_{j,\sigma}$, onde $y_{j,\sigma} = 1$ se, e somente se, $s'_{1,opt}[j] = \sigma$ e $y_{j,\sigma} = 0$, caso contrário.

Observe que este modelo é bastante semelhante àquele apresentado por Ben-Dor *et al*. [Ben-Dor *et al*., 1997]. Neste caso entretanto, as desigualdades (5.3) especificam que a distância entre cada seqüência de S e a solução ótima $s'_{1,opt}s''_1$, deve ser menor ou igual a $d_{1,opt}$. As desigualdades (5.4) indicam que somente valores 0 ou 1 podem ser atribuídos às variáveis $y_{j,\sigma}$.

Note que é impossível computar $d_{1,opt}$ eficientemente, a menos que P=NP. Entretanto, pode-se afirmar que para uma solução ótima de (5.1)-(5.4) da forma $s'_{1,opt}s''_1$, tem-se $d_{opt} \le d_{1,opt}$. Além disso, note que $d_H(s_1,s_i) \le k$, $\forall\ i \in \{1,..,m\}$. Logo, se $s'_{1,opt}s''_1$ é uma solução ótima de (5.1)-(5.4), segue que $d_H(s'_{1,opt}s''_1, s_i) \le d_H(s_1,s_i) \le k$, $\forall\ i \in \{1,..,m\}$, ou seja $d_{opt} \le d_{1,opt} \le k$.

O algoritmo de Lanctot *et al*. [Lanctot *et al*., 1999] é também baseado na idéia de arredondamento randômico e irá buscar soluções do tipo $s'_H s''_1$. Ele pode ser descrito sinteticamente como a seguir:

Caso 1: Valor de k "pequeno".

Pode-se usar uma simples busca exaustiva para encontrar uma solução ótima $s'_{1,opt}s''_1$ para (5.1)-(5.4). Serão necessárias $|\Sigma|^k$ seqüências de tamanho k sobre o alfabeto Σ para encontrar uma solução ótima. Cada seqüência pode ser comparada às m seqüências de S em tempo $O(mk)$. O tempo de processamento total será então $O(|\Sigma|^k\, mk)$, polinomial em termos de m.

Caso 2: Valor de k "grande".

Resolve-se o modelo de relaxação linear associado à formulação (5.1)-(5.4). Seja $\bar{d}$ o valor da solução relaxada $\bar{y}_{j,\sigma}$. Claramente pode-se ver que $d_{1,opt} \geq \bar{d}$. A solução relaxada $\bar{y}_{j,\sigma}$ fracionária não representa necessariamente uma solução viável para (5.1)-(5.4). Em seguida, arredondam-se estes valores (arredondamento randômico) para 0 ou 1 visando a determinação de uma solução viável para (5.1)-(5.4). O algoritmo aproximativo é detalhado a seguir (Algoritmo 5.1).

Observe que a solução viável s_H obtida é uma solução heurística obtida através da relaxação linear de (5.1)-(5.4), e portanto, é também uma solução viável para o problema original.

No passo (1) é calculada a maior distância k entre duas seqüências do conjunto S. Se k é um valor pequeno, faz-se uma busca exaustiva para encontrar a solução ótima do problema, passo (2). Caso contrário, no passo (3), resolve-se o modelo de relaxação linear proposto para o PSMP, obtendo-se valores fracionários $\bar{y}_{j,\sigma}$. Estes valores serão utilizados como probabilidades no processo de arredondamento randômico para geração de uma solução inteira. A partir da solução inteira encontrada, compara-se com o resultado obtido no passo (1), retornando em seguida a melhor solução obtida (passo 4).

Observe que a complexidade deste algoritmo no passo 1 será polinomial em m (número de seqüências de S) mesmo quando $k = n$. Neste caso, o procedimento de força bruta, para k pequeno, terá complexidade igual a $O(m^{\ln|\Sigma|})$. No passo 2, para k grande, a complexidade da relaxação linear será $O(n^3L)$ onde L representa o número total de bits utilizados na entrada de

dados (para maiores detalhes sobre a complexidade do problema de programação linear, vide [Wright, 1997]). Segue portanto que a complexidade total será $O(\max\{m^{\ln|\Sigma|}; n^3 L\})$.

Entrada: $s_1, s_2, ..., s_m \in \sum^n$, $\delta > 1$ e $0 < \varepsilon < 1$ (para δ e ε constantes).

Saída: seqüência $s_H \in \sum^n$.

Início

1. Determinar $k = d_H(s_1,s_2) = \max\{d_H(s_i,s_j) / i \neq j \text{ e } i,j \in \{1,..,m\}\}$,
2. **Se** valor de k pequeno **então**

   Executar busca exaustiva {Caso 1}
3. **Senão** {Caso 2}
   1. Resolver o modelo de relaxação linear de (5.1)-(5.4), retornando os valores relaxados $\bar{y}_{j\sigma}$ e $\bar{d}$
   2. Construir seqüência s' a partir dos valores relaxados $\bar{y}_{j\sigma}$

      Para $j = 1, ..., k$ **faça**

      $\Pr(s'[j] = \sigma) = \bar{y}_{j,\sigma}$, onde $\sigma \in \Sigma$
   3. Calcular $d_H = \max\{d_H(s's''_1, s_i)\}$, para i=1..m
   4. Retornar $s_H = s_1$ se $d_H(s_1,s_2) \leq d_H$ ou $s_H = s's''_1$ caso contrário.

Fim

Algoritmo 5.1: Algoritmo 4/3(1+ε)-aproximado

5.3 Análise de Aproximação

A idéia central nesta abordagem é utilizar a formulação auxiliar (5.1)-(5.4) na solução do problema original, apoiando-se nos mesmos passos da heurística anterior; relaxação linear e arredondamento randômico. Observa-se entretanto, que a relaxação linear e o arredondamento randômico serão aqui aplicados no subconjunto de k posições, diferentemente da heurística de [Ben-Dor *et al*., 1997] que eram aplicados indistintamente para todas as n posições.

Logo, se $s_H = s's''_1$ é uma solução randômica $(1+\varepsilon)$-aproximada de (5.1)-(5.4) para algum $\varepsilon > 0$ (vide Figura 5.2), segue então que: $E(d_H) \leq (1+\varepsilon)d_{1,opt}$, onde d_H é variável aleatória representando o valor da solução heurística. Equivalentemente, para algum $\varepsilon > 0$, tem-se que $\Pr(d_H > (1+\varepsilon)d_{1,opt}) < 1$, em uma iteração do Algoritmo 5.1.

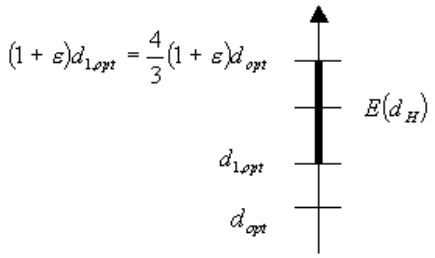


Figura 5.2: Razão de aproximação para o PSMP

Considere agora a seguinte proposição:

Teorema 5.1: $d_{1,opt} \leq 4/3 d_{opt}$

Prova: Suponha inicialmente:

$$d_{1,opt} \leq \beta d_{opt}, \text{ para algum } \beta \geq 1. \tag{5.5}$$

Considere agora $d_H(s_1,s_2) > \beta d_{opt}$ para algum $\beta \geq 1$ (caso contrário, se $d_H(s_1,s_2) \leq \beta d_{opt}$ basta fazer $s_H = s_1$ no Algoritmo 5.1 implicando diretamente em $d_{1,opt} \leq 4/3 d_{opt}$). Logo, como $d_H(s''_1,s''_2) = 0$, segue que:

$$d_H(s_1,s_2) = d_H(s'_1,s'_2) > \beta d_{opt}. \tag{5.6}$$

Pela desigualdade triangular, conclui-se que:

$$d_H\left(s'_1,s'_2\right) \leq d_H\left(s'_1,s'_{opt}\right) + d_H\left(s'_2,s'_{opt}\right) \tag{5.7}$$

onde $s_{opt} = s'_{opt}s''_{opt}$ é solução ótima do problema original. Portanto, substituindo (5.7) em (5.6) obtém-se:

$$d_H\left(s'_1, s'_{opt}\right) + d_H\left(s'_2, s'_{opt}\right) > \beta d_{opt}$$

Sem perda de generalidade, seja $d_H(s'_1, s'_{opt}) > (\beta/2)d_{opt}$. Como $d_H(s'_1, s'_{opt}) + d_H(s''_1, s''_{opt}) \leq d_{opt}$, conclui-se diretamente que:

$$d\left(s''_1, s''_{opt}\right) \leq d_{opt} - \frac{\beta}{2} d_{opt} = \left(\frac{2-\beta}{2} d_{opt}\right) \tag{5.8}$$

Agora, da desigualdade triangular:

$$d_H\left(s''_1, s''_i\right) \leq d_H\left(s''_1, s''_{opt}\right) + d_H\left(s''_{opt}, s''_i\right)$$

$$d_H\left(s''_{opt}, s''_i\right) \geq d_H\left(s''_1, s''_i\right) - d_H\left(s''_1, s''_{opt}\right), \ \forall\ i=1,...,m. \tag{5.9}$$

Como:

$$d_H\left(s_{opt}, s_i\right) = d_H\left(s'_{opt}, s'_i\right) + d_H\left(s''_{opt}, s''_i\right)$$

$$d_H\left(s'_{opt}, s'_i\right) = d_H\left(s_{opt}, s_i\right) - d_H\left(s''_{opt}, s''_i\right), \ \forall\ i=1,...,m. \tag{5.10}$$

Substituindo (5.9) em (5.10),

$$d_H\left(s'_{opt}, s'_i\right) \leq d_H\left(s_{opt}, s_i\right) - \left(d_H\left(s''_1, s''_i\right) - d_H\left(s''_1, s''_{opt}\right)\right)$$

$$d_H\left(s'_{opt}, s'_i\right) \leq d_H\left(s_{opt}, s_i\right) - d_H\left(s''_1, s''_i\right) + d_H\left(s''_1, s''_{opt}\right), \ \forall\ i=1,...,m \tag{5.11}$$

Como $d_H\left(s_{opt}, s_i\right) \leq d_{opt}$, substituindo (5.8) em (5.11) obtém-se:

$$d_H\left(s'_{opt}, s'_i\right) \leq d_{opt} + \left(\frac{2-\beta}{2}\right)d_{opt} - d_H\left(s''_1, s''_i\right)$$

Portanto:

$$d_{1,opt} \leq d_H\left(s'_{opt}\, s''_1, s_i\right) = d_H\left(s'_{opt}, s'_i\right) + d_H\left(s''_1, s''_i\right) \leq \left(\frac{4-\beta}{2}\right)d_{opt} \quad , \forall\ i=1,...,m.$$

Logo, $d_{1,opt} \leq \left(\frac{4-\beta}{2}\right)d_{opt}$ (5.12)

Como $\beta \geq 1$, segue de (5.5) e (5.12) que:

$$\beta = \frac{4-\beta}{2} = 4/3$$ ■

Lembre-se que o objetivo inicial é provar que $\Pr(d_H > (1+\varepsilon)d_{1,opt}) < 1$, para algum $\varepsilon > 0$. Assim, sem perda de generalidade, do Teorema anterior considere $d_{1,opt} = 4/3 d_{opt}$. Portanto, se B representa um *mau* evento e $\varepsilon > 0$, deve-se provar que:

$$\Pr(B) = \Pr\left(\ d_H > \frac{4}{3}(1+\varepsilon)d_{opt}\ \right) = \Pr\left(d_H > (1+\varepsilon)d_{1,opt}\right) < 1$$

Seja B_i um evento representando falha de uma solução $s_H = s's''_1$ com relação a uma determinada seqüência s_i, ou seja, B_i ocorre sempre que:

$$d_H\left(s', s'_i\right) > (1+\varepsilon)d_{1,opt} - d_H\left(s''_1, s''_i\right), \text{ para } i \in \{1,...,m\}.$$

Pode-se dizer que um *mau* evento B ocorre, sempre que B_i ocorre para pelo menos um índice $i=1,...,m$. Em outras palavras, espera-se que, para uma seqüência $s_i \in S$ e $\delta > 1$:

$$\Pr\left(B_i\right) \leq \frac{1}{\delta m}, \text{ para } i = 1..m \tag{5.13}$$

Desta forma:

$$\Pr(B) = \Pr\left(\bigcup_{i=1}^{m} B_i\right) \leq \frac{1}{\delta} < 1.$$

Observe, equivalentemente, que a probabilidade de sucesso $\Pr\left(\bigcap_{i=1}^{m} \overline{B_i}\right) > 0$, será sempre estritamente positiva, onde $\overline{B_i}$, indicando sucesso, representa o evento complementar de B_i.

Note ainda que, com alta probabilidade, pode-se garantir através do método de Monte Carlo que:

$$d_H(s', s'_i) \leq d_{1,opt} + \frac{4}{3}\varepsilon d_{opt} - d_H(s''_1, s''_i), \forall\ i=1,...,m.$$

Considere agora o seguinte resultado auxiliar:

Lema 5.1: $\mu_i = \mathrm{E}[d_H(s', s'_i)] \leq d_{1,opt} - d_H(s''_1, s''_i), \forall\ i=1,...,m.$

Prova: Considere a seguinte notação:

Seja $Y_i = d_H\left(s', s'_i\right) = \sum_{j=1}^{k} d_H\left(s'[j], s_i[j]\right)$. Note que as variáveis Y_i, para i=1..m, podem ser vistas como somas de variáveis aleatórias independentes. Assim:

$$E\left[d_H\left(s', s'_i\right)\right] = E\left[\sum_{j=1}^{k} d_H\left(s'[j], s_i[j]\right)\right]$$

$$= \sum_{j=1}^{k} \sum_{\sigma \in \Sigma} \Pr\left(s'[j] = \sigma\right) d_H\left(\sigma, s_i[j]\right)$$

$$= \sum_{j=1}^{k} \sum_{\sigma \in \Sigma} \overline{y}_{j,\sigma}\ d_H\left(\sigma, s_i[j]\right)$$

$$E\left[d_H\left(s', s'_i\right)\right] + d_H\left(s''_1, s''_i\right) \leq \overline{d} \leq d_{1,opt}$$

Ou seja:

$$\mu_i = \mathrm{E}[d_H(s', s'_i)] \leq d_{1,opt} - d_H(s''_1, s''_i), \text{ para } i = 1, \ldots, m. \qquad \blacksquare$$

Como $d_{1,opt} = 4/3 d_{opt} \leq k$, tem-se da desigualdade de Chernoff-Hoeffding (Lema 3.2) e do Lema anterior que:

$$\begin{aligned}
\Pr(B_i) &= \Pr\left(d_H(s', s'_i) > d_{1,opt} - d_H(s''_1, s''_i) + \frac{4}{3}\varepsilon d_{opt} \right) \\
&\leq \Pr\left(d_H(s', s'_i) > \mu_i + \varepsilon d_{opt} \right) \\
&< \exp\left(-\frac{1}{3} d_{opt} \varepsilon^2 \right), \quad \forall\, i = 1, \ldots, m. \qquad (5.14)
\end{aligned}$$

De (5.13) e (5.14) espera-se que:

$$\frac{1}{\delta m} > \frac{1}{\exp\left(\frac{1}{3} d_{opt} \varepsilon^2\right)} \quad \Rightarrow \quad \exp\left(\frac{1}{3} d_{opt} \varepsilon^2\right) > \delta m \quad \Rightarrow \quad \frac{1}{3} d_{opt} \varepsilon^2 > \ln(\delta m)$$

Finalmente, se $\Pr(B) < 1$ então:

$$d_{opt} > \frac{3\ln(\delta m)}{\varepsilon^2} \qquad (5.15)$$

Portanto, como $k \leq 2 d_{opt}$ (pois $s_1 \in S$ é solução 2-aproximada) temos que, se:

$$d_{opt} \geq \frac{k}{2} > \frac{3\ln(\delta m)}{\varepsilon^2}$$

temos que (5.15) será verdadeira sempre que $k > \frac{6\ln(\delta m)}{\varepsilon^2}$.

Desta forma, garante-se Pr(B) = $1/\delta < 1$ (probabilidade de um *mau* evento) sempre que $k = d_H(s_1,s_2)$ for maior que $6\ln(\delta m)/\varepsilon^2$. Caso contrário, executa-se o procedimento exato de complexidade polinomial.

Apesar das heurísticas de Ben-Dor et al. (Capítulo 4) e Lanctot et al., apresentado neste capítulo, serem diferentes em suas razões de aproximação, existe um ponto comum entre os dois modelos utilizados. Isto ocorre quando em Lanctot et al. $k = n$, ou seja, a pior distância entre duas seqüências do conjunto S corresponde ao próprio tamanho da seqüência. Logo, pode-se dizer que os dois modelos utilizados são coincidentes. Desta forma, como Ben-Dor et al. trabalha com d_{opt}, isto é, com o problema original, pode-se considerar, neste caso específico, que em Lanctot et al. tem-se um esquema de aproximação polinomial, visto que, $d_{1,opt} = d_{opt}$. Entretanto, para aquelas instâncias onde $k < n$ vale a relação $d_{1,opt} \leq 4/3 d_{opt}$.

A análise de *derandomização* no algoritmo de [Lanctot *et al*., 1999] pode ser realizada analogamente àquela desenvolvida no capítulo anterior para o algoritmo de Ben-Dor et al.

Capítulo 6

Esquema de aproximação polinomial para o PSMP

6.1 Introdução

No esquema de aproximação para o PSMP, foram utilizadas algumas das características de cada um dos algoritmos anteriores (Capítulo 4 e Capítulo 5). Uma descrição simplificada da heurística é dada a seguir. Dado um subconjunto r (r fixo) de seqüências $s_{i_1},...,s_{i_r}$ do conjunto $S = \{s_1,s_2,...,s_m\}$, considere os caracteres onde todas as r seqüências coincidem. Intuitivamente, existe uma grande chance de que estes caracteres, nas suas respectivas posições, estejam próximos da solução ótima. É possível mostrar que isto ocorre para pelo menos um subconjunto de r seqüências. Assim, considerando esses caracteres como parte da solução heurística, aplica-se à otimização apenas as posições onde as r seqüências não coincidem. Muito parecida com a estratégia de [Lanctot et al.,1999], a principal diferença está em como [Li *et al*., 2002] definem inicialmente a parte fixa da solução heurística (parte pré-determinada). No entanto, deve ser observado que o esquema de aproximação exige um maior tempo de processamento, comparado aos algoritmos anteriores, necessário para garantir a aproximação e a probabilidade de sucesso desejada. Será mostrado um algoritmo com razão de aproximação $1+O(1/r)$ e complexidade polinomial igual a $m^{O(r)}$.

6.2 Algoritmo de Li et al. [Li et al., 2002]

Considere r seqüências $s_{i_1},...,s_{i_r}$ (onde $r < m$) de S escolhidas arbitrariamente. O conjunto de posições coincidentes nas r seqüências será representado por $Q_{i_1,...,i_r}$ (ou simplesmente Q_r). Por outro lado, o conjunto complementar de posições, será representado por $P_{i_1,...,i_r} = \{1,...,n\} - Q_{i_1,...,i_r}$ (ou simplesmente P_r). Similar ao algoritmo de [Lanctot et al.,1999] (Capítulo 5) pode-se assumir o conjunto P_r de posições como sendo as primeiras $|P_r|$

posições das seqüências de S. Conseqüentemente, o conjunto Q_r representa as últimas $|Q_r|$ posições das seqüências de S.

Seja $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_p\}$ um alfabeto finito de símbolos ($p \geq 2$), $S = \{s_1, s_2, ..., s_m\}$ um conjunto de seqüências, onde cada seqüência $s_i \in \Sigma^n$, e seja $s_H = ys_{i_1}\big|_{Q_r}$ a seqüência que se deseja encontrar (solução heurística), onde y é uma seqüência desconhecida de tamanho $|P_r|$. Define-se, para cada caracter $\sigma \in \Sigma$ e cada caracter $y[j]$, para $j = 1..|P_r|$, uma variável binária $y_{j,\sigma}$, onde $y_{j,\sigma}=1$ se, e somente se, $y[j]=\sigma$, caso contrário $y_{j,\sigma}=0$.

Li *et al*. [Li *et al*., 2002] introduzem inicialmente o seguinte modelo de programação linear inteira auxiliar:

$$d_{Q_r,opt} = \min d \tag{6.1}$$

$$s.a \begin{cases} \sum_{\sigma \in \Sigma} y_{j,\sigma} = 1 & , j = 1..|P_r| \quad (6.2) \\ \sum_{j=1}^{|P_r|} \sum_{\sigma \in \Sigma} y_{j,\sigma}\, \chi(\sigma, s_i[j]) \leq d - d_H\left(s_{i_1}\big|_{Q_r}, s_i\big|_{Q_r}\right) & , i = 1..m \quad (6.3) \\ y_{j,\sigma} \in \{0,1\} & , j = 1..|P_r|; \sigma \in \Sigma \quad (6.4) \end{cases}$$

No modelo apresentado, (6.1) representa a função objetivo que se deseja minimizar. As restrições (6.2) asseguram que somente um símbolo $\sigma \in \Sigma$ será selecionado em cada uma das $|P_r|$ posições da seqüência y. As desigualdades (6.3) especificam que a distância entre cada seqüência de S e $ys_{i_1}\big|_{Q_r}$, deve ser menor ou igual a d. Além disso, $\chi(\sigma, s_i[j]) = 0$, se $s_i[j] = \sigma$ e $\chi(\sigma, s_i[j]) = 1$, caso contrário. Finalmente, as restrições (6.4) indicam que somente valores 0 ou 1 podem ser atribuídos às variáveis $y_{j,\sigma}$.

Note que é impossível computar $d_{Q_r,opt}$ eficientemente, a menos que P=NP. Entretanto, pode-se afirmar que para uma solução ótima de (6.1)-(6.4) da forma $ys_{i_1}\big|_{Q_r}$, tem-se $d_{opt} \leq d_{Q_r,opt}$.

O algoritmo de Li *et al*. [Li *et al*., 2002], pode ser descrito sinteticamente como a seguir:

Para cada subconjunto de r seqüências $s_{i_1},...,s_{i_r}$ de S, executam-se as etapas a seguir. Primeiramente, descobre-se o conjunto de posições Q_r, onde as r seqüências coincidem. Conseqüentemente, obtém-se o conjunto P_r de posições, onde as r seqüências diferem.

Caso 1: Valor de $|P_r|$ "pequeno".

Pode-se usar uma simples busca exaustiva para encontrar uma solução ótima $ys_{i_1}\big|_{Q_r}$ para (6.1)-(6.4). Serão necessárias $|\Sigma|^{|P_r|}$ seqüências de tamanho $|P_r|$ sobre o alfabeto Σ para encontrar uma solução ótima. Cada seqüência pode ser comparada às m seqüências de S em tempo $O(m|P_r|)$. Como veremos adiante $|P_r|$ será igual a $O(\ln m)$ e portanto, o tempo de processamento total será $O(m^{\ln|\Sigma|}\, m\ln m)$, polinomial em termos de m e n.

Caso 2: Valor de $|P_r|$ "grande".

Resolve-se o modelo de relaxação linear associado à formulação (6.1)-(6.4). Seja $\bar{d}$ o valor da solução relaxada $\bar{y}_{j,\sigma}$, não representando necessariamente uma solução viável para (6.1)-(6.4). Em seguida, arredondam-se estes valores (arredondamento randômico) para 0 ou 1 visando a determinação de uma solução viável para (6.1)-(6.4). O algoritmo aproximativo é detalhado a seguir.

Observe que a solução s_H obtida é uma solução heurística do problema auxiliar (6.1)-(6.4), e portanto, uma solução para o problema original.

O passo (1) será executado $\binom{m}{r}$ vezes, onde cada iteração representa um subconjunto de r seqüências do conjunto S. No passo (1.a), encontra-se o conjunto Q_r de posições onde as r seqüências coincidem. Conseqüentemente, obtém-se o conjunto P_r. Se $|P_r|$ é um valor pequeno, faz-se uma busca exaustiva para encontrar a solução ótima para o problema auxiliar (6.1)-(6.4), passo (1.b). Caso contrário, no passo (1.c), resolve-se o modelo de relaxação linear proposto para o PSMP, obtendo-se valores fracionários $\bar{y}_{j,\sigma}$. Estes valores serão utilizados como probabilidades no processo de arredondamento randômico para geração de uma solução inteira. Para cada iteração do passo (1), armazena-se sempre a melhor solução. Na etapa (2), cada seqüência do conjunto S será considerada solução do problema, desta

forma, obtém-se a melhor solução dentre as seqüências do conjunto *S*. Finalmente, na terceira e última etapa, retorna-se a melhor solução das duas etapas anteriores (1) e (2).

Entrada: $s_1, s_2, ..., s_m \in \sum^n$, um inteiro $r \geq 2$ e um número arbitrariamente pequeno $\varepsilon > 0$.

Saída: seqüência $s_H \in \sum^n$.

Início

1. **Para** cada subconjunto de r seqüências $\{ s_{i_1}, ..., s_{i_r} \}$ do conjunto S **faça**
   a. Encontrar o conjunto Q_r e P_r. Onde
      $$Q_r = \left\{ j \mid s_{i_1}[j] = ... = s_{i_r}[j] \; e \; j \in \{1,...,n\} \right\}, \; P_r = \{1...n\} - Q_r$$
   b. **Se** $|P_r|$ pequeno **então**
      Utilizar a estratégia da busca exaustiva. {Caso 1}
   c. **senão** {Caso 2}
      i. Resolver a relaxação do modelo (6.1)-(6.4). Utilizar a relaxação como probabilidade para o arredondamento randômico, encontrando uma seqüência s' de tamanho $|P_r|$.
      ii. Seja a seqüência s_H composta de $s_H|_{Q_r} = s_{i_1}|_{Q_r}$ e $s_H|_{P_r} = y$. Calcular a maior distância d_{H1} de s_H a s_i (para $i = 1..m$) onde s_H é solução heurística.
      iii. Atualiza melhor solução.
2. **Para** $i = 1..m$ **faça**
   Calcula a melhor distância d_{H2} utilizando-se uma seqüência $s_H = s_i$ do conjunto S.
3. Retorna a melhor solução $d_H = \min \{d_{H1}, d_{H2}\}$ obtida nas etapas acima.

Fim.

Algoritmo 6.1: Esquema de aproximação polinomial para o PSMP

6.3 Análise de Aproximação

A idéia central nesta abordagem é utilizar a formulação auxiliar (6.1)-(6.4) na solução do problema original, apoiando-se nas mesmas idéias apresentadas nas heurísticas anteriores; relaxação linear e arredondamento randômico. Similar à heurística de [Lanctot et al.,1999], que utiliza relaxação e o arredondamento randômico em um subconjunto de *k* posições, [Li *et al*., 2002] aplicam as mesmas idéias no conjunto P_r de posições. Logo, para algum subconjunto de índices $\{i_1,..., i_r\} \subset \{1,...,n\}$ escolhido convenientemente, se $s_H = ys_{i_1}\big|_{Q_r}$ é uma solução randômica (1+ε)-aproximada de (6.1)-(6.4) para algum $\varepsilon > 0$ (vide Figura 6.1), segue então que: $E(d_H) \leq (1+\varepsilon)\, d_{Q_r,opt}$, onde d_H = max $\{d_H(\, ys_{i_1}\big|_{Q_r}\, ,\, s_i);\ i=1..m\}$ é variável aleatória representando o valor da função objetivo. Equivalentemente, para algum $\varepsilon > 0$, pode-se provar que em uma iteração do Algoritmo 6.1:

$$\Pr(d_H > (1+\varepsilon)\, d_{Q_r,opt}) \leq \Pr(d_H > (1+\varepsilon+\rho)\, d_{opt}) < 1, \text{ para algum } \rho \in [0,1].$$

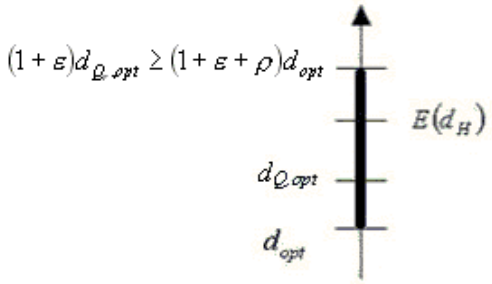


Figura 6.1: Razão de aproximação do PTA para o PSMP

Antes de provar o resultado acima, será garantida, inicialmente, a existência de índices $i_1, i_2, ..., i_r$ tal que os caracteres do conjunto Q_r representem uma boa aproximação da solução $s_{opt}\big|_{Q_r}$ (solução ótima restrita a Q_r), onde para todo $s_l \in S$,

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq \frac{1}{2r-1} d_{opt} \qquad (6.5)$$

Para mostrar (6.5), s_{i_1} e s_{opt} não serão comparadas diretamente em Q_r. Na verdade, considera-se o conjunto $J(l)=\left\{j \in Q_{i_1,i_2,\ldots,i_r} \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_{i_1}[j] \neq s_{opt}[j]\right\}$, para todo $s_l \in S$. Desta forma, será mostrado que para cada $s_l \in S$, $|J(l)| \leq 1/(2r\text{-}1)$.

Seja $p_{i_1,i_2,\ldots,i_k} = d_H\left(s_{i_1}\big|_{Q_k}, s_{opt}\big|_{Q_k}\right)$, para qualquer $2 \leq k < r$ e $1 \leq i_1, i_2, \ldots, i_k \leq m$. Define-se ainda $\rho_k = \min_{1\leq i_1,\ldots,i_k \leq m} \frac{p_{i_1,\ldots,i_k}}{d_{opt}}$. Note que, aumentando k, o número de posições onde todos s_{i_j} diferem (para j=1,...,k), diminui. Conseqüentemente, o conjunto Q_k diminui, e portanto, ρ_k também diminui.

A estratégia é mostrar que:

1) $|J(l)| \leq (\rho_k - \rho_{k+1})d_{opt}$, para todo $s_l \in S$ e $2 \leq k \leq r$.
2) Pelo menos uma das sentenças $\rho_2\text{-}\rho_3$, $\rho_3\text{-}\rho_4$, ..., $\rho_r\text{-}\rho_{r+1}$ é no máximo $1/(2r\text{-}1)$.

Seja $\rho_0 = \max_{1\leq i,j\leq m} \frac{d_H(s_i, s_j)}{d_{opt}}$, pela desigualdade triangular $\rho_0 \leq 2$ (no pior caso). O Lema a seguir, garante a desigualdade (6.5) para $\rho_0 > 1+1/(2r\text{-}1)$, isto é, quando a pior distância entre duas seqüências de S não encontra um bom resultado, representado aqui por $(1+1/(2r\text{-}1))d_{opt}$. Caso contrário, simplesmente utiliza-se como resultado a pior distância entre duas seqüências de S.

Lema 6.1: Para um r constante, onde $2 \leq r < n$, se $\rho_0 > 1+1/(2r\text{-}1)$, então existem índices $i_1, \ldots, i_r$, tal que para todo $1 \leq l \leq n$,

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq \frac{1}{2r-1} d_{opt}$$

Prova: Primeiramente, considere o seguinte resultado auxiliar, Parte 1.

Parte 1 Para todo valor de k, onde $2 \leq k \leq r$ e r constante, existem índices $1 \leq i_1, i_2, \ldots, i_r \leq m$, tal que para todo $s_l \in S$,

$$|J(l)| \leq (\rho_k - \rho_{k+1}) d_{opt}.$$

Prova: Considere os índices $1 \leq i_1, ..., i_k \leq m$, onde $p_k = \rho_k \, d_{opt}$. Note que isso sempre será verdade, pois para todas as combinações de $\binom{n}{k}$ existirão índices $1 \leq i_1, ..., i_k \leq m$, onde $p_k = \rho_k \, d_{opt}$. Então, para todo $1 \leq i_{k+1}, i_{k+2}, ..., i_r \leq m$ e $1 \leq l \leq m$, tem-se:

$$\begin{aligned}
|J(l)| &= \left| \left\{ j \in Q_{i_1,...,i_r} \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_{i_1}[j] \neq s_{opt}[j] \right\} \right| \\
&\leq \left| \left\{ j \in Q_{i_1,...,i_k} \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_{i_1}[j] \neq s_{opt}[j] \right\} \right| \qquad (6.6) \\
&= \left| \left\{ j \in Q_{i_1,...,i_k} \,\middle|\, s_{i_1}[j] \neq s_{opt}[j] \right\} - \left\{ j \in Q_{i_1,...,i_k} \,\middle|\, s_{i_1}[j] = s_l[j] \; e \; s_{i_1}[j] \neq s_{opt}[j] \right\} \right| \\
&= \left| \left\{ j \in Q_{i_1,...,i_k} \,\middle|\, s_{i_1}[j] \neq s_{opt}[j] \right\} - \left\{ j \in Q_{i_1,...,i_k,l} \,\middle|\, s_{i_1}[j] \neq s_{opt}[j] \right\} \right| \\
&= \left| \left\{ j \in Q_{i_1,...,i_k} \,\middle|\, s_{i_1}[j] \neq s_{opt}[j] \right\} \right| - \left| \left\{ j \in Q_{i_1,...,i_k,l} \,\middle|\, s_{i_1}[j] \neq s_{opt}[j] \right\} \right| \qquad (6.7) \\
&= p_{i_1,...,i_k} - p_{i_1,...,i_k,l} \\
|J(l)| &\leq (\rho_k - \rho_{k+1}) d_{opt}
\end{aligned}$$

Observe que a desigualdade (6.6) se baseia em $Q_r \subseteq Q_k$. A igualdade (6.7) pode ser justificada porque o conjunto $\{ j \in Q_{k,l} \,|\, s_{i_1}[j] \neq s_{opt}[j] \} \subseteq \{ j \in Q_k \,|\, s_{i_1}[j] \neq s_{opt}[j] \}$. ■

Considere agora o limite superior para r-1 termos, $\min\{\rho_2-\rho_3, \rho_3-\rho_4, ..., \rho_r-\rho_{r+1}\}$. Somando-se os r-1 termos, têm-se: $(\rho_2-\rho_3) + (\rho_3-\rho_4) + ... + (\rho_r-\rho_{r+1}) = \rho_2 - \rho_{r+1} \leq \rho_2$. Se $\rho_k = \min p_k/d_{opt}$ e $p_k \leq d_{opt}$, tem-se $\rho_k \leq 1$. Logo, $\rho_2 \leq 1$.

Portanto, pelo menos um dos termos $(\rho_2-\rho_3) + (\rho_3-\rho_4) + ... + (\rho_r-\rho_{r+1})$ será $\leq 1/(r\text{-}1)$. Nota-se que $1/(r\text{-}1)$ já representa um limite superior para os r-1 termos. No entanto, [Li *et al*., 2002] mostram um limite melhor considerando a média ponderada de r termos, como a seguir.

Parte 2 Para $2 \le r < m$, $\min\{\rho_0\text{-}1, \rho_2\text{-}\rho_3, \rho_3\text{-}\rho_4, ..., \rho_r\text{-}\rho_{r+1}\} \le 1/(2r\text{-}1)$.

Prova: Considerando, $1 \le i,j \le n$, onde $d_H(s_i,s_j) = \rho_0\, d_{opt}$ (pior caso), e o conjunto de posições onde s_i e s_j diferem (conjunto P_r), pela desigualdade triangular, tem-se:

$$d_H\left(s_i\big|_{P_r}, s_j\big|_{P_r}\right) \le d_H\left(s_i\big|_{P_r}, s_{opt}\big|_{P_r}\right) + d_H\left(s_j\big|_{P_r}, s_{opt}\big|_{P_r}\right)$$

$$\rho_0 d_{opt} \le d_H\left(s_i\big|_{P_r}, s_{opt}\big|_{P_r}\right) + d_H\left(s_j\big|_{P_r}, s_{opt}\big|_{P_r}\right)$$

Sem perda de generalidade, seja $d_H\left(s_i\big|_{P_r}, s_{opt}\big|_{P_r}\right) \ge (\rho_0/2) d_{opt}$. Logo, no conjunto de posições onde s_i e s_j coincidem (conjunto Q_r), as diferenças entre s_i e s_{opt} serão dadas por:

$$d_H\left(s_i\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) + d_H\left(s_i\big|_{P_r}, s_{opt}\big|_{P_r}\right) = d_H\left(s_i, s_{opt}\right)$$

$$d_H\left(s_i\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \le d_{opt} - (\rho_0/2) d_{opt}$$

$$d_H\left(s_i\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \le (1 - \rho_0/2) d_{opt} \qquad (6.8)$$

Assim, seja $\rho_k = \min p_k/d_{opt}$, onde $p_k = d_H\left(s_{i_1}\big|_{Q_r}, s_{opt}\big|_{Q_r}\right)$, de (6.8), tem-se que:

$$\rho_k = \min p_k/d_{opt} \le (1\text{-}\rho_0/2) d_{opt}/d_{opt}$$

$$\rho_k \le 1\text{-}\rho_0/2$$

$$\rho_2 \le 1\text{-}\rho_0/2$$

Portanto, obtém-se da somatória abaixo,

$$\frac{(1/2)(\rho_0 - 1) + (\rho_2 - \rho_3) + (\rho_4 - \rho_5) + ... + (\rho_r - \rho_{r+1})}{(1/2) + r - 1}$$

$$\le \frac{(1/2)\rho_0 + \rho_2 - 1/2}{r - 1/2} \le \frac{1}{2r - 1}$$

Assim, no mínimo um dos termos $\{\rho_0\text{-}1, \rho_2\text{-}\rho_3, \rho_3\text{-}\rho_4, ..., \rho_r\text{-}\rho_{r+1}\}$ é menor ou igual a $1/(2r\text{-}1)$. ■

Se $\rho_0 > 1 + 1/(2r\text{-}1)$, então pela Parte 2, existe no mínimo um k, onde $2 \le k < r$, tal que $\rho_k\text{-}\rho_{k+1} \le 1/(2r\text{-}1)$. Pela Parte 1, $|J(l)| \le 1/(2r\text{-}1)$. Assim, existem no máximo $1/(2r\text{-}1)$ posições em Q_r onde s_l difere de s_{i_1} enquanto coincide com s_{opt}, como demonstrado a seguir.

Considerando o conjunto de posições $Q'_r = \left\{ j \in Q_{i_1,\ldots,i_r} \,\middle|\, s_{i_1}[j] \neq s_l[j] \right\}$, para todo $s_l \in S$, tem-se do conjunto $J(l)$ aplicado em Q'_r que:

$$|J(l)| = \left| \left\{ j \in Q_{i_1,\ldots,i_r} \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_{i_1}[j] \neq s_{opt}[j] \right\} \right| = d_H\left(s_{i_1}\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right)$$

e do conjunto de posições onde s_l difere de s_{i_1} enquanto coincide com s_{opt} (representado por (6.9)) também aplicado em Q'_r que:

$$\left| \left\{ j \in Q_{i_1,\ldots,i_r} \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_l[j] = s_{opt}[j] \right\} \right| \tag{6.9}$$

$$= d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right) = d_H\left(s_l\big|_{Q'_r}, s_{i_1}\big|_{Q'_r}\right) - d_H\left(s_l\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right) \tag{6.10}$$

Observe na Figura 6.2, o detalhamento da passagem de (6.9) para (6.10). A função (6.9) está representada por 6.2(a) e a função (6.10) por 6.2(b).

Pela desigualdade triangular, $d_H\left(s_l\big|_{Q'_r}, s_{i_1}\big|_{Q'_r}\right) - d_H\left(s_l\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right) \le d_H\left(s_{i_1}\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right)$. Logo,

$$\left| \left\{ j \in Q_r \,\middle|\, s_{i_1}[j] \neq s_l[j] \; e \; s_l[j] = s_{opt}[j] \right\} \right| \le |J(l)|$$

Nota-se que para o conjunto Q_r e Q'_r o resultado anterior é o mesmo. Segue, também que $Q'_r \subseteq Q_r$. Conseqüentemente, qualquer distância entre duas seqüências em Q'_r será menor ou igual a distância em Q_r. Portanto, $d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \ge d_H\left(s_l\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right)$.

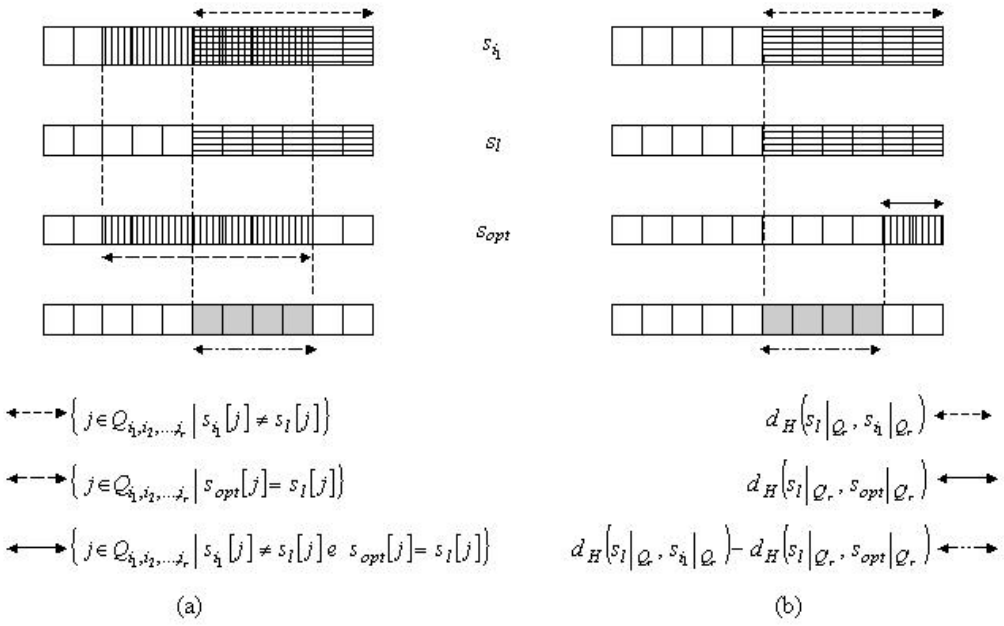


Figura 6.2: (6.9) = (6.10)

Finalmente, tem-se:

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q'_r}, s_{opt}\big|_{Q'_r}\right)$$

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq \left|\left\{ j \in Q_r \mid s_{i_1}[j] \neq s_l[j] \; e \; s_l[j] = s_{opt}[j] \right\}\right|$$

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq |J(l)| \leq \frac{1}{2r-1} d_{opt}$$

$$d_H\left(s_l\big|_{Q_r}, s_{i_1}\big|_{Q_r}\right) - d_H\left(s_l\big|_{Q_r}, s_{opt}\big|_{Q_r}\right) \leq \frac{1}{2r-1} d_{opt} \qquad \blacksquare$$

O Lema 6.1 mostrou como encontrar uma boa aproximação à uma solução ótima no conjunto de posições Q_r para algum i_1, i_2, ..., i_r . O Lema 6.4 mostrará como usar a solução do Lema 6.1 para construir uma boa solução para todo conjunto de posições n, ou seja, uma solução no formato $s_H = ys_{i_1}\big|_{Q_r}$ como descrito inicialmente.

Seja $P_{i_1,\ldots,i_k} = \{1,\ldots,n\} - Q_{i_1,\ldots,i_k}$, considere agora, os seguintes resultados auxiliares.

Lema 6.2: $|P_k| \leq k d_{opt}$

Prova: Considere o conjunto P_k de posições onde as $s_{i_1},...,s_{i_k}$ seqüências diferem. Seja q uma posição em P_k, existe então uma seqüência s_{i_j}, onde $j = 1..k$, tal que $s_{i_j}[q] \neq s_{opt}[q]$, caso contrário teríamos $s_{i_1}[q] = ... = s_{i_k}[q] = s_{opt}[q]$ e portanto q pertenceria a Q_k. Logo,

$$|P_k| \leq \sum_{q=1}^{|P_k|} \sum_{j=1}^{k} d_H\left(s_{i_j}[q], s_{opt}[q]\right)$$

$$= \sum_{j=1}^{k} \sum_{q=1}^{|P_k|} d_H\left(s_{i_j}[q], s_{opt}[q]\right)$$

Como a distância $d_H\left(s_{i_j}, s_{opt}\right) \leq d_{opt}$, onde $j = 1..k$

$$\leq \sum_{j=1}^{k} d_{opt}$$

$$|P_k| \leq k\, d_{opt}$$ ■

Lema 6.3: $\mu_i = \mathrm{E}[d_H(y', s_i|_{P_k})] \leq d_{Q_k, opt} - d_H(s_{i_1}|_{Q_k}, s_i|_{Q_k})$, $\forall\ i=1,...,m$.

Prova: Para cada $1 \leq j \leq |P_k|$, independentemente, $y'_{j,\sigma} = 1$ com probabilidade $\bar{y}_{j,\sigma}$ e $y'_{j,\sigma'} = 0$ para todo $\sigma' \in \Sigma - \{\sigma\}$. Desta forma, assegura-se que apenas um único símbolo $\sigma \in \Sigma$ representará a posição j, onde $y'_{j,\sigma} = 1$. Logo, $\sum_{\sigma\in\Sigma} \chi(\sigma, s_i[j]) y'_{j,\sigma}$ obtém valores 0 ou 1 aleatoriamente. Desde que o arredondamento seja independente para os diferentes j's, $\sum_{\sigma\in\Sigma} \chi(\sigma, s_i[j]) y'_{j,\sigma}$ são variáveis 0-1 aleatórias independentes para $1 \leq j \leq |P_k|$, temos então:

$$E\left[d_H\left(y', s_i|_{P_k}\right)\right] = E\left[\sum_{j=1}^{|P_k|} \sum_{\sigma\in\Sigma} \chi(\sigma, s_i[j]) y'_{j,\sigma}\right]$$

$$= \sum_{j=1}^{|P_k|} \sum_{\sigma\in\Sigma} \chi(\sigma, s_i[j]) E\left[y'_{j,\sigma}\right]$$

$$= \sum_{j=1}^{|P_k|} \sum_{\sigma \in \Sigma} \chi(\sigma, s_i[j]) \bar{y}_{j,\sigma}$$

$$E\left[d_H\left(y', s_i\big|_{P_k}\right)\right] + d_H\left(s_{i_1}\big|_{Q_k}, s_i\big|_{Q_k}\right) \leq \bar{d} \leq d_{Q_k,opt}$$

Ou seja:

$$\mu_i = \mathrm{E}[d_H(y',\ s_i\big|_{P_k})] \leq d_{Q_k,opt} - d_H(s_{i_1}\big|_{Q_k}, s_i\big|_{Q_k}),\ \text{para } i = 1, \ldots, m. \qquad \blacksquare$$

Lema 6.4: Seja $S = \{s_1, s_2, \ldots, s_m\}$, onde $|s_i| = n$ para $i=1,\ldots,m$. Considere s_{opt} uma solução ótima para o problema e $\max_{1 \leq i \leq n} d_H(s_i, s_{opt}) = d_{opt}$. Dada uma seqüência s' e um conjunto de posições Q de tamanho n-$O(d_{opt})$ tal que para cada $i=1,\ldots,m$,

$$d_H\left(s_i\big|_Q, s'\big|_Q\right) - d_H\left(s_i\big|_Q, s_{opt}\big|_Q\right) \leq \rho\, d_{opt} \tag{6.11}$$

onde $0 \leq \rho \leq 1$, pode-se obter uma solução com razão de aproximação $(1 + \rho + \varepsilon)d_{opt}$ em tempo polinomial para um valor $\varepsilon \geq 0$.

Prova: Considere $d_{Q,opt}$ solução ótima de (6.1)-(6.4), temos então (para $i = 1,\ldots,m$):

$$d_H(s_i|_Q, s_{opt}|_Q) + d_H(s_i|_P, s_{opt}|_P) = d_H(s_i, s_{opt})$$
$$d_H(s_i|_P, s_{opt}|_P) = d_H(s_i, s_{opt}) - d_H(s_i|_Q, s_{opt}|_Q)$$

Da desigualdade (6.11),

$$d_H(s_i|_P, s_{opt}|_P) \leq d_H(s_i, s_{opt}) - (d_H(s_i|_Q, s'|_Q) - \rho d_{opt})$$
$$d_H(s_i|_P, s_{opt}|_P) \leq d_{opt} - (d_H(s_i|_Q, s'|_Q) - \rho d_{opt})$$
$$d_H(s_i|_P, s_{opt}|_P) + d_H(s_i|_Q, s'|_Q) \leq d_{opt} + \rho d_{opt}$$
$$d_H(s_i|_P, s_{opt}|_P) + d_H(s_i|_Q, s'|_Q) \leq (1+\rho) d_{opt}$$
$$d_{Q,opt} \leq (1+\rho) d_{opt}$$

Lembre-se que se deseja provar que $\Pr(d_H > (1+\varepsilon)\, d_{Q,opt}) < 1$, para algum $\varepsilon > 0$. Assim, sem perda de generalidade, do resultado acima considere $d_{Q,opt} = (1+\rho)d_{opt}$. Portanto, se B representa um *mau* evento e $\varepsilon > 0$, deve-se provar que:

$$\begin{aligned}\Pr(B) &= \Pr\left(d_H > (1+\varepsilon)d_{Q,opt}\right) = \Pr\left(d_H > d_{Q,opt} + \varepsilon\, d_{Q,opt}\right) \\ &= \Pr\left(d_H > (1+\rho)d_{opt} + \varepsilon\, d_{Q,opt}\right) \\ &\le \Pr\left(d_H > (1+\rho)d_{opt} + \varepsilon\, d_{opt}\right) = \Pr\left(d_H > (1+\rho+\varepsilon)d_{opt}\right) < 1\end{aligned}$$

Desta forma, esperamos que:

$$\Pr(B) = \Pr\left(\bigcup_{i=1}^{m} B_i\right) < 1.$$

Onde B_i é um evento representando falha de uma solução $s_H = ys'\big|_Q$ com relação a uma determinada seqüência s_i, ou seja, B_i ocorre sempre que:

$$d_H\left(y, s_i\big|_P\right) > (1+\varepsilon)d_{Q,opt} - d_H\left(s'\big|_Q, s_i\big|_Q\right), \text{ para } i \in \{1,...,m\}.$$

Como $d_{Q,opt} = (1+\rho)d_{opt}$, segue que:

$$\begin{aligned}\Pr(B_i) &= \Pr\left(d_H\left(y, s_i\big|_P\right) > d_{Q,opt} - d_H\left(s'\big|_Q, s_i\big|_Q\right) + \varepsilon\, d_{Q,opt}\right) \\ &\le \Pr\left(d_H\left(y, s_i\big|_P\right) > d_{Q,opt} - d_H\left(s'\big|_Q, s_i\big|_Q\right) + \varepsilon\, d_{opt}\right)\end{aligned}$$

Pelo Lema 6.2, seja $\varepsilon' = \varepsilon/r$. Logo, $\varepsilon'|P| \le \varepsilon\, d_{opt}$.

$$\le \Pr\left(d_H\left(y, s_i\big|_P\right) > d_{Q,opt} - d_H\left(s'\big|_Q, s_i\big|_Q\right) + \varepsilon'|P|\right)$$

Do Lema 6.3, temos:

$$\le \Pr\left(d_H\left(y, s_i\big|_P\right) > \mu_i + \varepsilon'|P|\right)$$

Finalmente, da desigualdade de Chernoff-Hoeffding (Lema 3.2), temos:

$$\Pr(B_i) < \exp\left(-\frac{1}{3}|P|\varepsilon'^2\right), \quad \forall\ i=1,...,m.$$

Logo, $\Pr(B) < m\exp\left(-\frac{1}{3}|P|\varepsilon'^2\right) < 1$. Portanto:

$$|P| > \frac{(3\ln m)}{\varepsilon'^2}$$

Logo, se $|P| \geq (4\ln m)/\varepsilon'^2$ obtém-se uma solução s_H com custo $d_H \leq d_{Q,opt} + \varepsilon'|P|$ com probabilidade de sucesso $\geq 1 - m^{-1/3}$.

Se $|P| < (4\ln m)/\varepsilon'^2$, a busca exaustiva encontra a solução ótima para (6.1)-(6.4) em tempo polinomial.

Desde que $|P|$ seja da ordem de d_{opt}, onde $|P| \leq rd_{opt}$ para r constante (Lema 6.2), e $\varepsilon'=\varepsilon/r$, podemos obter $s_H = y_0 s'|_Q$ solução heurística para (6.1)-(6.4), onde y_0 é solução inteira de y, para todo $i=1,...,m$, ou seja:

$$\begin{aligned} d_H(s_i, s_H) &= d_H(s_i|_P, y_0) + d_H(s_i|_Q, s'|_Q) \\ &\leq d_{Q,opt} + \varepsilon'|P| \\ &\leq (1+\rho)d_{opt} + \varepsilon d_{opt} \\ &\leq (1 + \rho + \varepsilon)d_{opt} \end{aligned}$$ ■

Agora o algoritmo 6.1 será descrito em detalhes.

Teorema 6.1: O algoritmo 6.1 é um PTAS para o PSMP.

Prova: Dada uma instância do PSMP, seja s_{opt} uma solução ótima com distância d_{opt}, onde $d_{opt} = \max\{d_H(s_{opt}, s_i); i=1..m\}$. Seja P definido no passo (1.a) do algoritmo, logo pelo Lema 6.2, $P = O(d_{opt})$. Note que para r constante, como definido no Lema 6.1, os passos (1.b) e (1.c) são executados em tempo polinomial como provados no Lema 6.4. Obviamente, para r constante, é fácil ver que os outros passos também são executados em tempo polinomial.

Observe que se $\rho_0 \leq 1+1/(2r-1)$, pela definição de ρ_0, o passo (2) encontra uma solução com razão $\rho_0\, d_{opt} \leq 1+1/(2r-1)d_{opt}$. Entretanto, se $\rho_0 > 1+1/(2r-1)$, pelo Lema 6.1 e 6.4, o passo (1) encontra uma razão limitada superiormente por $(1+1/(2r-1)+\varepsilon)d_{opt}$.

Portanto, a razão de aproximação do algoritmo será $(1+1/(2r-1)+\varepsilon)d_{opt}$, provando o teorema. ■

Note por exemplo, que se $r = 2$ temos $\rho_0 = 1/(2r-1) = 1/3$ e portanto, teremos um algoritmo $(4/3+\varepsilon)$-aproximado igualando a razão de aproximação de Lanctot et al.[Lanctot et al.,1999] (vide Capítulo 5).

Conclusões

Neste trabalho, apresentamos alguns conceitos básicos e definições presentes na biologia molecular, visando uma maior compreensão de alguns dos problemas combinatórios mais freqüentes descritos na literatura (capítulo 2). Em particular, concentramos nossa atenção ao problema da seqüência mais próxima (PSMP).

Nosso objetivo foi estudar os principais algoritmos aproximativos existentes na literatura para o PSMP, em sua maioria baseados em métodos probabilísticos. Desta forma, fazemos uma descrição mais detalhada das técnicas de Monte Carlo, arredondamento randômico e *derandomização*, em particular o método das probabilidades condicionais, mostrando os avanços mais recentes obtidos até o momento. Outra contribuição importante foi o desenvolvimento da *derandomização* sugerida por Ben-Dor *et al*.[Ben-Dor *et al*., 1997], onde utilizamos o método dos estimadores pessimistas, determinando assim limitantes superiores para as probabilidades condicionais associadas.

Apresentamos de maneira detalhada, o algoritmo $(4/3+\varepsilon)$-aproximado de Lanctot et al.[Lanctot et al.,1999] e o esquema de aproximação polinomial de Li *et al*.[Li *et al*., 2002].

Como sugestão para trabalhos futuros podemos citar o problema da subseqüência mais próxima (PSSMP), uma extensão do PSMP, utilizando a mesma abordagem apresentada neste trabalho. Dado um conjunto $S = \{s_1,s_2,...,s_m\}$ de seqüências (todas de tamanho maior ou igual a n), sobre um alfabeto Σ, o PSSMP consiste em encontrar uma seqüência s_H de tamanho n, de forma que s_H minimize d onde, para cada $s_i \in S$ e y_i uma subseqüência de tamanho n de s_i, tenhamos $d_H(s_H, y_i) \leq d$, para i=1...m.

Uma outra questão a ser investigada é a utilização dos modelos de programação linear inteira apresentados recentemente por Pardalos *et al*. [Pardalos et al., 2004] combinados com as técnicas de arredondamento randômico e *derandomização*.

Outro aspecto a ser considerado, é a determinação de estimadores pessimistas para um alfabeto Σ (de cardinalidade $|\Sigma| > 2$) onde as distâncias $d_H(\sigma, s_i[j]) \in [0,D]$ para $s_i \in S$ e $D = max_{\alpha,\beta\in\Sigma}\{d_H(\alpha, \beta)\}$.

Neste caso, outras desigualdades (*tail inequalities*) deverão ser pesquisadas na literatura visando a determinação de um limitante superior para as probabilidades condicionais.

Referências Bibliográficas

[Alberts et al., 1994] Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J.D. Molecular Biology of the Cell. New York & London: Garland Publishing.

[Alizadeh et al., 1995] Alizadeh, F., Karp, R.M., Weisser, D.K. and Zweig, G. Physical mapping of chromosomes using unique probes. J. Comput. Biol., 2(2):159-184.

[Almeida & Setúbal, 2003] Almeida, N.F.Jr. e Setúbal, J.C. Ferramentas para comparação genômica, Anais do XXIII Congresso da Sociedade Brasileira de Computação, Campinas, SP, pp 13-20.

[Alon & Spencer, 1992] Alon, N. and Spencer, J. The Probabilistic Method. Wiley, New York.

[Altschul & Lipman, 1989] Altschul, S.F. and Lipman, D.J. Trees, stars, and multiple biological sequence alignment. SIAM Journal on Applied Mathematics, 49(1):197-209.

[Armen & Stein, 1995] Armen, C. and Stein, C. Short superstrings and the structure of overlapping strings. Journal of Computational Biology, 2(2):307-332.

[Armen & Stein, 1996] Armen, C. and Stein, C. A 2 2/3-approximation algorithm for the shortest superstring problem. In proceedings of the Seventh Symposium on Combinatorial Pattern Matching, volume 1075 of Lecture Notes in Computer Science, pages 87-103. Berlin: Springer-Verlag.

[Arora et al., 1995] Arora, S., Karger, D. and Karpinski, M. Polynomial time approximation schemes for dense instances of NP-hard problems. Proc. 27th Ann. ACM Symp. on Theory of Comp. , ACM, 284-293.

[Atlan & Koppel, 1990] Atlan, H. and Koppel, M. The cellular computer DNA: program or data? Bulletin of Mathematical Biology, 52(3):335-348.

[Bafna & Pevzner, 1996] Bafna, V. and Pevzner, P.A. Genome rearrangements and sorting by reversals. SIAM Journal on Computing, 25(2):272-289.

[Bains, 1986] Bains, W. Multan: a program to align multiple DNA sequences. Nucleic Acids Research, 14:159-177.

[Barton & Sternberg, 1987] Barton, G.J. and Sternberg, M.J.E. A strategy for the rapid multiple alignment of protein sequences. Journal of Molecular Biology, 198:327-337.

[Batzoglou, 2000] Batzoglou, S. Computational Genomics: Mapping, Comparison, and Annotation of Genomes. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT.

[Ben-Dor et al, 1997] Ben-Dor, A., Lancia, G., Perone, J. and Ravi, R. Banishing Bias from Consensus Sequences, Combinatorial Pattern Matching, 8th Annual Symposium, Springer-Verlag, Berlin.

[Berman & Hannenhalli, 1996] Berman, P. and Hannenhalli, S. Fast sorting by reversal. In Proceedings of the Seventh Symposium on Combinatorial Pattern Matching, volume 1075 of Lecture Notes in Computer Science, pages 168-185. Berlin: Springer-Verlag.

[Berman et al., 1997] Berman, P., Gumucio, D., Hardison, R., Miler, W. and Stojanovic, N. A linear-time algorithm for the 1-mismatch problem, Workshops on Algorithms and Data Structures, pp. 126-135.

[Blum et al., 1991] Blum, A., Jiang, T., Li, M., Tromp, J. and Yannakakis, M. Linear approximation of shortest superstrings. In Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, pages 328-336.

[Blum et al., 1994] Blum, A., Jiang, T., Li, M., Tromp, J. and Yannakakis, M. Linear approximation of shortest superstrings. Journal of the ACM, 41(4):630-647.

[Booth & Lueker, 1976] Booth, K.S. and Lueker, G.S. Testing of the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences, 13(3):335-379.

[Branden & Tooze, 1991] Branden, C. and Tooze, J. Introduction to protein structure. New York & London: Garland Publishing.

[Brown, 1999] Brown, T.A. Genomes. Wiley-Liss Bros.

[Cantor & Smith, 2000] Cantor, C. and Smith, C.L. Genomics: The Science and Technology Behind the Human Genome Project. Wiley-Interscience, New York.

[Caprara et al., 1995] Caprara, A., Lancia, G. and Ng, S.K. A column-generation-based branch-and-bound algorithm for sorting by reversals. Presented at the 4th DIMACS Implementation Challenge Work-shop.

[Caprara, 1997] Caprara, A. Sorting by reversals is difficult. In proceedings of the First Annual International Conference on Computational Molecular Biology.

[Carrillo & Lipman, 1988] Carrillo, H. and Lipman, D.J. The multiple sequence alignment problem in biology. SIAM J. Appl. Math, 48(5):1073-1082.

[Chao et al., 1994] Chao, K.M., Hardison, R.C. and Miller, W. Recent developments in linear-space alignment methods: a mini survey. Journal of Computational Biology, 1:271-291.

[Cieliebak et al, 2003] Cieliebak, M., Eidenbenz, S. and Penna, P. Noisy Data Make the Partial Digest Problem NP-Hard. In Proc. of the 3rd Workshop on Algorithms in Bioinformatics (WABI 2003), pages 111-123.

[Czumaj et al., 1994] Czumaj, A., Gasienec, L., Piotrow, M. and Rytter, W. Parallel and sequential approximations of shortest superstrings. In Proceedings of the Fourth Scandinavian Workshop on Algorithm Theory, pages 95-106.

[Dayhoff et al., 1978] Dayhoff, M., Schwartz, R.M. and Orcutt, B.C. A model of evolutionary change in proteins. In M.Dayhoff, editor, Atlas of Protein Sequence and Structure, volume 5, pages 345-352. National Biomedical Research Foundation, Silver Spring, MD, Supplement 3.

[Dakic, 2000] Dakic, T. On the turnpike problem. PhD thesis, Simon Fraser University.

[Dean & Staden, 1991] Dean, S. and Staden, R. A sequence assembly and editing program for efficient management of large projects. Nucleic Acids Research, 19(14):3907-3911.

[Devereux et al., 1984] Devereux, J., Haeberli, P. and Smithies, D. A comprehensive set of sequence analysis programs for the VAX. Nucleic Acids Research, 12:387-395.

[Dias, 2002] Dias, Z. Rearranjo de Genomas: Uma Coletania de Artigos. Tese de Doutorado, Unicamp.

[Doolittle et al., 1983] Doolittle, R.F., Hunkapiller, M.W., Hood, L.E., Devare, S.G., Robbins, K.C., Aaronson, S.A. and Antoniades, H.N. Simian sarcoma virus one gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor. Science, 221:275-277.

[Doolittle, 1985] Doolittle, R.F. Proteins. Scientific American, 253(4):74-83.

[Eppstein et al., 1992] Eppstein, D., Galil, Z., Giancarlo, R. and Italiano, G. Sparse dynamic programming I: Linear cost functions. Journal of the ACM, 39:519-545.

[Fauron & Havlik, 1989] Fauron, C. and Havlik, M. The maize mitochondrial genome of the normal type and the cytoplasmic male sterile type have very different organization. Current Genetics, 15:149-154.

[Fellows et al., 1993] Fellows, M.R., Hallett, M.T. and Wareham, H.T. DNA physical mapping: Three ways difficult. In Proceedings of the First Annual European Symposium on Algorithms, volume 726 of Lecture Notes in Computer Science, pages 157-168. Berlin: Springer-Verlag.

[Feng & Doolittle, 1987] Feng, D. and Doolittle, R. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of Molecular Evolution, 25:351-360.

[Ferreti et al., 1996] Ferreti, V., Nadeau, J.H. and Sankoff, D. Original synteny. In Proceedings of the Seventh Symposium on Combinatorial Pattern Matching, number 1075 in Lecture Notes on Computer Science, pages 159-167. Berlin: Springer-Verlag.

[Fitch et al., 1983] Fitch, W.M., Smith, T.F. and Ralph, W.W. Mapping the order of DNA restriction fragments. Gene, 22:19-29.

[Formaneck, 2003] Formaneck, S. Methods of Multiple Sequence Alignment. Department of Combinatorics & Optimization, University of WaterLoo.

[Frances & Litman, 1997] Frances, M. and Litman, A. On Covering Problems of Codes. Theory of Computing Systems, vol. 30, pp. 113-119.

[Frenkel, 1991] Frenkel, K.A. The human genome project and informatics. Communications of the ACM, 34(11).

[Fulkerson & Gross, 1965] Fulkerson, D.R. and Gross, O.A. Incidence matrices and interval graphs. Pacific Journal of Mathematics, 15(3):835-855.

[Gallant et al., 1980] Gallant, J., Maier, D. and Storer, J. On finding minimal length superstrings. Journal of Computer and System Science, 20:50-58.

[Garey et al., 1972] Garey, M.R., Graham, R.L. and Ullman, J.D. Worst case analysis of memory allocation algorithms. In Proc. of the 4th ACM Symp. on Theory of Computing. 143-150.

[Gasieniec et al., 1999] Gasieniec, L., Jansson, J. and Lingas, A. Efficient approximation algorithms for the Hamming center problem, Proc. 10th ACM-SIAM Symposium on Discrete Algorithms, pp. S905-S906.

[Gates & Papadimitriou, 1979] Gates, W.H. and Papadimitriou, C.H. Bounds for sorting by prefix reversal. Discrete Mathematics, 27:47-57.

[George et al., 1990] George, D.G., Barker, W.C. and Hunt, L.T. Mutation data matrix and its uses. In Doolittle [51], pages 333-351.

[Gingerias et al., 1979] Gingerias, T., Milazzo, J., Sciaky, D. and Roberts, R. Computer programs for assembly of DNA sequences. Nucleic Acids Research, 7:529-545.

[Goldberg et al., 1993] Goldberg, P.W., Golumbic, M.C., Kaplan, H. and Shamir, R. Three strikes against physical mapping of DNA. Unpublished manuscript.

[Goldstein & Waterman, 1987] Goldstein, L. and Waterman, M.S. Mapping DNA by stochastic relaxation. Advances in Applied Mathematics, 8:194-207.

[Golumbic et al., 1994] Golumbic, M.C., Kaplan, H. and Shamir, R. On the complexity of DNA physical mapping. Adv. Appl. Math., 15:251-261.

[Green & Green, 1991] Green, E.D. and Green, P. Sequence-tagged site (STS) content mapping of human chromosomes: Theoretical considerations and early experiences. PCR Methods and Appl., pages 77-90.

[Greenberg & Istrail, 1994] Greenberg, D. and Istrail, S. The chimeric mapping problem: Algorithmic strategies and performance evaluation on synthetic genomic data. Computers and Chemistry, 18(3):207-220.

[Greenberg & Istrail, 1995] Greenberg, D. and Istrail, S. Physical mapping by STS hybridization: Algorithmic strategies and the challenge of software evaluation. Journal of Computational Biology, 2(2):219-274.

[Gupta et al., 1995] Gupta, S.K., Kececioglu, J., Schäffer, A.A. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. Journal of Computational Biology, 2(3):459-472.

[Gusfield et al., 1992] Gusfield, D., Landau, G.M. and Schieber, B. An efficient algorithm for the all pairs suffix-prefix problem. Information Processing Letters, 41:181-185.

[Gusfield, 1993] Gusfield, D. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bulletin of Mathematical Biology, 55(1):141-154.

[Gusfield, 1994] Gusfield, D. Faster implementation of a shortest superstring approximation. Information Processing Letters, 51:271-274.

[Gusfield, 1997] Gusfield, D. Algorithms on Strings, Trees, and Sequences. Cambridge University Press.

[Hannenhalli & Pevzner, 1995a] Hannenhalli, S. and Pevzner, P.A. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In Proceedings of Twenty-Seventh Annual ACM Symposium on Theory of Computing, pages 178-189.

[Hannenhalli & Pevzner, 1995b] Hannenhalli, S. and Pevzner, P.A. Transforming men into mice (polynomial algorithm for genomic distance problem). In Proceedings of the IEEE Thirty-Sixth Annual Symposium on Foundations of Computer Science, pages 581-592.

[Hannenhalli & Pevzner, 1999] Hannenhalli, S. and Pevzner, P. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). Journal of ACM 46(1):1-27.

[Heijne, 1987] von Heijne, G. Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit? New York: Academic Press.

[Hertz & Stormo, 1995] Hertz, G. and Stormo, G. Identification of Consensus Patterns in Un-aligned DNA and Protein Sequences: A Large-Deviation Statistical Basis for Penalizing Gaps, in Proceedings of the 3rd International Conference on Bioinformatics and Genome Research, p201-216.

[Higgins et al., 1996] Higgins, D.G., Thompson, J.D. and Gibson, T.J. Using CLUSTAL for multiple sequence alignments. Methods in Enzymology, 266:383-402.

[Hirschberg, 1975] Hirschberg, D. A linear space algorithm for computing maximal common subsequences. Communications of the ACM, 18:341-343.

[Ho et al., 1990] Ho, S.T., Allison, L. and Yee, C.N. Restriction site mapping for three or more enzymes. Comp. Appl. Biosciences, 6:195-204.

[Hofstadter, 1979] Hofstadter, D. Gödel, Escher, Bach. New York: Basic Books.

[Hsu, 1992] Hsu, W.L. A simple test for the consecutive ones property. In Proceedings of the International Symposium on Algorithms & Computation (ISAAC).

[Huang et al., 1990] Huang, X., Hardison, R.C. and Miller, W. A space-efficient algorithm for local similarities. Computer Applications in the Biosciences, 6(4):373-381.

[Huang, 1992] Huang, X. A contig assembly program based on sensitive detection of fragment overlaps. Genomics, 14:18-25.

[Huang, 1996] Huang, X. An improved sequence assembly program. Genomics, 33:21-31.

[Jiang et al., 1996] Jiang, T., Jiang, Z. and Breslauer, D. Rotation of periodic strings and short superstrings. Technical Report, Max-Planck-Institut f. Informatik, Saarrucken, Germany.

[Johnson, 1974] Johnson, D.S. Approximation algorithms for combinatorial problems. J. Comput. System Sci., 9, 256-278.

[Kaplan et al., 1994] Kaplan, H., Shamir, R. and Tarjan, R.E. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In Proceedings of the IEEE Thirty-Fifth Annual Symposium on Foundations of Computer Science, pages 780-791.

[Karp, 1993] Karp, R.M. Mapping the Genome: some combinatorial problems arising in molecular biology. ACM Symposium on Theory of Computing'93, 278-285.

[Kececioglu, 1993] Kececioglu, J. The maximum weight trace problem in multiple sequence alignment. In Proc. 4-th Symp. Combinatorial Pattern Matching, pages 106-119. Springer-Verlag.

[Kececioglu & Sankoff, 1995] Kececioglu, J. and Sankoff, D. Exact and approximation algorithms for the inversion distance between two permutations. Algorithmica, 13:180-210.

[Kececioglu & Myers, 1995] Kececioglu, J.D. and Myers, E.W. Combinatorial algorithms for DNA sequence assembly. Algorithmica, 13:7-51.

[Koonin & Dolja, 1993] Koonin, E.V. and Dolja, V.V. Evolution and taxonomy of positive-strand RNA viruses: implications of comparative analysis of amino acid sequences. Critical Reviews in Biochemistry and Molecular Biology, 28(5):375-430.

[Kosaraju et al., 1994] Kosaraju, R., Park, J. and Stein, C. Long tours and short superstrings. In Proceedings of the IEEE Thirty-Fifth Annnual Symposium on Foundations of Computer Science, pages 166-177.

[Lancia, 2004] Lancia, G. Applications to Computational Molecular Biology, Kluwer International Series in Operations Research and Management Science, Volume on Modeling for Discrete Optimization, (G. Appa and P. Williams eds).

[Lanctot et al., 1999] Lanctot, K., Li, M., Ma, B., Wang, S. and Zhang, L. Distinguishing string selection problems. Proc. 10th ACM-SIAM Symp. On Discrete Algorithms, pp. 633-642.

[Lawrence & Reilly, 1990] Lawrence, C. and Reilly, A. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins, **7**, 41-51.

[Lewin, 1994] Lewin, B. Genes V. Oxford: Oxford University Press.

[Lemke & Werman, 1988] Lemke, P. and Werman, M. On the complexity of inverting the autocorrelation function of a finite integer sequence, and the problem of locating n points on a line, given the unlabelled distances between them. Preprint 453, Institute for Mathematics and its Application IMA.

[Lewontin, 1993] Lewontin, R. Biology as Ideology. New York: HarperPerennial.

[Li, 1990] Li, M. Towards a DNA sequencing theory. In Proc. 31-th Annual IEEE Symp. Found. Comput. Sci., pages 125-134.

[Li et al., 2002] Li, M., Ma, B. and Wang, L. [2002] On the closest string and substring problems. Journal of the ACM, 49(2):157-171.

[Lucas et al., 1991] Lucas, K., Busch, M., Mossinger, S. and Thompson, J.A. An improved microcomputer program for finding gene or gene family-specific Abd-Elsalam 95 oligonucleotides suitable as primers for polymerase chain reactions or as probes. Comput. Appl. Biosci. 7: 525-529.

[Ma, 2000] Ma, B. A polynomial time approximation scheme for the closest substring problem. In Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM), volume 1848 of Lecture Notes in Computer Science, pages 99-107.

[Mathews & Holde, 1990] Mathews, C.K. and van Holde, K.E. Biochemistry. Redwood City, CA: Benjamin/Cummings.

[Mayraz & Shamir, 1999] Mayraz, G. and Shamir, R. Construction of physical maps from oligonucleotide fingerprints data. Journal of Computational Biology, 6(2):237-252.

[Meidanis & Munuera, 1996] Meidanis, J. and Munuera, E.G. A theory for the consecutive ones property. In Proceedings of the Third South American Workshop on String Processing, volume 4 of International Informatics Series, pages 194-202. Carleton University Press.

[Meidanis & Setúbal, 1997] Setúbal, J.C. and Meidanis, J. Introduction to Computacional Molecular Biology. PWS Publishing Company.

[Messing et al., 1981] Messing, J., Crea, R. and Seeburg, P.H. A system for shotgun DNA sequencing. Nucleic Acids Research, 9:309-321.

[Meyer, 1983] Meyer, P. Probabilidade: Aplicações à Estatística. (2ª edição) Livros Técnicos e Científicos Editora S.A.

[Mott et al., 1993] Mott, R., Grigoriev, A., Maier, J.H.E. and Lehrach, H. Algorithms and software tools for ordering clone libraries: application to the mapping of the genome of schzosaccharomyces pombe. Nucleic Acid Research, 21(8):1965-1974.

[Motwani & Raghavan, 1995] Motwani, R. and Raghavan, P. Randomized Algorithms, Cambrigde Univ. Press.

[Myers & Miller, 1988] Myers, E.W. and Miller, W. Optimal alignments in linear space. Computer Applications in the Biosciences, 4(1):11-17.

[Needleman & Wunsch, 1970] Needleman, S.B. and Wunsch, C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48:443-453.

[NewBerg & Naor, 1993] NewBerg, L.A. and Naor, D. A lower bound on the number of solutions to the exact probed partial digest problem. Adv. Appl. Math. 14:172-185.

[Okura, 2002] Okura, V.K. Bioinformática de Projetos Genoma de Bactérias. Tese de M.Sc., Universidade Estadual de Campinas.

[Palmer, 1987] Palmer, J.D. Chloroplast DNA evolution and biosystematic uses of chloroplast DNA variation. The American Naturalist, 130:S6-S29, Supplement.

[Palmer & Herbon, 1987] Palmer, J.D. and Herbon, L.A. Unicircular structure of the brassica hirta mitochondrial genome. Current Genetics, 11:565-570.

[Palmer et al., 1988] Palmer, J.D., Osório, B. and Thompson, W.F. Evolutionary significance of inversions in legume chloroplast DNAs. Current Genetics, 14:65-74.

[Pandurangan & Ramesh, 2002] Pandurangan, G. and Ramesh, H. The Restriction Mapping Problem Revisited. Journal of Computer and System Sciences (special issue on Computational Biology), 65, 526-544 (invited paper).

[Pardalos et al., 2004] Pardalos, P.M., Meneses, C.N., Lu, Z., Oliveira, C.A.S. Optimal Solutions for the Closest String Problem via Integer Programming. To appear in INFORMS Journal on Computing.

[Pearson & Miller, 1992] Pearson, W.R. and Miller, W. Dynamic programming algorithms for biological sequence comparison. In L. Brand and M. L. Johnson, editors, Numerical Computer Methods, volume 210 of Methods in Enzymology, pages 575-601. New York: Academic Press.

[Peltola et al., 1984] Peltola, H., Söderlund, H. and Ukkonen, E. SEQAIDS: A DNA sequence assembling program based on a mathematical model. Nucleic Acids Research, 12:307-321.

[Pevzner, 1992] Pevzner, P.A. DNA physical mapping, flows in networks and minimum cycles mean in graphs. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 8:99-112.

[Pevzner, 1995] Pevzner, P.A. DNA physical mapping and alternating Eulerian cycles in colored graphs. Algorithmica, 13(1/2):77-105.

[Pevzner, 2000] Pevzner, P. Computational Molecular Biology - An Algorithmic Approach. MIT Press,

[Posfai et al., 1989] Posfai, J., Bhagwat, A.S., Posfai, G. and Roberts, R.J. Predictive motifs derived from cytosine methyltransferases. Nucleic Acids Res 17 (7), 2421-2435.

[Poustka et al., 1986] Poustka, A., Pohl, T., Barlow, D.P., Zehetner, G., Craig, A., Michiels, F., Ehrich, E., Frischauf, A.M. and Lehrach, H. Molecular approaches to mammalian genetics. Cold Spring Harbor Symposium on Quantitative Biology, 51:131-139.

[Proutski & Holme, 1996] Proutski, V. and Holme, E.C. Primer Master: a new program for the design and analysis of PCR primers. CABIOS 12:253-255.

[Raghavan & Thompson, 1987] Raghavan, P., Thompson, C.D. Randomized Rounding: Provably good algorithms and algorithmics proofs. Combinatorica 7, 365-374.

[Raghavan, 1988] Raghavan, P. A probabilistic construction of deterministic algorithms: Approximating packing integer programs. Journal of Computer and System Sciences, 37:130-143.

[Robbins, 1992] Robbins, R.J. Challenges in the human genome project. IEEE Engineering in Medicine and Biology, 11(1):25-34.

[Rosenblatt & Seymour, 1982] Rosenblatt, J. and Seymour, P. The structure of homometric sets. SIAM Journal of Algorithms and Discrete Mathematics, 3(3):343-350.

[Rosenfeld et al., 1984] Rosenfeld, I., Ziff, E. and van Loon, V. DNA for beginners. Writers and Readers.

[Sankoff, 1975] Sankoff, D. Minimal mutation trees of sequences. SIAM Journal on Applied Mathematics, 28:35-42.

[Sankoff & Kruskal, 1983] Sankoff, D. and Kruskal, J.B. Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison. Reading, MA: Addison-Wesley.

[Sankoff, 1985] Sankoff, D. Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM Journal on Applied Mathematics, 45:810-825.

[Sankoff, 1993] Sankoff, D. Analytical approaches to genomic evolution. Biochimie, 75(409-413).

[Schmitt & Waterman, 1991] Schmitt, W. and Waterman, M.S. Multiple solutions of DNA restriction mapping problem. Advances in Applied Mathematics, 12:412-427.

[Sellers, 1974] Sellers P. H. On the theory and computation of evolutionary distances. SIAM J. Appl. Math. 26:787-793.

[Sim & Park, 2001] Sim, J.S. and Park, K. The Consensus String Problem for a Metric is NP-Complete. J. of Discrete Algorithms, 2(1), 115–121.

[Simon, 1987] Simon, I. Sequence comparison: some theory and some practice. In Proceedings of the LITP Spring School on Theoretical Computer Science, volume 377 of Lecture Notes in Computer Science, pages 79-92. Berlin: Springer-Verlag.

[Sivakumar, 2002] Sivakumar, D. Algorithmic Derandomization via Complexity Theory. In Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, pages 619-626. ACM Press.

[Skiena et al., 1990] Skiena, S.S., Smith, W.D. and Lemke, P. Reconstructing sets from interpoint distances. In Proc. 6-th Ann. ACM Symp. on Computational Geometry, pages 332-339.

[Skiena & Sundaram, 1994] Skiena, S.S. and Sundaram, G. A Partial Digest Approach to Restriction Site Mapping. Bulletin of Mathematical Biology, 56(2), pp. 275-294.

[Smith & Waterman, 1981] Smith, T.F. and Waterman, M.S. Identification of common molecular subsequences. Journal of Molecular Biology, 147:195-197.

[Srinivasan, 1999] Srinivasan, A. Approximation algorithms via randomized rounding: a survey. Series in Advanced Topics in Mathematics, pages 9-71.

[Staden, 1979] Staden, R. A strategy of DNA sequencing employing computer programs. Nucleic Acids Research, 6:2601-2610.

[Stefik, 1978] Stefik, M. Inferring DNA structure from segmentation data. Artificial Intelligence, 11:85-114.

[Stormo, 1990] Stormo, G.D. Consensus patterns in DNA. InDoolittle, R. F., ed., Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences, Methods inEnzymology, volume 183. Academic Press. 211-221.

[Stormo & Hartzell, 1991] Stormo, G.D. and Hartzell, G.W. Identifying protein-binding sites from unaligned DNA fragments,Proc. Natl. Acad. Sci. USA, 88:5699–5703.

[Tamarin, 1991] Tamarin, R. Principles of Genetics. Duduque, IA: Wm. C. Brown.

[Tarhio & Ukkonen, 1998] Tarhio, J. and Ukkonen, E. A greedy approximation algorithm for constructing shortest common superstrings. Theoretical Comput. Sci., 57:131-145.

[Taylor, 1987] Taylor, W.R. Multiple sequence alignment by a pairwise algorithm. Computer Applications in Biosciences, 3:81-87.

[Teng & Yao, 1993] Teng, S.-H. and Yao, F. Approximating shortest superstrings. In Proceedings of the IEEE Thirty-Fourth Annual Symposium on Foundations of Computer Science, pages 158-165.

[Turner, 1989] Turner, J. Approximation algorithms for the shortest common superstring problem. Information and Computation, 83:1-20.

[Vigron, 1996] Vigron, M. Near-optimal sequence alignment. Currenty Opinion in Struct. Biol., 6(3):346-352.

[Zehetner et al., 1988] Zehetner, G., Frischauf, A. and Lehrach, H. Approaches to restriction map determination, pages 147-164. IRL Press, Oxford. M. J. Bishop and C. J. Rawling (eds.), Nucl. Acid and Protein Sequence Analysis, Pratical Approaches.

[Zhang et al., 1994] Zhang, Z., Raghavachari, B., Hardison, R. and Miller, W. Chaining multiple-alignment blocks. J. Comput. Biol., 1:217-226.

[Wang & Jiang, 1994] Wang, L. and Jiang, T. On the complexity of multiple sequence alignment. J. Comput. Biol., 1:337-348.

[Waterfield et al., 1983] Waterfield, M.D., Scrace, G.T., Whittle, N., Stroobant, P., Johnsson, A., Wasteson, A., Westermark, B., Heldin, C.H., Huang, J.S. and Deuel, T.F. Platelet-derived growth factor is structurally related to the putative transforming protein p28sis of simian sarcoma virus. Nature, 304:35-39.

[Waterman et al., 1976] Waterman, M.S., Smith, T.F. and Beyer, W.A. Some biological sequence metrics. Advances in Mathematics, 20:367-387.

[Waterman & Perlwitz, 1984] Waterman, M.S. and Perlwitz, M.D. Line Geometries for Sequence Comparisons. Bull Math Biol;46(4):567-577.

[Waterman et al., 1984] Waterman, M.S., Arratia, R. and Galas, D.J. Pattern Recognition in Several Sequences: Consensus and Alignment. Bull. Math. Biol. 46, 515-527.

[Waterman & Griggs, 1986] Waterman, M.S. and Griggs, J.R. Interval graphs and maps of DNA. Bulletin of Mathematical Biology, 48(2):189-195.

[Waterman, 1986] Waterman, M.S. Multiple sequence alignment by consensus. Nucleic Acids Res. 14(22):9095-102.

[Waterman, 1989] Waterman, M.S. Mathematical Methods for DNA Sequences. Boca Raton, FL: CRC Press. Editor 1989.

[Waterman, 1995] Waterman, M.S. Introduction to Computational Biology, Chapman and Hall.

[Watson & Crick, 1953a] Watson, J.D. and F.H.C. Crick. Molecular structure of nucleic acid. A structure for deoxyribose nucleic acid. Nature 171:737-738.

[Watson & Crick, 1953b] Watson, J.D. and F.H.C. Crick. Genetic implications of the structure of deoxyribonucleic acid. Nature 171:964-967.

[Watson et al., 1987a] Watson, J.D. et al. Molecular Biology of the Gene, volume 1. Redwood City, CA: Benjamin/Cummings.

[Watson et al., 1987b] Watson, J.D. et al. Molecular Biology of the Gene, volume 2. Redwood City, CA: Benjamin/Cummings.

[Watterson et al., 1982] Watterson, G.A., Ewens, W.J., Hall, T.E. and Morgan, A. The chromosome inversion problem. Journal of Theoretical Biology, 99:1-7.

[Wright, 1997] Wright, S.J. Primal-Dual Interior-Point Methods. SIAM – Society for Industrial and Applied Mathematics.