Universidade Estadual de Campinas
Instituto de Matemática Estatística e Computação Científica
Departamento de Matemática Aplicada

Reconstrução e Classificação de Estruturas Espaciais via Otimização Contínua: Ênfase em Proteínas

Rodrigo Silva Lima*
Doutorado em Matemática Aplicada - Campinas - SP

Orientador: Prof. Dr. José Mario Martínez Pérez
Coorientador: Profa. Dra. Margarida Pinheiro Mello

Campinas, Janeiro de 2012

∗ Este trabalho recebeu apoio financeiro da FAPESP (processo 06/04053-8).

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

Rodrigo Silva Lima

Reconstrução e Classificação de Estruturas Espaciais via Otimização Contínua: Ênfase em Proteínas

TESE de DOUTORADO APRESENTADA
AO INSTITUTO DE MATEMÁTICA,
ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DA UNICAMP PARA OBTENÇÃO DO TÍTULO DE
DOUTOR EM MATEMÁTICA APLICADA

ORIENTADOR: Prof. Dr. José Mario Martínez Pérez
COORIENTADOR: Profa. Dra. Margarida Pinheiro Mello

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DE DOUTORADO
DEFENDIDA PELO ALUNO RODRIGO SILVA LIMA, E ORIENTADA PELO
PROF. DR. JOSÉ MARIO MARTÍNEZ PÉREZ

Prof. Dr. José Mario Martínez Pérez

Profa. Dra. Margarida Pinheiro Mello

CAMPINAS, 2012

FICHA CATALOGRÁFICA ELABORADA POR ANA REGINA MACHADO – CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA – UNICAMP

L628r

Lima, Rodrigo Silva, 1982-
Reconstrução e classificação de estruturas espaciais via otimização contínua : ênfase em proteínas / Rodrigo Silva Lima. – Campinas, SP : [s.n.], 2012.

Orientador: José Mario Martínez Pérez.
Coorientador: Margarida Pinheiro Mello.
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

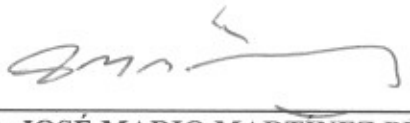
1. Proteínas – Modelos matemáticos. 2. Otimização matemática. 3. Programação não-linear. 4. Imagem tridimensional. I. Martínez Pérez, José Mario, 1948-. II. Mello, Margarida Pinheiro, 1957-. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em inglês: Reconstruction and classification of spatial structures via continuous optimization : emphasis on proteins
Palavras-chave em inglês:
Proteins - Mathematical models
Mathematical optimization
Nonlinear programming
Three-dimensional imaging
Área de concentração: Matemática Aplicada
Titulação: Doutor em Matemática Aplicada
Banca examinadora:
José Mario Martínez Pérez [Orientador]
Carlile Campos Lavor
Sandra Augusta Santos
Nelson Maculan Filho
Juliano de Bem Francisco
Data da defesa: 20-01-2012
Programa de Pós-Graduação: Matemática Aplicada

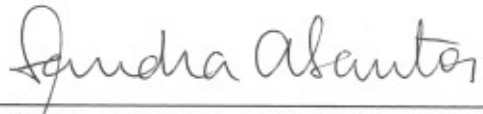
Tese de Doutorado defendida em 20 de janeiro de 2012 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.



Prof(a). Dr(a). JOSÉ MARIO MARTÍNEZ PÉREZ

Prof(a). Dr(a). CARLILE CAMPOS LAVOR



Prof(a). Dr(a). SANDRA AUGUSTA SANTOS

Prof(a). Dr(a). NELSON MACULAN FILHO

Prof(a). Dr(a). JULIANO DE BEM FRANCISCO

Dedico este trabalho aos meus pais Getúlio e Bernadete, ao meu irmão Rogério e à minha esposa Luciana.

Agradecimentos

Agradeço:

A meus pais que sempre me apoiaram em meus estudos.

À minha esposa Luciana por estar sempre ao meu lado me incentivando.

Ao prof. José Mario Martínez pela orientação e confiança em meu trabalho.

À profa. Margarida Mello por todos os anos que me orientou e ajudou na Unicamp.

A Leandro Martínez pela ajuda com os conceitos químicos.

Aos funcionários Ednaldo, Tânia e Lívia por me ajudarem sempre que precisei.

Aos meus amigos pelo incentivo e torcida.

À FAPESP pelo apoio financeiro.

"Talvez não tenhamos conseguido fazer o melhor, mas lutamos para que o melhor fosse feito. Ainda não somos o que deveríamos ser, ainda não somos o que iremos ser, mas graças a Deus não somos o que éramos."

Martin Luther King

Resumo

Neste trabalho estudamos inicialmente o problema da reconstrução 3D de uma proteína dadas as distâncias entre pares de átomos de sua estrutura. Formulamos a situação como um problema de otimização não linear com função objetivo contínua no domínio de variáveis e mostramos através de experimentos computacionais que a estrutura original da proteína é recuperada mesmo quando admitimos conhecidas apenas um subconjunto de distâncias intra-átomos. Em seguida, estudamos problema da representação de um conjunto de proteínas comparadas em relação às suas estruturas tridimensionais. Propomos algumas formulações para este problema onde as proteínas são representadas por objetos em espaços euclidianos e elaboramos também um procedimento para classificar proteínas novas sem a necessidade de realizar exaustivas comparações estruturais envolvendo as proteínas analisadas.

Palavras-chave: proteínas, modelos matemáticos, otimização matemática, programação não linear, imagem tridimensional.

Abstract

In this work we initially study the problem of reconstruct the 3D structure of a protein given the distances between pairs of its atoms. We formulate this situation as a nonlinear optimization problem with a continuous objective function over the domain of variables. We show by computational experiments that the original protein structure is recovered even when we do not use all the distances between its atoms. Next, we study the problem of representing a set of proteins. The proteins are compared with respect to their 3D structures. We propose some formulations to this problem, where the proteins are represented by objects in euclidean spaces and we elaborate also a form of use these representations to classify new proteins without perform many comparisons between the analyzed structures.

Keywords: proteins, mathematical models, mathematical optimization, nonlinear programming, three-dimensional image.

Sumário

Notações utilizadas

- $s = \{s_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$: conjunto de escores resultantes da comparação de n proteínas duas a duas,
- $\Delta = \{\hat{d}_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$: disparidades criadas a partir do conjunto s, que deverão ser realizadas por objetos em um determinado espaço euclidiano,
- d_{ij}: distância euclidiana final entre os objetos i e j,
- $\langle \cdot, \cdot \rangle$: produto interno euclidiano,
- $\| \cdot \|$: norma euclidiana,
- $\mathbb{1}^T = (1, 1, \ldots, 1)^T$: vetor cujas componentes são todas iguais a 1,
- $\mathrm{diag}(M)$: vetor com os elementos da diagonal principal de uma matriz quadrada M,
- I_n: matriz identidade de ordem n.
- $\lceil x \rceil$: número inteiro imediatamente superior ao número real x.
- $|P|$: cardinalidade do conjunto P.
- $\nabla f(x)$: gradiente da função $f(x)$.
- 1 angstrom: equivale a 1×10^{-10} m.

Outras notações, quando necessárias, serão introduzidas no decorrer do texto.

Capítulo 1

Introdução

O problema de representar conjuntos de proteínas comparadas aparece na literatura de duas formas diferentes. Na primeira, pesquisadores criam maneiras de representar semelhanças entre pares de proteínas, identificando-as por pontos no plano ou no espaço de tal forma que proteínas semelhantes fiquem próximas. Na segunda, criam-se bancos de dados com informações sobre as comparações realizadas para que sejam utilizadas na classificação de novas proteínas. Tanto em uma quanto em outra forma de representação, o resultado final é um "mapa" que armazena semelhanças e diferenças entre as proteínas comparadas. A utilidade do primeiro mapa consiste apenas em permitir uma visualização gráfica das semelhanças entre as proteínas analisadas. Já o segundo mapa pode ser usado para agilizar o processo de classificação de novas proteínas.

Neste capítulo fazemos uma revisão dos trabalhos existentes na literatura sobre comparação e classificação de proteínas. Começamos definindo alguns conceitos básicos de biologia molecular que serão freqüentemente utilizados ao longo do texto. Em seguida, discutimos técnicas empregadas na comparação de proteínas e apresentamos os principais pacotes computacionais que realizam este tipo de tarefa. Analisamos também características de algumas bases de dados disponíveis na rede que classificam proteínas de forma automática. Discutimos brevemente as ferramentas matemáticas que estão por trás das técnicas de visualização e, por fim, apresentamos nossa proposta de trabalho e mostramos como o texto está organizado.

1.1 Proteínas

Proteínas são compostos orgânicos constituídos de *aminoácidos* que estão conectados entre si por *ligações peptídicas* formando uma cadeia. A estrutura de uma proteína "começa" em um aminoácido (definido por convenção como aminoácido *N-terminal*) e "termina" no aminoácido que está mais longe do primeiro em termos de sua posição na cadeia (aminoácido *C-terminal*). Em geral, os aminoácidos são moléculas contendo os grupos funcionais *amina* (NH2) e *carboxila* (COOH). A Figura 1.1 ilustra sua estrutura. Os grupos amina e carboxila estão presos ao mesmo átomo de carbono, também conhecido como *carbono alfa* (Cα) e a

letra G indica a presença de um *substituinte orgânico* (grupo de átomos preso ao carbono Cα). Cada aminoácido dentro da cadeia é chamado *resíduo* e é comumente representado por uma letra maiúscula. Por exemplo, A, L e V são, respectivamente, siglas para *Alanina*, *Leucina* e *Valina*.

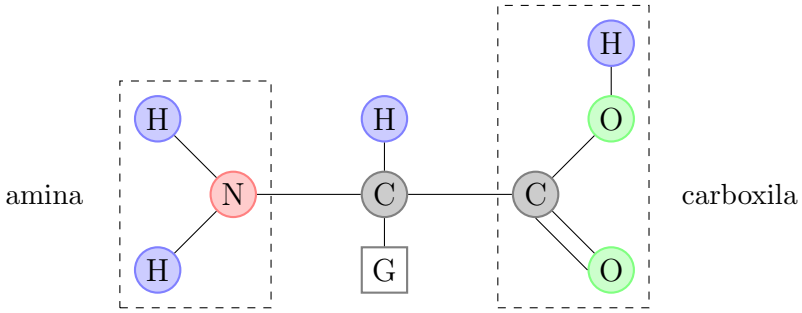


Figura 1.1: Estrutura geral de um aminoácido.

O estudo da configuração tridimensional de uma proteína é fundamental para o conhecimento do papel que ela desempenha nos organismos e é tema constante de pesquisas [28, 38]. Para fins didáticos, é comum dividirmos a proteína em quatro estruturas:

- **Primária**: são as sequências de todos os aminoácidos que formam a proteína. A estrutura primária é representada por uma sequência de letras maiúsculas que correspondem aos aminoácidos. As letras são colocadas na mesma ordem em que os aminoácidos ocorrem na cadeia da proteína analisada. Desta forma, podemos pensar na estrutura primária de uma proteína como sendo uma palavra.

- **Secundária**: são as sequências de aminoácidos que se organizam no espaço tridimensional em forma de hélices ou fitas.

- **Terciária**: é a disposição tridimensional de cada estrutura secundária, ou seja, a posição e orientação espacial das hélices ou fitas.

- **Quaternária**: é o modo como estão conectadas no espaço várias moléculas de proteínas.

Várias proteínas já foram descobertas e muitas outras surgem a cada dia. Como a quantidade de proteínas conhecidas é muito grande, faz-se necessária a criação de um banco de dados para guardar e administrar todas as informações relevantes sobre cada uma delas. O banco de dados mais conhecido e utilizado por pesquisadores da área de bioquímica é o *Protein Data Bank* (PDB) (`http://www.rcsb.org/pdb/`) [12]. Nele estão arquivados dados de 77 101 proteínas (última atualização do banco de dados: 08/11/2011). O *site* permite aos usuários realizar buscas simples ou avançadas, baseadas em informações sobre sequências de aminoácidos, estrutura tridimensional e funções de uma determinada proteína. As moléculas podem ser visualizadas e vários tipos de arquivos podem ser baixados gratuitamente. Como exemplo, as Figuras 1.2 e 1.3 foram obtidas ao realizarmos uma busca

no PDB pelo nome *caspase-1*. A Figura 1.2 mostra a configuração tridimensional (estrutura terciária) da molécula. Observe a presença de hélices e fitas em sua estrutura. Como dissemos anteriormente, as sequências de aminoácidos que definem estas formas são as estruturas secundárias da proteína. A Figura 1.3 mostra um trecho da sequência de aminoácidos

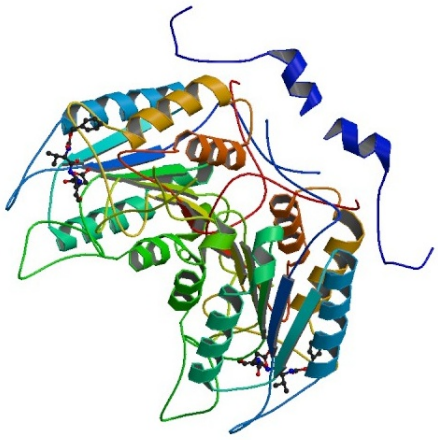


Figura 1.2: Configuração 3D da molécula *caspase-1*.

(estrutura primária) de uma cadeia da molécula *caspase-1*. Cada aminoácido (resíduo) é representado por uma letra maiúscula. De acordo com esta figura, é possível saber quais aminoácidos formam hélices. Por exemplo, a sequência LEEAQR pertence à primeira hélice (linha ondulada localizada na primeira fila de cima para baixo). Já as setas que apontam para a direita indicam a presença de fitas. A primeira seta amarela (localizada na primeira fila de cima para baixo) é uma fita formada pela sequência de aminoácidos LALIIC.

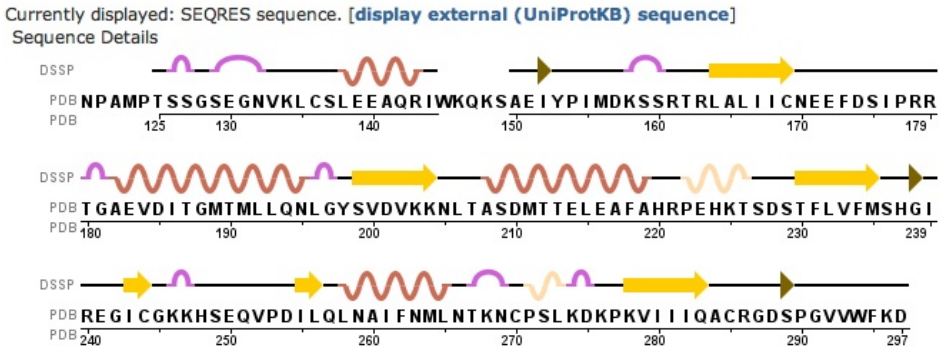


Figura 1.3: Trecho de uma sequência de aminoácidos da *caspase-1*.

Cada proteína cadastrada no PDB possui um único código de identificação (sigla). Este código é composto por quatro caracteres: o primeiro caracter é sempre um número inteiro

no intervalo $[1, 9]$ enquanto os três últimos podem ser números inteiros em $[0, 9]$ ou letras. Por exemplo, `1mbn`, `2hhd` e `9ins` são, respectivamente, siglas para mioglobina, hemoglobina humana e insulina. Essa regra permite a criação de 419 904 códigos distintos. As proteínas cadastradas no PDB representam apenas 18% desse total. Como o PDB é constantemente atualizado com proteínas novas, será preciso num futuro próximo redefinir a forma de identificação das estruturas depositadas neste banco de dados.

Além do PDB, outras bases de dados utilizadas por cientistas são SCOP (*Structural Classification of Proteins*) [10, 71] e CATH (*Class, Architecture, Topology and Homologous superfamily*) [18, 75]. Estas bases também possuem diversos recursos para que usuários possam estudar as estruturas e funções das proteínas, além de ser possível comparar uma nova proteína com as proteínas da base de dados armazenada em cada servidor. Muitos trabalhos na literatura são dedicados ao desenvolvimento de métodos para comparação e classificação de proteínas. A seguir, faremos uma breve discussão desses métodos, destacando algumas de suas características.

1.2 Comparação e classificação de proteínas

Semelhanças entre proteínas são detectadas basicamente por meio de duas técnicas: alinhamento de sequências de aminoácidos e alinhamento de estruturas tridimensionais. Em ambos os casos o grau de semelhança entre pares de proteínas comparadas é dado por um número real não negativo denominado *escore*. Em geral, essas técnicas de comparação estabelecem escores que são diretamente proporcionais às similaridades do par de proteínas analisado, isto é, quanto maior é a semelhança entre duas proteínas, maior é o valor do escore correspondente. É importante frisar que não há um padrão para determinar escores de conjuntos de proteínas, ou seja, existem na literatura diferentes modos para estabelecer o valor de um escore.

A técnica do alinhamento de sequências de aminoácidos (estruturas primárias) consiste em comparar trechos de proteínas distintas para tentar identificar aminoácidos que se repetem. Considere, como exemplo, as duas sequências de letras indicadas na Figura 1.4. Neste caso, a semelhança é diretamente proporcional ao número de aminoácidos que ocupam a mesma posição em ambas as sequências analisadas (letras em destaque).

seq. 1: $F\,T\,F\,T\,A\,L\,I\,L\,L\,A\,V\,A\,V$

seq. 2: $F\,S\,S\,T\,A\,L\,S\,L\,L\,A\,S\,A\,V$

Figura 1.4: Alinhamento de estruturas primárias.

São raros os casos de sequências de letras que podem ser comparadas por inspeção. Por essa razão, existem algoritmos eficientes para tratar o problema computacionalmente. Entre os algoritmos mais conhecidos para realizar este tipo de tarefa estão *BLAST* [4], *CLUSTAL* [40, 58] e *FASTA* [77]. *BLAST* emprega uma estratégia que, inicialmente, verifica a existência

de pequenos trechos de letras que se repetem entre duas sequências de aminoácidos. Somente depois desse processo, o algoritmo inicia a etapa de alinhamento. *BLAST* é baseado no trabalho de T. S. Smith e M. S. Waterman [84] e mais detalhes sobre o seu funcionamento podem ser encontrados em [29]. O programa *CLUSTAL* permite o alinhamento de várias sequências de aminoácidos de diferentes proteínas ao mesmo tempo. Ele constrói uma árvore a partir de uma matriz de escores obtida pelo alinhamento sequencial de pares de proteínas. Por fim, esta árvore é utilizada para a obtenção dos alinhamentos múltiplos. O programa *FASTA*, também baseado em [84], encontra regiões de semelhança entre sequências de aminoácidos ou identifica duplicações locais dentro de uma sequência isolada. Independentemente das particularidades de cada algoritmo, todos empregam heurísticas na determinação dos alinhamentos.

O alinhamento de estruturas tridimensionais das proteínas também é um campo de intensa pesquisa. A cada ano cresce o número de novos métodos que utilizam diferentes representações para as proteínas, novas maneiras de medir escores são criadas e diferentes estratégias de otimização são empregadas. De acordo com H. Hasegawa e L. Holm em [38], existe uma forte tradição na obtenção de alinhamentos tridimensionais através da sobreposição rígida de moléculas, ou seja, as proteínas a serem alinhadas são tratadas como corpos rígidos. Assim, para medir o grau de semelhança entre duas estruturas no espaço, é necessário aplicar a uma delas um movimento rígido, de forma que as estruturas finais fiquem sobrepostas. Na realização dessa tarefa, as proteínas são representadas pelas coordenadas tridimensionais de seus átomos. Em particular, elas podem ser representadas de maneira simplificada pelas coordenadas dos átomos de carbono alfa (Cα), como ilustra a Figura 1.5. Esta representação preserva as principais características do arranjo espacial dos aminoácidos na estrutura da proteína. Entre os principais métodos de alinhamento que seguem esta

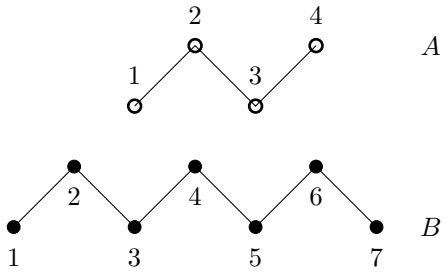


Figura 1.5: Proteínas representadas pelos átomos C_α.

metodologia estão *SSM* [53], *MAMMOTH* [76] e *LOVOALIGN* [7, 8, 9, 35, 70]. Outros métodos como *DALI* [41], *TOPOFIT* [63] e *FAST* [97] trabalham com mapas de contato, grafos e matrizes de distâncias. Em particular, o *software LOVOALIGN* é o único entre os métodos citados acima que não emprega heurísticas para realizar um alinhamento 3D. Os alinhamentos em *LOVOALIGN* são obtidos via algoritmos de otimização contínua.

A ação típica de um pesquisador que se depara com uma nova proteína é querer compará-la com as proteínas depositadas no PDB. Esta tarefa tem como objetivo conhecer todas as proteínas da base de dados que são parecidas com a proteína nova em termos de sequências de aminoácidos ou estruturas tridimensionais. Como o PDB possui 77 101 proteínas cadastradas, a classificação manual torna-se impossível de ser realizada. Com o intuito de automatizar este trabalho, surgem programas capazes de estabelecer classificações entre proteínas. Alguns programas utilizam estratégias que comparam uma proteína com todas as proteínas de uma base pré-definida. Entre eles estão *YAKUSA* [15], *GANGSTA+* [36], *MATRAS* [49], *SSM* [53], *TOPOFIT* [63], *TOPS++FATCAT* [90]. Já os pacotes *VOROMETRIC* [79] e *DALILITE* [42] possuem uma base formada por proteínas cujas relações de semelhança duas a duas são conhecidas. Assim, uma proteína nova pode ser comparada com apenas um subconjunto de proteínas desta base, sem que seja necessário investigar todas as proteínas disponíveis. Este tipo de comparação só é possível porque os programas utilizam eficientes algoritmos de busca em árvores de dados para obter rapidamente os resultados [39, 88].

1.3 Visualização de semelhanças entre proteínas

Suponha que n proteínas foram comparadas duas a duas por algum método de alinhamento e um conjunto de escores $s = \{s_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$ foi gerado. Mais adiante daremos detalhes de como estes escores podem ser gerados. Vários trabalhos existentes na literatura são dedicados ao estudo de métodos para visualizar os resultados das comparações entre proteínas. Um dos métodos de maior popularidade é denominado *Ajuste Multidimensional* (do inglês *Multidimensional Scaling*) [14, 21, 54, 55, 56]. Neste método, um conjunto de n objetos comparados é representado por pontos em um espaço de dimensão pequena, preferencialmente $\mathbb{R}^2$ ou $\mathbb{R}^3$, de tal forma que pontos próximos correspondam a objetos semelhantes. A cada par de objetos é atribuído um número $\hat{d}_{ij} \geq 0$ que mede o grau de discrepância entre eles, isto é, quanto maior for o valor de $\hat{d}_{ij}$, maior será a diferença entre os objetos i e j. Dessa forma, o problema consiste em alocar os pontos no espaço e fazer com que as distâncias euclidianas d_{ij} entre pares de pontos sejam tão próximas quanto possível dos valores $\hat{d}_{ij}$, isto é, $d_{ij} \approx \hat{d}_{ij}$, para todo $i \neq j$. No caso que estamos tratando, os objetos são proteínas e os valores $\hat{d}_{ij}$ são criados para que sejam inversamente proporcionais aos escores s_{ij}.

O problema clássico do ajuste de distâncias entre objetos parte da hipótese de que o conjunto $\Delta = \{\hat{d}_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$ é realizável por pontos em algum espaço $\mathbb{R}^m$. Quando essa hipótese é verdadeira, as coordenadas dos pontos que representam os objetos podem ser obtidas analiticamente [20, 27, 33, 34]. Porém, na maioria das aplicações, $\hat{d}_{ij}$ não é necessariamente uma distância entre pontos, pois seu valor pode indicar apenas quão parecidos ou diferentes são dois objetos i e j. Em situações como esta, o problema pode ser formulado como um problema de otimização não linear cuja função objetivo minimiza a soma dos quadrados das diferenças entre d_{ij} e $\hat{d}_{ij}$. Algumas condições podem ser adicionadas a esta formulação para melhorar a qualidade da solução obtida. Uma delas consiste em pedir

que a ordem dos elementos do conjunto Δ seja preservada na solução final, ou seja,

$$\text{se } \hat{d}_{ij} \leq \hat{d}_{k\ell} \text{ então } d_{ij} \leq d_{k\ell}.$$

Alguns autores trabalharam na visualização de semelhanças entre famílias de proteínas empregando o modelo clássico, como é o caso dos trabalhos de D. Agrafiotis *et al* [1] e J. Hou *et al* [44, 45, 82, 83]. Em [1], os autores propõem um algoritmo que combina técnicas de ajuste de distâncias entre pontos com redes neurais. Já nos trabalhos [44, 45, 82, 83], é proposta uma maneira de medir distâncias entre pares de proteínas comparadas baseada nos ângulos de torção de suas estruturas 3D. Então, uma representação em $\mathbb{R}^3$ é construída para uma família de proteínas retiradas da base SCOP. Há ainda trabalhos que investem em modelos bidimensionais sem que distâncias entre pontos sejam ajustadas efetivamente. Em [37] um problema de otimização sem restrições é construído com o objetivo de minimizar o produto escalar entre dois vetores. As coordenadas de um vetor são números que medem o grau de semelhança entre pares de um conjunto de proteínas e o outro vetor tem coordenadas dadas por

$$\gamma e^{(-\tau \|x^i - x^j\|/2)^2},$$

onde x^i, x^j são pontos no plano e γ, τ são constantes. Assim, ao minimizar o produto escalar as exponenciais presentes na função objetivo asseguram que proteínas parecidas fiquem próximas na configuração final. No trabalho [30] os autores propõem um problema de otimização cujas variáveis são as entradas de uma matriz que deve ser semi-definida positiva. Assim, técnicas de programação semi-definida são empregadas na resolução do problema.

1.4 Escore *Structal* e rotina *LOVOALIGN*

Os modelos que propomos neste trabalho têm como ponto de partida um conjunto de proteínas comparadas em relação às suas estruturas tridimensionais. Para avaliar o grau de semelhança entre os pares de proteínas que aparecem ao longo do texto, adotamos o *escore Structal* [51, 87]. Sejam

$$A = \{a_1, a_2, \ldots, a_{n_A}\} \text{ e } B = \{b_1, b_2, \ldots, b_{n_B}\}$$

duas proteínas com n_A e n_B átomos, respectivamente, onde cada átomo é representado por suas coordenadas tridimensionais. Por exemplo, na proteína A temos que $a_i \in \mathbb{R}^3$ para todo $i = 1, 2, \ldots, n_A$. Uma *subcadeia* de uma proteína é um subconjunto de átomos arranjados de acordo com a ordem em que aparecem em sua estrutura. Por exemplo, uma subcadeia em A de comprimento k pode ser denotada por

$$Q_A = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}, \text{ onde } 1 \leq i_1 < i_2 < \ldots < i_k \leq n_A.$$

Dizemos que esta subcadeia possui um *intervalo* se existirem dois índices consecutivos i_ℓ e $i_{\ell+1}$ tais que $i_\ell + 1 < i_{\ell+1}$. Sejam Q_A e Q_B duas subcadeias com comprimento k das proteínas A e B, respectivamente. Uma *correspondência* (*ou bijeção*) entre Q_A e Q_B associa pares de

átomos das duas proteínas que ocupam a mesma posição nas respectivas subcadeias. Nesse sentido, o escore *Structal* entre Q_A e Q_B é definido por

$$s_{AB} = \sum_{\ell=1}^{k} \frac{20}{1 + \|a_{i_\ell} - b_{j_\ell}\|^2/5} - 10\ n_g, \tag{1.1}$$

onde n_g é o número de intervalos presentes na correspondência.

A comparação estrutural entre as proteínas A e B se dá pela procura de subcadeias Q_A e Q_B que sejam semelhantes e tenham o maior comprimento possível. Isto significa que devemos procurar por uma bijeção entre os átomos das proteínas que maximize o valor do escore *Structal* (1.1). De acordo com a fórmula (1.1), se A e B possuem estruturas idênticas $\|a_{i_\ell} - b_{j_\ell}\| = 0$ para todos os índices i_ℓ, j_ℓ presentes na bijeção e $n_g = 0$. Consequentemente, o escore entre as proteínas será $s_{AB} = 20n_A$. Se A e B diferem em trechos de suas estruturas, teremos $s_{AB} \in (0, 20\min\{n_A, n_B\}]$. Em qualquer dos casos anteriores, se dividirmos s_{AB} pelo número de átomos da menor proteína do par comparado, obteremos um número no intervalo $(0, 20]$. Ao longo do texto, nos referiremos a este número como *escore Structal normalizado*.

O método *Structal* para alinhamento estrutural de proteínas consiste em maximizar o escore *Structal* (1.1) [52, 87]. Para isso, cada proteína é representada pelas coordenadas $3D$ de seus átomos de carbono alfa (C_α) e os seguintes passos são aplicados iterativamente:

1. Dada uma orientação espacial para duas estruturas, obter, via programação dinâmica, a correspondência (bijeção) que maximiza o escore (1.1).

2. Dada a correspondência entre os átomos do par de estruturas, obter uma transformação de corpo rígido através da solução de um problema *Procrustes*.

A estratégia de programação dinâmica utilizada no passo 1 é o melhor procedimento para obter a bijeção que maximiza o escore *Structal* [72]. No entanto, esta técnica requer um número de operações que cresce quadraticamente com o número de átomos do par de proteínas envolvido [16]. O problema *Procrustes* no passo 2 consiste em encontrar uma transformação de corpo rígido que sobreponha as estruturas do par de proteínas analisado. Esta transformação é determinada através da minimização de um desvio conhecido na literatura por *RMSD* (do inglês *root mean square deviation*) [32, 48, 50].

Martínez *et al.* [70] perceberam algumas situações em que o método *Structal* converge para alinhamentos que não maximizam o escore (1.1) ou oscila entre dois alinhamentos distintos. Para contornar essas dificuldades, os autores propuseram substituir o problema *Procrustes* pela resolução de um problema de otimização do menor valor ordenado [9, 69]. Como resultado dessa pesquisa surgiu a rotina *LOVOALIGN* para comparação estrutural de proteínas [7, 8]. Esta rotina, escrita em *Fortran* 77, está disponível para *download* gratuito em `www.ime.unicamp.br/~martinez/lovalign`.

Em todos os modelos para a construção de mapas que apresentaremos ao longo do texto, utilizaremos *LOVOALIGN* para obter os escores *Structal* dos conjuntos de proteínas. Em alguns modelos utilizaremos estes escores para estabelecer discrepâncias entre pares de

proteínas i e j, isto é, criaremos números reais $\hat{d}_{ij}$ não negativos inversamente proporcionais aos valores s_{ij}. Então, representaremos as proteínas por objetos em um espaço $\mathbb{R}^m$ de tal forma que as distâncias entre pares desses objetos sejam aproximadamente iguais aos números $\hat{d}_{ij}$. Já em outros modelos, identificaremos cada proteína por um vetor em $\mathbb{R}^m$ e empregaremos os escores *Structal* normalizados para ajustar produtos escalares entre pares desses vetores. A seguir, mostramos como nosso trabalho está organizado, descrevendo resumidamente o conteúdo de cada capítulo.

1.5 Organização do trabalho

De acordo com o que expusemos anteriormente, podemos pensar em um "mapa de proteínas" como sendo uma base de dados que permite aos usuários comparar e classificar rapidamente uma nova proteína baseados em informações pré-estabelecidas, ou como sendo uma configuração de pontos em $\mathbb{R}^2$ ou $\mathbb{R}^3$ que corresponde a uma família de proteínas comparadas. No primeiro caso, a base de dados deve ser construída a partir de um conjunto de proteínas comparadas duas a duas e essas informações são usadas para que critérios rápidos de comparação sejam desenvolvidos. No segundo caso, os escores das comparações são utilizados para estabelecer discrepâncias entre pontos que representam as proteínas. Então, o conjunto de pontos é distribuído no plano ou no espaço de tal forma que as discrepâncias estipuladas sejam realizadas. A solução desse problema é um "mapa visual" que permite a identificação de proteínas parecidas.

Nosso objetivo neste trabalho é construir, via procedimentos de otimização contínua, uma representação (mapa) para conjuntos de proteínas comparadas. Pretendemos utilizar essa representação para classificar uma nova proteína sem precisar compará-la com todas as proteínas representadas. E, uma vez classificada, a nova proteína será incluída ao mapa e deverá ficar próxima das proteínas que com ela se parecem.

Os problemas que formulamos e resolvemos nesta pesquisa possuem a forma geral:

$$\begin{array}{ll} \min & f(x) \\ \text{s. a} & \ell \leq x \leq u, \end{array} \tag{1.2}$$

onde $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ é uma função contínua, x, ℓ e u são vetores em $\mathbb{R}^n$. Para resolver (1.2) utilizamos rotinas de otimização numérica [22, 68, 74]. Realizamos diversos experimentos computacionais com os modelos propostos. Em alguns modelos, os resultados obtidos nos testes não foram satisfatórios e, por essa razão, foram logo descartados. Em outros, obtivemos sucesso relativo e isto nos incentivou a investigá-los mais a fundo.

Nosso trabalho está dividido do seguinte modo: no Capítulo 2 estudamos inicialmente o problema do ajuste de distâncias entre pares de pontos. Como aplicação deste problema, investigamos a tarefa de reconstruir a estrutura tridimensional de uma proteína dadas as distâncias entre seus átomos. Mostramos por meio de um teorema quais são as condições necessárias e suficientes para que um conjunto de números reais positivos sejam distâncias

euclidianas entre pares de pontos em um determinado espaço. Através de exemplos, discutimos as dificuldades de utilizar este modelo para construir representações de conjuntos de proteínas comparadas. Em seguida, propomos algumas modificações do problema na tentativa de obter melhores resultados. Formulamos situações em que proteínas comparadas são representadas por círculos, segmentos de reta e poligonais formadas por dois segmentos. Fazemos experimentos numéricos com cada formulação e discutimos as principais dificuldades observadas.

No Capítulo 3 abandonamos a meta de ajustar distâncias e passamos a investigar o ajuste de escores. Nos modelos que apresentamos neste capítulo, as proteínas são representadas por vetores em um determinado espaço euclidiano $\mathbb{R}^m$ e nossa tarefa consiste em ajustar produtos escalares e escores. Mostramos, neste caso, que o problema sempre possui solução qualquer que seja o número de proteínas utilizado. Realizamos vários experimentos numéricos com as formulações propostas e discutimos os resultados obtidos. Uma das formulações mostrou-se mais promissora nos experimentos e, por essa razão, foi escolhida para ser utilizada nas tarefas de construção de mapas e inclusão de novas proteínas em mapas. Neste capítulo elaboramos também um procedimento para localizar em um conjunto de proteínas comparadas um subconjunto de proteínas que sejam semelhantes a uma proteína nova. Este procedimento visa evitar a comparação da proteína nova com todas as proteínas do conjunto, minimizando desta maneira o esforço computacional. Fizemos testes para validar esta estratégia e analisamos os resultados obtidos.

No Capítulo 4 apresentamos um modelo de construção de mapas de proteínas baseado na teoria da otimização do menor valor ordenado. Utilizando esta mesma teoria, propomos também um modelo para incluir novas proteínas em mapas. Realizamos experimentos computacionais para validar os modelos e obtivemos resultados bastante razoáveis. Por fim, no Capítulo 5 concluímos o trabalho e apresentamos as referências bibliográficas utilizadas em nossa pesquisa.

Capítulo 2

Problemas de distâncias euclidianas

O problema do ajuste de distâncias entre pontos é estudado desde as primeiras décadas do século XX [81]. Admitindo-se conhecidas as distâncias entre pares de um conjunto de n pontos, a tarefa de ajustá-las consiste em encontrar as coordenadas dos pontos em um espaço euclidiano $\mathbb{R}^m$ de tal forma que as distâncias entre os pontos no espaço realizem as distâncias dadas. Este problema aparece em diversas áreas do conhecimento, como por exemplo, psicologia [14, 56], engenharias [85, 93, 94, 95] e química [31, 59, 60, 61, 64, 65, 66, 92]. No primeiro caso, o conjunto de pontos pode representar um grupo de pessoas de determinada cidade ou região e as distâncias podem medir uma característica presente nos indivíduos analisados. Assim, dois pontos próximos em uma representação correspondem a pessoas que compartilham da mesma característica. No segundo caso, os pontos podem ser considerados como as posições em que torres de transmissão de sinais devem ser erguidas de modo que as distâncias entre elas respeitem valores pré-estabelecidos. E por fim, um problema muito conhecido em química é o problema da geometria de distâncias moleculares que consiste em determinar a configuração de uma molécula no espaço tridimensional, conhecidas as distâncias entre seus átomos.

Nos exemplos citados acima, as distâncias entre pontos devem ser euclidianas, ou seja, se x^i, x^j são pontos em $\mathbb{R}^m$, a distância entre eles é dada por:

$$d_{ij} = \left[\sum_{k=1}^{m} (x_k^i - x_k^j)^2 \right]^{\frac{1}{2}}. \tag{2.1}$$

Uma pergunta natural que surge diante destes exemplos é a seguinte: dado um conjunto $\Delta = \{\hat{d}_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$ com $N = n(n-1)/2$ números reais não negativos, é sempre possível encontrar uma configuração de pontos em um determinado espaço tal que as distâncias euclidianas entre pares desses pontos sejam dadas pelos elementos de Δ? Problemas que tratam dessa questão são chamados *problemas de distância euclidiana*. A seguir, estudaremos algumas propriedades desses problemas e veremos uma condição necessária e suficiente para que a pergunta acima tenha resposta positiva. Em seguida, apresentaremos os modelos que propusemos para realizar distâncias entre proteínas comparadas de um conjunto, discutiremos os resultados computacionais obtidos e os prós e contras de cada modelo.

2.1 Matrizes de distâncias euclidianas

Definição 2.1 *Uma matriz $D = [d_{ij}] \in \mathbb{R}^{n\times n}$ é chamada Matriz de Distâncias Euclidianas se existem n pontos $x^1, x^2, \ldots, x^n \in \mathbb{R}^m$, com $m \leq n-1$, tais que*

$$\|x^i - x^j\| = d_{ij}, \text{ para todo } 1 \leq i < j \leq n.$$

Deste modo, o conjunto $\{x^1, x^2, \ldots, x^n\}$ é dito ser uma realização em $\mathbb{R}^m$ da matriz D.

Fica claro pela definição acima que se D é uma matriz de distâncias euclidianas, então D é simétrica com diagonal nula, isto é, $d_{ii} = 0,\ i = 1, \ldots, n$. O conjunto de matrizes de distâncias euclidianas de ordem n é comumente denotado por EDM(n) (sigla para *Euclidean Distance Matrices of Order n*). Assim, dada uma matriz simétrica $\widehat{D}$ de ordem n com entradas não negativas e diagonal nula, estamos interessados em saber que condições devem ser cumpridas para que esta matriz pertença ao conjunto EDM(n). O teorema abaixo, devido a Schoenberg [81], estabelece um critério de verificação:

Teorema 2.1 *Seja $\widehat{D} = [\hat{d}_{ij}] \in \mathbb{R}^{n\times n}$ uma matriz simétrica com entradas não negativas e diagonal nula. Sejam também $M \in \mathbb{R}^{n\times n}$ e $V \in \mathbb{R}^{n\times(n-1)}$ tais que*

$$M = [(\hat{d}_{ij})^2] \quad e \quad V = \begin{bmatrix} -\mathbb{1}^T \\ I_{n-1} \end{bmatrix}. \tag{2.2}$$

Então, $\widehat{D}$ é uma matriz de distâncias euclidianas se, e somente se,

$$V^T M V \leq 0. \tag{2.3}$$

Prova: Vamos mostrar primeiramente que a condição é necessária. Suponha $\widehat{D} \in$ EDM(n). Então, existem n pontos $x^1, x^2, \ldots, x^n$ em $\mathbb{R}^m$, $m \leq (n-1)$, tais que

$$\|x^i - x^j\| = \hat{d}_{ij}, \quad \forall\ i, j. \tag{2.4}$$

Elevando ambos os lados de (2.4) ao quadrado, segue que

$$\|x^i - x^j\|^2 = \|x^i\|^2 + \|x^j\|^2 - 2\langle x^i, x^j\rangle = (\hat{d}_{ij})^2. \tag{2.5}$$

Seja $X \in \mathbb{R}^{m\times n}$ tal que a j-ésima coluna contém as coordenadas de x^j. Então, considerando a relação (2.5), podemos reescrever M da seguinte forma:

$$M = \text{diag}(X^T X)\ \mathbb{1}^T + \mathbb{1}\ \text{diag}(X^T X)^T - 2X^T X, \tag{2.6}$$

onde $\mathbb{1}$ e diag($X^T X$) são vetores em $\mathbb{R}^n$. Logo, usando (2.6) e o fato de que

$$V^T \mathbb{1} = 0,$$

obtemos

$$V^T MV = -2V^T X^T XV = -2(XV)^T(XV) \leq 0. \tag{2.7}$$

Provemos agora que a condição é suficiente. Suponha que $V^T MV \leq 0$. Então, segue que $-V^T MV \geq 0$ e, como esta matriz admite decomposição espectral, podemos escrever

$$-V^T MV = Q\Lambda Q^T \Longrightarrow -\frac{1}{2}V^T MV = \left(\frac{1}{\sqrt{2}}Q\Lambda^{1/2}\right)\left(\frac{1}{\sqrt{2}}\Lambda^{1/2}Q^T\right) = XX^T, \tag{2.8}$$

onde $X \in \mathbb{R}^{(n-1)\times(n-1)}$ e posto$(X) = m \leq n-1$. Denotando as linhas da matriz X por $x^2, x^3, \ldots, x^n \in \mathbb{R}^{n-1}$, da igualdade de matrizes em (2.8) obtemos

$$\begin{aligned} (\hat{d}_{1i})^2 &= \|x^i\|^2, \quad i = 2, \ldots, n, \\ (\hat{d}_{1i})^2 + (\hat{d}_{1j})^2 - (\hat{d}_{ij})^2 &= \|x^i\|^2 + \|x^j\|^2 - \|x^i - x^j\|^2, \quad i \neq j. \end{aligned} \tag{2.9}$$

Logo, de (2.9) concluímos que:

$$\|x^i\| = \hat{d}_{1i} \quad \text{para} \quad i = j \quad \text{e} \quad \|x^i - x^j\| = \hat{d}_{ij} \quad \text{para} \quad i \neq j, \tag{2.10}$$

onde $2 \leq i < j \leq n$. Assim, de acordo com (2.10), o ponto $x^1 = 0$ (origem do sistema de coordenadas) juntamente com os pontos $x^2, x^3, \ldots, x^n$ formam uma realização para a matriz $\widehat{D}$. Portanto, $\widehat{D} \in \text{EDM}(n)$. ■

A condição suficiente dada pelo Teorema 2.1 garante a realização de n pontos em um espaço euclidiano de dimensão no máximo igual a $n-1$. Além disso, as coordenadas dos pontos encontrados não são únicas. De fato, qualquer configuração $\{\tilde{x}^1, \ldots, \tilde{x}^n\}$ obtida de $\{x^1, \ldots, x^n\}$ por um movimento rígido, também será uma realização para a matriz $\widehat{D}$. A seguir, ilustraremos o teorema através de dois exemplos.

Exemplo 1: Considere a tarefa de alocar quatro pontos em $\mathbb{R}^3$ cujas distâncias dois a dois são dadas pelas entradas da matriz

$$\widehat{D} = \begin{pmatrix} 0 & \sqrt{2} & \sqrt{2} & 1 \\ \sqrt{2} & 0 & \sqrt{2} & 1 \\ \sqrt{2} & \sqrt{2} & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{2.11}$$

Neste exemplo

$$-\frac{1}{2}V^T MV = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \geq 0. \tag{2.12}$$

Decompondo (2.12) da mesma forma que em (2.8), obtemos

$$X = \begin{pmatrix} -1.21313 & -1.21313 & -0.88807 \\ -0.70711 & 0.70711 & 0 \\ -0.16826 & -0.16826 & 0.45970 \end{pmatrix}. \tag{2.13}$$

Portanto, as colunas de X juntamente com o vetor nulo formam uma configuração em $\mathbb{R}^3$ que realiza as distâncias $\hat{d}_{ij}$ dadas pelas entradas da matriz (2.11). A Figura 2.1 mostra a disposição espacial dos pontos com relação ao sistema de coordenadas canônico de $\mathbb{R}^3$. Nesta figura temos

$$\begin{aligned} x^1 &= (0,0,0)^T, \\ x^2 &= (-1.21313, -0.70711, -0.16826)^T, \\ x^3 &= (-1.21313, 0.70711, -0.16826)^T, \\ x^4 &= (-0.88807, 0, 0.45970)^T. \end{aligned} \tag{2.14}$$

Notemos que a configuração encontrada não é única. De fato, $\tilde{x}^1 = (0,0,1)^T$, $\tilde{x}^2 = (1,0,0)^T$, $\tilde{x}^3 = (0,0,1)^T$ e $\tilde{x}^4 = (0,0,0)^T$ formam outra realização para $\widehat{D}$.

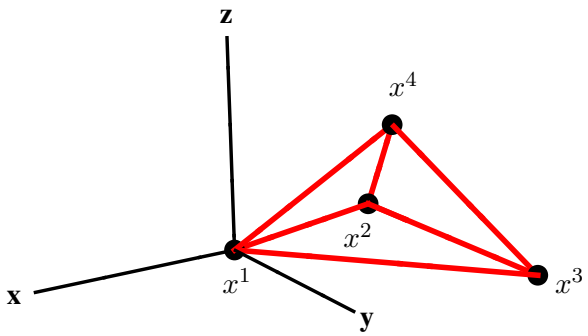


Figura 2.1: Uma configuração de pontos que realiza as distâncias de $\widehat{D}$.

Exemplo 2: Considere agora o problema de colocar três pontos em $\mathbb{R}^2$, de forma que as distâncias entre pares desses pontos sejam dadas pela matriz

$$\widehat{D} = \begin{pmatrix} 0 & 0.1 & 0.1 \\ 0.1 & 0 & 10.2 \\ 0.1 & 10.2 & 0 \end{pmatrix}. \tag{2.15}$$

A matriz

$$-\frac{1}{2}V^T M V = \begin{pmatrix} -0.01 & 52.01 \\ 52.01 & -0.01 \end{pmatrix} \tag{2.16}$$

não é semi-definida positiva pois um de seus autovalores é negativo. Portanto, pelo Teorema 2.1, não é possível determinar uma configuração de três pontos no plano tal que as distâncias de (2.15) sejam realizadas.

Na prática, verificar se $\widehat{D} \in \mathrm{EDM}(n)$ não é tarefa fácil, sobretudo se a quantidade de pontos considerada é grande. O número de operações necessárias para verificar se uma matriz pertence a EDM(n) é da ordem de n^3 [46]. Com efeito, a construção da matriz $V^T MV$ requer n operações e, a propriedade de positivo-definição é verificada via fatoração de Cholesky, que é da ordem de $n^3/3$ operações. J. B. Saxe mostra em [80] que o problema de alocar pontos em um espaço euclidiano n-dimensional é fortemente NP-difícil. J. Dattorro elaborou uma rotina para verificar se uma matriz $\widehat{D}$ simétrica, com diagonal nula e entradas não negativas, é uma matriz de distâncias euclidianas. A rotina, denominada *isedm*, pode ser obtida gratuitamente em `http://www.convexoptimization.com/wikimization`.

Antes de apresentarmos os modelos que propusemos para a construção de mapas de proteínas comparadas, faremos um breve estudo sobre o Problema da Geometria de Distâncias Moleculares, destacando algumas contribuições existentes na literatura. Também apresentamos e discutimos alguns experimentos que fizemos na tentativa de resolver o problema utilizando a rotina de otimização numérica *GENCAN* [13].

2.2 O problema da geometria de distâncias moleculares

Suponha uma molécula com n átomos. O problema da geometria de distâncias moleculares consiste em recuperar a estrutura tridimensional desta molécula a partir do conjunto de distâncias $\hat{d}_{ij}$ entre seus pares de átomos. Ou seja, precisamos determinar uma configuração de n pontos $\{x^1, x^2, \ldots, x^n\} \subset \mathbb{R}^3$ tais que $\|x^i - x^j\| = \hat{d}_{ij}$ para todo $1 \leq i < j \leq n$. Se todas as distâncias $\hat{d}_{ij}$ são conhecidas exatamente, o Teorema 2.1 nos garante que existe uma configuração de pontos que as realiza. Em particular, Q. Dong e Z. Wu constróem em [25] um algoritmo que resolve o problema em tempo linear quando todas as distâncias intra-átomos são conhecidas exatamente. No entanto, em situações práticas, apenas um subconjunto de distâncias intra-átomos é conhecida. Nestes casos, o problema então passa a ser o de completar o conjunto de distâncias, determinando as distâncias desconhecidas, e encontrar uma configuração de pontos que realize todas as distâncias.

Na literatura existem basicamente duas maneiras distintas de tratar o problema numericamente. Na primeira delas, pesquisadores formulam um problema de otimização não linear e empregam métodos contínuos para obterem uma solução aproximada [3, 31, 89]. Na segunda, o problema é resolvido usando técnicas de otimização discreta e teoria de grafos [59, 60, 61, 62, 64].

Nesta seção utilizamos a rotina de otimização numérica *GENCAN* para realizar alguns experimentos com o problema

$$\begin{aligned} \min \quad & \sum_{i,j} (\|x^i - x^j\| - \hat{d}_{ij})^2 \\ \text{s. a} \quad & x^i \in \mathbb{R}^3, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{2.17}$$

Notemos que as variáveis do problema (2.17) são as coordenadas dos pontos $x^i \in \mathbb{R}^3$ e a função objetivo não é diferenciável em configurações de pontos tais que $x^i = x^j$ para algum i e j. No entanto, como estamos interessados em realizar distâncias $\hat{d}_{ij}$ sempre maiores que zero, não corremos o risco de obter configurações com pontos coincidentes e enfrentar problemas numéricos com a aplicação de um algoritmo de minimização que requer derivadas primeiras analíticas. Esta afirmação foi demonstrada por Jan De Leeuw em [23].

Nos experimentos que vamos apresentar mais adiante, as soluções obtidas por *GENCAN* para o problema (2.17) foram comparadas com a estrutura verdadeira da molécula analisada. Esta comparação foi feita da seguinte forma: dadas a configuração original de átomos e uma solução numérica de (2.17), determinamos um movimento rígido que sobrepõe as duas estruturas de maneira ótima, ou seja, mantendo fixa uma das estruturas, o movimento é aplicado à outra estrutura na tentativa de fazer com que elas coincidam exatamente. O problema de encontrar este movimento rígido é conhecido na literatura como *Procrustes* ou *RMSD* (do inglês *Root Mean Square Deviation*) [32, 50] e é descrito a seguir.

Sejam M_0 e M_1 matrizes em $\mathbb{R}^{n\times p}$, o problema *Procrustes* consiste em determinar uma matriz ortogonal $Q \in \mathbb{R}^{p\times p}$ que minimiza a função

$$g(Q) = \|M_0 - M_1 Q\|_F, \tag{2.18}$$

onde $\|\cdot\|_F$ é a norma de Frobenius para matrizes. No caso em que estamos trabalhando, n é o número de átomos e $p = 3$ é a dimensão do espaço euclidiano. A matriz ortogonal Q^* que minimiza (2.18) pode ser determinada analiticamente. G. Golub e C. Van Loan em [32] utilizam uma decomposição em valores singulares para determinar Q^*. Já S. Kearsley em [50] emprega quatérnios unitários para encontrar Q^*. A diferença entre as duas formas de resolução do problema *Procrustes* está no fato de que a decomposição em valores singulares não garante que a matriz ortogonal Q^* seja uma matriz de rotação. Há casos em que Q^* pode ser uma matriz de reflexão ($\det(Q^*) = -1$). O mesmo não ocorre na formulação proposta por Kearsley pois quatérnios unitários são usados para descrever rotações. Neste caso, a matriz Q^* terá sempre determinante igual a 1, sendo portanto uma matriz de rotação. Mais detalhes sobre quatérnios e rotações podem ser encontrados nas referências [19, 43, 57, 67]. Nos experimentos que apresentamos a seguir, investigamos se as soluções obtidas por *GENCAN* para o problema (2.17) diferem da configuração original por um movimento de rotação ou reflexão. Para isso, resolvemos o problema *Procrustes* usando as duas formulações descritas acima: via decomposição em valores singulares e via quatérnios unitários.

2.2.1 Primeiros experimentos numéricos

Para realizar os experimentos com o problema (2.17), retiramos seis proteínas do *Protein Data Bank* e tomamos somente as coordenadas dos carbonos alfa (C_α) presentes em cada estrutura. A sigla de cada proteína e o número de átomos C_α considerados estão indicados na Tabela 2.1. Propusemos quatro baterias de testes cujos detalhes são discutidos a seguir. Os experimentos foram rodados em um computador Apple com as seguintes características: sistema operacional MAC OS X 10.5, processador Intel Core 2 Duo com 2.4GHz e 2GB de memória.

Proteína	n_{C_α}
1AMU	509
1OOH	126
2012	407
3RAT	124
6PAX	133
1lmO	3535

Tabela 2.1: Proteínas utilizadas nos experimentos.

Primeira bateria de testes

Neste primeiro conjunto de experimentos, admitimos conhecidas todas as distâncias $\hat{d}_{ij}$ entre os pares de átomos C_α. Tomando as cinco primeiras proteínas da Tabela 2.1, rodamos cada problema 20 vezes, partindo de um ponto inicial gerado aleatoriamente, e apresentamos na Tabela 2.2 somente os testes em que *GENCAN* obteve o menor valor de função. As colunas da Tabela 2.2 obedecem a seguinte legenda: *ndist* é o número total de distâncias entre pares de átomos, *nvar* é o número de variáveis do problema, *iter* e *avalf* são, respectivamente, o total de iterações e avaliações de função realizadas, $f(x^*)$ é o valor final de função e $t(s)$ é o tempo medido em segundos. A coluna *rotação* indica a quantidade de rodadas em que *GENCAN* obteve uma solução que difere da configuração original por um movimento rígido cuja matriz ortogonal é de rotação. Já a coluna *reflexão*, indica o total de rodadas em que a solução numérica difere da configuração original por um movimento rígido onde a matriz ortogonal é de reflexão. Em todos os testes indicados *GENCAN* parou com a norma do gradiente menor que 10^{-4}.

Na Tabela 2.2, os valores finais de função pequenos mostram que *GENCAN* conseguiu encontrar uma configuração de pontos em $\mathbb{R}^3$ que realiza todas as distâncias indicadas na coluna *ndist*. Os testes com a proteína 6PAX chamaram mais atenção pois em 6 das 20 rodadas realizadas constatamos que a rotina parou em um ponto cujo valor final de função é da ordem de 10^3 mas a norma do gradiente neste ponto é inferior a 10^{-4}. Com efeito, nestas rodadas pudemos observar que algumas distâncias não foram realizadas satisfatoriamente.

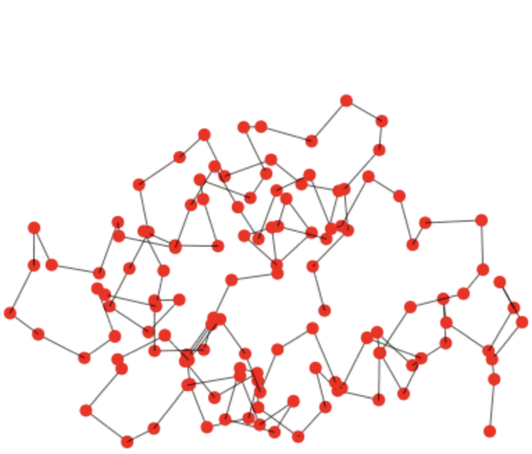
Proteína	ndist	nvar	iter	avalf	$f(x^*)$	$t(s)$	rotação	reflexão
1AMU	129286	1527	20	39	1.21E−19	1.76	13	7
1OOH	7875	378	14	30	3.61E−19	0.08	12	8
2012	82621	1221	17	36	1.10E−19	0.99	12	8
3RAT	7626	372	17	27	3.84E−19	0.11	13	7
6PAX	8778	399	18	42	7.27E−19	0.21	6	8
1lmO	6246345	10605	26	68	5.59E−21	120.23	101	99

Tabela 2.2: Geometria de distâncias moleculares – primeiro conjunto de testes.

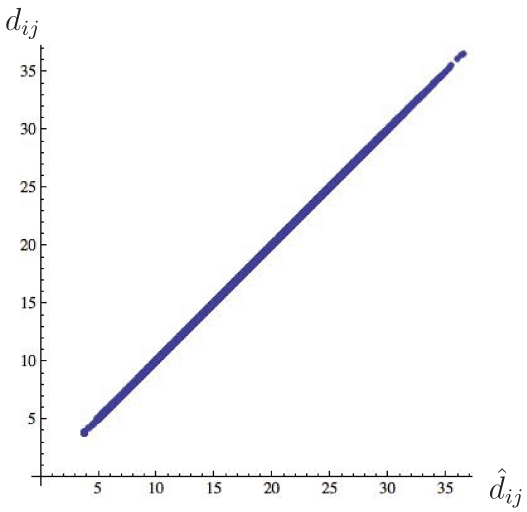
Com a proteína 1lmO realizamos um teste um pouco diferente: admitindo conhecidas todas as distâncias entre os pares de átomos, realizamos 200 rodadas de *GENCAN* fornecendo pontos iniciais diferentes. A última linha da Tabela 2.2 indica o melhor resultado obtido por *GENCAN* ao final de todas as rodadas. O experimento durou cerca de sete horas para executar as 200 rodadas e aproximadamente dois minutos para resolver o problema indicado na tabela. Observemos que em praticamente metade dos experimentos (101 rodadas) a rotina consegue recuperar a estrutura da proteína. Nas outras 99 rodadas a solução obtida por *GENCAN* difere da configuração original por um movimento rígido cuja matriz ortogonal é de reflexão.

Escolhemos a proteína 1OOH para ilustrar um teste em que *GENCAN* obtém uma configuração de pontos que difere da estrutura original por um movimento rígido cuja matriz ortogonal é de reflexão. A Figura 2.2 a) mostra a configuração original de 1OOH com 126 carbonos alfa. Cada átomo está representado por um ponto em $\mathbb{R}^3$ e pontos consecutivos estão ligados por segmentos de reta. Já a Figura 2.2 b) compara as distâncias originais (eixo $\hat{d}_{ij}$) e as distâncias obtidas numericamente (eixo d_{ij}). Notemos que os ajustes obtidos são perfeitos, pois todos os pares ordenados $(\hat{d}_{ij}, d_{ij})$ estão sobre a reta $y = x$. As análises realizadas com o problema *Procrustes* estão indicadas nas Figuras 2.3 a) e 2.3 b). Embora tenhamos considerado todos os átomos para determinar os movimentos rígidos, construímos as figuras tomando apenas os 10 primeiros átomos C_α de 1OOH.

A Figura 2.3 a) mostra a sobreposição da configuração obtida por *GENCAN* (pontos verdes) com a configuração original (pontos vermelhos que não aparecem na imagem). Para determinarmos a sobreposição resolvemos o problema *Procrustes* pela formulação proposta por H. Golub e C. Van Loan e obtivemos um movimento rígido cuja matriz ortogonal é de reflexão. Já a Figura 2.3 b) mostra a sobreposição das duas configurações obtida pela resolução do problema *Procrustes* formulado por S. Kearsley. Nesse caso, a matriz ortogonal é de rotação. Fica evidente na Figura 2.3 b) que a solução obtida por *GENCAN* (pontos verdes) é a imagem refletida da configuração original (pontos vermelhos). Sendo assim, não é possível determinar um movimento de rotação que transforme uma estrutura na outra. As soluções obtidas na resolução do problema *Procrustes* (2.18) e os valores finais da função $g(Q)$ estão indicados na Tabela 2.3.

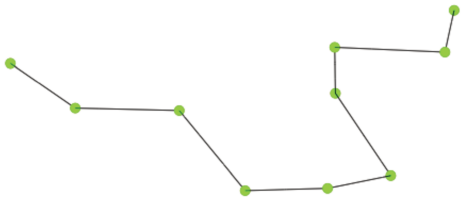


a) Estrutura original de 1OOH

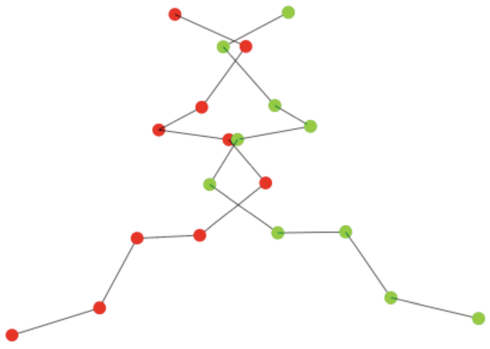


b) 7875 distâncias ajustadas

Figura 2.2: Análises realizadas com a proteína 1OOH.



a) Reflexão: estruturas sobrepostas



b) Rotação: estruturas não sobrepostas

Figura 2.3: Proteína 1OOH: análise via resolução do problema *Procrustes*.

Procrustes via Decomposição em Valores Singulares

$$Q_{\text{ref}} = \begin{pmatrix} 0.545563 & 0.463014 & -0.698555 \\ 0.835074 & -0.229917 & 0.49979 \\ -0.0708004 & 0.856012 & 0.512085 \end{pmatrix}, \quad g(Q_{\text{ref}}) = 2.87132\text{E}{-10},$$

Procrustes via Quatérnios Unitários

$$Q_{\text{rot}} = \begin{pmatrix} 0.583586 & 0.810877 & 0.0436581 \\ 0.798626 & -0.563372 & -0.211681 \\ -0.147052 & 0.158401 & -0.976363 \end{pmatrix}, \quad g(Q_{\text{rot}}) = 1.37579\text{E}{+02}.$$

Tabela 2.3: Resultados do problema *Procrustes* envolvendo `1OOH`.

Segunda bateria de testes

Neste conjunto de experimentos consideramos as cinco primeiras proteínas indicadas na Tabela 2.1 e todas as distâncias $\hat{d}_{ij}$ entre seus pares de átomos C_α. No entanto, para simular erros, perturbamos cada distância $\hat{d}_{ij}$ com um número gerado aleatoriamente no intervalo $[-\rho, \rho]$, onde $|\rho| \leq 1$. Para cada proteína e cada valor de ρ fixos, resolvemos (2.17) 20 vezes partindo de um ponto inicial diferente. Em todas as rodadas a rotina parou com a norma do gradiente menor que 10^{-4}. A Tabela 2.4 indica apenas os testes em que *GENCAN* atingiu os menores valores de função. Notemos nesta tabela que à medida que aumentamos o valor de ρ os valores finais de função também aumentam por um fator de 10^2. Os números totais de iterações, avaliações de função e tempo de execução, indicados na Tabela 2.5, foram muito semelhantes aos valores obtidos nos experimentos da primeira bateria de testes.

Proteína	$\rho = 10^{-5}$	$\rho = 10^{-4}$	$\rho = 10^{-3}$	$\rho = 10^{-2}$	$\rho = 10^{-1}$	$\rho = 1$
`1AMU`	4.263864E−06	4.263864E−04	4.263864E−02	4.263864E+00	4.263864E+02	4.263871E+04
`1OOH`	2.527256E−07	2.527256E−05	2.527256E−03	2.527253E−01	2.527222E+01	2.526905E+03
`2O12`	2.709833E−06	2.709833E−04	2.709833E−02	2.709833E+00	2.709808E+02	2.709746E+04
`3RAT`	2.453024E−07	2.453024E−05	2.453024E−03	2.453025E−01	2.453041E+01	2.453186E+03
`6PAX`	2.829028E−07	2.829028E−05	2.829028E−03	2.829023E−01	2.828973E+01	2.828446E+04

Tabela 2.4: Valores finais de função em testes com simulação de erros.

Ilustramos a seguir o teste realizado com a proteína `3RAT` em que fixamos $\rho = 1$ e obtivemos como valor final de função 2.453186E+03 (quinta linha e última coluna da Tabela 2.4). A Figura 2.4 a) mostra a estrutura original de `3RAT` com 124 átomos de carbono alfa. A Figura 2.4 b) mostra a comparação via *Procrustes* da estrutura original (pontos vermelhos) com a configuração obtida numericamente (pontos verdes). Na figura estão indicados apenas os 10 primeiros átomos C_α. Ao resolvermos o problema *Procrustes* (2.18) por ambas as

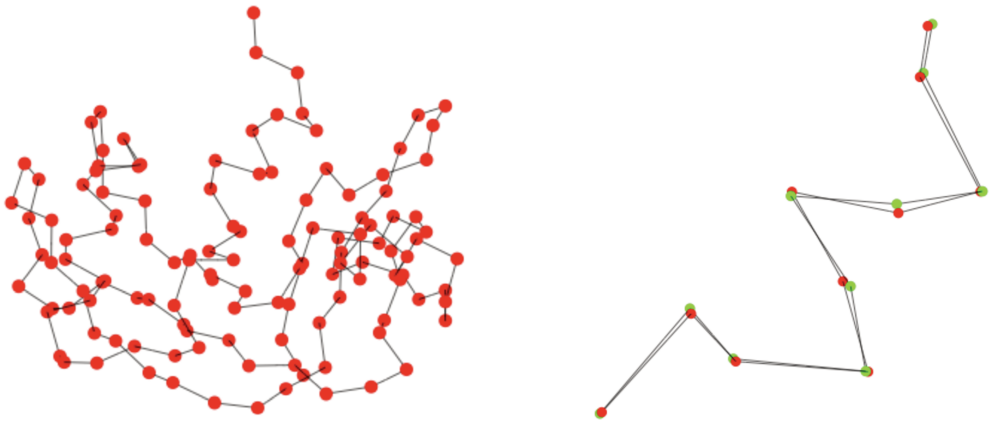
	$\rho = 10^{-5}$			$\rho = 10^{-4}$			$\rho = 10^{-3}$		
Proteína	iter	avalf	$t(s)$	iter	avalf	$t(s)$	iter	avalf	$t(s)$
1AMU	18	33	1.75	17	35	1.60	17	34	1.50
1OOH	16	40	0.10	20	38	0.11	17	30	0.10
2O12	21	52	1.31	21	45	1.53	18	34	1.25
3RAT	16	29	0.11	17	34	0.13	17	35	0.13
6PAX	16	34	0.17	22	48	0.26	19	50	0.15

	$\rho = 10^{-2}$			$\rho = 10^{-1}$			$\rho = 1$		
Proteína	iter	avalf	$t(s)$	iter	avalf	$t(s)$	iter	avalf	$t(s)$
1AMU	20	34	1.79	24	63	1.76	18	40	1.84
1OOH	15	33	0.09	13	27	0.08	16	36	0.09
2O12	22	51	1.50	18	27	1.27	22	45	1.76
3RAT	15	19	0.11	16	30	0.12	15	28	0.10
6PAX	17	41	0.17	18	41	0.19	21	56	0.22

Tabela 2.5: Desempenho de *GENCAN* em testes com simulação de erros.

formulações discutidas, obtivemos a mesma matriz de rotação Q. A matriz e o valor final da função $g(Q)$ (2.18) são, respectivamente, dados por:

$$Q = \begin{pmatrix} -0.728719 & -0.651336 & 0.211497 \\ 0.141985 & -0.44583 & -0.883785 \\ 0.669932 & -0.614001 & 0.417364 \end{pmatrix}, \quad g(Q) = 2.02778.$$



a) Estrutura original de 3RAT b) Comparação via *Procrustes*

Figura 2.4: Análises realizadas com a proteína 3RAT.

Podemos notar pela Figura 2.4 b) que apesar de termos perturbado os valores $\hat{d}_{ij}$ com números gerados no intervalo [-1, 1], conseguimos obter um movimento rígido que deixa as

estruturas muito próximas. A Figura 2.5 mostra um gráfico onde o eixo das abscissas é formado pelas distâncias perturbadas com erros $(\hat{d}_{ij} + \rho_{ij})$, onde $\rho_{ij} \in [-1, 1]$, e o eixo das ordenadas é formado pelas distâncias obtidas após os ajustes (d_{ij}). Apesar do valor final de função não ser pequeno (da ordem de 10^3), notamos que os pares ordenados $(\hat{d}_{ij} + \rho_{ij}, d_{ij})$ concentram-se em torno da reta $y = x$.

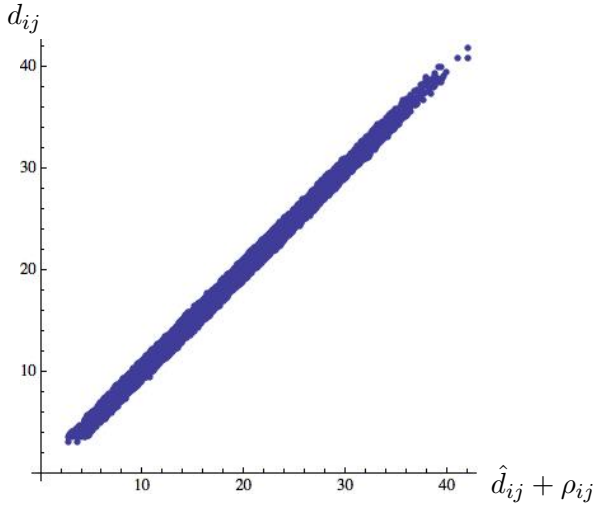


Figura 2.5: Ajuste de 7626 distâncias com erros.

Terceira bateria de testes

Neste conjunto de experimentos utilizamos as proteínas indicadas na Tabela 2.1. No entanto, não consideramos mais todas as distâncias entre pares de átomos para resolver o problema (2.17). Tomamos agora somente as distâncias entre átomos consecutivos na estrutura 3D da proteína e as distâncias menores que um valor fixo d_{fix}. Fizemos testes variando o parâmetro d_{fix} e analisamos os resultados com base na resolução do problema *Procrustes*. Para cada proteína e cada valor de d_{fix} rodamos o problema (2.17) 50 vezes partindo de um ponto inicial diferente. Em todas as rodadas *GENCAN* parou com a norma do gradiente menor que 10^{-4}.

As Tabelas 2.6 e 2.7 mostram apenas as informações referentes aos resultados dos testes em que a rotina de otimização atingiu os menores valores de função objetivo. Na Tabela 2.6 estão indicados o número total de distâncias entre pares de átomos (*ndist*), o número total de distâncias consideradas na resolução de (2.17) (*nd*) e o valor final de função objetivo $(f(x^*))$. Já a Tabela 2.7 mostra o desempenho de *GENCAN* na resolução dos problemas. Nos testes com $d_{\text{fix}} = 15$, *GENCAN* consegue encontrar configurações de pontos que ajustam todas as distâncias entre os pares de átomos usando menos de 36 % do total de distâncias conhecidas. As configurações obtidas nestes experimentos com $d_{\text{fix}} = 15$ diferem da configuração original ora por um movimento rígido envolvendo rotação, ora por um movimento rígido envolvendo reflexão. Este comportamento já era esperado, de acordo com o que vimos nos resultados de

testes anteriores. *GENCAN* realizou mais iterações e avaliações de função nos experimentos com $d_{\text{fix}} = 6$, como podemos observar na Tabela 2.7. As Tabelas 2.6 e 2.7 também trazem

		$d_{\text{fix}} = 6$		$d_{\text{fix}} = 10$		$d_{\text{fix}} = 15$	
Proteína	ndist	nd	$f(x^*)$	nd	$f(x^*)$	nd	$f(x^*)$
1AMU	129286	1614	8.438471E+00	4650	2.514611E+02	14193	5.554569E−17
1OOH	7875	386	1.929190E−01	1074	3.657396E−17	2824	1.198010E−18
2O12	82621	1221	4.402781E+00	3899	5.413680E+00	11083	2.446779E−13
3RAT	7626	393	1.189479E+00	1045	2.463908E−10	2724	2.767508E−18
6PAX	8778	386	4.858342E−03	916	2.438740E+02	2292	1.311720E−13

		$d_{\text{fix}} = 10$		$d_{\text{fix}} = 9$		$d_{\text{fix}} = 8$	
Proteína	ndist	nd	$f(x^*)$	nd	$f(x^*)$	nd	$f(x^*)$
1lmO	6246345	1085393	1.393982E−17	839689	6.667112E−15	677434	4.904778E−13

Tabela 2.6: Distâncias consideradas e valores finais de função.

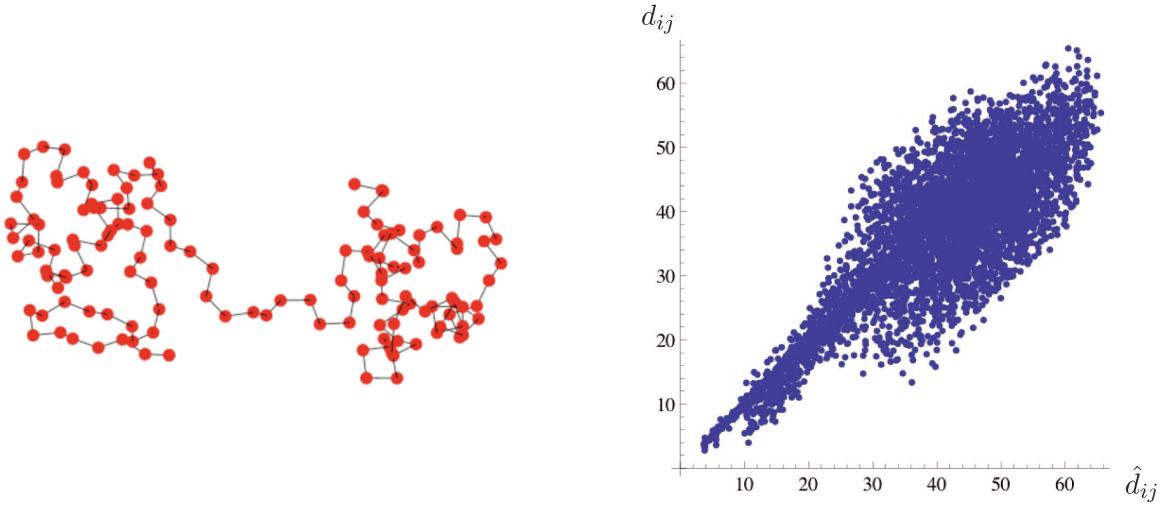
	$d_{\text{fix}} = 6$			$d_{\text{fix}} = 10$			$d_{\text{fix}} = 15$		
Proteína	iter	avalf	$t(s)$	iter	avalf	$t(s)$	iter	avalf	$t(s)$
1AMU	121	428	48.88	52	173	2.53	64	192	2.88
1OOH	150	304	8.490	34	93	0.15	18	35	0.08
2O12	328	733	137.91	57	176	3.06	36	127	1.60
3RAT	111	244	5.19	44	115	0.45	28	78	0.14
6PAX	66	160	3.54	142	326	6.34	49	84	1.80

	$d_{\text{fix}} = 10$			$d_{\text{fix}} = 9$			$d_{\text{fix}} = 8$		
Proteína	iter	avalf	$t(s)$	iter	avalf	$t(s)$	iter	avalf	$t(s)$
1lmO	63	245	103.51	65	235	111.52	35	98	66.41

Tabela 2.7: Desempenho de *GENCAN* na resolução dos problemas.

os resultados de alguns experimentos realizados com proteína 1lmO. Conseguimos recuperar a estrutura original desta proteína tomando 3 valores diferentes para o parâmetro d_{fix}: 10, 9 e 8. Nestas três situações o número de distâncias (nd) utilizadas na resolução dos problemas não ultrapassa 18 % do total de distâncias ($ndist$).

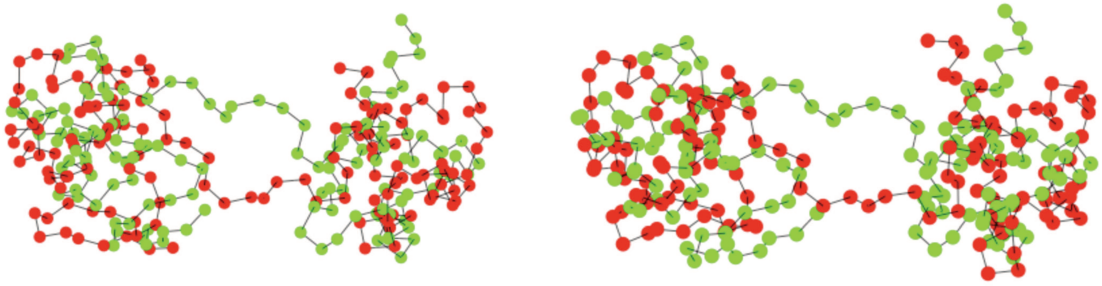
Ilustramos os resultados de um teste realizado com a proteína 6PAX. A Figura 2.6 a) mostra a estrutura original de 6PAX com 133 átomos de carbono alfa e a Figura 2.6 b) mostra um gráfico onde as distâncias originais entre os pares de átomos estão no eixo $\hat{d}_{ij}$ e, as distâncias obtidas numericamente estão no eixo d_{ij}. Neste experimento fornecemos para a rotina de otimização somente as distâncias entre átomos consecutivos e as distâncias menores que $d_{\text{fix}} = 10$, totalizando 916 distâncias (aproximadamente 10 % do total de distâncias). O valor final de função obtido foi 243.874 e, como podemos perceber pelo gráfico, vários valores d_{ij} não foram ajustados satisfatoriamente.



a) Estrutura original de `6PAX` b) Tentativa de ajustar 8778 distâncias

Figura 2.6: Análises realizadas com a proteína `6PAX`.

Os emaranhados indicados nas Figuras 2.7 a) e 2.7 b) mostram a comparação, via *Procrustes*, da estrutura original de `6PAX` (pontos vermelhos) com a estrutura obtida por *GENCAN* (pontos verdes). Para obter a Figura 2.7 a) resolvemos o problema *Procrustes* utilizando decomposição em valores singulares e obtivemos um movimento rígido envolvendo uma matriz de reflexão. No caso da Figura 2.7 b) aplicamos quatérnios unitários e obtivemos um movimento rígido envolvendo uma matriz de rotação. As matrizes ortogonais obtidas e os valores finais da função (2.18) estão indicados na Tabela 2.8. De modo geral, os experimentos que fizemos nesta terceira bateria de testes nos mostraram que é possível recuperar a estrutura 3D de uma proteína utilizando apenas um subconjunto de distâncias entre pares de átomos e uma rotina de otimização contínua. Em particular, a rotina *GENCAN* conseguiu resolver muito rapidamente todos os problemas propostos.



a) *Procrustes* – reflexão b) *Procrustes* – rotação

Figura 2.7: Comparação das estruturas via *Procrustes*.

Procrustes via Decomposição em Valores Singulares
$Q_{\text{ref}} = \begin{pmatrix} 0.619471 & -0.528366 & -0.580591 \\ -0.756835 & -0.59837 & -0.262972 \\ 0.208462 & -0.602315 & 0.770558 \end{pmatrix}, \quad g(Q_{\text{ref}}) = 1.17023\text{E+}02,$
Procrustes via Quatérnios Unitários
$Q_{\text{rot}} = \begin{pmatrix} 0.584185 & -0.546667 & -0.599903 \\ 0.787414 & 0.20257 & 0.582189 \\ -0.196741 & -0.812478 & 0.548792 \end{pmatrix}, \quad g(Q_{\text{rot}}) = 1.30287\text{E+}02.$

Tabela 2.8: Resultados do problema *Procrustes* envolvendo `6PAX`.

Quarta bateria de testes

Dada uma proteína qualquer com n átomos de carbono alfa, assumiremos agora que sua estrutura é articulável. Sem perda de generalidade, colocamos o primeiro átomo C_α na origem do sistema de coordenadas tridimensional e mantemos fixo o segmento ℓ_{12} que une os dois primeiros átomos (veja a Figura 2.8). Vamos admitir também que o ângulo entre dois segmentos consecutivos é fixo. Sendo assim, os movimentos permitidos à estrutura são os seguintes: o segmento ℓ_{23} poderá rodar de um ângulo θ_1 ao redor do eixo definido pelo segmento ℓ_{12}. O segmento ℓ_{34} poderá rodar de um ângulo θ_2 ao redor do eixo definido pelo segmento ℓ_{23} e, procedendo seqüencialmente dessa maneira, o último segmento (que une os átomos $n-1$ e n) rodará de um ângulo θ_{n-2} ao redor do eixo definido pelo penúltimo segmento (que une os átomos $n-2$ e $n-1$).

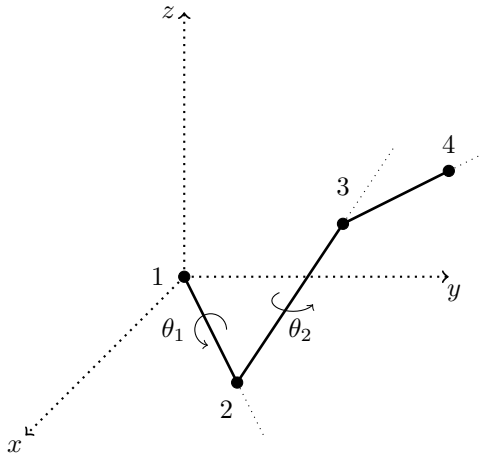


Figura 2.8: Proteína modelada como uma estrutura articulável.

Ao modelar a proteína como uma estrutura articulável, admitimos conhecidas as distâncias entre dois átomos de carbono alfa consecutivos (comprimentos dos segmentos que unem os átomos i e $i+1$). Formulamos então o problema de recuperar a estrutura 3D de uma proteína como o problema de determinar os ângulos θ_i, $1 \leq i \leq n-2$, tais que as distâncias entre os pares de átomos C_α menores que um valor fixo d_{fix} sejam realizadas efetivamente. Lembremos que o problema (2.17) resolvido nas baterias de testes anteriores tinha como variáveis as coordenadas dos átomos. Agora as variáveis passam a ser os $n-2$ ângulos que definem os movimentos de rotação. Para determinarmos a orientação espacial da estrutura da proteína, precisamos fazer uma composição de rotações. Seja $\omega = (\omega_1, \omega_2, \omega_3)^T$ um vetor unitário em $\mathbb{R}^3$ que define o eixo de rotação e $\theta \in [0, 2\pi]$ o ângulo de rotação. A matriz que descreve uma rotação de θ em torno de ω é dada por

$$R(\theta, \omega) = \begin{pmatrix} c_\theta + \omega_1^2 v_\theta & \omega_1\omega_2 v_\theta - \omega_3 s_\theta & \omega_1\omega_3 v_\theta + \omega_2 s_\theta \\ \omega_1\omega_2 v_\theta + \omega_3 s_\theta & c_\theta + \omega_2^2 v_\theta & \omega_2\omega_3 v_\theta - \omega_1 s_\theta \\ \omega_1\omega_3 v_\theta - \omega_2 s_\theta & \omega_2\omega_3 v_\theta + \omega_1 s_\theta & c_\theta + \omega_3^2 v_\theta \end{pmatrix}, \tag{2.19}$$

onde $s_\theta = \text{sen}(\theta)$, $c_\theta = \cos(\theta)$ e $v_\theta = 1 - \cos(\theta)$. Outras maneiras de descrever rotações em $\mathbb{R}^3$ podem ser encontradas em [86].

Para testar o problema (2.17) reformulado com ângulos, tomamos a proteína `1AMU` e realizamos alguns experimentos variando o número de átomos em sua estrutura. Por exemplo, em um teste consideramos apenas os 10 primeiros átomos C_α consecutivos da cadeia de `1AMU`. Em outro, consideramos os 20 primeiros átomos C_α consecutivos e assim por diante. Empregamos também dois valores diferentes para o parâmetro d_{fix}: 6 e 15. Utilizamos a rotina *GENCAN* na resolução dos problemas com o vetor gradiente calculado via diferenças finitas. Cada problema foi resolvido 50 vezes partindo de um ponto inicial gerado aleatoriamente. A Tabela 2.9 mostra apenas os resultados dos testes em que *GENCAN* atingiu o menor valor de função. As colunas nC_α e *ndist* indicam, respectivamente, a quantidade de átomos C_α consecutivos de `1AMU` empregados na resolução dos problemas e o número total de distâncias entre estes pares de átomos. Para cada valor de d_{fix}, a coluna *nd* indica a quantidade de distâncias efetivamente utilizadas. E, por fim, as colunas *iter*, *avalf*, $t(s)$ e $f(x^*)$ mostram o desempenho de *GENCAN* na resolução dos problemas.

		$d_{\text{fix}} = 6$					$d_{\text{fix}} = 15$				
nC_α	ndist	nd	iter	avalf	$t(s)$	$f(x^*)$	nd	iter	avalf	$t(s)$	$f(x^*)$
10	45	22	1	12	0.004	5.899460E−30	43	4	14	0.011	2.892934E−29
20	190	46	2	15	0.044	4.910996E−29	111	2	12	0.067	4.677829E−28
30	435	72	1	8	0.103	3.571215E−28	180	18	54	1.300	5.465923E−28
50	1225	131	4	20	1.772	3.730164E−28	381	69	175	111.063	4.030227E−27

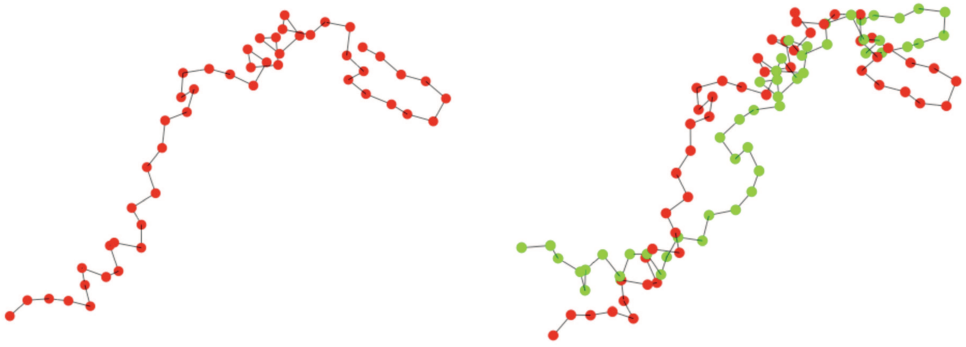
Tabela 2.9: Testes realizados com átomos C_α consecutivos de `1AMU`.

Como podemos observar pela Tabela 2.9, *GENCAN* consegue encontrar configurações de pontos com valor final de função pequeno. Nos testes com $d_{\text{fix}} = 15$, a rotina obtém configurações que realizam todas as distâncias entre pares de átomos. Já nos experimentos

com $d_{\text{fix}} = 6$, embora todas as distâncias utilizadas (nd) sejam ajustadas efetivamente, observamos casos em que *GENCAN* obteve configurações diferentes da estrutura original. Com efeito, este comportamento era esperado devido ao tipo de movimento articulado que permitimos às estruturas analisadas. Ilustramos a seguir o teste realizado com os 50 primeiros átomos de carbono alfa consecutivos de `1AMU` e $d_{\text{fix}} = 6$ (última linha da Tabela 2.9). A Figura 2.9 a) mostra a estrutura original com 50 átomos e a Figura 2.9 b) mostra a comparação, via *Procrustes*, das configurações original (pontos vermelhos) e numérica (pontos verdes). Resolvemos o problema *Procrustes* utilizando as duas formulações discutidas anteriormente e obtivemos a mesma matriz de rotação. A matriz obtida e o valor final de função (2.18) do problema *Procrustes* são, respectivamente, dados por:

$$Q = \begin{pmatrix} -0.161113 & 0.246074 & 0.955767 \\ 0.942579 & -0.248700 & 0.222921 \\ 0.292555 & 0.936801 & -0.191876 \end{pmatrix}, \quad g(Q) = 46.3852.$$

Por fim, a Figura 2.10 compara as distâncias originais $\hat{d}_{ij}$ com as distâncias d_{ij} obtidas após os ajustes. Notemos no gráfico que apesar de vários valores d_{ij} não serem ajustados satisfatoriamente, os pares ordenados $(\hat{d}_{ij}, d_{ij})$ concentram-se em torno da reta $y = x$. De um modo geral, o comportamento de *GENCAN* nos testes com o problema reformulado (2.17) foi muito semelhante ao comportamento observado nas baterias de experimentos anteriores: constatamos casos em que a rotina de otimização obteve uma configuração de pontos igual à imagem refletida da configuração original (a menos de translação).



a) `1AMU` – 50 átomos C_α b) Comparação via *Procrustes*

Figura 2.9: Análises realizadas com a proteína `1AMU`.

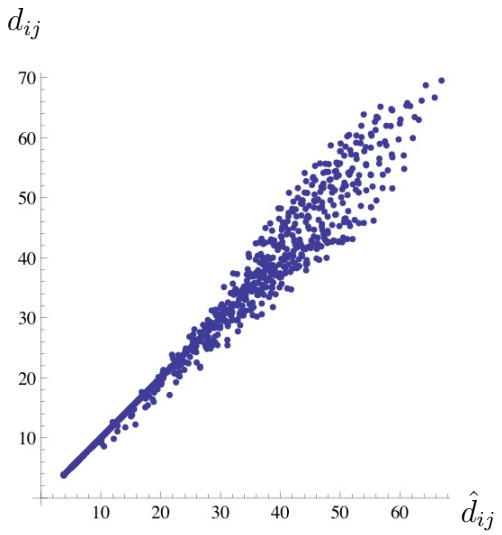


Figura 2.10: Tentativa de ajustar 1225 distâncias (50 átomos).

2.2.2 Comparando *GENCAN* e *MD-jeep*

Nesta seção contrastamos os desempenhos de duas rotinas utilizadas para resolver instâncias do problema da geometria de distâncias moleculares: *GENCAN* e *MD-jeep*. A rotina *GENCAN*, já usada nos experimentos anteriores, emprega uma estratégia de otimização contínua para resolver problemas do tipo

$$\text{minimizar } f(x) \quad \text{restrita a} \quad \ell \leq x \leq u.$$

GENCAN integra o pacote de otimização numérica *ALGENCAN* [5, 6] e pode ser obtida gratuitamente no endereço eletrônico `www.ime.usp.br/~egbirgin/tango`. A rotina *MD-jeep*, desenvolvida especificamente para resolver problemas de geometria de distâncias moleculares, adota um procedimento de otimização combinatória para reconstruir a estrutura 3D de uma proteína [61, 62, 64, 65]. *MD-jeep*, elaborada por Mucherino *et. al*, também é uma rotina gratuita e pode ser obtida em `www.antoniomucherino.it/en/mdjeep.php`.

Para realizar as comparações com os dois *softwares*, retiramos do *Protein Data Bank* oito arquivos de proteínas e tomamos apenas as coordenadas dos átomos N, C_α e C de cada estrutura para realizar os testes. Adaptamos *GENCAN* para resolver (2.17) partindo de diversos pontos iniciais gerados aleatoriamente. Também pedimos para *MD-jeep* imprimir em arquivos de texto todas as soluções encontradas com tolerância de 10^{-3}. Em cada experimento, fornecemos para ambas as rotinas somente as coordenadas dos pares de átomos com distância menor ou igual a um valor fixo d_{fix}.

Os desempenhos de *GENCAN* e *MD-jeep* na resolução dos problemas foram comparados em relação a três medidas:

a) tempo de CPU (em segundos),

b) erro envolvido na solução obtida:

$$E_{\text{sol}} = \frac{1}{n_d} \sum_{i,j} \frac{|d_{ij} - \hat{d}_{ij}|}{\hat{d}_{ij}}, \tag{2.20}$$

onde $\hat{d}_{ij}$ e d_{ij} são, respectivamente, a distância verdadeira e a distância ajustada entre os pares de átomos i e j e n_d é o total de distâncias consideradas.

c) escore *Structal* normalizado obtido por alinhar, via *LOVOALIGN*, as soluções numéricas com as estruturas originais das proteínas.

A Tabela 2.10 mostra os resultados obtidos nos experimentos e suas colunas possuem a seguinte legenda: *nat* indica o número de átomos N, C_α, C presentes em cada proteína, *ndist* é a quantidade total de distâncias entre os pares de átomos e *nd* é o número total de distâncias menores ou iguais ao valor fixo d_{fix}. Com relação à rotina *GENCAN* devemos salientar que cada problema foi rodado inúmeras vezes partindo de pontos iniciais diferentes, de modo que os resultados indicados na Tabela 2.10 são referentes à rodada em que a rotina obteve uma solução satisfatória. O critério de parada de *GENCAN* em todos os experimentos foi o da norma do gradiente menor que a tolerância de 10^{-4}. A rotina *MD-jeep* forneceu sempre duas soluções em cada teste. Na tabela estão indicados apenas os resultados correspondentes às soluções de *MD-jeep* que apresentaram os maiores valores de escore *Structal* normalizado. Como podemos perceber, ambas as rotinas conseguem obter configurações finais bastante satisfatórias para as proteínas testadas. Com efeito, notemos que os escores são muito próximos de 20, indicando que as estruturas comparadas são bastante semelhantes.

					GENCAN			*MD-jeep*		
Proteína	nat	ndist	nd	d_{fix}	$t(s)$	E_{sol}	escore	$t(s)$	E_{sol}	escore
1CRN	138	9453	1250	6.0	1.41	9.63E−08	19.99	0.001	9.63E−05	19.99
1PTQ	150	11175	1263	6.0	1.65	9.53E−04	19.89	0.001	9.78E−05	19.99
2ERL	120	7140	1136	6.0	0.44	4.17E−08	19.99	0.001	8.65E−05	19.99
1PPT	108	5778	1039	6.5	0.74	6.15E−04	19.95	0.001	9.51E−05	19.99
1PHT	249	30876	2631	6.5	5.60	3.66E−08	19.99	0.002	9.34E−05	19.99
1HOE	222	24531	2715	7.0	0.79	1.85E−10	19.99	0.002	8.12E−05	19.99
3RAT	372	69006	4567	7.0	3.37	1.53E−09	19.99	0.004	8.82E−05	19.99
1A7O	291	42195	4472	8.0	33.48	6.37E−10	19.99	0.003	7.59E−05	19.99

Tabela 2.10: Comparando *GENCAN* e *MD-jeep*.

A partir dos experimentos que realizamos nesta seção, pudemos verificar a existência de dois tipos de solução para o Problema da Geometria de Distâncias Moleculares: uma configuração de pontos que coincide exatamente com a estrutura original e uma configuração obtida da estrutura original por um movimento de reflexão. Com efeito, ao formularmos o problema como um problema de otimização não linear (2.17) construímos uma função

objetivo levando em consideração apenas informações escalares (conjunto de distâncias). Para excluirmos a possibilidade de obter imagens refletidas, deveríamos também exigir que a orientação da configuração original de pontos fosse preservada. Outras propriedades e exemplos de problemas envolvendo matrizes de distâncias euclidianas podem ser encontrados em [2, 11, 20, 27, 33, 34, 73].

No restante deste trabalho nos dedicaremos a estudar maneiras de representar proteínas comparadas. Não temos apenas o objetivo de construir mapas de proteínas para visualizar semelhanças entre elas. Queremos também aproveitar estes mapas para estabelecer critérios de classificação para novas proteínas. Começaremos nosso estudo investigando a construção de mapas no plano cartesiano. Nesta etapa representamos as proteínas por círculos. Posteriormente, investimos em modelos de mapas para o espaço tridimensional em que as proteínas são representadas por segmentos de retas e conjuntos de pontos. E, por fim, trabalhamos com espaços de dimensão maior que 4. Em todos os experimentos utilizamos a rotina *LOVOALIGN* para comparar as proteínas com relação a semelhanças estruturais e gerar os escores *Structal* correspondentes às comparações.

2.3 Proteínas representadas por círculos

Passaremos agora a estudar o problema de representar proteínas comparadas em um espaço euclidiano. Vamos admitir que um conjunto de n proteínas foram comparadas duas a duas e um conjunto de escores $s = \{s_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$ foi criado a partir das comparações. Vamos supor também que um conjunto de números reais não negativos Δ é criado a partir dos escores, isto é, $\Delta = \{\hat{d}_{ij} \in \mathbb{R}_+ \mid \hat{d}_{ij} = h(s_{ij})\}$, onde $h : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ faz com que $\hat{d}_{ij}$ seja inversamente proporcional a s_{ij}. Como não existe na literatura um padrão para determinar os escores s_{ij}, o conjunto Δ pode não ser realizável por pontos em nenhum espaço euclidiano. Isto significa que o Teorema 2.1 pode falhar na procura por uma configuração de pontos que realizem os números $\hat{d}_{ij}$. Esta dificuldade nos motivou na busca por outras formas geométricas de representar proteínas comparadas.

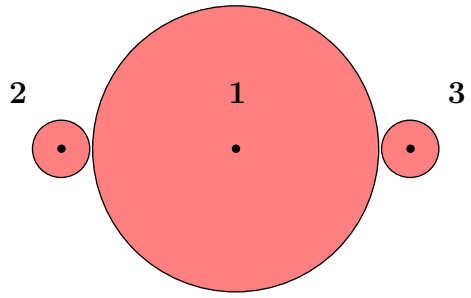


Figura 2.11: Círculos no plano realizam as distâncias do Exemplo 2.

Voltando ao **Exemplo 2** da Seção 2.1, conseguiremos ajustar as distâncias da matriz (2.15) se utilizarmos círculos para representar os objetos ao invés de pontos. A Figura 2.11

mostra três círculos no plano numerados de 1 a 3, com raios iguais a 5 cm, 1 cm, 1 cm, respectivamente. As distâncias entre os círculos 1 e 2, 1 e 3 são iguais a 0.1 cm e a distância entre os círculos 2 e 3 é de 10.2 cm. Inspirados neste pequeno exemplo, decidimos investigar a possibilidade de representar um conjunto de proteínas comparadas por círculos no plano. Os resultados desse estudo são apresentados a seguir.

Dado um conjunto de n proteínas comparadas pelo *software LOVOALIGN* [7, 8, 9, 70], seja Δ um conjunto de $N = n(n-1)/2$ números reais não negativos obtidos a partir dos escores *Structal* resultantes do processo de comparação. Nos modelos que apresentaremos nesta seção, cada proteína comparada é representada por um círculo no plano. Denotando, respectivamente, por C^i, r_i, o centro e o raio do i-ésimo círculo, queremos posicionar n círculos em $\mathbb{R}^2$ de tal forma que as distâncias dois a dois estejam tão próximas quanto possível dos elementos de Δ. Os experimentos apresentados nesta seção foram rodados em um computador Apple com as seguintes características: sistema operacional MAC OS X 10.5, processador Intel Core 2 Duo com 2.4GHz e 2GB de memória.

2.3.1 Modelo 1: ajustando todas as distâncias

Nosso primeiro modelo recai no seguinte problema de minimização:

$$\begin{aligned} \min \quad & f(C^i, r_i) \\ s.a \quad & r_i \geq 0,\ i = 1, \ldots, n, \end{aligned} \tag{2.21}$$

onde

$$f(C^i, r_i) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\|C^i - C^j\| - (r_i + r_j) - \hat{d}_{ij} \right)^2. \tag{2.22}$$

Notemos que as variáveis do problema de otimização são os centros e os raios dos círculos. Além disso, devemos observar que as derivadas da função (2.22) não estão definidas para configurações em que $C^i = C^j$, para todo $1 \leq i < j \leq n$. No entanto, podemos contornar essa dificuldade redefinindo a função objetivo do seguinte modo:

$$\widehat{f}(C^i, r_i) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\|C^i - C^j\|^2 - (r_i + r_j + \hat{d}_{ij})^2 \right)^2. \tag{2.23}$$

A função (2.23) cumpre o mesmo papel da função (2.22), ou seja,

$$\widehat{f}(C^i, r_i) = 0 \Leftrightarrow \|C^i - C^j\| - (r_i + r_j) = \hat{d}_{ij},$$

para todo $i \neq j$ e suas derivadas de primeira ordem estão bem definidas em todo o domínio.

Experimentos numéricos

Fizemos inicialmente experimentos com dados artificiais empregando as formulações de função (2.22) e (2.23) para resolver o problema (2.21). Os problemas foram resolvidos com

a rotina de otimização numérica *GENCAN* [13]. Cada teste foi rodado a partir de um único ponto inicial gerado aleatoriamente, isto é, não empregamos nenhum tipo de estratégia *multistart* para a obtenção destes primeiros resultados. Utilizamos como configurações artificiais as seguintes quantidades de círculos: 20, 30, 50, 100, 200, 300 e 400. Os centros destes círculos foram gerados em caixas $[a, b] \times [c, d]$ de diferentes tamanhos. Por essa razão, dividimos os experimentos em dois conjuntos: os de "caixas grandes" e os de "caixas pequenas". Os raios das configurações originais também foram gerados de forma aleatória dentro de um intervalo $[r_{\min}, r_{\max}]$. Em todos os experimentos cuidamos para que os círculos gerados não se interceptassem dois a dois. A Tabela 2.11 mostra as caixas e os intervalos utilizados para gerar os raios.

Caixas grandes			Caixas pequenas		
Círculos	Centros	Raios	Círculos	Centros	Raios
100	$[-1500, 1500] \times [-1500, 1500]$	$[20, 100]$	20	$[-100, 100] \times [-100, 100]$	$[1, 10]$
200	$[-1500, 1500] \times [-1500, 1500]$	$[20, 50]$	30	$[-100, 100] \times [-100, 100]$	$[1, 10]$
300	$[-1500, 1500] \times [-1500, 1500]$	$[10, 30]$	50	$[-100, 100] \times [-100, 100]$	$[1, 10]$
400	$[-1500, 1500] \times [-1500, 1500]$	$[10, 30]$	100	$[-500, 500] \times [-500, 500]$	$[1, 25]$

Tabela 2.11: Dados utilizados para gerar configurações artificiais de círculos.

Os resultados dos experimentos com caixas grandes estão resumidos na Tabela 2.12. Em cada teste utilizamos uma formulação diferente para a função objetivo (formulações (2.22) e (2.23)). Para comparar o desempenho de *GENCAN* na resolução dos problemas computamos o número total de iterações realizadas (*niter*), o número total de avaliações de função (*avalf*), o valor final da função objetivo ($f(x^*)$) e o critério de parada da rotina (*inform*).

		Função (2.22)				**Função (2.23)**			
Teste	Círculos	niter	avalf	$f(x^*)$	inform	niter	avalf	$f(x^*)$	inform
1	100	30	65	2.3220E−12	1	49	204	3.5883E−15	6
2	100	45	95	1.3192E−17	1	38	152	3.8561E−15	6
3	200	28	63	3.5435E−15	1	41	121	1.4111E−14	6
4	200	33	71	7.5878E−13	1	54	219	1.5823E−14	6
5	300	32	70	6.0262E−16	1	54	287	4.0676E−14	6
6	300	29	64	8.2604E−13	1	112	1731	3.4415E−14	6
7	400	35	77	5.9501E−12	1	39	167	6.7445E−14	6
8	400	29	74	1.2064E−12	1	47	200	6.3574E−14	6
9	400	59	143	1.3881E−14	1	766	20000	1.0947E+11	8
10	400	32	80	2.4027E−17	1	63	422	6.7477E−14	6

Tabela 2.12: Resultados de experimentos com caixas grandes.

Os valores de *inform* que aparecem na Tabela 2.12 têm o seguinte significado: 1 indica que *GENCAN* convergiu com a norma *sup* do gradiente projetado contínuo menor que a tolerância $\epsilon = 10^{-4}$; 6 indica que o algoritmo parou porque o tamanho de passo utilizado na busca linear era pequeno, e, por fim, 8 significa que o número máximo de avaliações de função objetivo foi atingido. Observando a Tabela 2.12 podemos notar que todos os problemas resolvidos com a função (2.22) apresentaram resultados satisfatórios, ou seja, em todos os testes o valor final da função objetivo é menor que 1.0E−10. Quando utilizamos a função (2.23) notamos que, com exceção do teste 9, o critério de parada dos demais testes é 6. Apesar disso, os valores finais da função objetivo nestes testes foram bastante razoáveis. Devemos notar também que *GENCAN* realiza menos avaliações de função ao resolver os problemas com a função (2.22).

Apresentamos a seguir resultados numéricos referentes aos testes com caixas pequenas. Utilizamos 20, 30, 50 e 100 círculos, realizando dez testes com cada quantidade. Em cada teste uma configuração sintética de círculos foi gerada e as duas formulações para a função objetivo foram empregadas na resolução dos problemas. Novamente não empregamos estratégia *multistart* na obtenção dos resultados, ou seja, um único ponto inicial foi fornecido para a rotina de otimização. A Tabela 2.13 mostra o valor final de função obtido por *GENCAN*. O número total de iterações e avaliações de função está indicado na Tabela 2.14. Nestes experimentos o critério de parada do algoritmo, comum a todos os testes, foi o da norma do gradiente projetado menor que 10^{-4} (inform = 1). Observando a Tabela 2.13 notamos que no teste 9 com 30 círculos utilizando a função (2.22) o algoritmo parou em um ponto estacionário pois, apesar do critério de parada obtido ser inform = 1, o valor final de função é da ordem de 1.0E+05. Analisando os resultados da Tabela 2.14 vemos que os números totais de iterações e avaliações de função são pequenos e não variam muito de uma formulação para outra. Apesar da função (2.22) não ser diferenciável em alguns pontos, não detectamos erros numéricos provenientes da não-diferenciabilidade. Em todos os experimentos que envolveram essa função, *GENCAN* parou em pontos cujas derivadas estavam bem definidas.

Investigando as soluções dos experimentos anteriores, percebemos que as configurações finais de círculos obtidas por *GENCAN* diferiam das configurações originais apenas por um movimento rígido. Para ilustrar esse comportamento, apresentamos abaixo duas figuras construídas com os resultados do teste 4 com caixa pequena e 100 círculos (veja Tabela 2.13). A Figura 2.12 a) mostra a configuração original com sete círculos destacados e a Figura 2.12 b) mostra o resultado obtido por *GENCAN* ao resolver o problema com a função objetivo (2.23). Notemos que a configuração final pode ser obtida por aplicar à configuração original um movimento de rotação no sentido horário seguido por um pequeno deslocamento.

Nestes experimentos, a tarefa de ajustar um conjunto de n círculos no plano só foi possível porque os elementos do conjunto Δ foram gerados artificialmente, ou seja, sabíamos de antemão as posições dos círculos e, para todo $1 \leq i < j \leq n$, definimos

$$\hat{d}_{ij} = \|C^i - C^j\| - (r_i + r_j). \tag{2.24}$$

	20 círculos		**30 círculos**	
Teste	**Função (2.22)**	**Função (2.23)**	**Função (2.22)**	**Função (2.23)**
1	2.9745E−11	3.3123E−21	4.6073E−11	7.9937E−14
2	1.3993E−09	6.5133E−19	7.1367E−17	8.0600E−21
3	6.4232E−15	2.8227E−17	6.3485E−15	1.2585E−20
4	4.0063E−10	3.3820E−21	6.6192E−10	1.1955E−15
5	1.2952E−10	4.1734E−21	8.1647E−18	6.0449E−21
6	3.4205E−14	9.4563E−15	2.8297E−11	8.4112E−20
7	3.9685E−16	1.0277E−15	2.3422E−10	9.9173E−18
8	1.6262E−09	1.0292E−16	5.2573E−18	3.2765E−17
9	5.1887E−11	6.6841E−15	1.1235E+05	2.2487E−15
10	1.4734E−13	3.3588E−18	4.0906E−16	4.5196E−17

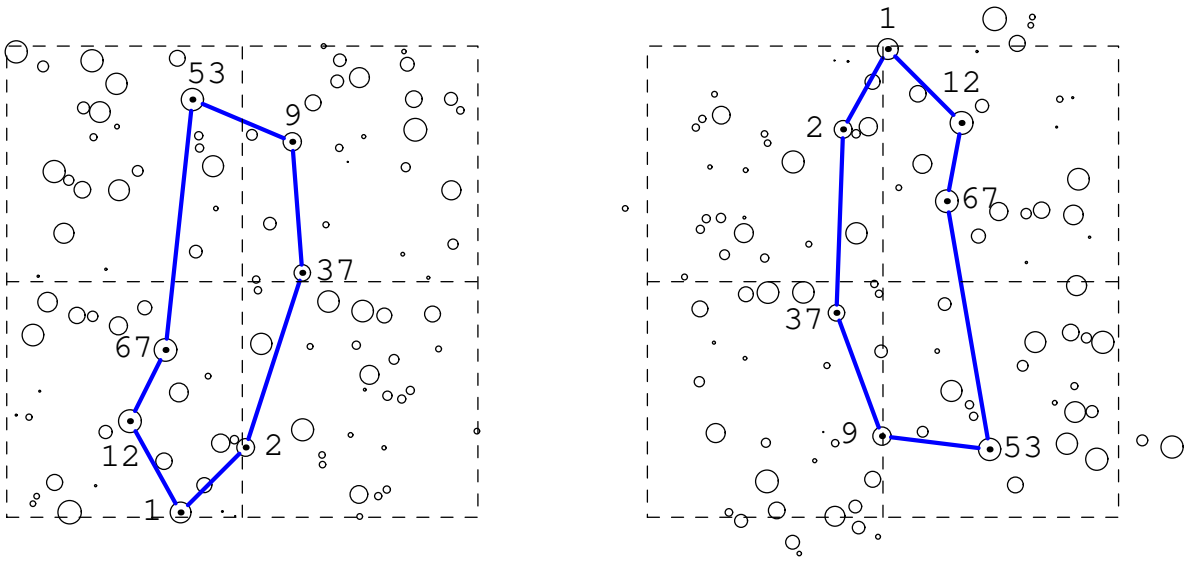
	50 círculos		**100 círculos**	
Teste	**Função (2.22)**	**Função (2.23)**	**Função (2.22)**	**Função (2.23)**
1	1.5803E−16	5.6325E−20	3.3720E−16	7.4979E−17
2	5.4362E−16	3.0544E−20	1.1688E−11	4.6050E−17
3	5.6256E−12	2.2526E−20	4.2477E−15	4.7051E−17
4	3.6120E−15	5.2614E−14	7.8042E−14	5.8135E−17
5	1.0347E−15	1.6519E−15	2.9856E−10	4.6509E−16
6	2.9217E−10	1.9418E−17	8.3047E−11	3.9373E−17
7	1.8584E−17	1.4643E−15	2.6646E−14	4.5354E−17
8	1.0070E−09	2.1525E−16	2.5575E−17	5.8499E−17
9	2.2713E−17	3.4684E−20	6.7667E−18	5.1460E−17
10	2.9966E−14	2.2604E−19	1.3164E−16	5.0185E−17

Tabela 2.13: Valor final de função objetivo em testes com caixas pequenas.

	20 círculos				**30 círculos**			
	Função (2.22)		**Função (2.23)**		**Função (2.22)**		**Função (2.23)**	
Teste	niter	avalf	niter	avalf	niter	avalf	niter	avalf
1	23	52	38	81	26	53	25	58
2	26	64	31	74	34	76	27	73
3	21	55	25	56	29	56	36	89
4	32	84	33	79	21	43	22	50
5	38	99	29	68	21	57	26	58
6	20	36	23	58	25	60	33	83
7	27	64	22	49	22	42	22	51
8	27	56	28	73	26	55	32	75
9	24	53	21	56	25	48	33	76
10	35	74	22	54	31	61	29	67

	50 círculos				**100 círculos**			
	Função (2.22)		**Função (2.23)**		**Função (2.22)**		**Função (2.23)**	
Teste	niter	avalf	niter	avalf	niter	avalf	niter	avalf
1	19	36	24	52	27	66	26	64
2	26	58	23	57	34	83	33	75
3	32	72	30	72	31	72	32	86
4	28	73	26	68	27	65	29	76
5	26	56	25	67	38	93	25	58
6	40	106	29	75	36	90	38	101
7	31	66	26	60	30	73	32	86
8	23	57	28	61	29	65	34	79
9	37	88	25	50	31	73	28	64
10	24	58	23	59	36	77	26	66

Tabela 2.14: Iterações e avaliações de função objetivo em testes com caixas pequenas.



a) Configuração original b) Solução obtida por *GENCAN*

Figura 2.12: Resultados obtidos no teste com 100 círculos sintéticos.

Como os círculos que criamos não se interceptavam dois a dois, tínhamos necessariamente $\hat{d}_{ij} > 0$. No entanto, como queremos utilizar círculos para representar proteínas comparadas, precisamos criar um conjunto Δ a partir dos escores resultantes das comparações e, neste caso, não temos garantia de que o problema (2.21) possa ser resolvido adequadamente. A seguir, propomos um modelo que tentará alocar círculos no plano exigindo apenas que um subconjunto de Δ seja realizável.

2.3.2 Modelo 2: ajustando um subconjunto de distâncias

Suporemos agora que o conjunto Δ é obtido a partir dos escores de n proteínas. Então, fixando um número positivo d_{fix}, vamos compará-lo com cada elemento de Δ da seguinte forma: se $\hat{d}_{ij} < d_{\text{fix}}$ exigimos que os círculos i e j tenham no plano uma distância que esteja tão próxima quanto possível do valor $\hat{d}_{ij}$. Caso contrário, deixamos que a distância entre os círculos i e j assuma qualquer valor maior ou igual a d_{fix}. Para que estas propriedades sejam cumpridas, construímos a função:

$$
\begin{aligned}
f(C^i, r_i) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{ij}^2, \\
\text{onde} \qquad f_{ij} &= \begin{cases} \|C^i - C^j\|^2 - (\hat{d}_{ij} + r_i + r_j)^2, \text{ se } \hat{d}_{ij} < d_{\text{fix}}, \\ \\ \max\{0, (d_{\text{fix}} + r_i + r_j)^2 - \|C^i - C^j\|^2\}, \text{ caso contrário.} \end{cases}
\end{aligned} \tag{2.25}
$$

Notemos que (2.25) é contínua, uma vez diferenciável e vale zero quando os requisitos descritos acima forem satisfeitos. Então, nosso novo problema de otimização consiste em minimizar a função (2.25) sujeita às restrições de não negatividade dos raios dos círculos, isto é, $r_i \geq 0,\ \forall\, i$. A escolha do valor do parâmetro d_{fix} é, em princípio, arbitrária. Gostaríamos que seu valor fosse fixado de tal maneira que apenas as discrepâncias entre proteínas parecidas fossem ajustadas. Desse modo, não faz sentido escolher um valor muito grande para d_{fix}, porque dessa forma estaríamos dando o mesmo grau de importância para discrepâncias entre proteínas que não possuem semelhança alguma. A escolha de um valor adequado para d_{fix} pode permitir um ajuste razoável dos círculos no plano.

Experimentos numéricos

Na tentativa de validar o modelo utilizamos um conjunto de 20 proteínas comparadas, que foram extraídas de `www.ime.unicamp.br/~martinez/lovoalign`. Desse total, criamos subconjuntos de 10 e 15 proteínas. Os escores resultantes da comparação entre as proteínas foram obtidos com o auxílio da rotina *LOVOALIGN*. As discrepâncias $\hat{d}_{ij}$ foram calculadas em função dos escores da seguinte forma:

$$\hat{d}_{ij} = \frac{10^3}{s_{ij}},\ 1 \leq i < j \leq n. \tag{2.26}$$

Os problemas foram resolvidos numericamente pela rotina *GENCAN* empregando uma estratégia *multistart*, isto é, um mesmo problema foi resolvido várias vezes com chutes iniciais diferentes, porém mantendo constante o valor de d_{fix}. Nos casos em que todas as rodadas de um mesmo problema fracassavam (valor final de função objetivo grande), diminuíamos o valor de d_{fix} e tentávamos resolver o problema novamente. Esse procedimento foi repetido até que a função objetivo ficasse menor que a tolerância $\epsilon = 10^{-2}$. As configurações iniciais (chutes iniciais) utilizadas nos testes foram geradas aleatoriamente: os centros dos círculos foram criados dentro da caixa $[-500, 500] \times [-500, 500]$ e os raios foram gerados no intervalo $[5, 10]$. Para evitar configurações finais com círculos de raios muito grandes impusemos um limitante superior nas variáveis correspondentes aos raios. A Tabela 2.15 mostra os resultados de três

nprot	d_{fix}	ndist	nd	$f(x^*)$
10	20	45	11	4.2312E−17
15	19	105	17	2.3112E−15
20	18	190	24	1.7749E−22

Tabela 2.15: Resultados obtidos em três testes com proteínas.

testes em que obtivemos sucesso. As quantidades indicadas em cada coluna têm o seguinte significado: *nprot* é quantidade de proteínas utilizadas, d_{fix} é o valor escolhido para o cálculo da função (2.25), *ndist* é o número total de discrepâncias entre pares de proteínas, *nd* é a

quantidade de discrepâncias $\hat{d}_{ij}$ menores que d_{fix} ajustadas efetivamente e $f(x^*)$ é o valor da função objetivo na solução final.

De acordo com a Tabela 2.15 vemos que em cada teste o número de discrepâncias realizadas ($\hat{d}_{ij} < d_{\text{fix}}$) é pequeno se comparado ao total ($ndist$) de cada problema. Em vários experimentos com 10, 15 e 20 proteínas tentamos, sem sucesso, aumentar o valor de d_{fix} para ajustar um número maior de valores $\hat{d}_{ij}$. Porém, à medida que aumentávamos o valor deste parâmetro, *GENCAN* não conseguia resolver o problema, isto é, não encontrava uma configuração de círculos correspondente a um valor pequeno de função objetivo. Vamos exemplificar as dificuldades encontradas nesse modelo tomando como exemplo os resultados numéricos obtidos com 10 proteínas. A Figura 2.13 mostra um histograma das distâncias calculadas pela fórmula (2.26). O eixo das abscissas indica algumas faixas de valores para $\hat{d}_{ij}$ e o eixo das ordenadas mostra a freqüência com que essas faixas ocorrem. Por exemplo, podemos observar que apenas dois valores $\hat{d}_{ij}$ são menores que 5 e que a maioria está entre 15 e 35. Baseados neste histograma tentamos resolver o problema atribuindo valores para d_{fix} iguais a 40, 35, 30, 25, 20, mas só obtivemos sucesso nos testes fazendo $d_{\text{fix}} = 20$.

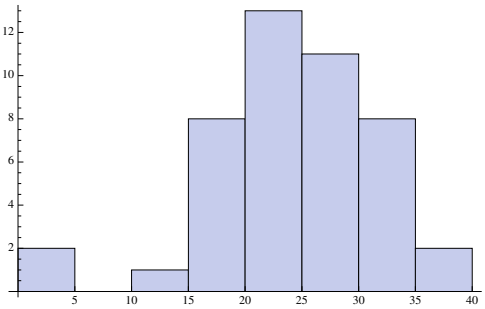


Figura 2.13: Histograma dos valores $\hat{d}_{ij}$ para um conjunto de 10 proteínas comparadas.

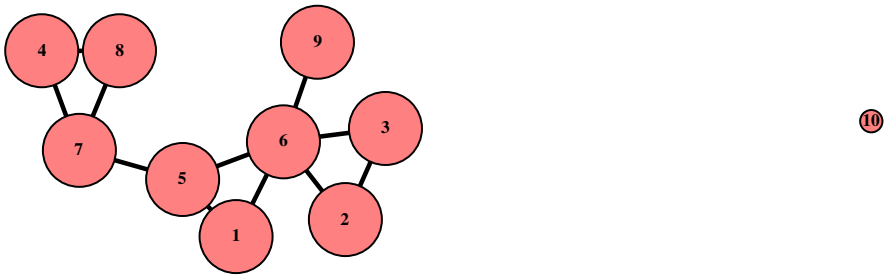


Figura 2.14: Representação bidimensional para um conjunto de 10 proteínas comparadas.

A Figura 2.14 mostra a configuração final de círculos obtida por *GENCAN* considerando $d_{\text{fix}} = 20$. Os círculos cujas distâncias são menores que d_{fix} estão conectados por linhas cheias. Podemos observar que os círculos 4 e 8, 1 e 5, representam as proteínas com maior grau de semelhança (escore grande), por estarem mais próximos no desenho. O círculo 10 está mais afastado dos demais círculos e não estabelece conexão com nenhum deles. De fato, a discrepância entre a proteína 10 e qualquer outra proteína representada na figura é maior que 20. Notemos também que os círculos numerados de 1 a 9 possuem aproximadamente o mesmo raio. O valor final de função para a configuração ilustrada na figura acima foi da ordem de 1.0E−17 (veja Tabela 2.15). Apesar disso, não consideramos este resultado satisfatório, pois gostaríamos de obter configurações com um número maior de distâncias realizadas. Em alguns experimentos empregamos também a seguinte fórmula para calcular as discrepâncias a partir dos escores:

$$\hat{d}_{ij} = s_{\max} - s_{ij},\ 1 \leq i < j \leq n, \tag{2.27}$$

onde $s_{\max} = \max\{s_{ij}\}$, mas os resultados numéricos foram muito semelhantes aos apresentados.

Ao trabalhar com este modelo também encontramos dificuldades que surgem do fato de não sabermos se os números $\hat{d}_{ij}$ calculados a partir dos escores s_{ij} são de fato distâncias que podem ser realizadas por círculos no plano. Lembremos que os testes bem sucedidos do Modelo 2.3.1 se deviam ao fato de empregarmos conjuntos de números gerados artificialmente. Tentaremos contornar essas dificuldades propondo a seguir um novo modelo que exigirá um pouco menos: as distâncias finais entre os círculos deverão respeitar uma ordem definida sobre os elementos de Δ. Pretendemos com isso tentar relaxar a condição de que as distâncias finais sejam ajustadas efetivamente aos elementos de Δ. A ideia de fazer com que uma ordem para Δ seja mantida na configuração final aparece na literatura em alguns trabalhos de J. B. Kruskal relacionados a *Ajuste Mutidimensional* [54, 55, 56].

2.3.3 Modelo 3: distâncias respeitando uma ordem

Suponha que os elementos do conjunto Δ estão em ordem crescente, isto é,

$$0 \leq \hat{d}_{i_1 j_1} \leq \hat{d}_{i_2 j_2} \leq \ldots \leq \hat{d}_{i_k j_k} \leq \hat{d}_{i_{k+1} j_{k+1}} \leq \ldots \leq \hat{d}_{i_N j_N}, \tag{2.28}$$

onde $N = n(n-1)/2$. Desejamos agora encontrar uma configuração de círculos no plano tal que as distâncias dois a dois respeitem apenas a ordem (2.28), ou seja, se $\hat{d}_{ij} \leq \hat{d}_{k\ell}$, então as distâncias finais entre os círculos i e j (d_{ij}) e entre os círculos k e ℓ ($d_{k\ell}$) deverão ser tais que

$$d_{ij} \leq d_{k\ell}. \tag{2.29}$$

Assim, de acordo com (2.28), definimos a função objetivo deste novo modelo como:

$$f(C^i, r_i) = \max\{0, d_{i_1 j_1} - d_{i_2 j_2}\}^2 + \ldots + \max\{0, d_{i_{N-1} j_{N-1}} - d_{i_N j_N}\}^2, \tag{2.30}$$

onde

$$d_{i_k j_k} = \|C^{i_k} - C^{j_k}\| - (r_{i_k} + r_{j_k}).$$

Notemos que os valores $\hat{d}_{ij}$ não entram no cálculo da função (2.30), apenas a ordem em que eles foram colocados é considerada. O problema de otimização que resolveremos aqui também consiste em minimizar (2.30) sujeita à não negatividade das variáveis que representam os raios dos círculos.

Experimentos numéricos

Para testar o novo modelo propusemos inicialmente um problema artificial que chamamos de *Mapa de Professores*. O problema consiste em construir um mapa bidimensional com um conjunto de professores pesquisadores de forma que aqueles que possuem um número grande de artigos publicados em comum devem ficar próximos na representação. Cada professor é então representado por um círculo e o problema a ser resolvido recai no que já discutimos anteriormente. Escolhemos seis professores do Departamento de Matemática Aplicada (DMA) do IMECC-Unicamp e coletamos o número de artigos (n_{ij}) que os professores i e j publicaram juntos. Estas informações foram retiradas dos currículos Lattes dos respectivos professores no dia 08/09/2008 e estão indicadas na Tabela 2.16. O conjunto de discrepâncias Δ foi determinado de acordo com a seguinte relação:

$$\hat{d}_{ij} = \begin{cases} \dfrac{10}{n_{ij}}, & \text{se } n_{ij} \neq 0, \\ \\ M \gg 0, & \text{caso contrário,} \end{cases} \tag{2.31}$$

onde M é um número grande e arbitrário (utilizamos $M = 15$ em nossos testes).

Antes de realizarmos qualquer teste numérico com o modelo e os dados coletados, decidimos tentar resolver o problema "a mão". Colocamos os valores $\hat{d}_{ij}$ em ordem crescente e, utilizando papel, régua, lápis e compasso, ajustamos seis círculos de forma que as distâncias

Prof.	**1**	**2**	**3**	**4**	**5**	**6**
1	0	1	2	12	0	15
2	1	0	0	7	1	5
3	2	0	0	1	0	4
4	12	7	1	0	5	13
5	0	1	0	5	0	0
6	15	5	4	13	0	0

Tabela 2.16: Artigos publicados em comum por seis professores do DMA.

entre eles respeitassem a ordem do conjunto Δ. Então, sabendo da existência de uma solução para este problema, partimos para os experimentos computacionais. Os problemas foram resolvidos pela rotina de otimização numérica *NMinimize* do *software Mathematica* (`www.wolfram.com`). Ao resolver o problema com os dados da Tabela 2.16 obtivemos uma configuração indesejável como mostra a Figura 2.15. Apesar do valor final da função objetivo ser pequeno nesta solução (1.06378E−24), podemos perceber que todos os círculos se interceptam. De fato, na função do modelo não existe nenhuma condição para impedir interseção entre círculos. Para evitar esta situação, introduzimos na função (2.30) penalizações das

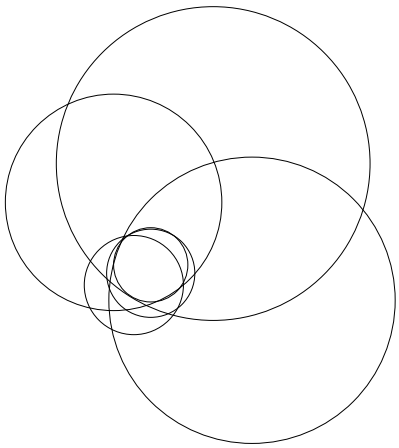


Figura 2.15: Resultado não satisfatório para o problema do mapa de professores.

restrições

$$||C^i - C^j|| - (r_i + r_j) \geq \epsilon, \ \forall \ i \neq j, \ \text{com } \epsilon > 0, \tag{2.32}$$

cujo papel é tentar impedir interseções entre círculos na configuração final. A Figura 2.16 mostra a solução obtida quando as penalizações são adicionadas à função objetivo. Utilizamos no teste $\epsilon = 0.5$ e o valor final de função foi 1.72655E−16. Notemos que os círculos não mais se interceptam e possuem diferentes tamanhos. A Figura 2.17 mostra um gráfico das distâncias originais $\hat{d}_{ij}$ (colocadas em ordem crescente) pelas distâncias d_{ij} obtidas numericamente. Como podemos observar, a curva que liga os pontos é não decrescente, indicando

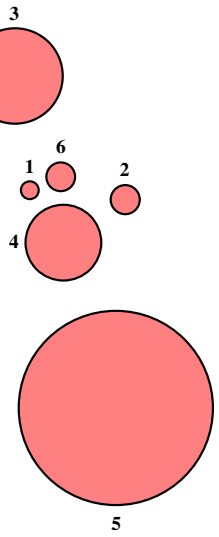


Figura 2.16: Resultado satisfatório para o problema do mapa de professores.

que as distâncias finais respeitam a ordem do conjunto Δ.

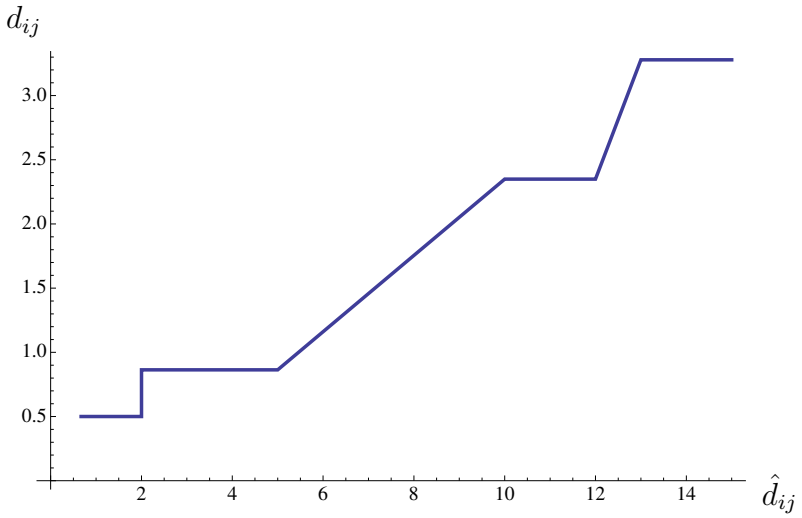


Figura 2.17: Gráfico $\hat{d}_{ij} \times d_{ij}$: "distâncias" entre professores.

Fizemos também alguns testes com conjuntos pequenos de proteínas que foram retiradas do *site* `www.ime.unicamp.br/~martinez/lovoalign` e comparadas via *LOVOALIGN*. Nestes testes utilizamos a função objetivo (2.30) com as restrições penalizadas (2.32). Os resultados numéricos obtidos não foram bons. A Figura 2.18 ilustra a situação para um conjunto de 10 proteínas comparadas. Neste experimento a rotina *NMinimize* atingiu o número

máximo de iterações permitidas e não conseguiu obter uma configuração satisfatória. O valor final de função objetivo foi 9.38788. Notemos pela figura que os círculos ficam muito próximos e alguns deles possuem raios muito pequenos. Tentamos rodar o mesmo problema várias vezes empregando uma estratégia *multistart*, mas não obtivemos sucesso em nenhuma rodada. A Figura 2.19 mostra um gráfico $\hat{d}_{ij} \times d_{ij}$ construído com os resultados obtidos neste experimento. A linha cheia que liga os pontos revela que a ordem original do conjunto Δ não foi preservada na configuração final.

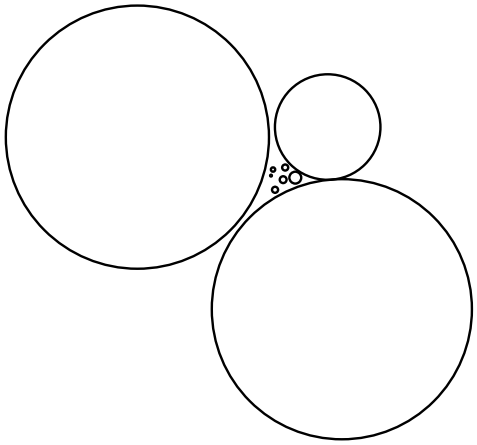


Figura 2.18: Resultado indesejável para um teste com 10 proteínas.

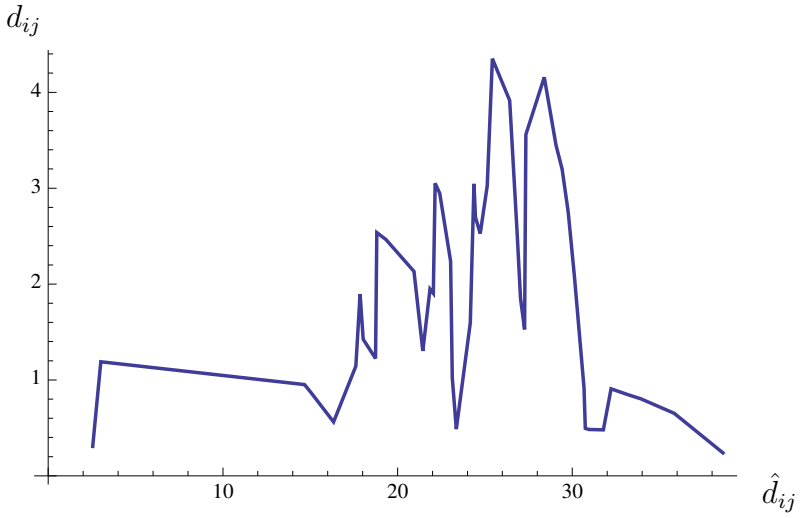


Figura 2.19: Gráfico $\hat{d}_{ij} \times d_{ij}$: ordem original não é preservada.

Nos três modelos que propusemos nesta seção tentamos representar um conjunto de proteínas comparadas por círculos no plano, ajustando distâncias entre pares desses círculos. Constatamos que a dificuldade comum a todos os modelos foi ajustar um conjunto de números $\hat{d}_{ij}$ criados a partir dos escores de proteínas comparadas. A estratégia de deixar livre os raios para que os círculos pudessem variar de tamanho também não ajudou na busca por configurações razoáveis. Além disso, como trabalhamos com uma representação bidimensional, tínhamos poucos graus de liberdade para ajustar as distâncias. A seguir, investigamos algumas maneiras de representar proteínas comparadas em $\mathbb{R}^3$. Começamos representando cada proteína por um segmento de reta de comprimento fixo, posteriormente permitimos que cada segmento tenha comprimento variável e, por fim, representamos cada proteína por uma poligonal formada por dois segmentos de comprimento fixo.

2.4 Proteínas representadas por segmentos de reta

Os experimentos com círculos nos mostraram que é difícil, em alguns casos impossível, o ajuste de todos os elementos do conjunto Δ. Para tentar contornar as dificuldades observadas nos testes bidimensionais propomos agora a seguinte estratégia: cada proteína será representada por um segmento de reta em $\mathbb{R}^3$ e nossa tarefa consistirá em ajustar somente os valores $\hat{d}_{ij}$ correspondentes a proteínas que possuem alguma semelhança entre si.

2.4.1 Modelo 1: segmentos de comprimento fixo

Representando cada proteína por um segmento de reta, nosso objetivo consiste em alocar os segmentos no espaço tridimensional de tal forma que somente as disparidades ($\hat{d}_{ij}$) correspondentes a proteínas semelhantes sejam efetivamente ajustadas. Para distribuir os segmentos no espaço, aplicaremos a cada um deles um movimento rígido. O i-ésimo segmento sofrerá a ação de uma sequência de três rotações simples em torno dos eixos coordenados, cujos ângulos de rotação são dados pela tripla $\theta^i = (\theta^i_1,\ \theta^i_2,\ \theta^i_3)^T$, seguida de um deslocamento $r^i = (r^i_1, r^i_2, r^i_3)^T$. Desse modo, denotando por $u^i, v^i \in \mathbb{R}^3$ as coordenadas iniciais dos pontos extremos do i-ésimo segmento de reta, após a aplicação do movimento rígido obteremos as seguintes coordenadas:

$$\begin{aligned} \hat{u}^i &= R(\theta^i)u^i + r^i, \\ \hat{v}^i &= R(\theta^i)v^i + r^i, \end{aligned} \tag{2.33}$$

onde

$$R(\theta^i) = \begin{pmatrix} -s^i_1 s^i_2 c^i_3 + c^i_1 c^i_2 & c^i_1 s^i_2 c^i_3 + s^i_1 c^i_2 & s^i_2 s^i_3 \\ -s^i_1 c^i_2 c^i_3 - c^i_1 s^i_2 & c^i_1 c^i_2 c^i_3 - s^i_1 s^i_2 & c^i_2 s^i_3 \\ s^i_1 s^i_3 & -c^i_1 s^i_3 & c^i_3 \end{pmatrix} \tag{2.34}$$

é a matriz que descreve a sequência de rotações realizadas em que $s^i_k \equiv \text{sen}(\theta^i_k)$ e $c^i_k \equiv \cos(\theta^i_k)$, para $k = 1, 2, 3$.

Neste modelo os segmentos de reta que representam as proteínas deverão movimentar-se em $\mathbb{R}^3$ de tal forma que as distâncias finais dois a dois (d_{ij}) respeitem uma das seguintes condições para $d_{\text{fix}} > 0$ fixo:

$$\begin{aligned} d_{ij} &\geq d_{\text{fix}} \quad \text{se} \quad \hat{d}_{ij} \geq d_{\text{fix}}, \\ d_{ij} &\approx \hat{d}_{ij} \quad \text{se} \quad \hat{d}_{ij} < d_{\text{fix}}. \end{aligned} \tag{2.35}$$

Ou seja, se a discrepância $\hat{d}_{ij}$ entre as proteínas i e j for menor que o valor fixo d_{fix}, a distância final d_{ij} entre os segmentos de reta correspondentes deverá ser aproximadamente igual a $\hat{d}_{ij}$. Caso contrário, d_{ij} poderá assumir qualquer valor maior ou igual a d_{fix}. Portanto, o problema de otimização que pretendemos resolver é dado por:

$$\min \sum_{i,j \,\in\, I_1} \max\{0, d_{\text{fix}}^2 - d_{ij}^2\}^2 + \sum_{i,j \,\in\, I_2} (d_{ij}^2 - \hat{d}_{ij}^2)^2 \ , \tag{2.36}$$

onde I_1, I_2 são conjuntos de índices tais que

$$\begin{aligned} I_1 &= \{i \neq j \in \mathbb{N} \mid \hat{d}_{ij} \geq d_{\text{fix}}\}, \\ I_2 &= \{i \neq j \in \mathbb{N} \mid \hat{d}_{ij} < d_{\text{fix}}\}. \end{aligned} \tag{2.37}$$

Notemos que, em (2.36), d_{ij} é uma função dos ângulos e deslocamentos associados aos segmentos de reta i e j, isto é, $d_{ij} \equiv d_{ij}(\theta^i, \theta^j, \omega^i, \omega^j)$. Assim, dado um conjunto de n proteínas, o problema a ser resolvido tem um total de $6n$ variáveis: três ângulos de rotação e três coordenadas de deslocamento para cada proteína. Além disso, como a função objetivo é não negativa, os minimizadores globais do problema (2.36) correspondem a configurações de segmentos cujas distâncias finais satisfazem uma das condições (2.35), pois em tais configurações os somatórios que definem a função valem aproximadamente zero.

Distância entre dois segmentos de reta

Um passo importante na resolução do problema (2.36) é o cálculo da distância entre dois segmentos de reta, fixadas as suas posições e orientações no espaço. O valor desta distância, como ilustra a Figura 2.20, pode ser determinado resolvendo-se o seguinte problema de minimização:

$$\begin{aligned} \min \quad & \|z^i - z^j\|^2 \\ \text{s. a} \quad & z^i \in \overline{\hat{u}^i \hat{v}^i}, \\ & z^j \in \overline{\hat{u}^j \hat{v}^j}. \end{aligned} \tag{2.38}$$

Como $z^i = \hat{u}^i + t_i\ (\hat{v}^i - \hat{u}^i)$ para algum $t_i \in [0,1]$ e $z^j = \hat{u}^j + t_j\ (\hat{v}^j - \hat{u}^j)$ para algum $t_j \in [0,1]$, o problema (2.38) é facilmente convertido no problema

$$\begin{aligned} \min \quad & g(t_i, t_j) \\ \text{s. a} \quad & 0 \leq t_i \leq 1, \\ & 0 \leq t_j \leq 1, \end{aligned} \tag{2.39}$$

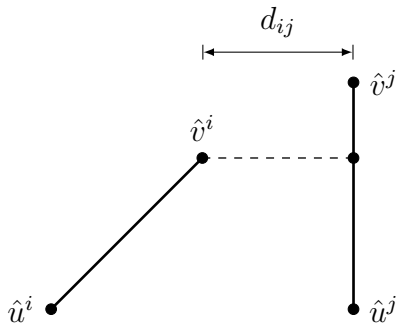


Figura 2.20: Distância mínima entre dois segmentos de reta.

onde $g(t_i, t_j)$ é uma função quadrática nas variáveis t_i e t_j cuja expressão é:

$$g(t_i, t_j) = t_i^2\|q_1\|^2 + t_j^2\|q_2\|^2 - 2t_it_j\langle q_1, q_2\rangle + 2t_i\langle q_1, q_3\rangle - 2t_j\langle q_2, q_3\rangle + \|q_3\|^2, \tag{2.40}$$

com

$$\begin{aligned} q_1 &= \hat{v}^i - \hat{u}^i, \\ q_2 &= \hat{v}^j - \hat{u}^j, \\ q_3 &= \hat{u}^i - \hat{u}^j. \end{aligned} \tag{2.41}$$

Embora o problema (2.39) possa ser resolvido por um método numérico, optamos por resolvê-lo analiticamente seguindo as ideias desenvolvidas no livro de D. H. Eberly [26]. Os pontos críticos da função $g(t_i, t_j)$ são dados pela solução do sistema linear $\nabla g(t_i, t_j) = 0$:

$$\begin{cases} t_i\|q_1\|^2 - t_j\langle q_1, q_2\rangle &= -\langle q_1, q_3\rangle, \\ -t_i\langle q_1, q_2\rangle + t_j\|q_2\|^2 &= \langle q_2, q_3\rangle. \end{cases} \tag{2.42}$$

Notemos que a matriz de coeficientes de (2.42), que denotaremos por M_0, possui determinante não negativo. De fato,

$$\begin{aligned} \det(M_0) &= \|q_1\|^2\|q_2\|^2 - \langle q_1, q_2\rangle^2 \\ &= \|q_1\|^2\|q_2\|^2(1 - \cos^2\gamma) \\ &= \|q_1\|^2\|q_2\|^2 \operatorname{sen}^2\gamma \\ &= \|q_1 \times q_2\|^2 \geq 0, \end{aligned} \tag{2.43}$$

onde γ é o ângulo entre os vetores q_1 e q_2. Portanto, para resolver o problema (2.39) devemos analisar dois casos:

- **Caso 1**: $\det(M_0) > 0$ - implica que os vetores q_1 e q_2 não são paralelos e o sistema linear (2.42) possui única solução dada por:

$$\begin{aligned} t_i^* &= \frac{\langle q_1, q_2\rangle\langle q_2, q_3\rangle - \langle q_1, q_3\rangle\|q_2\|^2}{\|q_1 \times q_2\|^2}, \\ t_j^* &= \frac{-\langle q_1, q_2\rangle\langle q_1, q_3\rangle + \langle q_2, q_3\rangle\|q_1\|^2}{\|q_1 \times q_2\|^2}. \end{aligned} \tag{2.44}$$

- **Caso 2**: $\det(M_0) = 0$ - implica que q_1 e q_2 são paralelos e, consequentemente, (2.42) possui infinitas soluções.

No **Caso 1** sabemos que $(t_i^*, t_j^*)^T$ é o único minimizador global da função $g(t_i, t_j)$, porém, este minimizador pode ocorrer fora do conjunto viável $[0,1] \times [0,1]$. Quando isto acontece devemos procurar por um minimizador na fronteira desta região. A Figura 2.21 mostra o plano cartesiano dividido em nove setores enumerados de 0 a 8. Se $(t_i^*, t_j^*)^T$ pertencer à Região 0, então os dois pontos que realizam a menor distância entre os segmentos i e j são pontos interiores desses segmentos. Suponha, por exemplo, que $(t_i^*, t_j^*)^T$ pertence à Região 1. Nesta situação fazemos $t_i^* = 1$ e a função $g(t_i, t_j)$ passa a ter apenas uma variável, isto é, $\hat{g}(t_j) = g(1, t_j)$. Assim, o minimizador de $\hat{g}(t_j)$ será um ponto no interior do intervalo $[0,1]$, onde $\hat{g}'(t_j) = 0$, ou será um ponto extremo do mesmo, ou seja, $t_j^* = 0$ ou $t_j^* = 1$. Uma análise semelhante pode ser feita para as Regiões 3, 5 e 7. Agora, suponha que o minimizador ocorre na Região 2. Então, devemos analisar o sinal das componentes do vetor gradiente de $g(t_i, t_j)$ no ponto $(1,1)^T$ para decidir se minimizamos na face $t_i = 1$ ou $t_j = 1$. Um procedimento parecido deve ser aplicado às Regiões 4, 6 e 8.

No **Caso 2**, aproveitando o fato de que q_1 e q_2 são paralelos, projetamos ortogonalmente os pontos extremos de um segmento na reta suporte gerada pelo outro segmento. Por exemplo, se cada ponto extremo do segmento j for projetado na reta suporte gerada pelo segmento i, podemos escrever:

$$\begin{aligned} \hat{u}_j &= \hat{u}_i + \sigma_0 q_1 + q_0, \\ \hat{u}_j + q_2 &= \hat{u}_i + \sigma_1 q_1 + q_0, \end{aligned} \tag{2.45}$$

onde σ_0, σ_1 são escalares reais a serem determinados, q_1, q_2 são dados por (2.41) e q_0 é um vetor ortogonal a q_1. Tomando o produto interno com q_1 de ambos os lados de cada igualdade acima, podemos resolver as equações resultantes para as incógnitas σ_0 e σ_1. Assim, obtidos os valores destes escalares, tudo o que devemos fazer é comparar os intervalos $[\min\{\sigma_0, \sigma_1\},\ \max\{\sigma_0, \sigma_1\}]$ e $[0,1]$. Se estes dois intervalos são disjuntos, a distância mínima ocorre para pontos extremos dos segmentos i e j, como ilustra a Figura 2.22 a). Caso contrário, existem infinitos pares de pontos que realizam a menor distância. Neste caso, escolhemos um ponto extremo de um segmento e um ponto interior de outro. Esta situação está ilustrada na Figura 2.22 b).

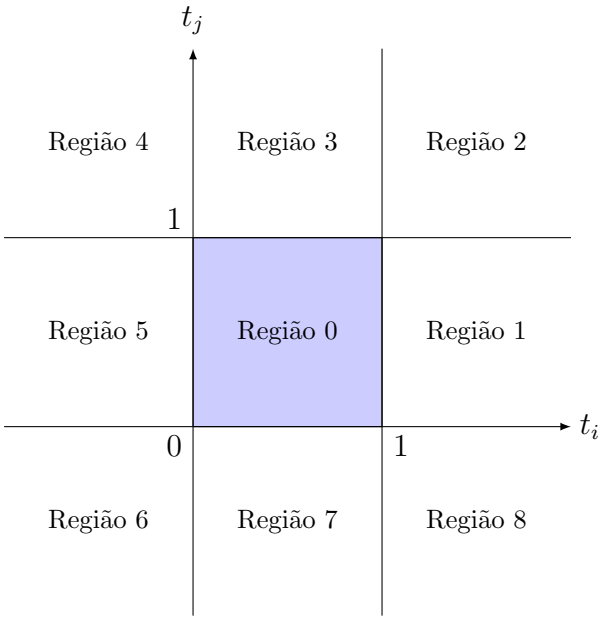
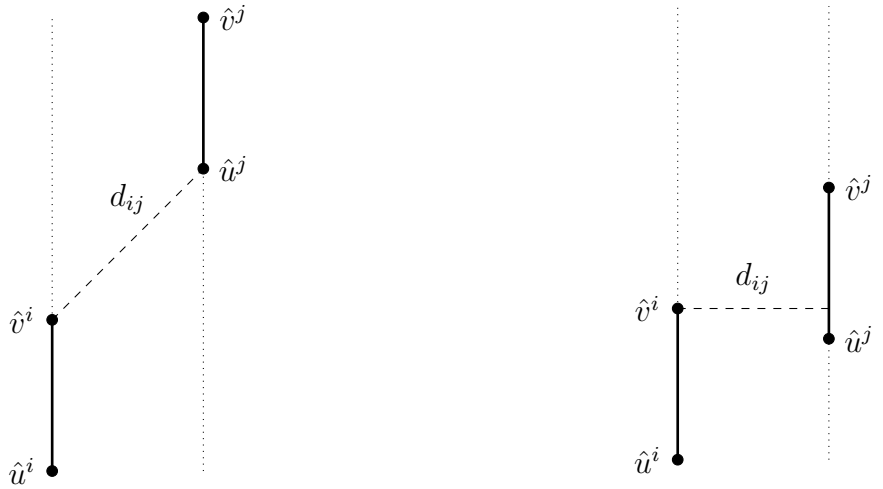


Figura 2.21: Regiões do plano cartesiano que devem ser analisadas.



a) $[\min\{\sigma_0, \sigma_1\},\ \max\{\sigma_0, \sigma_1\}] \cap [0, 1] = \emptyset$ b) $[\min\{\sigma_0, \sigma_1\},\ \max\{\sigma_0, \sigma_1\}] \cap [0, 1] \neq \emptyset$

Figura 2.22: Distância mínima no caso de segmentos paralelos.

De acordo com o que acabamos de expor, podemos construir um algoritmo que determina analiticamente a distância mínima entre dois segmentos de reta através da análise de casos. Dadas as coordenadas dos pontos extremos de dois segmentos, o código deve inicialmente calcular o determinante (2.43) para decidir se os segmentos são ou não paralelos. Se os segmentos não são paralelos, os cálculos realizados são aqueles descritos no **Caso 1**, ou seja, o algoritmo verifica se o minimizador global da função $g(t_i, t_j)$ está na região viável. Em caso afirmativo, a distância mínima é determinada. Caso contrário, a fronteira da região é testada. Por outro lado, se os segmentos são paralelos, o código procede da maneira descrita no **Caso 2**. Os experimentos que apresentamos a seguir foram rodados em um computador Apple com as seguintes especificações: sistema operacional MAC OS X 10.5, processador Intel Core 2 Duo com 2GHz e 2GB de memória.

Experimentos numéricos

Nestes testes as proteínas foram retiradas de `www.ime.unicamp.br/~martinez/lovoalign`, comparadas via *LOVOALIGN* e os elementos de Δ foram estipulados da seguinte forma:

$$\hat{d}_{ij} = 20 - s_{ij},\ 1 \leq i < j \leq n, \tag{2.46}$$

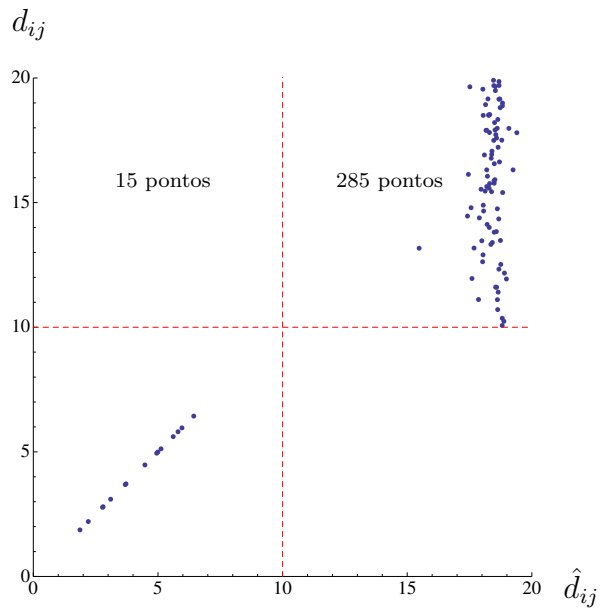
onde s_{ij} é agora o *escore Structal normalizado* resultante da comparação das proteínas i e j. Este novo escore, cujo valor está sempre no intervalo $(0, 20]$, é dado pela razão entre o escore *Structal* fornecido por *LOVOALIGN* e o número de átomos da menor proteína do par comparado, conforme explicamos na Seção 1.4 do Capítulo 1. Então, de acordo com a relação (2.46) quanto mais próximo de 20 for o valor de s_{ij}, maior será a semelhança entre as proteínas i e j e, consequentemente, menor será o valor de $\hat{d}_{ij}$. Os resultados computacionais foram obtidos com o auxílio da rotina *GENCAN*. Em cada teste medimos o número de iterações, avaliações de função e valor final de função obtidos pela rotina. As colunas das tabelas que mostraremos a seguir possuem o seguinte significado: *nprot* é o número de proteínas comparadas, d_{fix} é um parâmetro real positivo e fixo, n_{tot} é o número total de discrepâncias, n_{I_1} é a quantidade de distâncias finais maiores ou iguais a d_{fix}, n_{I_2} são as distâncias finais menores que d_{fix} (efetivamente ajustadas), *niter* é o número total de iterações, *avalf* é o número total de avaliações de função e $f(x^*)$ é o valor final de função objetivo. Em todas as rodadas, *GENCAN* parou com a norma do gradiente menor que 10^{-4}.

A Tabela 2.17 mostra alguns dos resultados obtidos nos experimentos em que os segmentos de reta possuem o mesmo comprimento. Em cada teste investigamos o valor de d_{fix} que deveria ser fixado para fazer com que todas as discrepâncias $\hat{d}_{ij}$ inferiores a este valor fossem efetivamente ajustadas. Cada problema foi resolvido para um valor diferente do parâmetro até que o valor final de função objetivo fosse menor que a tolerância $\epsilon = 10^{-2}$. Como podemos observar pelos resultados da tabela, à medida que o número de proteínas consideradas aumenta, precisamos diminuir o valor de d_{fix} para que seja possível encontrar uma configuração de segmentos que produza um valor pequeno de função. Notemos ainda que os número totais de iterações e avaliações de função atingidos por *GENCAN* aumentam nos problemas com quantidades maiores de proteínas. Podemos observar também que o

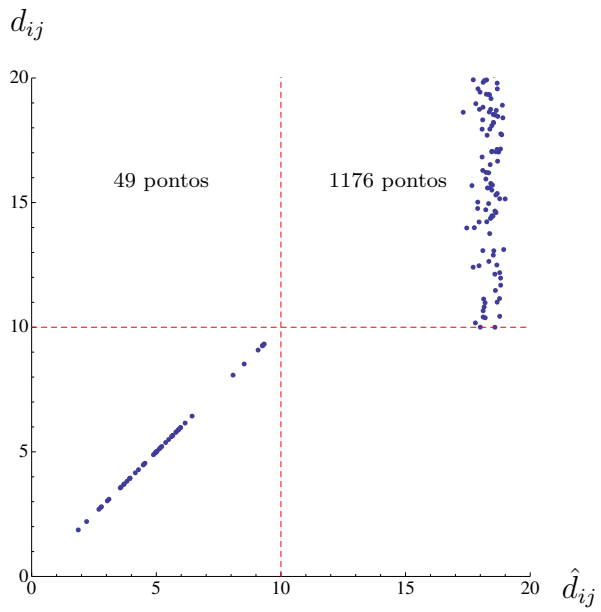
nprot	d_{fix}	n_{tot}	n_{I_1}	n_{I_2}	niter	avalf	$f(x^*)$
10	10	45	43	2	14	60	5.212504E−11
25	10	300	285	15	33	115	5.076322E−19
50	10	1225	1176	49	299	1612	3.160316E−20
75	10	2775	2698	77	276	8261	7.836608E+02
100	10	4950	4827	123	504	3720	5.645881E+02
75	5.7	2775	2722	53	179	1007	4.491551E−14
100	4.6	4950	4890	60	287	2022	2.338183E−14

Tabela 2.17: Resultados de testes com segmentos de comprimento fixo.

número de discrepâncias efetivamente ajustadas (n_{I_2}) é pequeno se comparado ao número total de discrepâncias do problema (n_{tot}). As Figuras 2.23 a) e 2.23 b) mostram gráficos $\hat{d}_{ij} \times d_{ij}$ construídos com as distâncias finais obtidas nos testes com 25 e 50 proteínas, respectivamente. As linhas tracejadas indicam o valor de distância fixo (d_{fix}) adotado em ambos os experimentos. As quantidades de pontos presentes em cada gráfico também estão indicadas nas figuras: a Figura 2.23 a) mostra que 15 valores $\hat{d}_{ij}$ foram efetivamente ajustados e 285 possuem valor maior ou igual a 10 (alguns pontos não aparecem no gráfico por terem a coordenada y maior que 20). Já na Figura 2.23 b) podemos notar que 49 valores $\hat{d}_{ij}$ são ajustados satisfatoriamente.



a) 25 proteínas - 300 distâncias



a) 50 proteínas - 1225 distâncias

Figura 2.23: Gráficos $\hat{d}_{ij} \times d_{ij}$: segmentos de comprimento fixo.

A Figura 2.24 mostra a configuração final de segmentos obtida no teste com 25 proteínas. Os segmentos mais próximos possuem distância menor que $d_{\text{fix}} = 10$ e representam proteínas semelhantes. Já os segmentos mais afastados representam proteínas que nada se parecem. Fizemos esta classificação com base nos valores dos escores *Structal* normalizados obtidos no processo de comparação das proteínas envolvidas nos experimentos.

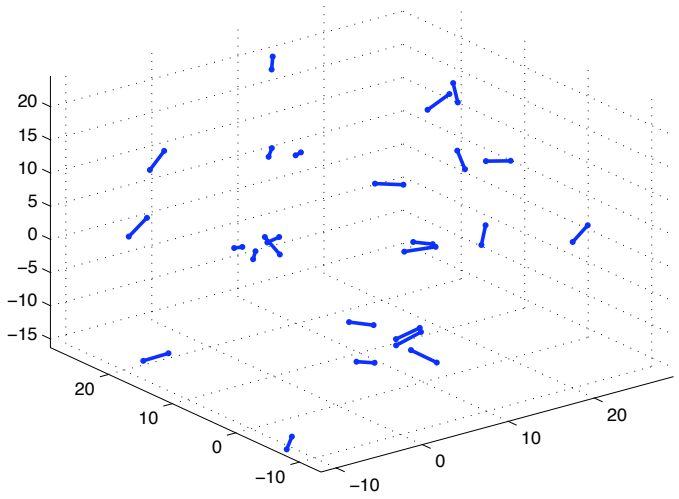


Figura 2.24: 25 segmentos de comprimento fixo.

2.4.2 Modelo 2: segmentos de comprimento variável

No Modelo 2.4.1 apresentado anteriormente, assumimos que os segmentos de reta possuem comprimento fixo. Constatamos pelos resultados numéricos que não foi possível manter constante o valor do parâmetro d_{fix} em todos os experimentos e vimos a necessidade de diminuí-lo em testes com quantidades maiores de proteínas. Como consequência deste fato, conseguimos realizar efetivamente uma quantidade de valores $\hat{d}_{ij}$ muito pequena em relação ao número total de elementos dos conjuntos Δ que apareceram nos problemas. A partir de agora, tentaremos não diminuir muito o valor de d_{fix} nos experimentos. Para isso, vamos aumentar os graus de liberdade do problema permitindo que os comprimentos dos segmentos sejam variáveis durante o processo de ajuste das discrepâncias $\hat{d}_{ij}$. Neste caso, após aplicarmos aos pontos extremos do i-ésimo segmento uma rotação $R(\theta^i)$ e um deslocamento r^i, permitimos que seu tamanho aumente por um fator $\alpha_i > 0$, isto é,

$$\|\alpha_i(\hat{v}_i - \hat{u}_i)\| = \alpha_i\|\hat{v}_i - \hat{u}_i\|, \tag{2.47}$$

onde $1 \leq \alpha_i \leq \alpha_{\max}$, como ilustra a Figura 2.25.

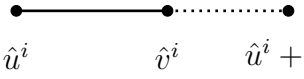


Figura 2.25: Segmento com comprimento variável.

Assim, o problema que passamos a resolver minimiza uma função semelhante à utilizada no Modelo 2.4.1, só que agora com 7 variáveis para cada proteína representada: três ângulos, três coordenadas de deslocamento e um escalar real.

Experimentos numéricos

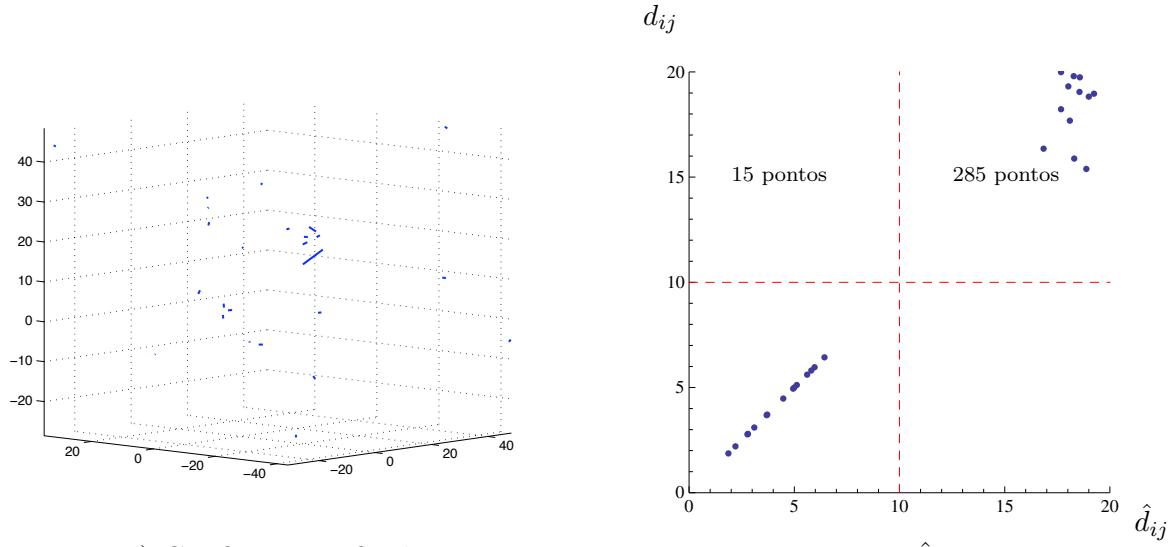
Empregamos nestes experimentos os mesmos conjuntos de proteínas comparadas que figuraram nos testes com o Modelo 2.4.1. Utilizamos também a mesma forma de gerar o conjunto Δ com os escores *Structal* normalizados. Os problemas foram resolvidos numericamente pela rotina *GENCAN* e as tabelas seguem as mesmas legendas definidas anteriormente. A Tabela 2.18 mostra os resultados obtidos nos experimentos. Inicialmente tomamos $d_{\text{fix}} = 10$ e obtivemos bons ajustes somente para os testes com 10, 25 e 50 proteínas. Assim como nos experimentos realizados com o Modelo 2.4.1, observamos a necessidade de diminuir o valor do parâmetro d_{fix} para quantidades de proteínas superiores a 50. Os problemas com 75 e 100 proteínas foram resolvidos para vários valores de d_{fix} até encontrarmos uma solução satisfatória. As duas últimas linhas da Tabela 2.18 mostram os melhores resultados obtidos nestes testes. Em todos os experimentos indicados na Tabela 2.18, o critério de parada de *GENCAN* foi o da norma do gradiente menor que 10^{-4}.

nprot	d_{fix}	n_{tot}	n_{I_1}	n_{I_2}	niter	avalf	$f(x^*)$
10	10	45	43	2	11	16	1.961699E−20
25	10	300	285	15	55	134	3.214463E−19
50	10	1225	1176	49	300	1202	1.395316E−16
75	10	2775	2698	77	348	1813	4.212664E+01
100	10	4950	4827	123	518	3347	3.169610E+02
75	5.8	2775	2720	55	162	738	2.939243E−13
100	4.6	4950	4890	60	184	951	1.930541E−14

Tabela 2.18: Resultados de testes com segmentos de comprimento variável.

A Figura 2.26 a) mostra a configuração final para o teste em que 25 proteínas são representadas por segmentos de comprimento variável. Notemos nesta figura que apenas um segmento é visivelmente maior que os demais. Neste teste, tomando $d_{\text{fix}} = 10$ conseguimos ajustar efetivamente apenas 15 discrepâncias $\hat{d}_{ij}$ de um total de 300. E este é o mesmo valor que obtivemos no teste em que o comprimento dos segmentos foi mantido fixo (veja a terceira linha da Tabela 2.17). A Figura 2.26 b) mostra o gráfico de discrepâncias originais

versus distâncias finais para este experimento. A grande maioria dos valores d_{ij} são maiores que 20 e, por isso, pontos correspondentes a estas distâncias não aparecem na figura. No gráfico também estão indicadas as quantidades de distâncias d_{ij} efetivamente ajustadas e de distâncias maiores que $d_{\text{fix}} = 10$.



a) Configuração final

b) Gráfico $\hat{d}_{ij} \times d_{ij}$

Figura 2.26: 25 segmentos de comprimento variável.

De acordo com estes resultados, podemos notar que a estratégia de tornar variáveis os comprimentos dos segmentos não contribui muito bem para o ajuste de um número maior de números $\hat{d}_{ij}$. Além disso, o comportamento de *GENCAN* na resolução dos problemas foi muito semelhante ao comportamento observado nos experimentos com o Modelo 2.4.1. A seguir, apresentaremos nossa última tentativa para ajustar disparidades entre proteínas representadas por segmentos de reta no espaço 3D. Aproveitaremos algumas características desenvolvidas nesta seção para construir um modelo cujos graus de liberdade sejam ainda maiores. A título de comparação, realizaremos uma bateria de testes com os mesmos conjuntos de proteínas empregados nos experimentos anteriores e analisaremos os resultados obtidos.

2.4.3 Modelo 3: poligonais formadas por dois segmentos

Esperávamos que a estratégia de tornar variáveis os comprimentos dos segmentos de reta facilitasse o ajuste das distâncias entre eles. Porém, nos deparamos com as mesmas dificuldades encontradas nos experimentos realizados com o Modelo 2.4.1. Desenvolveremos agora uma abordagem que aproveita as ideias apresentadas anteriormente, podendo facilitar a tarefa de ajustar distâncias entre proteínas representadas em $\mathbb{R}^3$. A estratégia consiste em representar cada proteína por uma poligonal formada por dois segmentos de reta e realizar somente as distâncias entre poligonais que correspondem a proteínas semelhantes.

Seguindo esse raciocínio, cada proteína é representada por uma poligonal constituída por dois segmentos de reta formando um ângulo de 90 graus entre si. Os segmentos possuem o mesmo comprimento, que é mantido fixo. Assim, o problema de otimização a ser resolvido é semelhante àquele que resolvemos em 2.4.1 cujas variáveis são ângulos de rotação e deslocamentos para cada proteína. Porém, as distâncias que devem ser ajustadas são distâncias entre poligonais. Para determiná-las, calculamos as distâncias entre pares de segmentos de poligonais distintas e tomamos o menor valor. Ou seja, se ℓ_{i1}, ℓ_{i2} e ℓ_{j1}, ℓ_{j2} são, respectivamente, os segmentos que formam as poligonais i e j, a distância mínima entre elas é dada por

$$d_{ij} = \min\{d_{\ell_{i1}\ell_{j1}},\ d_{\ell_{i1}\ell_{j2}},\ d_{\ell_{i2}\ell_{j1}},\ d_{\ell_{i2}\ell_{j2}}\}, \tag{2.48}$$

como ilustra a Figura 2.27. Notemos que o cálculo de d_{ij} envolve quatro cálculos de distância entre segmentos de reta e estas distâncias são calculadas através dos procedimentos descritos em 2.4.1.

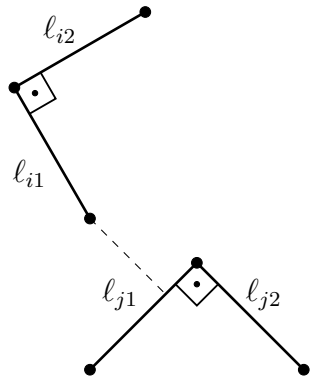


Figura 2.27: Distância mínima entre duas poligonais.

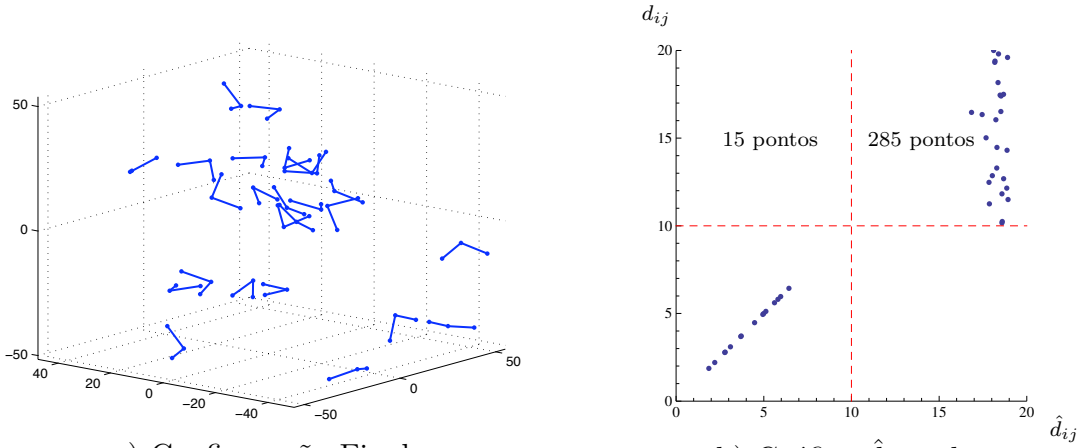
Experimentos numéricos

Empregamos novamente nestes experimentos os mesmos conjuntos de proteínas comparadas utilizados nos Modelos 2.4.1 e 2.4.2. O conjunto Δ também foi gerado da mesma forma descrita anteriormente, isto é, a partir dos escores *Structal* normalizados resultantes da comparação entre pares de proteínas. A Tabela 2.19 mostra o desempenho de *GENCAN* na resolução de problemas-teste envolvendo cinco quantidades de proteínas. A sigla de cada coluna desta tabela é a mesma adotada nas tabelas dos testes anteriores. Considerando inicialmente $d_{\text{fix}} = 10$, conseguimos ajustar efetivamente um conjunto de n_{I_2} distâncias nos testes com 10, 25 e 50 proteínas. No entanto, foi preciso diminuir o valor de d_{fix} nos demais experimentos para conseguirmos obter um valor final pequeno de função objetivo. Observemos nos testes com 75 e 100 proteínas que *GENCAN* atinge o número máximo de avaliações de função permitidas (avalf = 20000) para $d_{\text{fix}} = 10$. Nestes dois testes o número total de avaliações de função diminui quando d_{fix} também é diminuido.

nprot	d_{fix}	n_{tot}	n_{I_1}	n_{I_2}	niter	avalf	$f(x^*)$
10	10	45	43	2	22	29	1.291167E−12
25	10	300	285	15	41	95	3.605217E−18
50	10	1225	1176	49	192	1118	1.395717E−15
75	10	2775	2698	77	1594	20000	1.741954E+01
100	10	4950	4827	123	1603	20000	5.343404E+02
75	5.7	2775	2722	53	195	1066	8.945253E−14
100	4.7	4950	4888	62	195	1156	2.232349E−16

Tabela 2.19: Resultados de testes com poligonais de dois segmentos.

A Figura 2.28 a) mostra a configuração final obtida no teste em que 25 proteínas são representadas por poligonais de dois segmentos com $d_{\text{fix}} = 10$. Nesta figura, as poligonais mais próximas representam proteínas semelhantes e as distâncias entre elas são menores que 10. Podemos constatar isso observando o gráfico da Figura 2.28 b) construído com os dados obtidos neste experimento. No eixo das abscissas estão as disparidades originais e, no eixo das ordenadas, estão as distâncias obtidas numericamente. Notemos neste gráfico que apenas 15 valores $\hat{d}_{ij}$ são ajustados efetivamente. Obtivemos a mesma quantidade de ajustes nos modelos anteriores empregando esta mesma quantidade de proteínas. A Tabela 2.20 traz uma comparação do número de graus de liberdade (variáveis) dos modelos propostos neste capítulo para a representação de proteínas comparadas.



a) Configuração Final

b) Gráfico $\hat{d}_{ij} \times d_{ij}$

Figura 2.28: 25 poligonais de comprimento fixo.

Neste capítulo vimos que o modelo clássico dos pontos para a representação de proteínas só se torna viável mediante as condições do Teorema 2.1. Por essa razão, propusemos modelos diferentes dos existentes na literatura para representar conjuntos de proteínas comparadas. Começamos nossa investigação pelos modelos de círculos, mas os resultados numéricos obti-

modelos	**dimensão**	**graus de liberdade**
círculos	2	3
segto. comp. fixo	3	6
segto. comp. var.	3	7
poligonais	3	6

Tabela 2.20: Modelos para representação de proteínas do Capítulo 2.

dos não foram muito satisfatórios. Posteriormente, passamos a representar as proteínas por segmentos de reta em $\mathbb{R}^3$. Neste caso, obtivemos resultados satisfatórios após a introdução de um parâmetro d_{fix} que controlava o número de distâncias efetivamente ajustadas. Pudemos observar que o comportamento dos testes foi praticamente o mesmo, independendo do número de graus de liberdade de cada formulação. Constatamos também que o parâmetro d_{fix} não pode ficar fixo em todos os testes e precisou ser aumentado nos experimentos com mais de 50 proteínas. A grande dificuldade que notamos foi o fato de só conseguirmos ajustar efetivamente uma pequena parcela dos valores $\hat{d}_{ij}$ criados a partir dos escores. De modo geral, os resultados obtidos nos experimentos com os modelos para construção de mapas não foram muito surpreendentes. Por essa razão, não nos preocupamos em registrar os tempos de execução de cada experimento. No entanto, devemos ressaltar que a rotina *GENCAN* gastou mais tempo nos testes em que o ajuste de todos os valores $\hat{d}_{ij}$ era exigido.

No próximo capítulo abandonaremos a estratégia de ajustar distâncias entre objetos geométricos e apresentaremos modelos que tentarão ajustar diretamente os escores *Structal* obtidos na comparação das proteínas. As principais ideias desenvolvidas no Capítulo seguinte são: representar proteínas por vetores em algum espaço euclidiano $\mathbb{R}^m$ e construir problemas de otimização onde os escores sejam ajustados aos produtos escalares entre os vetores. Começamos fazendo testes em dimensão pequena e, em seguida, aumentamos a dimensão na tentativa de ajustar um número maior de escores. Realizamos experimentos computacionais com os modelos e discutimos os resultados obtidos.

Capítulo 3

Mapas de proteínas em $\mathbb{R}^m$

3.1 Ajuste de escores entre vetores

Neste capítulo trataremos do problema de ajustar escores entre um conjunto de proteínas comparadas. Para isso, associaremos as proteínas a vetores de algum espaço euclidiano $\mathbb{R}^m$ e procederemos da seguinte forma: dadas duas proteínas i, j e um escore resultante da comparação entre elas s_{ij}, queremos encontrar dois vetores $p^i, p^j \in \mathbb{R}^m$ tais que seu produto escalar seja aproximadamente igual a s_{ij}, ou seja,

$$\langle p^i, p^j \rangle \approx s_{ij}. \tag{3.1}$$

Assim, dadas n proteínas comparadas e os escores $s = \{s_{ij} \in \mathbb{R}_+ \mid 1 \leq i < j \leq n\}$, nossa tarefa consiste em encontrar n vetores em algum espaço $\mathbb{R}^m$ cujos produtos escalares dois a dois satisfaçam a relação (3.1).

Definido dessa maneira, o problema descrito acima possui a seguinte propriedade: se a dimensão do espaço utilizado para representar as proteínas é igual ao número de proteínas consideradas, é sempre possível obter neste espaço uma configuração de vetores tais que

$$\langle p^i, p^j \rangle = s_{ij}, \text{ para todo } 1 \leq i < j \leq n. \tag{3.2}$$

Provaremos esta afirmação na proposição a seguir.

Proposição 3.1 *Sejam n o número de proteínas que queremos representar e s o conjunto de escores obtidos por comparar as proteínas duas a duas. Então, é possível determinar n vetores em um espaço euclidiano de dimensão n tais que os produtos escalares entre pares desses vetores satisfazem (3.2).*

Prova: Seja $S_0 = [s_{ij}]$ uma matriz simétrica de ordem n tal que s_{ii} é o escore obtido por comparar a proteína i com uma cópia idêntica a ela, para todo $i = 1, 2, \ldots, n$, e s_{ij} é o escore correspondente à comparação das proteínas i e j para todo $i \neq j$. Como S_0 é simétrica, seus autovalores são reais. Definimos $S_1 = S_0 + \lambda I$, com $\lambda > |\lambda_{\min}|$, onde $\lambda_{\min}$ é o menor autovalor

de S_0 e I é a matriz identidade de ordem n. Por construção, os autovalores de S_1 são não negativos e os autovetores associados são os mesmos de S_0. Assim, podemos escrever:

$$S_1 = U\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}U^T,$$

onde U e Λ são matrizes de ordem n formadas, respectivamente, pelos autovetores e autovalores de S_1. Esta decomposição nos permite definir a matriz $X = U\Lambda^{\frac{1}{2}}$ de ordem n cujas linhas fornecem as coordenadas de n vetores em $\mathbb{R}^n$. Portanto, $S_1 = XX^T$ é uma matriz tal que o i-ésimo elemento da diagonal principal corresponde ao quadrado da norma do vetor que representa a proteína i e os elementos que estão fora da diagonal são iguais aos escores das proteínas comparadas. ■

A seguir, apresentamos algumas formulações para o problema do ajuste de produtos escalares e escores. Nestas formulações utilizamos os escores *Structal* normalizados que foram definidos no Capítulo 1. Os experimentos numéricos deste capítulo foram realizados em um computador Apple com as seguintes características: sistema operacional MAC OS X versão 10.5, processador Intel Core 2 Duo com 2.4GHz e 2GB de memória.

3.1.1 Modelo 1: ajustando todos os escores

Propomos resolver inicialmente um problema de otimização sem restrições nas variáveis cujo objetivo é minimizar a soma dos quadrados das diferenças entre produtos escalares e escores, ou seja,

$$\min \sum_{i<j} (\langle p^i, p^j \rangle - s_{ij})^2. \tag{3.3}$$

Assim, se temos um conjunto de n proteínas comparadas, as variáveis do problema (3.3) são as coordenadas de n vetores em algum espaço $\mathbb{R}^m$, totalizando nm variáveis. A função que será minimizada é não negativa e continuamente diferenciável. Deste modo, ao resolver (3.3), buscamos por configurações de vetores que produzam um valor de função tão próximo de zero quanto possível para que os ajustes entre produtos escalares e escores sejam bons.

Experimentos numéricos

Fizemos experimentos numéricos para investigar a possibilidade de ajustar os produtos escalares em espaços de dimensão pequena. Para validar o modelo (3.3), fixamos inicialmente a quantidade de proteínas em 10 e variamos a dimensão dos espaços de vetores de 2 a 10. Os dados das proteínas usadas nos testes foram retirados de `www.ime.unicamp.br/~martinez/lovoalign` e as proteínas foram comparadas pela rotina *LOVOALIGN*. Utilizamos dois pacotes de otimização numérica na resolução dos problemas: *GENCAN* e *MCS*. *MCS*, diferentemente de *GENCAN*, utiliza uma estratégia de otimização global para minimizar funções. Este pacote foi desenvolvido para *MATLAB* por A. Neumaier e pode ser obtido gratuitamente em `www.mat.univie.ac.at/~neum`. Os detalhes sobre a teoria envolvida em *MCS* podem ser encontrados em [47]. Para obter os resultados da Tabela 3.1 não empregamos nenhum tipo de estratégia *multistart*. Esta tabela apenas mostra, a título de comparação,

os valores finais de função $f(x^*)$ obtidos para cada valor fixo de dimensão (*dim*). Podemos observar pela tabela que as soluções obtidas pelos *softwares* foram praticamente iguais nos problemas com dimensão 2, 3 e muito próximas nos problemas de dimensão 4, 5. Notemos também que à medida que a dimensão do espaço de vetores aumenta, o valor final de função correspondente é cada vez menor.

	GENCAN	MCS
dim	$f(x^*)$	$f(x^*)$
2	21.2996	21.2996
3	1.1383	1.1383
4	0.3530	0.3571
5	0.1277	0.1460
6	3.3147E−04	2.1000E−03
7	8.7780E−04	3.2606E−04
8	2.4328E−13	5.3840E−29
9	1.4040E−19	7.1491E−30
10	1.5345E−19	1.1887E−21

Tabela 3.1: Comparando *GENCAN* e *MCS* em testes com 10 proteínas.

Nos testes que apresentamos a seguir aumentamos a quantidade de proteínas e utilizamos somente *GENCAN* para resolver os problemas. Esta escolha é devida ao fato de que *MCS* não resolve problemas com número muito grande de variáveis. Empregamos uma estratégia *multistart* na resolução dos problemas. Alguns dos resultados obtidos neste conjunto de testes estão indicados na Tabela 3.2. Nesta tabela, *nprot* é a quantidade de proteínas consideradas, *dim* é a dimensão do espaço, *niter* e *avalf* são, respectivamente, o número total de iterações e avaliações de função e $f(x^*)$ é o valor final de função obtido por *GENCAN*. De acordo com os resultados, a função objetivo diminui bastante de valor quando buscamos soluções em espaços de dimensão grande. Também podemos constatar nestes experimentos a validade da afirmação feita na Proposição 3.1, pois, quando a dimensão do espaço é igual ao número de proteínas, é sempre possível obter uma solução para o problema (3.3). Contudo, *GENCAN* encontrou dificuldades em alguns casos. Repare, por exemplo, no total de iterações e avaliações de função realizadas no problema com 75 proteínas em dimensão 50. Neste caso *GENCAN* parou porque atingiu o número máximo de avaliações de função. Já nos problemas em que a dimensão foi igual ao número de proteínas, a rotina realizou um número pequeno de iterações e avaliações de função.

A Figura 3.1 mostra gráficos construídos a partir dos resultados obtidos nos testes com 50 proteínas. No eixo x de cada gráfico estão os escores *Structal* normalizados do conjunto de proteínas colocados em ordem crescente (s_{ij}). No eixo y estão os respectivos produtos escalares calculados a partir das coordenadas finais dos vetores obtidos no processo de otimização ($\langle p^i, p^j \rangle$). Portanto, os pares ordenados são $(s_{ij}, \langle p^i, p^j \rangle)$. Quanto mais próximos os pontos estão da reta $y = x$ (também indicada nos gráficos), melhor é o ajuste. Notemos

que em $\mathbb{R}^3$ o ajuste é ruim, porém, à medida que a dimensão do espaço aumenta, os pontos tendem a ficar mais próximos da reta $y = x$.

Podemos perceber nestes experimentos que quando tentamos resolver o problema de otimização (3.3) em espaços de dimensão pequena temos poucos graus de liberdade para ajustar os vetores. Consequentemente, as soluções finais obtidas não são boas. A seguir, modificaremos um pouco nossa primeira formulação para tentar ainda obter bons ajustes em $\mathbb{R}^2$ ou $\mathbb{R}^3$. Para isso, faremos com que o problema reformulado tenha um número maior de graus de liberdade.

nprot = 25				nprot = 50			
dim	niter	avalf	$f(x^*)$	dim	niter	avalf	$f(x^*)$
3	11	13	4.141313E+02	3	12	20	1.668080E+04
10	2637	11533	4.354575E+00	10	2065	10474	1.145376E+02
20	2889	2000	3.475844E−03	20	3905	19967	2.622674E+01
25	25	77	2.742733E−21	50	15	27	4.886821E−22

nprot = 75				nprot = 100			
dim	niter	avalf	$f(x^*)$	dim	niter	avalf	$f(x^*)$
3	12	17	4.511840E+03	3	12	14	7.348332E+03
20	2621	20000	2.295061E+02	20	1353	9501	5.371149E+02
50	2185	20000	1.858455E+01	50	2256	19930	1.546689E+02
75	19	54	3.263620E−21	100	25	76	2.001732E−19

Tabela 3.2: Resultados de *GENCAN* para quatro quantidades de proteínas.

3.1.2 Modelo 2: ajustando escores e parâmetros

Nossa proposta agora consiste em modificar a formulação anterior para aumentar os graus de liberdade do problema sem que a dimensão do espaço de vetores seja muito grande. Para isso, tentaremos minimizar o quadrado da diferença entre um escore e uma função do correspondente produto escalar. Dessa forma, introduzindo a função real

$$\phi(t) = \sum_{\ell=1}^{n_p} c_\ell t^{\alpha_\ell}, \text{ para } t \geq 0, \tag{3.4}$$

com $c_\ell \geq 0$, $\alpha_\ell \geq 1$, $\ell = 1, \ldots, n_p$, propomos resolver o seguinte problema de otimização:

$$\begin{aligned} &\min \quad \sum_{i<j} \left(\ \phi(\ \langle p^i, p^j \rangle \ ) - s_{ij} \ \right)^2, \\ &\text{s. a} \quad p^i_k \geq 0, \ \ 1 \leq k \leq m \ \text{ e } \ 1 \leq i < j \leq n, \end{aligned} \tag{3.5}$$

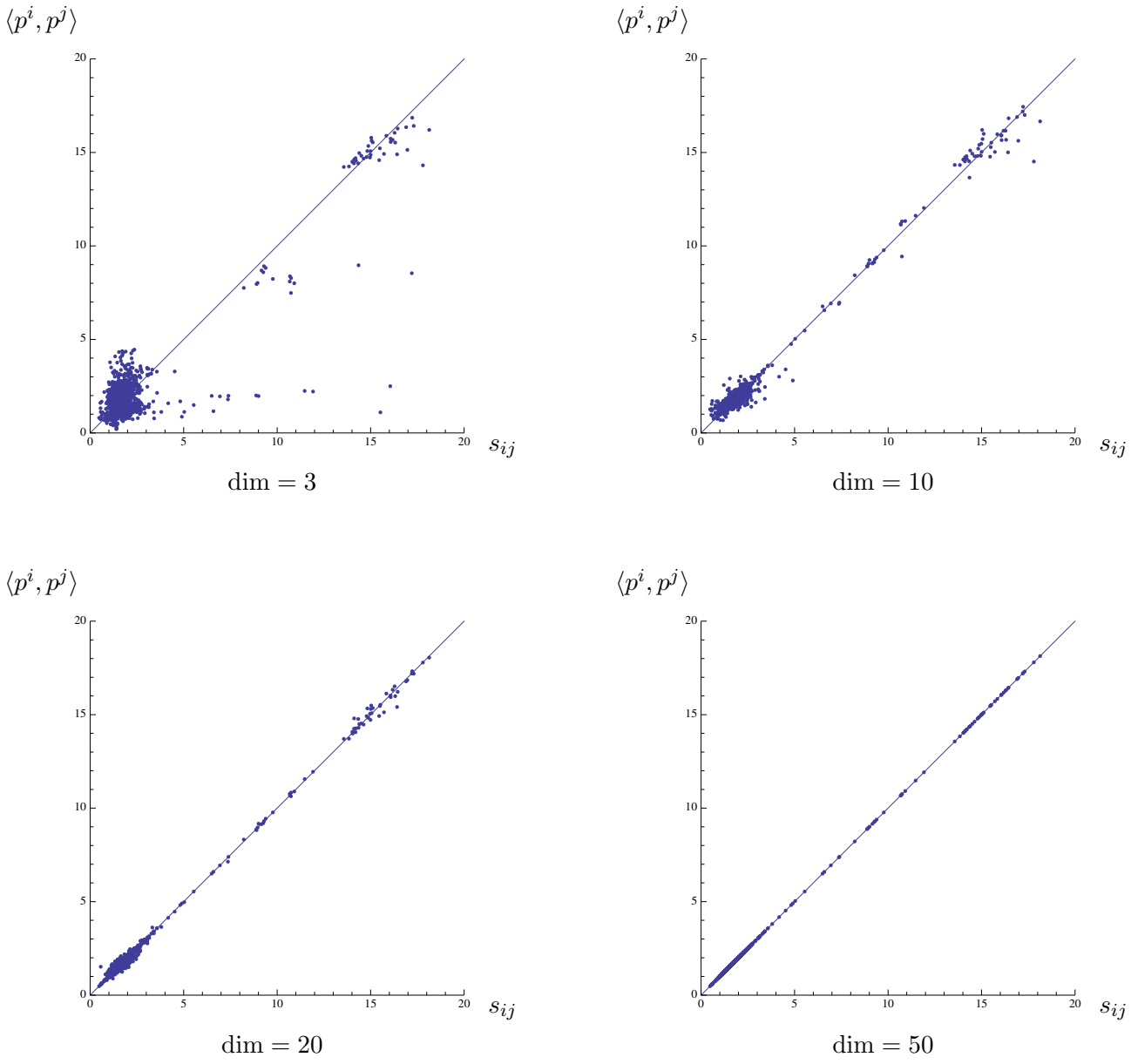


Figura 3.1: Gráficos obtidos nos testes com 50 proteínas (1225 escores).

onde m é a dimensão do espaço. As restrições de não negatividade das coordenadas do vetores são para forçar que $\phi(t)$ seja uma função não decrescente, ou seja, quanto maior o valor de $\langle p^i, p^j \rangle$, maior será o valor de $\phi(\ \langle p^i, p^j \rangle\ )$. Se temos um conjunto de n proteínas, o problema definido acima possui um total de $mn + 2n_p$ variáveis: n vetores em $\mathbb{R}^m$, n_p coeficientes c_ℓ e n_p expoentes α_ℓ. Então, ao resolver (3.5) as coordenadas dos vetores e os parâmetros da função $\phi(t)$ são ajustados ao mesmo tempo. Assim, podemos aumentar arbitrariamente a quantidade de termos de $\phi(t)$ mantendo fixa a dimensão m. Verificaremos a seguir se esta estratégia é útil para conseguir bons resultados de ajustes em $\mathbb{R}^2$ ou $\mathbb{R}^3$.

Experimentos numéricos

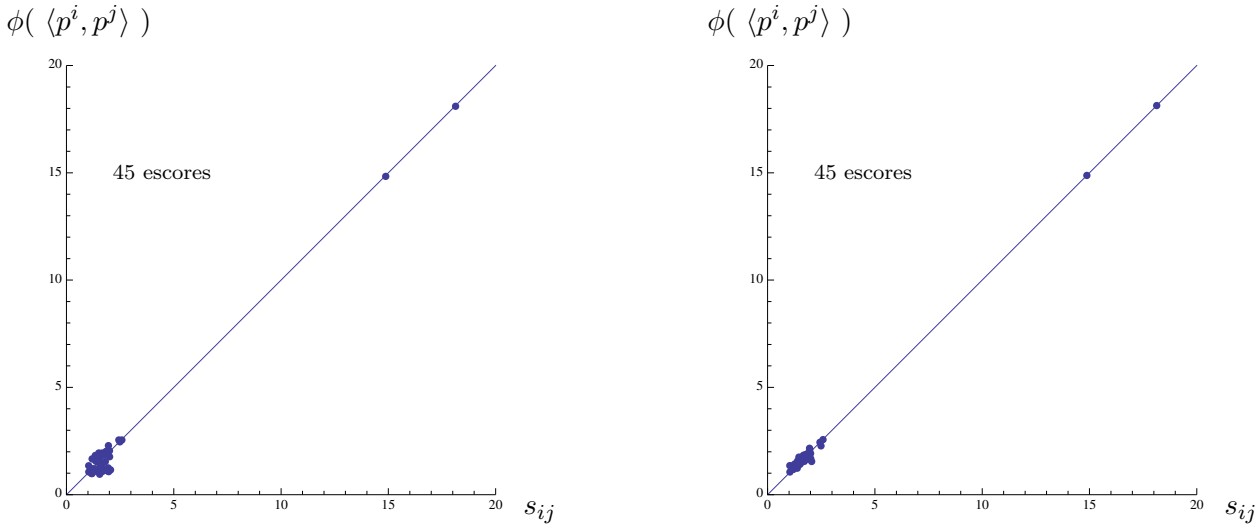
Para validar o modelo tomamos as mesmas 10 proteínas usadas nos experimentos com o Modelo 3.1.1 e utilizamos *GENCAN* na resolução dos problemas. Criamos dois conjuntos de testes. Cada problema foi resolvido mil vezes, sendo que em cada rodada um chute inicial diferente foi fornecido para a rotina de otimização. No primeiro conjunto fixamos a dimensão do espaço e variamos a quantidade de termos da função $\phi(t)$(parâmetro n_p). A Tabela 3.3 mostra, para cada teste, o menor valor de função obtido ao final das mil rodadas. A coluna n_p indica a quantidade de termos considerados para definir $\phi(t)$. Por fim, *niter*, *avalf* indicam, respectivamente, os números totais de iterações e avaliações de função realizados por *GENCAN* e $f(x^*)$ é o valor final de função.

	dim = 2			dim = 3		
n_p	niter	avalf	$f(x^*)$	niter	avalf	$f(x^*)$
5	119	682	6.050502E+00	46	90	1.138318E+00
10	186	558	6.050502E+00	68	151	1.138318E+00
20	324	1975	6.050502E+00	97	242	1.138318E+00
50	95	379	6.050502E+00	4005	20000	1.073184E+01

Tabela 3.3: Testes com 10 proteínas e diferentes quantidades de termos para $\phi(t)$.

De acordo com os resultados da Tabela 3.3 podemos notar que apesar de aumentarmos o número de variáveis do problema com a introdução da função $\phi(t)$, o menor valor de função obtido foi igual para todos os testes de mesma dimensão, com exceção do último experimento em dimensão 3. As Figuras 3.2 a) e 3.2 b) ilustram, respectivamente, os resultados obtidos nos testes com dimensão 2 e 3 e $n_p = 5$ (terceira linha da Tabela 3.3). A quantidade de escores ajustados também está indicada em cada gráfico.

No segundo conjunto de testes fixamos em 5 a quantidade de termos da função $\phi(t)$ e variamos a dimensão do espaço de vetores. Também realizamos mil rodadas para cada problema, fornecendo ao algoritmo um ponto inicial diferente a cada vez. A Tabela 3.4 mostra os resultados correspondentes ao menor valor de função obtido ao final das rodadas. Notemos nestes resultados que o valor da função objetivo só diminui consideravelmente quando a dimensão do espaço é superior a 5. Este comportamento é semelhante àquele



a) dim = 2 e $f(x^*) = 6.050502$ b) dim = 3 e $f(x^*) = 1.138318$

Figura 3.2: Resultados de testes com 10 proteínas e 5 termos para $\phi(t)$.

obtido nos testes realizados com o Modelo 3.1.1, onde ajustávamos diretamente os produtos escalares aos escores. Pelos resultados da Tabela 3.3 notamos também que nada adiantou aumentar o número de parâmetros da função $\phi(t)$ na tentativa de conseguir pequenos valores finais de $f(x)$ em $\mathbb{R}^2$ ou $\mathbb{R}^3$. Acreditamos que a dificuldade dessa formulação se deva ao fato de tentarmos ajustar ao mesmo tempo variáveis de natureza bastante diferente: coordenadas de vetores e parâmetros de uma função. Para tentar melhorar os resultados obtidos e realizar testes com quantidades maiores de proteínas, podemos ainda tentar resolver o problema (3.5) aplicando uma estratégia conhecida por *separação de variáveis*, cujo objetivo é reduzir o número de variáveis do problema. A seguir, detalhamos esta estratégia e apresentamos os testes computacionais realizados.

$n_p = 5$			
dim	niter	avalf	$f(x^*)$
3	46	90	1.138318E+00
4	10000	19359	3.507503E−01
5	569	2392	3.086025E−02
7	420	717	1.111408E−08
10	79	174	2.936309E−08

Tabela 3.4: Testes com 10 proteínas e diferentes dimensões de espaço.

Experimentos numéricos aplicando separação de variáveis

Como o problema (3.5) possui dois conjuntos distintos de variáveis (coordenadas de vetores e parâmetros de uma função), podemos aproveitar sua estrutura para resolvê-lo de uma forma

diferente. Denotando por x_p o vetor que contém as coordenadas dos vetores que representam n proteínas e por x_α o vetor que contém os parâmetros da função $\phi(t)$, isto é,

$$\begin{aligned} x_p &= (p_1, p_2, \ldots, p_n)^T, \\ x_\alpha &= (c_1, \alpha_1, c_2, \alpha_2, \ldots, c_{n_p}, \alpha_{n_p})^T, \end{aligned} \tag{3.6}$$

o problema (3.5) pode ser reescrito da seguinte forma:

$$\begin{aligned} \min \quad & f(x_p, x_\alpha) \\ \text{s. a} \quad & x_p \in V_1, \\ & x_\alpha \in V_2, \end{aligned} \tag{3.7}$$

onde V_1 e V_2 formam o domínio da função $f(x_p, x_\alpha)$. A estratégia para resolver (3.7) é a seguinte:

(i) Para $\widehat{x}_\alpha \in V_2$ fixo, defina $x_p(\widehat{x}_\alpha) = \text{argmin}\ \{f(x_p, \widehat{x}_\alpha)$ s. a $x_p \in V_1\}$,

(ii) Resolva o problema

$$\begin{aligned} \min\ \widehat{f}(x_\alpha) &= f(x_p(x_\alpha),\ x_\alpha) \\ \text{s. a}\ \ x_\alpha &\in V_2. \end{aligned} \tag{3.8}$$

Após separarmos as variáveis de (3.5), o problema que devemos resolver é dado por (3.8). Notemos que o problema reformulado possui um número menor de variáveis. Porém, a função $\widehat{f}$ é um pouco mais complicada que a função objetivo de (3.7), pois, em geral, não se consegue uma expressão analítica para $x_p(x_\alpha)$.

Nos experimentos numéricos utilizando a técnica de separação de variáveis usamos dois pacotes de otimização numérica: *BOBYQA* e *GENCAN*. O pacote *BOBYQA* (*Bound Optimization By Quadratic Approximation*) resolve problemas de otimização com restrições de caixa sem que derivadas de primeira ordem da função objetivo sejam fornecidas (ver [24, 78]). Decidimos usá-lo na resolução do problema (3.8) para não nos preocuparmos com os cálculos das primeiras derivadas da função $\widehat{f}(x_\alpha)$, que envolvem logaritmos. Em cada avaliação de função realizada por *BOBYQA* utilizamos *GENCAN* para determinar $x_p(x_\alpha)$. Realizamos testes com cinco quantidades de proteínas, também retiradas de `www.ime.unicamp.br/~martinez/lovoalign` e utilizamos *LOVOALIGN* para compará-las. Para cada conjunto de proteínas fixado, fizemos experimentos considerando dois valores de n_p e três valores para a dimensão do espaço de vetores. Cada problema foi resolvido cem vezes sendo que em cada rodada um ponto inicial diferente foi gerado. A Tabela 3.5 mostra o menor valor de função obtido por teste ao final das rodadas. Nos testes com 10 proteínas notamos que o valor final de função é pequeno quando a dimensão do espaço considerada é 5 e 10, independentemente do valor adotado para o parâmetro n_p. Nos demais testes os valores de dimensão e do parâmetro n_p considerados não foram suficientes para fazer com que a função atingisse um

valor menor que 1.0E−02. O comportamento de *GENCAN* na resolução dos subproblemas foi semelhante ao que já havíamos observado anteriormente: em alguns testes o algoritmo parou porque atingiu o número máximo de iterações permitidas e em outros testes parou porque atingiu o número máximo de avaliações de função objetivo.

	$n_p = 5$			$n_p = 10$		
nprot	dim = 3	dim = 5	dim = 10	dim = 3	dim = 5	dim = 10
10	1.1586276E+00	3.207695E−02	9.115795E−10	1.157679+00	3.237525E−02	4.704389E−11
25	3.321477E+02	4.681135E+01	5.843332E+01	3.697164E+02	3.829086E+02	1.101549E+01
50	1.494711E+03	7.894396E+02	4.999429E+02	1.376844E+03	1.525777E+04	2.029660E+02
75	2.661344E+03	2.791524E+03	2.746218E+03	4.116157E+03	2.821704E+03	8.899012E+02
100	4.892202E+04	3.971207E+03	4.919083E+04	4.855438E+03	4.825231E+03	4.958534E+04

Tabela 3.5: Resultados obtidos pela estratégia de separação de variáveis.

Os resultados numéricos obtidos pela aplicação da técnica de separação de variáveis não foram muito surpreendentes. Com efeito, os valores finais de função são bastante parecidos com os valores obtidos pela estratégia de ajustar ao mesmo tempo os dois conjuntos de variáveis. Em pequenas dimensões ($\mathbb{R}^2$ ou $\mathbb{R}^3$) praticamente não foi possível obter uma configuração com valor final de função pequeno. Analisando as dificuldades desse modelo, bem como do modelo anterior, chegamos à conclusão de que estamos dando o mesmo grau de importância tanto para escores de proteínas parecidas quanto para escores de proteínas que não possuem semelhança alguma. Então, como fizemos no caso do ajuste de distâncias, propomos a seguir um modelo que tentará ajustar somente os escores correspondentes a proteínas que possuem semelhanças significativas entre si.

3.1.3 Modelo 3: ajustando um subconjunto de escores

Nesta abordagem exigimos menos dos escores relativos a proteínas que não se parecem e construímos um modelo que tenta ajustar somente os escores de proteínas semelhantes. Para isso, fixamos um valor $s_{\text{fix}} > 0$ e impomos uma das seguintes condições para cada s_{ij}:

$$\begin{aligned} &\text{se } s_{ij} \leq s_{\text{fix}}, \quad \text{devemos encontrar dois vetores } p^i, p^j \in \mathbb{R}^m \text{ tais que } \langle p^i, p^j \rangle \leq s_{\text{fix}}, \\ &\text{se } s_{ij} > s_{\text{fix}}, \quad \text{devemos encontrar dois vetores } p^i, p^j \in \mathbb{R}^m \text{ tais que } \langle p^i, p^j \rangle \approx s_{ij}. \end{aligned} \tag{3.9}$$

Ou seja, os produtos escalares referentes a escores maiores que s_{fix} serão efetivamente ajustados. Já os produtos escalares correspondentes a escores menores ou iguais a s_{fix} poderão assumir qualquer valor que não ultrapasse s_{fix}. A função objetivo deste modelo é construída para que valha zero se uma das condições (3.9) forem satisfeitas. Portanto, nosso novo

problema de otimização é dado por:

$$\begin{aligned} \min \quad & \sum_{i,j \,\in\, I_1} \max\{0, \langle p^i, p^j \rangle - s_{\text{fix}}\}^2 \quad + \sum_{i,j \,\in\, I_2} \left(\langle p^i, p^j \rangle - s_{ij} \right)^2 \\ \text{s. a} \quad & p_k^i \geq 0, \quad 1 \leq k \leq m \quad \text{e} \quad 1 \leq i < j \leq n, \end{aligned} \tag{3.10}$$

onde I_1 é o conjunto de índices das proteínas cujos escores são menores ou iguais a s_{fix}, I_2 corresponde aos índices das proteínas com escores maiores que s_{fix}. Para referência futura, denotaremos por f_1 a parte da função objetivo que corresponde ao somatório sobre os índices de I_1 e por f_2 o somatório sobre os índices de I_2. Desse modo temos $f(x) = f_1(x) + f_2(x)$. Consideramos inicialmente as restrições de não negatividade nas variáveis do problema (3.10) para evitar produtos escalares negativos, pois, se um determinado valor de escore s_{ij} é menor que s_{fix}, qualquer valor negativo para o correspondente produto escalar satisfaz a condição $\langle p^i, p^j \rangle < s_{\text{fix}}$.

Experimentos numéricos

No primeiro conjunto de experimentos tomamos $s_{\text{fix}} = 10$ e investigamos o valor de dimensão adequado para um ajuste razoável entre produtos escalares e escores. Trabalhamos com seis quantidades de proteínas comparadas via *LOVOALIGN*. As proteínas foram retiradas de `www.ime.unicamp.br/~martinez/lovoalign` e a rotina *GENCAN* foi utilizada na resolução dos problemas. Na Tabela 3.6 abaixo, *nprot* é a quantidade de proteínas e n_{esc} é o número total de escores *Structal* normalizados resultantes da comparação de cada conjunto de proteínas. As duas últimas colunas desta tabela indicam, respectivamente, a quantidade de escores que são menores ou iguais e maiores que o escore fixo.

nprot	n_{esc}	$s_{ij} \leq s_{\text{fix}}$	$s_{ij} > s_{\text{fix}}$
10	45	43	2
25	300	285	15
50	1225	1176	49
75	2775	2698	77
100	4950	4827	123
200	19900	19473	427

Tabela 3.6: Divisão do conjunto de escores para $s_{\text{fix}} = 10$.

Nos testes que apresentamos a seguir, empregamos uma estratégia *multistart*. Cada problema foi resolvido cem vezes e em cada rodada fornecemos para a rotina *GENCAN* um ponto inicial gerado de maneira aleatória. Os valores indicados nas linhas das tabelas a seguir correspondem aos testes em que obtivemos o melhor (menor) valor de função objetivo de um total de cem rodadas. A Tabela 3.7 mostra os valores finais de função obtidos nos experimentos com escore fixo $s_{\text{fix}} = 10$. Nesta tabela a coluna *dim* indica as dimensões

consideradas para os espaços de vetores. Como era esperado, à medida que a dimensão aumenta obtemos valores finais de função cada vez menores. Com exceção do teste com 200 proteínas, o valor dim = 7 é adequado para resolver os demais problemas, pois os valores finais de $f(x)$ obtidos para esta dimensão são menores que 10^{-2}. Outras informações como os números totais de iterações (*niter*), avaliações de função (*avalf*) estão indicados na Tabela 3.8. Esta última tabela também mostra os tempos de execução (medidos em segundos) de cada problema. Notemos que nos problemas com maior quantidade de proteínas e dimensão maior que 4 a rotina *GENCAN* realiza um número maior de avaliações de função e, consequentemente, demanda um tempo maior de execução. No teste com 100 proteínas e dimensão 7, *GENCAN* atingiu o número máximo de avaliações de função permitidas. Já nos demais experimentos o critério de parada foi o da norma do gradiente menor que a tolerância 10^{-4}.

	nprot					
dim	10	25	50	75	100	200
3	1.094955E−11	5.012041E−01	3.727699E+01	1.381199E+02	2.524430E+02	1.406232E+03
4	1.581315E−13	1.165863E−13	1.240525E+00	3.975321E+01	1.087435E+02	7.189665E+02
5	1.342019E−14	2.000261E−19	6.256323E−03	8.380050E−01	1.958207E+01	4.124285E+02
6	2.098763E−21	2.880469E−13	7.924842E−15	2.301182E−16	2.449959E−01	1.924290E+02
7	1.157762E−12	1.031441E−13	3.695400E−11	2.114099E−09	9.235962E−03	6.750150E+01

Tabela 3.7: Valores finais de função dos testes com restrições nas variáveis.

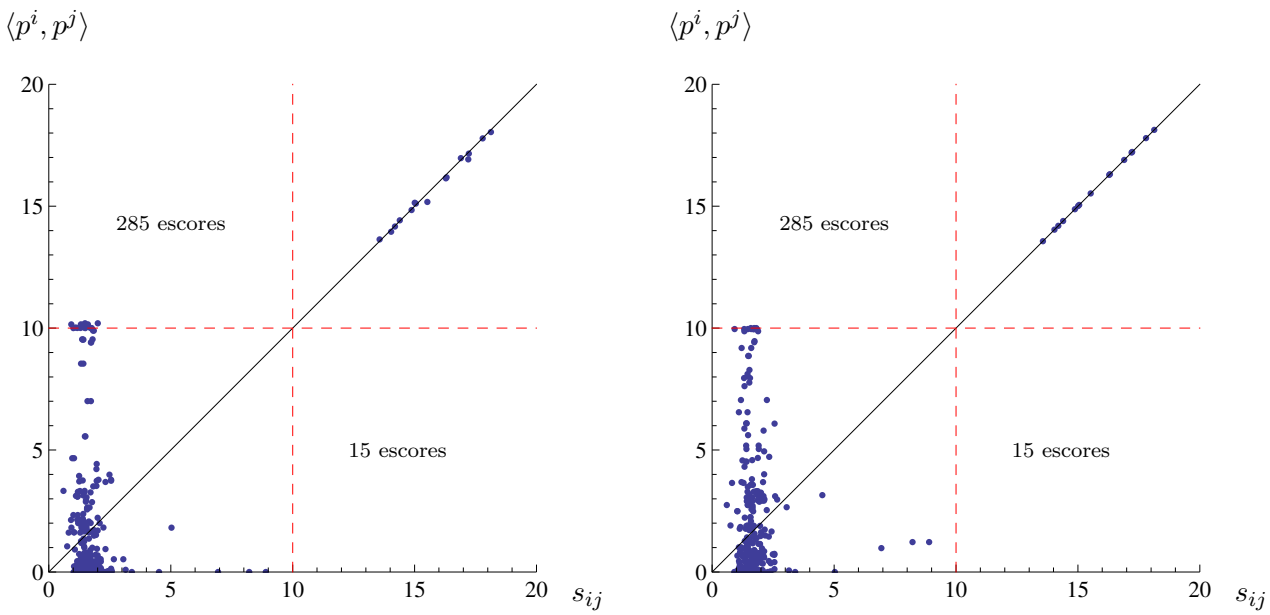
As Figuras 3.3 a) e 3.3 b) mostram, respectivamente, gráficos construídos a partir dos resultados dos experimentos com 25 proteínas para valores de dimensão 3 e 4. Na Figura 3.3 a) podemos observar que as condições (3.9) são razoavelmente satisfeitas. Neste teste, os valores finais das partes 1 e 2 da função objetivo foram $f_1(x) = 0.210853$ e $f_2(x) = 0.290350$, respectivamente. Porém, ao aumentarmos a dimensão do espaço para 4 conseguimos obter melhores resultados (veja a Figura 3.3 b)) pois os valores finais das partes 1 e 2 de $f(x)$ são $f_1(x) = 1.093965\text{E}{-16}$ e $f_2(x) = 1.164769\text{E}{-13}$, respectivamente. Nos gráficos também estão indicadas, de cada lado da reta vertical $s_{ij} = 10$, as quantidades de produtos escalares que satisfazem as condições (3.9), ou seja, do lado esquerdo está o total de produtos escalares menores ou iguais a s_{fix} e do lado direito está o total de produtos escalares efetivamente ajustados.

A Figura 3.4 mostra dois pontos de vista diferentes para a configuração final de vetores obtida no teste com 25 proteínas e dimensão 3. A cada ponto na figura está associado um vetor $p^i = (p_1^i, p_2^i, p_3^i)^T$ que emana da origem do sistema de eixos coordenados (linhas tracejadas). Quanto mais afastados estão os vetores no desenho, menor é o produto escalar entre eles e, consequentemente, menor é a semelhança entre as proteínas que eles representam. Observando com um pouco mais de atenção a Figura 3.4, notamos que alguns vetores possuem tamanho reduzido e se concentram em torno da origem do sistema de coordenadas. Este mesmo comportamento foi observado até nos experimentos com a dimensão do

	nprot								
	10			25			50		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	9	42	0.004	35	116	0.008	56	179	0.024
4	7	33	0.004	43	118	0.008	114	607	0.160
5	6	39	0.000	41	162	0.016	1057	7778	6.844
6	11	38	0.000	36	131	0.016	168	745	0.684
7	7	44	0.000	24	84	0.004	1313	10539	7.180

	nprot								
	75			100			200		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	39	125	0.028	59	178	0.060	86	242	0.488
4	146	480	0.260	106	317	0.400	138	461	1.348
5	187	613	0.504	304	1293	2.632	232	730	2.380
6	678	4243	15.844	436	1620	6.236	307	911	6.280
7	217	1374	4.424	2349	20000	128.556	506	1533	20.309

Tabela 3.8: Outras informações dos testes com restrições nas variáveis.



a) dim = 3 b) dim = 4

Figura 3.3: Gráficos referentes aos testes com 25 proteínas e $s_{\text{fix}} = 10$.

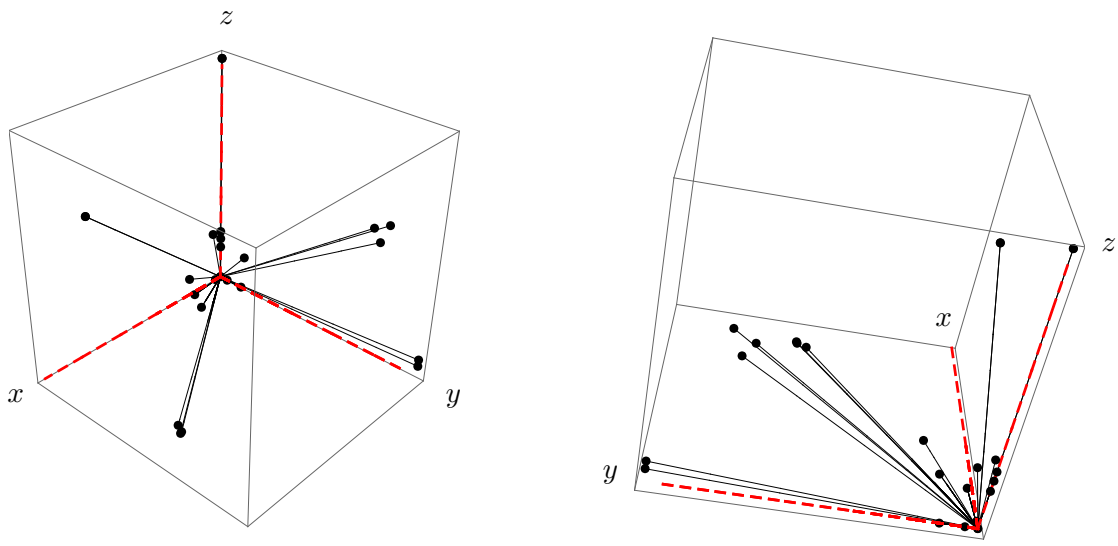


Figura 3.4: Duas perspectivas de uma representação 3D com 25 proteínas.

espaço de vetores maior que 3. Para evitar este tipo de dificuldade, decidimos retirar do Modelo 3.1.3 as restrições que forçam as coordenadas dos vetores a permanecerem no ortante positivo:

$$p_k^i \geq 0,\ 1 \leq k \leq m \text{ e } 1 \leq i < j \leq n. \tag{3.11}$$

Retirando essas restrições, o problema passa a ser irrestrito nas variáveis p_k^i e os produtos escalares finais entre vetores que correspondem a proteínas com pouca ou nenhuma semelhança, podem assumir valores negativos. Isso não nos traz dificuldades na interpretação dos resultados, pois, quanto mais negativo for o produto escalar entre dois vetores $p^i, p^j \in \mathbb{R}^m$, menos relacionadas estão as proteínas correspondentes.

Utilizando os mesmos conjuntos de proteínas dos experimentos anteriores, realizamos testes com esta pequena variação do Modelo 3.1.3. Empregamos novamente uma estratégia *multistart*: rodamos cada problema cem vezes e colocamos nas tabelas os resultados referentes à rodada com menor valor final de função objetivo. Mantivemos em todas as rodadas o parâmetro $s_{\text{fix}} = 10$. A Tabela 3.9 mostra os valores finais de função obtidos. Observemos, por exemplo, o teste com 200 proteínas e dimensão 6. Neste teste o valor final de função obtido por *GENCAN* foi $f(x) = 4.365591$. Para o mesmo conjunto de proteínas, a Tabela 3.7 mostra o valor $f(x) = 192.4290$. Ou seja, ao retiramos do problema as restrições de não negatividade contribuimos para uma ligeira melhora nos resultados.

A Tabela 3.10 traz outras informações referentes aos experimentos realizados com esta variação do Modelo 3.1.3. Nesta tabela *dim* é a dimensão do espaço de vetores considerado, *niter* é o número total de iterações, *avalf* é o número total de avaliações de função objetivo e $t(s)$ é o tempo de execução medido em segundos. Com exceção do último teste com 100 e 200 proteínas (última linha da Tabela 3.10), onde *GENCAN* parou porque atingiu o número máximo de avaliações de função permitidas, o critério de parada nos demais experimentos

foi o da norma do gradiente menor que 10^{-4}.

	nprot					
dim	10	25	50	75	100	200
3	7.250321E−14	2.247543E−15	3.828029E−01	6.201106E−01	5.176705E+00	2.696470E+02
4	6.635546E−13	9.910505E−13	5.872790E−03	1.616577E−02	2.291639E+00	1.750138E+01
5	5.951508E−22	5.829329E−21	4.069083E−07	3.543946E−04	7.186238E−01	7.875880E+00
6	1.041079E−10	4.617248E−11	8.730313E−13	2.716402E−08	7.593816E−02	4.365591E+00
7	3.845608E−14	1.135802E−16	6.036439E−14	5.460451E−12	7.634443E−03	2.346628E+00

Tabela 3.9: Valores finais de função dos testes sem restrições nas variáveis.

	nprot								
	10			25			50		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	5	37	0.000	14	85	0.004	41	232	0.092
4	6	27	0.000	19	67	0.008	564	5089	3.648
5	6	11	0.000	22	123	0.012	344	3090	3.888
6	6	8	0.000	14	58	0.008	64	349	0.272
7	10	24	0.000	14	58	0.008	49	227	0.176

	nprot								
	75			100			200		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	112	767	0.959	919	10133	38.074	113	818	10.792
4	217	1769	4.188	433	3570	16.4130	1130	11172	469.457
5	1202	10395	48.699	388	3178	19.549	702	7076	303.030
6	744	8171	39.274	256	2232	19.705	456	4382	249.815
7	88	518	1.368	1406	20000	293.034	1664	20000	1722.739

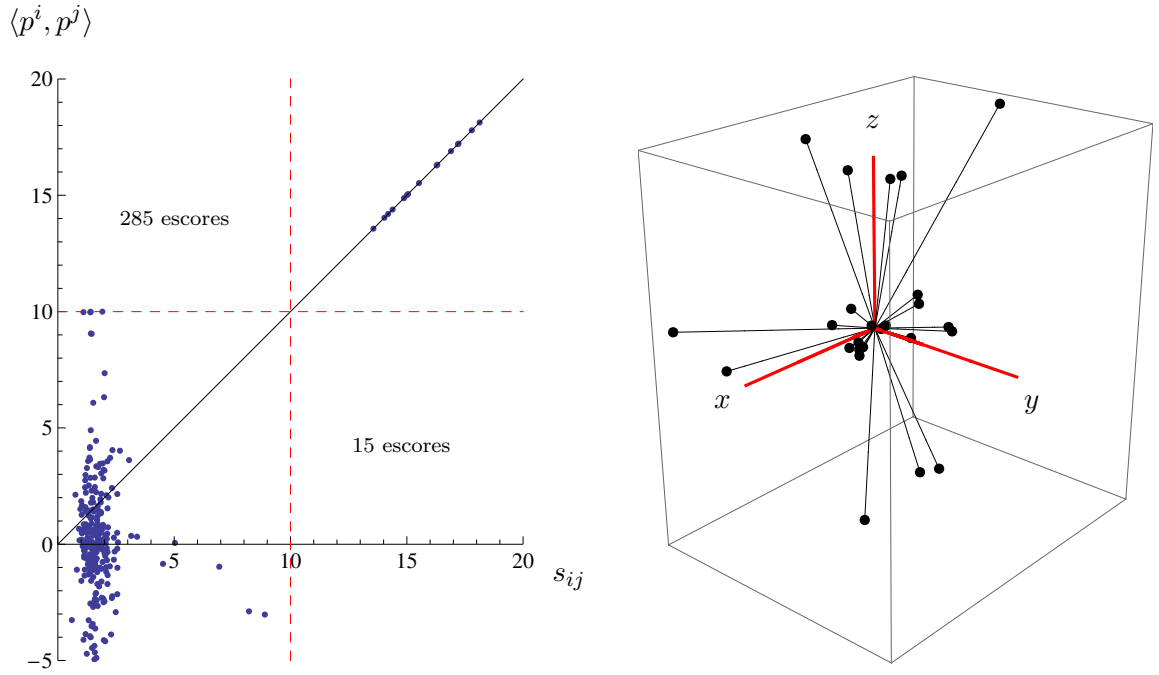
Tabela 3.10: Outras informações dos testes sem restrições nas variáveis.

Os resultados numéricos obtidos no teste com 25 proteínas e dimensão 3 estão ilustrados nas Figuras 3.5 a) e 3.5 b). A Figura 3.5 a) mostra o gráfico dos escores *Structal* normalizados (s_{ij}) *versus* produtos escalares dos vetores ($\langle p^i, p^j \rangle$). Notemos na figura a presença de vários produtos escalares negativos devido ao fato de retirarmos do Modelo 3.1.3 as restrições de não negatividade das coordenadas dos vetores. A configuração final de vetores em $\mathbb{R}^3$ correspondentes a este gráfico está ilustrada na Figura 3.5 b). Cada vetor i representado tem uma extremidade na origem do sistema de coordenadas e outra extremidade no ponto de coordenadas (p_1^i, p_2^i, p_3^i). Vetores próximos neste desenho representam proteínas semelhantes. Podemos observar também na figura que alguns vetores possuem tamanho pequeno e se

aproximam do vetor nulo. Este comportamento também foi notado nos experimentos onde as coordenadas dos vetores estavam restritas ao ortante positivo.

A obtenção de uma representação para um conjunto de proteínas comparadas onde algumas proteínas são representadas por vetores que estão próximos do vetor nulo não parece ser uma boa estratégia, sobretudo se quisermos utilizar essa representação para classificar novas proteínas, como faremos mais adiante. Sendo assim, propomos agora uma ligeira modificação do modelo que estamos tratando. Pediremos que todos os vetores que representam as proteínas tenham o mesmo tamanho, ou seja, pertençam a uma esfera de $\mathbb{R}^m$ centrada na origem e de raio fixo. Pretendemos com esta imposição impedir que os vetores tenham norma próxima de zero.

A seguir, apresentamos os detalhes do modelo modificado, realizamos testes numéricos para validá-lo e discutimos os resultados obtidos. Em seguida, utilizamos este modelo para classificar novas proteínas. Nesta estratégia de classificação, adaptamos um algoritmo de clusterização que seleciona no conjunto de proteínas representadas aquelas que são mais parecidas com a proteína que queremos classificar.



a) $s_{ij} \times \langle p^i, p^j \rangle$ b) 25 vetores em $\mathbb{R}^3$

Figura 3.5: Teste com 25 proteínas: Modelo 3.1.3 sem restrições nas variáveis.

3.2 Construção de mapas de proteínas na esfera

Nos experimentos que seguem, além de retirarmos do problema (3.10) as restrições de não negatividade das variáveis, vamos adicionar à função objetivo penalizações das restrições

$$\|p^i\| = \sqrt{20}, \quad \forall\ i = 1, \ldots, n. \tag{3.12}$$

Procedendo dessa forma agora, nosso novo objetivo consiste em encontrar n vetores em $\mathbb{R}^m$ que tenham norma igual a $\sqrt{20}$ e satisfaçam as condições (3.9). A escolha para o tamanho dos vetores ser $\sqrt{20}$ é devida ao fato de trabalharmos com os escores *Structal* normalizados na realização dos experimentos. Como já dissemos anteriormente (veja o Capítulo 1), estes escores assumem valores no intervalo (0, 20]. Então, no caso de termos duas proteínas idênticas em um conjunto, o escore *Structal* normalizado entre elas será igual a 20 e o vetor $p^i \in \mathbb{R}^m$ que a representa deve satisfazer a equação

$$\langle p^i, p^i \rangle = \|p^i\|^2 = 20. \tag{3.13}$$

Portanto, o problema de otimização que passamos a resolver é o seguinte:

$$\min \sum_{i,j\ \in\ I_1} \max\{0, \langle p^i, p^j \rangle - s_{\text{fix}}\}^2 + \sum_{i,j\ \in\ I_2} (\langle p^i, p^j \rangle - s_{ij})^2 + \sum_{i=1}^{n} (\|p^i\|^2 - 20)^2, \tag{3.14}$$

onde I_1 é o conjunto de índices das proteínas cujos escores são menores ou iguais a s_{fix} e I_2 é o conjunto de índices das proteínas cujos escores são maiores que s_{fix}.

Para testar esta modificação do Modelo 3.1.3 utilizamos os mesmos conjuntos de proteínas dos experimentos anteriores. Empregamos novamente uma estratégia *multistart* resolvendo cada problema cem vezes e colocamos nas tabelas somente os resultados referentes à rodada em que *GENCAN* obteve o menor valor final de função objetivo. O escore fixo usado em todos os testes foi $s_{\text{fix}} = 10$ e o critério de parada de *GENCAN* foi o mesmo em todos os experimentos: norma do gradiente menor que 10^{-4}. A Tabela 3.11 mostra os valores finais de função obtidos pela rotina. Já os números de iterações, avaliações de função e tempo de execução (medido em segundos) estão indicados na Tabela 3.12.

	nprot					
dim	10	25	50	75	100	200
3	4.755576E−16	2.102044E+02	1.434096E+03	2.976164E+03	4.594501E+03	1.162422E+04
4	2.204832E−16	3.137366E+00	4.564829E+02	1.505599E+03	2.636576E+03	7.852111E+03
5	4.000253E−12	2.990529E−12	2.679639E+01	3.574485E+02	1.019176E+03	4.859577E+03
6	6.664151E−15	5.076109E−18	8.984436E+00	1.916856E+01	2.356764E+03	2.383340E+03
7	3.779368E−16	5.641667E−11	3.032802E+00	7.724641E+00	2.149591E+01	8.667124E+02

Tabela 3.11: Valores finais de função para problemas com esferas de raio $\sqrt{20}$.

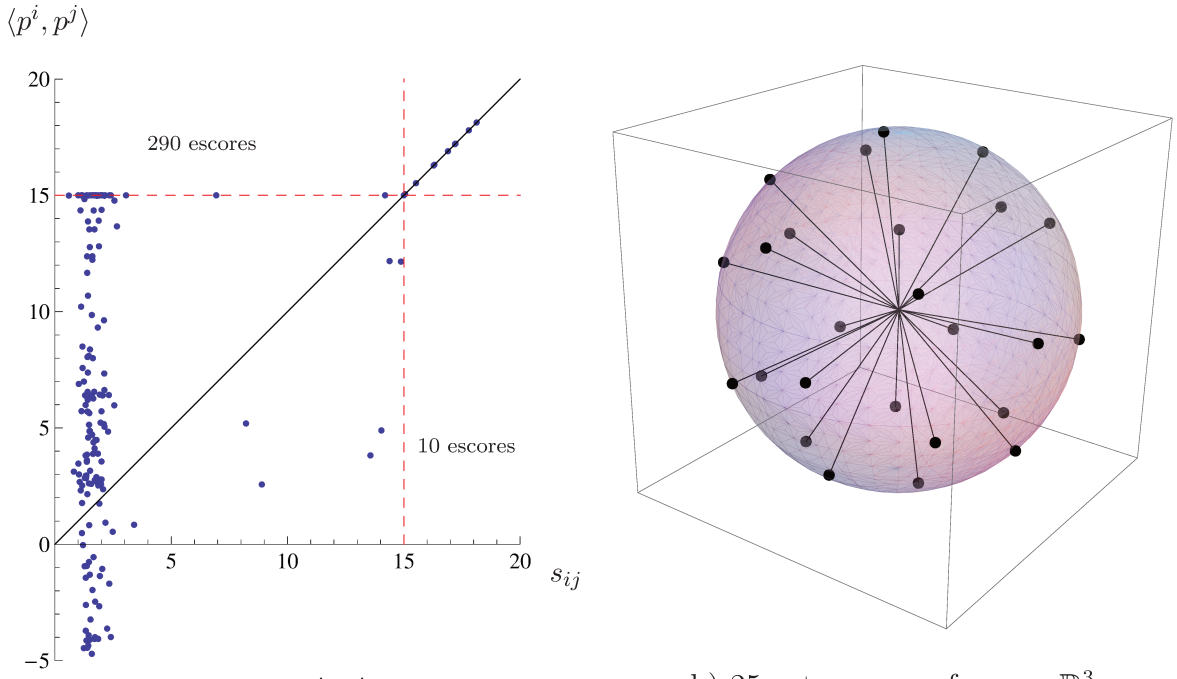
	nprot								
	10			25			50		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	8	16	0.000	29	112	0.012	51	222	0.116
4	12	34	0.004	82	482	0.136	36	137	0.064
5	8	11	0.000	22	92	0.028	83	467	0.356
6	8	26	0.000	16	37	0.012	101	756	1.256
7	9	16	0.000	10	18	0.008	38	139	0.340

	nprot								
	75			100			200		
dim	niter	avalf	$t(s)$	niter	avalf	$t(s)$	niter	avalf	$t(s)$
3	38	138	0.112	43	164	0.448	63	302	2.888
4	57	285	0.216	57	245	0.608	122	630	2.792
5	88	431	0.892	135	747	0.664	164	909	10.936
6	250	1766	3.772	164	1028	12.084	230	1367	20.933
7	111	905	6.336	954	10449	68.668	186	1199	36.098

Tabela 3.12: Outros dados de *GENCAN* nos testes com esferas de raio $\sqrt{20}$.

Na Tabela 3.12 notamos nos testes com 50, 75, 100 e 200 proteínas que a dimensão do espaço igual a 7 não é suficiente para que *GENCAN* encontre uma configuração de vetores com valor final de função menor que 10^{-3}. Em relação aos testes sem restrições nas variáveis (Tabela 3.10), os tempos de execução da rotina diminuíram bastante nos experimentos com restrições penalizadas para 100 e 200 proteínas. Podemos notar isso comparando os tempos da Tabela 3.12 com os tempos indicados na Tabela 3.10, onde fizemos testes com os mesmos conjuntos de proteínas. As Figuras 3.6 a) e 3.6 b) ilustram os resultados numéricos de um teste onde 25 proteínas são representadas por vetores pertencentes a uma esfera de $\mathbb{R}^3$ centrada em $(0, 0, 0)$ e de raio $\sqrt{20}$. Neste experimento foi preciso aumentar o parâmetro s_{fix} para 15, pois mantendo o valor $s_{\text{fix}} = 10$ não conseguimos obter uma configuração de vetores com valor final de função pequeno, como indica a primeira linha da Tabela 3.11.

Os três modelos para construção de mapas apresentados neste capítulo têm em comum o fato das proteínas serem representadas por vetores em um espaço euclidiano de dimensão m. Nos dois primeiros modelos (3.1.1 e 3.1.2) a estratégia de ajustar todos os produtos escalares só é bem sucedida se aumentamos consideravelmente a dimensão do espaço onde os vetores são alocados. Com o Modelo 3.1.3 conseguimos reduzir um pouco a dimensão do espaço ajustando somente os escores correspondentes a proteínas com semelhança significativa. Uma pequena alteração foi testada neste modelo a fim de evitarmos configurações com vetores de norma próxima de zero. As restrições de norma constante impostas ao problema (3.14) e penalizadas na função objetivo foram satisfeitas de modo bastante razoável nos experimentos realizados. Estas restrições tornaram nosso modelo mais coerente, pois



a) $s_{ij} \times \langle p^i, p^j \rangle$

b) 25 vetores na esfera em $\mathbb{R}^3$

Figura 3.6: Ajustando produtos escalares na esfera 3D.

nos ajudaram a reduzir as indeterminações ocasionadas pela busca de soluções em espaços de dimensão grande.

3.2.1 Classificação de proteínas via mapas na esfera

Dos três modelos propostos neste capítulo, o mais promissor foi o Modelo (3.1.3) em que as proteínas comparadas são representadas por vetores em uma esfera contida em $\mathbb{R}^m$ e de raio $\sqrt{20}$. Nossa intenção agora é utilizar esta representação para classificar proteínas novas.

Suponha um conjunto $C_{\text{prot}} = \{A_1, A_2, \ldots, A_n\}$ com n proteínas comparadas em relação às estruturas tridimensionais e considere uma representação para este conjunto

$$P_{\text{prot}} = \{p^1, p^2, \ldots, p^n\}, \text{ com } p^i \in \mathbb{R}^m \text{ e } \|p^i\| = \sqrt{20} \text{ para todo } i = 1, 2, \ldots, n,$$

obtida através da resolução do problema (3.14). Dada uma proteína $A \notin C_{\text{prot}}$, desejamos incluí-la na representação P_{prot} de modo que A fique próxima de proteínas com estruturas tridimensionais semelhantes à sua. Para realizar este procedimento utilizaremos somente um subconjunto pequeno de proteínas de C_{prot}. Nosso objetivo é evitar que A seja comparada com todas as proteínas deste conjunto a fim de reduzir o esforço computacional. Propomos então a seguinte estratégia:

(1) Selecionar em C_{prot} um subconjunto pequeno de n_A proteínas.

(2) Comparar, via *LOVOALIGN*, a estrutura 3D de A com as estruturas 3D das proteínas selecionadas e obter os escores *Structal* normalizados, isto é, determinar $s_{iA} \in (0, 20]$ para $i = 1, \ldots, n_A$.

(3) Utilizar as representações $p^i \in P_{\text{prot}}$ das n_A proteínas escolhidas no passo (1) para incluir a proteína A no mapa através da resolução do seguinte problema de otimização:

$$\begin{array}{ll} \min & (\|x_A\|^2 - 20)^2 + \sum_{i=1}^{n_A} (\ \langle p^i, x_A \rangle - s_{iA}\ )^2, \\ \text{s. a} & x_A \in \mathbb{R}^m. \end{array} \tag{3.15}$$

Uma vez resolvido o problema (3.15), estimamos os escores *Structal* normalizados entre A e as proteínas de C_{prot} que não foram selecionadas em (1) através do cálculo dos produtos escalares

$$\langle p^i, x_A \rangle \text{ para } i \in I, \tag{3.16}$$

onde I é o conjunto de índices das $n - n_A$ proteínas representadas que restaram em P_{prot}. Para escolher n_A proteínas em C_{prot}, propomos um algoritmo que divide C_{prot} em subconjuntos menores, chamados *clusteres*. Em seguida, realizamos buscas nestes *clusteres* com o objetivo de localizar proteínas estruturalmente semelhantes à proteína que queremos incluir na representação. Descrevemos a seguir os detalhes desse procedimento.

3.2.2 Algoritmo para seleção rápida de proteínas

A seleção de um subconjunto de proteínas de $C_{\text{prot}} = \{A_1, A_2, \ldots, A_n\}$ para a resolução do problema de inclusão (3.15) não deve ser feita de qualquer modo. Devemos escolher proteínas que tenham alguma semelhança com a proteína A que queremos incluir na representação P_{prot}. Deste modo, elaboramos um algoritmo que procura em C_{prot} proteínas semelhantes à proteína A, mas evita chamadas excessivas da rotina *LOVOALIGN* para comparar as estruturas tridimensionais. Nosso algoritmo é composto por duas fases: *fase de clusterização* e *fase de busca*. Na fase de clusterização, dividimos o conjunto C_{prot} em subconjuntos C_i (chamados *clusteres*), de forma que cada subconjunto contenha proteínas parecidas entre si. Adotamos a fórmula:

$$\text{disp}(A_i, A_j) = 20 - s_{ij},\ 1 \leq i < j \leq n, \tag{3.17}$$

para medir as disparidades entre os pares de proteínas, onde s_{ij} é o escore *Structal* normalizado entre as proteínas A_i e A_j. O procedimento utilizado nesta fase é uma adaptação do algoritmo proposto por Robert Clason em [17], que agrupa conjuntos de pontos no plano baseados na distância euclidiana entre eles. Em nossa versão adaptada, os pontos são substituídos pelos elementos do conjunto C_{prot} e as distâncias euclidianas são substituídas pelas disparidades calculadas de acordo com a fórmula (3.17).

O algoritmo da fase de clusterização (Algoritmo 1) começa escolhendo uma proteína A_i qualquer em C_{prot} para ser a líder do primeiro *cluster*. Em seguida, a proteína $A_j \in C_{\text{prot}}$ que estiver mais distante de A_i segundo a fórmula (3.17) é escolhida para ser a líder do segundo *cluster*. Então, cada proteína A_k restante em C_{prot} ($k \neq i, j$) é associada ao *cluster* da proteína líder mais próxima. Se uma das disparidades $\text{disp}(A_k, A_i)$ ou $\text{disp}(A_k, A_j)$ exceder a disparidade média entre A_i e A_j, a proteína A_k torna-se a proteína líder do terceiro *cluster* e o processo de comparações de disparidades e associação de proteínas aos *clusteres* recomeça. Caso contrário, o algoritmo termina. Todas as informações do processo de clusterização são armazenadas em três vetores: c, d e h. Por exemplo, se ao término do processo obtivermos $c(5) = 2$, $d(5) = 4.5$, isto significa que a proteína associada ao índice 5 pertence a um *cluster* liderado pela proteína associada ao índice 2 e a disparidade entre esse par de proteínas é de 4.5. O vetor h armazena apenas os índices das proteínas que foram escolhidas como líderes durante o processo de agrupamento. Dependendo da relação de semelhança entre os elementos de C_{prot}, o algoritmo pode criar *clusteres* contendo apenas uma proteína.

Algoritmo 1 Fase de Clusterização

Entrada: conjunto C_{prot} e escores *Structal* normalizados s_{ij}

Saída: vetores c, d e h contendo informações sobre os *clusteres*

1: Escolha $A_i \in C_{\text{prot}}$ e faça $\ell \leftarrow i$

2: Para $1 \leq j \leq n$ faça $c(j) \leftarrow \ell$

3: Faça $h(1) \leftarrow \ell$ e para $2 \leq j \leq n$ faça $h(j) \leftarrow 0$

4: Para $1 \leq j \leq n$: $d(j) \leftarrow \text{disp}(A_\ell, A_j)$ se $j \neq \ell$ e $d(j) \leftarrow 0$ caso contrário

5: Faça flag $\leftarrow 0$ e nc $\leftarrow 1$

6: **Enquanto** flag $= 0$ **faça**

7: nc $\leftarrow$ nc $+ 1$

8: $\ell \leftarrow$ índice em que ocorre $\max\{d_j\}$

9: $h(\text{nc}) \leftarrow \ell$

10: Para $1 \leq j \leq n$: $d_{\text{new}}(j) \leftarrow \text{disp}(A_\ell, A_j)$ se $j \neq \ell$ e $d_{\text{new}}(j) \leftarrow 0$ caso contrário

11: Para $1 \leq j \leq n$: se $d_{\text{new}}(j) < d(j)$ faça $d(j) \leftarrow d_{\text{new}}(j)$ e $c(j) \leftarrow \ell$

12: $d_{\max} \leftarrow \max\{d_j\}$

13: $d_{\text{media}} \leftarrow \left[\sum_{k \neq j}^{\text{nc}} \text{disp}(A_{h(k)}, A_{h(j)})\right] / N$, onde $N = \text{nc}(\text{nc} - 1)/2$

14: Se $d_{\max} < d_{\text{media}}$: flag $\leftarrow 1$. Volte ao passo 6.

15: **Fim**

Após o processo de clusterização tem início a fase de busca. Começamos esta fase sorteando aleatoriamente uma proteína A_i em C_{prot}. Em seguida, verificamos a qual *cluster* C_i ela pertence e utilizamos *LOVOALIGN* para compará-la com a proteína nova A. Se o

escore obtido nesta comparação é menor que $s_{\text{fix}} = 10$, declaramos que não existe semelhança significativa entre A e A_i, excluímos do próximo sorteio todas as proteínas que estão contidas em C_i e reiniciamos o processo de busca sorteando outra proteína em C_{prot}/C_i. Caso contrário, paramos a execução do algoritmo alegando que o *cluster* C_i contém proteínas que se parecem razoavelmente com a proteína nova. A vantagem desse procedimento é que toda vez que uma proteína A_i é selecionada no início da fase de busca, a rotina *LOVOALIGN* é invocada apenas uma vez para compará-la com a proteína nova, reduzindo desta maneira o número de comparações entre A e os elementos de C_{prot}. No pior dos casos, se cada elemento de C_{prot} constituir um *cluster*, *LOVOALIGN* será chamada n vezes, onde n é o número total de proteínas em C_{prot}.

Algoritmo 2 Fase de Busca

Entrada: conjunto C_{prot}, proteína $A \neq C_{\text{prot}}$ e vetor c obtido na fase de clusterização

Saída: conjunto C_{new} ou mensagem dizendo que nenhuma proteína semelhante foi localizada

1: Faça $n_{\text{lovo}} \leftarrow 0$, $s_{\text{fix}} \leftarrow 10$ e $C_{\text{new}} \leftarrow \{\ \}$
2: Faça $C_{\text{aux}} \leftarrow \{\ \}$. Se $|C_{\text{new}}| \neq 0$ vá para o passo 12
3: **Enquanto** $|C_{\text{prot}}| \neq 0$ **faça**
4: Escolha $A_i \in C_{\text{prot}}$ e faça $\ell \leftarrow i$
5: $s_{iA} = LOVOALIGN(A, A_i)$ $\quad \#LOVOALIGN$ usada como subrotina#
6: Faça $n_{\text{lovo}} \leftarrow n_{\text{lovo}} + 1$
7: Para $1 \leq j \leq n$: se $c(j) = c(\ell)$, faça $C_{\text{aux}} \leftarrow C_{\text{aux}} \cup \{A_j\}$
8: Se $s_{iA} \geq s_{\text{fix}}$: $C_{\text{new}} \leftarrow C_{\text{new}} \cup C_{\text{aux}}$
9: $C_{\text{prot}} \leftarrow C_{\text{prot}}/C_{\text{aux}}$
10: Volte ao passo 2
11: **Fim**
12: Pare. Exiba C_{new} ou imprima: "`nenhuma proteína semelhante encontrada`"

Apresentamos a seguir duas baterias de experimentos para validar nosso algoritmo. Na primeira, testamos separadamente as fases de clusterização e busca, empregando três conjuntos de proteínas comparadas. Computamos os tempos de execução de cada fase e analisamos os resultados obtidos. Na segunda bateria, resolvemos o problema de inclusão de proteínas utilizando os mapas construídos sobre a esfera em $\mathbb{R}^m$ (variação do Modelo 3.1.3). Assim, dados $C_{\text{prot}} = \{A_1, A_2, \ldots, A_n\}$, $A \notin C_{\text{prot}}$ e a representação $P_{\text{prot}} = \{p^1, p^2, \ldots, p^n\}$, com $p^i \in \mathbb{R}^m$ e $\|p^i\| = \sqrt{20}$, aplicamos inicialmente nosso algoritmo para selecionar em C_{prot} um subconjunto de n_A proteínas de acordo com os critérios que discutimos anteriormente. Em seguida, utilizamos os vetores p^i correspondentes às proteínas escolhidas para resolver o problema (3.15) e determinar x_A. Por fim, estimamos os escores *Structal* normalizados entre A e as proteínas restantes em C_{prot} usando (3.16). Para efeito de comparação, computamos

também os verdadeiros escores *Structal* normalizados entre A e todas as proteínas presentes em C_{prot} e analisamos os resultados graficamente.

3.2.3 Experimentos numéricos

Testes para validação do algoritmo de clusterização e busca

Os experimentos foram realizados com três conjuntos de proteínas retiradas do endereço `www.ime.unicamp.br/~martinez/lovoalign`. Inicialmente, utilizamos a rotina *LOVOALIGN* para comparar cada conjunto e obter os escores *Structal* normalizados necessários na fase de clusterização. Então, para cada conjunto de proteínas comparadas, testamos separadamente as duas fases de nosso algoritmo, isto é, primeiramente rodamos a fase de clusterização e, em seguida, testamos a fase de busca. A Tabela 3.13 abaixo mostra os resultados obtidos no processo de determinação dos *clusteres*. Nesta tabela *nprot* é a quantidade de proteínas presentes em cada conjunto C_{prot}, nc é o número de *clusteres* encontrados na fase de clusterização e $t(s)$ é o tempo (medido em segundos) necessário para a completa determinação desses *clusteres*. Cada experimento apresentado nesta tabela foi rodado apenas uma vez.

nprot	nc	$t(s)$
100	67	0.712525
200	125	4.405491
400	246	34.435182

Tabela 3.13: Validação da fase de clusterização.

Os resultados obtidos no processo de clusterização foram utilizados para testar o algoritmo da fase de busca. A Tabela 3.14 mostra os resultados referentes a estes experimentos. A tabela foi dividida em três partes, de acordo com a quantidade de proteínas (*nprot*) presentes em cada conjunto *clusterizado*. As colunas denotadas por "A" contêm as siglas das proteínas novas que não pertencem aos conjuntos utilizados. A coluna $|C_{\text{new}}|$ indica a quantidade de proteínas encontradas na fase de busca e que são razoavelmente semelhantes à proteína "A" por possuirem escores *Structal* normalizados maiores que $s_{\text{fix}} = 10$. A coluna n_{lovo} mostra o número de vezes que a rotina *LOVOALIGN* foi chamada durante a execução de cada teste. E por fim, a coluna $t(s)$ mostra o tempo (medido em segundos) de duração de cada busca. O tempo que *LOVOALIGN* despende em cada alinhamento realizado não está contabilizado no valor $t(s)$. Lembramos que o tempo registrado em cada teste depende da proteína escolhida para iniciar a fase de busca, já que esta escolha pode ser feita aleatoriamente.

Observemos, por exemplo, os resultados indicados na terceira linha da Tabela 3.14. No teste com o conjunto de 100 proteínas, foram necessárias apenas $n_{\text{lovo}} = 2$ chamadas de *LOVOALIGN* para que o algoritmo encontrasse $|C_{\text{new}}| = 12$ proteínas semelhantes à proteína `1h4w`. O número n_{lovo} também é pequeno no teste com a proteína `1fy8` e o conjunto de 400 proteínas: bastaram 8 comparações estruturais para que 32 proteínas semelhantes fossem

nprot = 100				nprot = 200				nprot = 400			
A	$\|C_{\text{new}}\|$	n_{lovo}	$t(s)$	A	$\|C_{\text{new}}\|$	n_{lovo}	$t(s)$	A	$\|C_{\text{new}}\|$	n_{lovo}	$t(s)$
1h4w	12	2	0.000967	19hc	0	125	0.063164	1fy8	32	8	0.007466
1npm	12	4	0.003106	1npm	20	6	0.003090	1A9WE	0	246	0.229368
1fiw	12	17	0.004349	1ykt	20	7	0.003599	1ykt	32	1	0.006027
2cst	2	25	0.006192	1trn	20	5	0.002915	1OBM	0	246	0.235059

Tabela 3.14: Validação da fase de busca.

localizadas. Repare agora no teste com a proteína 19hc e o conjunto de 200 proteínas (terceira linha da tabela). Neste caso o algoritmo parou porque percorreu todos os 125 *clusteres* e não detectou nenhuma proteína cujo escore *Structal* normalizado com relação a 19hc fosse maior que $s_{\text{fix}} = 10$. Por essa razão, obtivemos neste teste $|C_{\text{new}}| = 0$.

Testes com o problema de inclusão de proteínas na esfera

Realizamos alguns testes em mapas com 100 e 200 proteínas extraídas do endereço www.ime.unicamp.br/~martinez/lovoalign. Estas proteínas são as mesmas que figuram nos experimentos com os outros modelos apresentados neste capítulo. Inicialmente, construímos uma representação resolvendo o problema (3.14) com a rotina *GENCAN*. Fornecemos à rotina um ponto inicial gerado aleatoriamente e adotamos $s_{\text{fix}} = 10$ como o valor de escore fixo. Em seguida, retiramos do mesmo endereço eletrônico outras proteínas que não foram utilizadas nas representações e realizamos testes incluindo cada proteína no mapa separadamente. Para resolver cada problema de inclusão (3.15), selecionamos as n_A proteínas aplicando em C_{prot} o algoritmo de clusterização e busca discutido anteriormente. Os problemas de inclusão também foram resolvidos com a rotina *GENCAN*, partindo de um ponto inicial gerado de maneira aleatória.

Experimentos em um mapa com 100 proteínas

A Tabela 3.15 mostra os resultados dos testes com um mapa de 100 proteínas representadas em uma esfera de $\mathbb{R}^{12}$ (problema 3.14). Esta tabela foi dividida em duas partes: fase de construção do mapa e fase de inclusão das proteínas. Em ambas as partes, as colunas *niter*, *avalf* e $f(x^*)$ indicam, respectivamente, o número de iterações, avaliações de função e valor final de função objetivo atingidos pela rotina *GENCAN* na resolução dos problemas. Na fase de construção do mapa, a coluna n_{I_2} indica o número de escores que foram efetivamente ajustados aos produtos escalares e *ndim* é a dimensão do espaço de vetores adotado para representar as proteínas. Nesta fase, a coluna $t(s)$ indica o tempo que *GENCAN* levou para construir o mapa. Já na fase de inclusão, a coluna A mostra as siglas das proteínas que foram incluídas no mapa e a coluna n_A mostra a quantidade de proteínas representadas que foram selecionadas pelo algoritmo de clusterização e busca. Em todos os testes o critério de parada de *GENCAN* foi o da norma do gradiente menor que a tolerância de 10^{-4}. Os

tempos despendidos na fase de inclusão, indicados na coluna $t(s)$, são a soma dos tempos gastos pelas rotinas *LOVOALIGN* e *GENCAN*. É possível observar também que nesta fase o número de iterações e avaliações de função são pequenos. Além disso, os valores finais de função são bastante razoáveis.

Fase de construção do mapa					
n_{I_2}	ndim	niter	avalf	$t(s)$	$f(x^*)$
123	12	41	198	8.300519	2.127420E−15

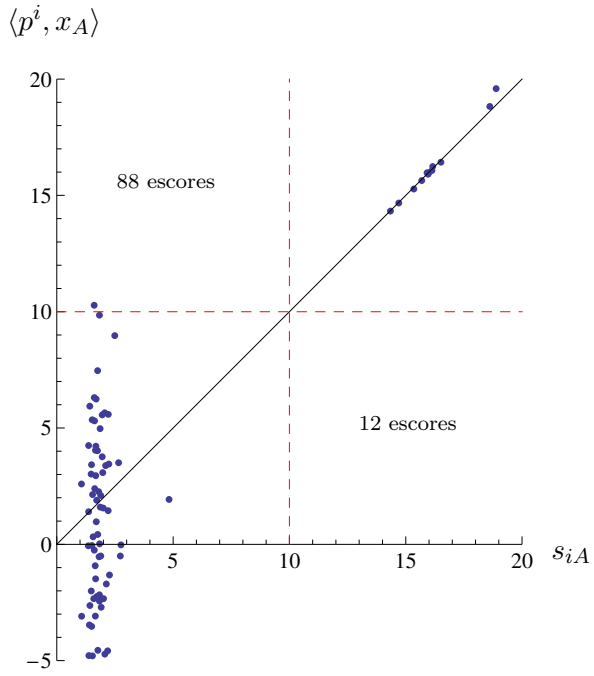
Fase de inclusão das proteínas					
A	n_A	niter	avalf	$t(s)$	$f(x^*)$
`1h4w`	12	12	13	0.936031	6.970889E−01
`1fiw`	12	12	13	1.185300	5.914756E+00
`2cst`	2	9	10	2.197715	2.126278E−13

Tabela 3.15: Experimentos de inclusão em um mapa com 100 proteínas.

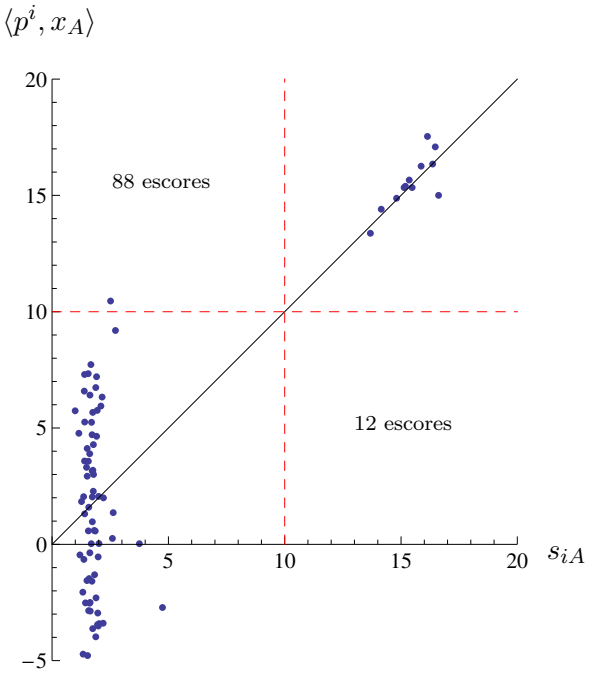
A Figura 3.7 ilustra os resultados numéricos obtidos na resolução do problema de inclusão das proteínas `1h4w`, `1fiw` e `2cst` no mapa com 100 proteínas comparadas. O gráfico da Figura 3.7 a) corresponde à inclusão de `1h4w` utilizando as coordenadas de $n_A = 12$ proteínas representadas. No eixo x deste gráfico estão os escores *Structal* normalizados verdadeiros (s_{iA}) obtidos pela comparação de `1h4w` com todas as 100 proteínas representadas no mapa. E, no eixo y, estão os produtos escalares entre x_A, vetor que representa `1h4w`, e os vetores p^i que representam as proteínas. A Figura 3.7 b) ilustra o experimento em que `1fiw` é incluída no mapa utilizando as coordenadas de 12 proteínas representadas. E, por fim, a Figura 3.7 c) ilustra o experimento em que `2cst` é incluída no mapa utilizando apenas duas proteínas representadas. Notemos nas figuras que os pares ordenados $(s_{iA}, \langle p^i, x_A \rangle)$ ocupam duas regiões bem definidas do plano cartesiano, indicando que as proteínas foram posicionadas razoavelmente no mapa. Na parte superior esquerda de cada gráfico estão os produtos escalares correspondentes aos escores menores que $s_{\text{fix}} = 10$ e, na parte inferior direita, estão os produtos escalares referentes aos escores mais significativos. Podemos observar também nas Figuras 3.7 a) e 3.7 b) que para $s_{iA} > 10$ os pontos tendem a se aglomerar em torno da reta $y = x$.

Experimentos em um mapa com 200 proteínas

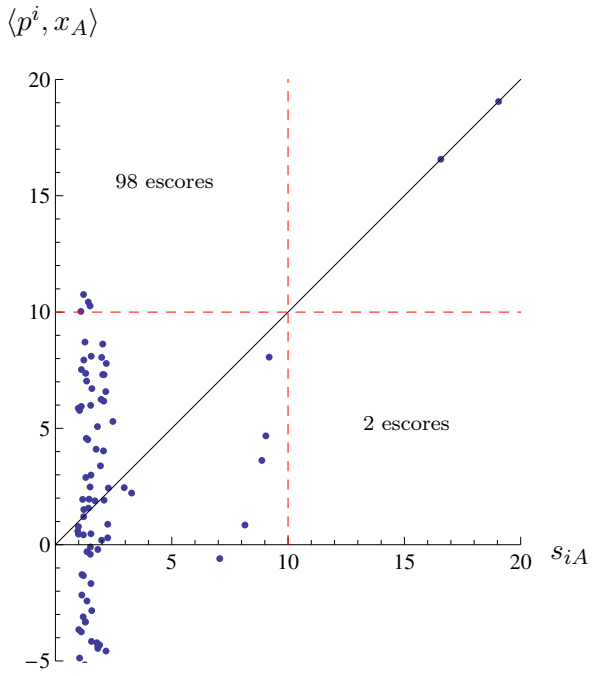
Nos mesmos moldes dos experimentos anteriores, realizamos alguns testes de inclusão em um mapa contendo 200 proteínas comparadas. Construímos uma representação para o conjunto de proteínas em um espaço de vetores de dimensão 20 e adotamos como escore fixo o valor $s_{\text{fix}} = 10$. Quatro proteínas não representadas foram selecionadas para serem adicionadas ao mapa. Os resultados numéricos tanto da fase de construção do mapa quanto da fase de inclusão de proteínas estão indicados na Tabela 3.16, cujas colunas seguem a mesma legenda



a) $A =$ `1h4w` e $n_A = 12$



b) $A =$ `1fiw` e $n_A = 12$



c) $A =$ `2cst` e $n_A = 2$

Figura 3.7: Incluindo `1h4w`, `1fiw` e `2cst` em um mapa com 100 proteínas.

definida na Tabela 3.15. O algoritmo de clusterização e busca selecionou novamente proteínas que foram utilizadas para resolver o problema (3.15). O critério de parada apresentado por *GENCAN* em todos os testes foi o da norma do gradiente menor que a tolerância de 10^{-4}. Analisando a Tabela 3.16 podemos notar que na fase de construção do mapa 427 escores

Fase de construção do mapa					
n_{I_2}	ndim	niter	avalf	$t(s)$	$f(x^*)$
427	20	81	479	155.71373	5.109079E−10

Fase de inclusão das proteínas					
A	n_A	niter	avalf	$t(s)$	$f(x^*)$
`1npm`	20	9	10	1.608511	1.101756E+00
`1trn`	20	11	12	1.616486	4.024869E−01
`1ykt`	20	9	10	3.597643	2.978858E+00
`1fy8`	20	10	11	3.474188	1.845143E+00

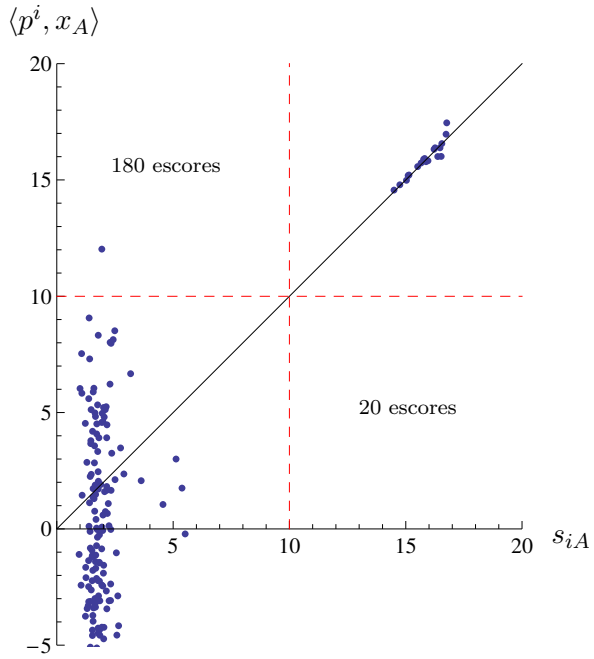
Tabela 3.16: Experimentos de inclusão em um mapa com 200 proteínas.

maiores que s_{fix} foram ajustados efetivamente. Como o valor final de função objetivo nesta fase é muito pequeno, a configuração final de vetores obtida é bastante satisfatória. Já na fase de inclusão de proteínas, o comportamento da rotina *GENCAN* foi muito semelhante ao obtido nos testes realizados com o mapa de 100 proteínas. Nesta fase, os tempos de execução de cada teste foram muito pequenos (observe a coluna $t(s)$). Notemos também que *GENCAN* realizou poucas iterações e avaliações de função até encontrar um minimizador para o problema.

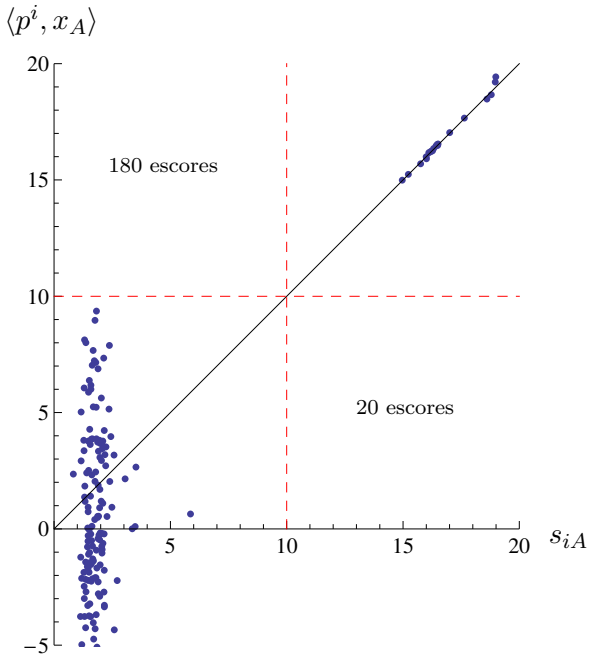
Assim como nos experimentos anteriores, a qualidade dos resultados obtidos foi avaliada mediante a construção de gráficos como os ilustrados na Figura 3.8. Notemos nos gráficos que a grande maioria dos pontos ocupa duas regiões distintas do plano previstas em nossos modelos. Os pontos que ficam à esquerda da reta vertical $s_{iA} = 10$ correspondem às proteínas que não são parecidas com a proteína incluída. Já os pontos que estão à direita da reta são referentes às proteínas mais semelhantes. Notemos neste último caso que os pontos tendem a ficar em torno da reta $y = x$, indicando que a proteína nova "A" está próxima das proteínas que possuem maior grau de semelhança com ela.

Tentativas para melhorar a qualidade dos ajustes

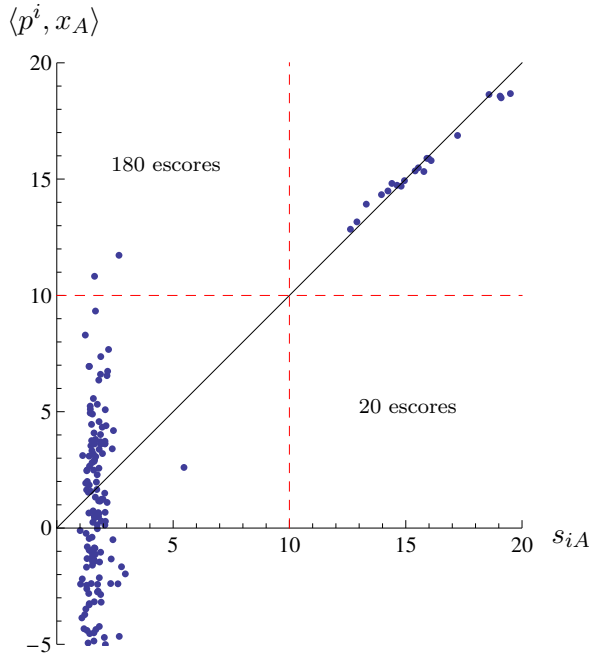
Nos experimentos com mapas na esfera, apesar de conseguirmos resolver muito rapidamente os problemas de inclusão de proteínas, percebemos que as estimativas para os escores *Structal* normalizados ($\langle p^i, x_A \rangle$) deixam um pouco a desejar. Repare, por exemplo, na Figura 3.8 a) que ilustra o teste de inclusão da proteína $A =$ `1npm` no mapa com 200 proteínas representadas. Neste gráfico, é possível notar claramente um par ordenado $(s_{iA}, \langle p^i, x_A \rangle)$ tal que $s_{iA} < 5$ e $\langle p^i, x_A \rangle > 10$. Como o escore *Structal* normalizado entre a proteína i e a



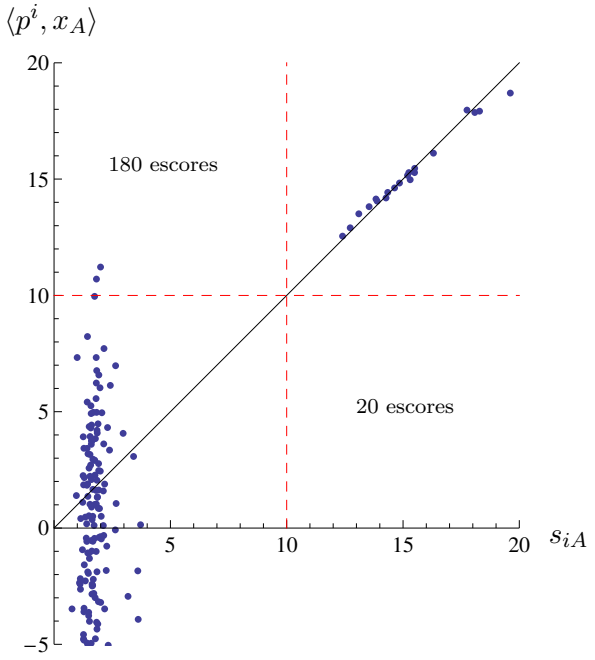
a) $A =$ `1npm` e $n_A = 20$



b) $A =$ `1trn` e $n_A = 20$



c) $A =$ `1ykt` e $n_A = 20$



d) $A =$ `1fy8` e $n_A = 20$

Figura 3.8: Incluindo `1npm`, `1trn`, `1ykt` e `1fy8` em um mapa com 200 proteínas.

proteína `1npm` (s_{iA}) é menor que 5, o escore estimado ($\langle p^i, x_A \rangle$) deveria assumir também um valor pequeno. Mostraremos agora, através de experimentos numéricos, que as estimativas para os escores tornam-se melhores quando aumentamos a dimensão do espaço de vetores. Para isso, retomaremos o problema de ajustar vetores na esfera, mas pediremos que todos os produtos escalares sejam ajustados aos escores efetivamente, isto é, construiremos mapas de vetores em $\mathbb{R}^m$ onde

$$\langle p^i, p^j \rangle \approx s_{ij} \quad \text{para todo } 1 \leq i < j < n,$$

e

$$\|p^i\| = \sqrt{20} \quad \text{para todo } i = 1, 2, \ldots, n,$$

e, em seguida, utilizaremos estes mapas para resolver os problemas de inclusão de proteínas. Lembremos que no início deste capítulo concentramos nossos esforços em resolver um problema muito semelhante a esse (problema 3.3 – Modelo 3.1.1). Naquele momento, pudemos constatar que a qualidade dos ajustes melhorava à medida que a dimensão do espaço de vetores também crescia.

Consideraremos agora espaços euclidianos $\mathbb{R}^m$ de dimensão maior e construiremos mapas de proteínas resolvendo o seguinte problema de otimização:

$$\begin{aligned} \min \quad & \sum_{i<j} \left(\langle p^i, p^j \rangle - s_{ij} \right)^2 + \rho \sum_{i=1}^{n} (\|p^i\|^2 - 20)^2, \\ \text{s. a} \quad & p^i \in \mathbb{R}^m \text{ para todo } i = 1, 2, \ldots, n. \end{aligned} \tag{3.18}$$

No problema (3.18), o primeiro somatório na função objetivo obriga os produtos escalares a se ajustarem aos escores entre os pares de proteínas e o segundo somatório pede que os vetores tenham norma $\sqrt{20}$. Neste último caso, $\rho > 0$ é um parâmetro de penalização fixo. Uma vez construída a representação para um conjunto de proteínas, a inclusão de uma proteína nova se dará pela resolução do problema (3.15) com o parâmetro de penalização ρ multiplicando o termo $(\|x_A\|^2 - 20)$.

As proteínas utilizadas nas construções dos mapas foram as mesmas que figuraram nos experimentos anteriores (conjuntos de 100 e 200 proteínas). Consideramos a dimensão dos espaços de vetores igual ao número de proteínas representadas. Os resultados dos experimentos estão indicados na Tabela 3.17. O valor do parâmetro de penalização utilizado na construção de cada mapa está indicado na coluna ρ e os desempenhos de *GENCAN* estão indicados nas colunas que têm a mesma legenda das tabelas anteriores. O critério de parada de *GENCAN* em todos os problemas foi o da norma do gradiente menor que a tolerância de 10^{-4}. As Figuras 3.9 e 3.10 ilustram, respectivamente, os resultados de inclusão nos mapas contendo 100 e 200 proteínas. Notemos que as estimativas ($\langle p^i, x_A \rangle$) para os escores *Structal* normalizados são melhores. Com efeito, o aumento da dimensão do espaço de vetores favorece os ajustes.

Mapa com 100 proteínas

Fase de construção					
ρ	ndim	niter	avalf	$t(s)$	$f(x^*)$
10^2	100	27	65	2.076399E+01	9.502828E+01

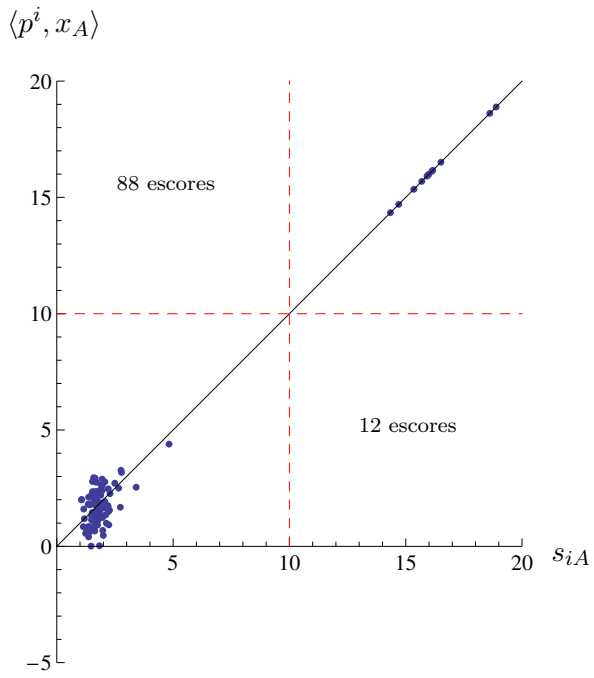
Fase de inclusão					
A	n_A	niter	avalf	$t(s)$	$f(x^*)$
`1h4w`	12	20	23	0.972824	2.070404E−17
`1fiw`	12	16	19	1.199057	9.932828E−13
`2cst`	2	21	26	2.203775	3.530633E−13

Mapa com 200 proteínas

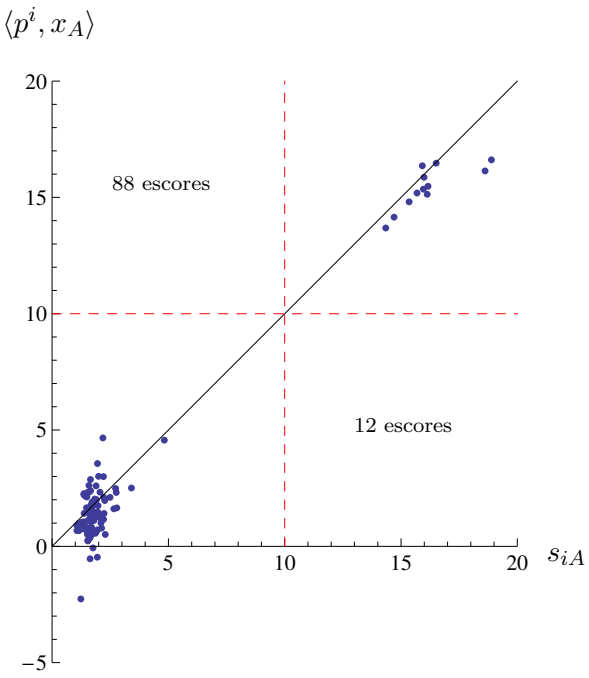
Fase de construção					
ρ	ndim	niter	avalf	$t(s)$	$f(x^*)$
10^3	200	46	93	2.999589E+02	1.200169E+03

Fase de inclusão					
A	n_A	niter	avalf	$t(s)$	$f(x^*)$
`1npm`	20	39	52	1.810288	2.222490E−16
`1trn`	20	35	45	1.774565	2.042827E−16
`1fy8`	20	62	92	3.514374	1.577072E+00
`1ykt`	20	48	66	3.683169	3.218125E+00

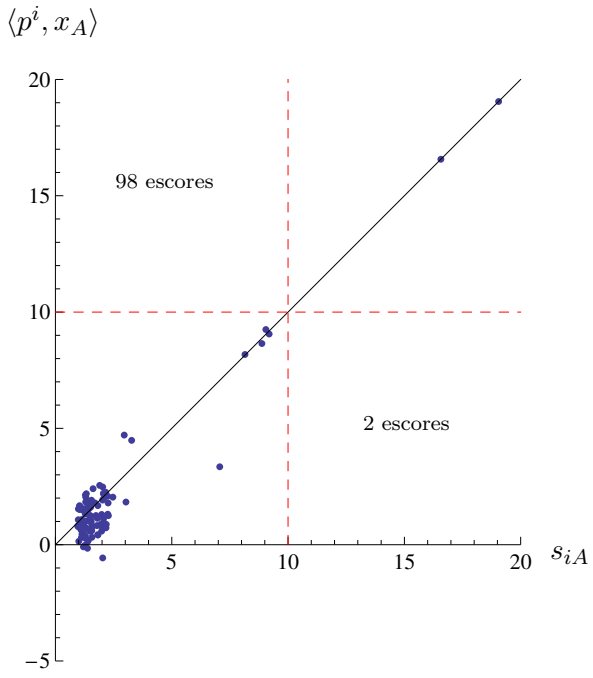
Tabela 3.17: Resultados de inclusão em mapas com dimensão maior.



a) $A =$ `1h4w` e $n_A = 12$

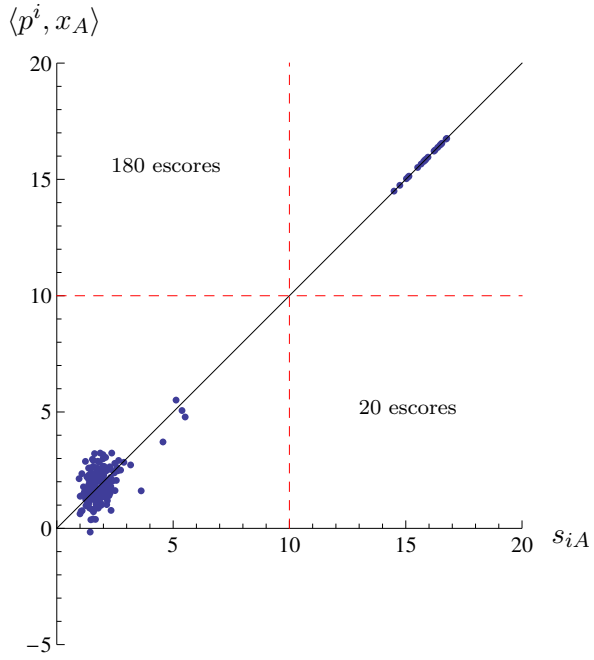


b) $A =$ `1fiw` e $n_A = 12$

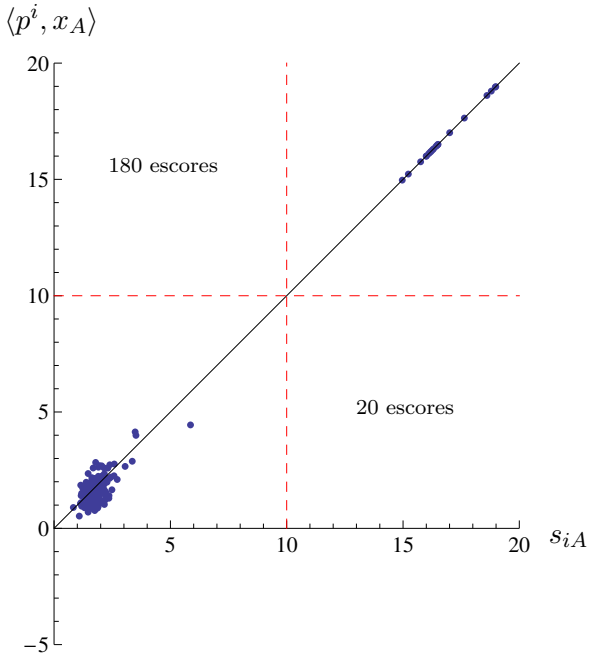


c) $A =$ `2cst` e $n_A = 2$

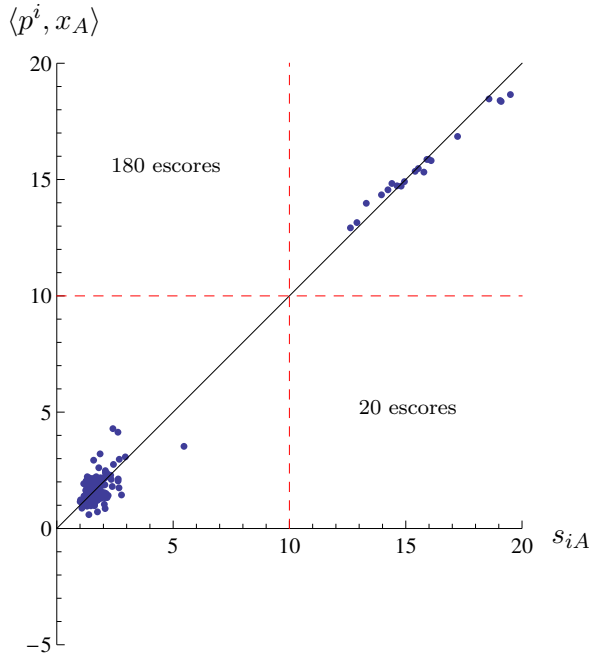
Figura 3.9: Inclusões em um mapa com 100 proteínas.



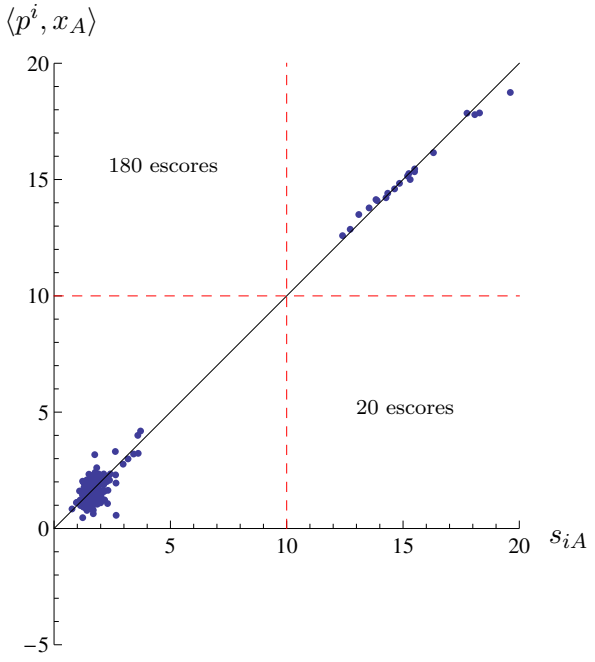
a) $A =$ `1npm` e $n_A = 20$



b) $A =$ `1trn` e $n_A = 20$



c) $A =$ `1ykt` e $n_A = 20$



d) $A =$ `1fy8` e $n_A = 20$

Figura 3.10: Inclusões em um mapa com 200 proteínas.

Considerações finais sobre o capítulo

Neste capítulo propusemos três modelos para a construção de mapas de proteínas comparadas. Em todos os modelos as proteínas eram representadas por vetores em um espaço $\mathbb{R}^m$ de tal forma que os produtos escalares entre pares desses vetores ficassem ajustados aos escores *Structal* normalizados das proteínas correspondentes. No Modelo 3.1.1 tentamos ajustar todos os escores do conjunto s. Percebemos pelos resultados dos experimentos que precisávamos aumentar a dimensão do espaço de vetores para conseguir ajustes razoáveis. Nestes testes, os melhores resultados foram obtidos em espaços com dimensão igual ao número de proteínas comparadas. Criamos em seguida o Modelo 3.1.2, onde aumentamos os graus de liberdade do problema ajustando os escores a uma função dos produtos escalares. Os testes com esse modelo não foram muito animadores pois, mesmo com o aumento dos graus de liberdade do problema obtivemos ajustes pouco satisfatórios. Por fim, no Modelo 3.1.3 decidimos ajustar apenas os escores mais significantes entre pares de proteínas. Este último modelo teve duas variantes, sendo a mais promissora aquela em que construímos mapas na esfera. As restrições de norma constante impostas ao Modelo 3.1.3 através de penalizações na função objetivo foram facilmente obedecidas nos testes realizados. Tentamos utilizar estes mapas para classificar novas proteínas evitando compará-las com todas as proteínas representadas. Os resultados numéricos obtidos nos problemas de inclusão nos mostraram que a proteína nova ficou razoavelmente próxima de proteínas parecidas. No entanto, as estimativas dos escores *Structal* normalizados entre a proteína incluída e as demais proteínas representadas não foram muito boas. Contornamos esta dificuldade construindo mapas na esfera impondo que a dimensão do espaço fosse igual ao número de proteínas representadas e todos os escores fossem ajustados. Os resultados de inclusão de proteínas nestes mapas foram melhores, como pudemos observar nos gráficos construídos.

Abandonaremos a partir de agora modelos que procuram ajustar distâncias e produtos escalares. No próximo capítulo investiremos em um modelo que tentará melhorar a qualidade dos ajustes sem aumentar muito a dimensão do espaço onde os mapas são construídos. Para isso, representaremos cada proteína por um conjunto de pontos em $\mathbb{R}^3$ e ajustaremos conjuntos de pontos distintos empregando uma fórmula semelhante à fórmula do escore *Structal* (1.1) definida no Capítulo 1. O problema de minimização que surge dessa formulação será resolvido com base na teoria da otimização do menor valor ordenado.

Capítulo 4

Mapas como problemas *LOVO*

Nos capítulos anteriores propusemos modelos para construir representações de conjuntos de proteínas onde ajustávamos diretamente distâncias entre pares de objetos (círculos, segmentos de reta) e produtos escalares entre pares de vetores. Nos modelos com ajustes de distâncias constatamos que o sucesso dos experimentos numéricos dependia fortemente das hipóteses do Teorema 2.1. Já nos modelos com ajustes de produtos escalares, percebemos a necessidade de aumentar consideravelmente a dimensão do espaço de vetores para conseguirmos bons resultados.

Neste capítulo propomos um modelo para a construção de mapas onde cada proteína é representada por um conjunto de pontos em $\mathbb{R}^3$. Uma fórmula semelhante à formula do escore *Structal* (1.1) definida no Capítulo 1 é então empregada para ajustar conjuntos distintos de pontos no espaço. Formulamos esta situação como um problema de otimização do menor valor ordenado e mostramos através de experimentos computacionais que os ajustes obtidos são bastante razoáveis. Em seguida, utilizamos o mesmo modelo para resolver o problema da inclusão de proteínas novas em mapas. Realizamos testes numéricos e contrastamos os resultados obtidos com resultados do capítulo anterior. Antes de descrevermos em detalhes o modelo proposto, faremos brevemente uma introdução sobre a teoria de problemas de otimização do menor valor ordenado.

4.1 Otimização do menor valor ordenado

Seja $I = \{1, 2, \ldots, m\}$ e considere o conjunto de funções $\{f_i\}_{i \in I}$, onde $f_i : \Omega \longrightarrow \mathbb{R}$ é contínuamente diferenciável em $\Omega \subseteq \mathbb{R}^n$ para todo $i \in I$. O problema de otimização do menor valor ordenado, também conhecido na literatura por problema *LOVO* (do inglês *Low Order Value Optimization*), é dado por

$$\begin{aligned} \min \quad & f_{\min}(x), \\ \text{s. a} \quad & x \in \Omega, \end{aligned} \tag{4.1}$$

onde

$$f_{\min}(x) = \min\{f_1(x), f_2(x), \ldots, f_m(x)\}. \tag{4.2}$$

A função (4.2) é diferenciável por partes pois cada f_i, com $i \in I$, é continuamente diferenciável em Ω. O exemplo a seguir retirado de [91] mostra um problema *LOVO* em uma variável.

Exemplo: Seja $\Omega = [-1, 1] \subset \mathbb{R}$ e considere as funções de valor real dadas por

$$\begin{aligned} f_1(x) &= -2x^3 + 2x^2 + x + 2, \\ f_2(x) &= x + 3, \\ f_3(x) &= 2x^2 + 2, \end{aligned}$$

para todo $x \in \Omega$. A Figura 4.1 ilustra os gráficos das funções f_1, f_2 e f_3 juntamente com gráfico de $f_{\min}(x)$. Notemos no gráfico de $f_{\min}(x)$ a existência de pontos onde a primeira derivada não está definida e pontos com primeira derivada nula.

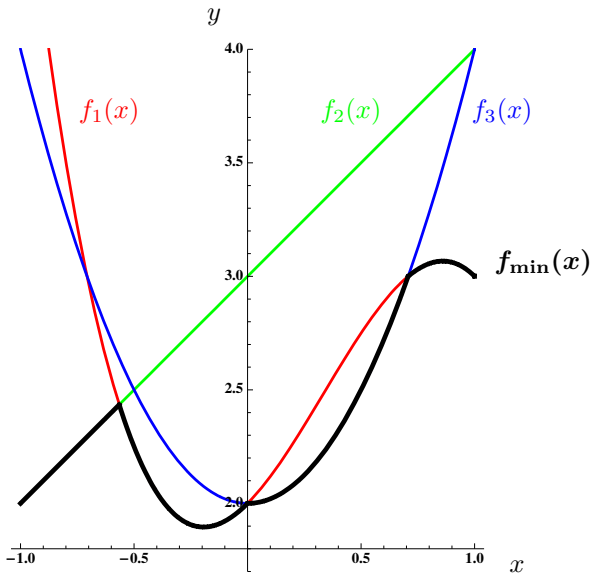


Figura 4.1: Exemplo de problema *LOVO* em uma variável.

Para resolver o problema (4.1) numericamente não poderíamos, em princípio, utilizar métodos de otimização contínua, dado que $f_{\min}(x)$ é diferenciável por partes em Ω. No entanto, F. S. Yano mostra em sua tese de doutorado [91] que as consequências da aplicação de tais métodos a problemas *LOVO* são menos severas. Em resumo, Yano prova que algoritmos para minimização irrestrita aplicados ao problema (4.1) com $\Omega = \mathbb{R}^n$ convergem para pontos com primeira derivada nula. Sendo assim, é possível utilizar qualquer algoritmo de otimização diferenciável para resolver o problema (4.1) numericamente. Em nossos experimentos utilizaremos a rotina *GENCAN*. Para isso, dado x_k em cada iteração de *GENCAN*, devemos fornecer $f_{\min}(x_k)$ e calcular $\nabla f_{\min}(x_k)$ como o gradiente da função $f_i(x_k)$ que realiza o mínimo.

Segundo Martínez *et al.* em [7, 8], o alinhamento estrutural de proteínas, discutido no Capítulo 1, é um exemplo de situação que pode ser modelada como um problema (4.1) sem restrições no domínio das variáveis, ou seja, $\Omega = \mathbb{R}^n$. Este é o princípio básico da rotina *LOVOALIGN*, amplamente utilizada neste trabalho para comparar as estruturas tridimensionais das proteínas. Mais aplicações da teoria *LOVO* podem ser encontradas em [9, 69, 91] e detalhes sobre o funcionamento de *LOVOALIGN* podem ser obtidos diretamente no endereço `www.ime.unicamp.br/~martinez/lovoalign`. A seguir descrevemos o novo modelo para a construção de mapas de proteínas comparadas onde utilizamos a filosofia *LOVO*.

4.2 Construção de mapas via formulação *LOVO*

Consideremos novamente um conjunto de n proteínas comparadas em relação às suas estruturas tridimensionais. Vamos denotar esse conjunto por $C_{\text{prot}} = \{A_1, A_2, \ldots, A_n\}$. Sejam

$$I_{\text{prot}} = \{1, 2, \ldots, n\}$$

o conjunto de índices associados aos elementos de C_{prot} e

$$s_n = \{s_{ij} \in (0, 20] \mid i, j \in I_{\text{prot}}\}$$

os escores *Structal* normalizados resultantes da comparação das n proteínas de C_{prot} duas a duas. No modelo que propomos agora cada proteína $A_i \in C_{\text{prot}}$ é representada por um conjunto de n_{p_i} pontos

$$P_i = \{x_1^i, x_2^i, \ldots, x_{n_{p_i}}^i\} \;\text{ com }\; x_k^i \in \mathbb{R}^3 \;\; \forall\, k = 1, 2, \ldots, n_{p_i}.$$

Vamos impor que a quantidade de pontos contida em cada conjunto P_i seja proporcional ao número de átomos de carbono alfa (N_{C_i}) presentes na estrutura de A_i. Em nossos experimentos adotaremos

$$n_{p_i} = \left\lceil \frac{N_{C_i}}{N_0} \right\rceil, \tag{4.3}$$

onde N_0 é um número natural maior ou igual a 1.

Suponha que duas estruturas P_i, P_j sejam conhecidas em $\mathbb{R}^3$. Definiremos o grau de semelhança entre elas como o maior escore *Structal* normalizado que pode ser calculado considerando todas as bijeções consecutivas sem intervalos entre os pontos de P_i e P_j (veja a Figura 4.2). Chamaremos este escore de "*gap-free*". Como n_{p_i} e n_{p_j} são os números de pontos de P_i e P_j respectivamente, o número total de bijeções consecutivas sem intervalos entre as duas estruturas é

$$nb_{ij} = |n_{p_i} - n_{p_j}| + 1, \tag{4.4}$$

e a fórmula para o cálculo do escore *gap-free* entre P_i e P_j será dada por

$$gf(P_i, P_j) = \frac{1}{n_{ij}} \sum_{k}^{n_{ij}} \frac{20}{1 + \|x_k^i - x_k^j\|^2/5}, \tag{4.5}$$

onde $n_{ij} = \min\{n_{p_i}, n_{p_j}\}$. Repare que a fórmula (4.5) difere da fórmula (1.1) apresentada no Capítulo 1 somente pela ausência do termo que considera o número de intervalos presentes nas bijeções entre as estruturas analisadas.

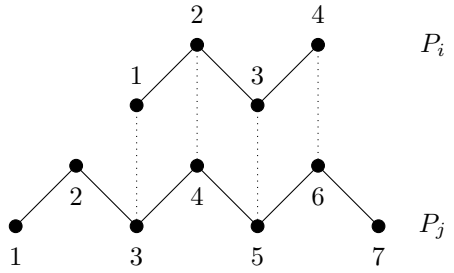


Figura 4.2: Bijeção consecutiva sem intervalos entre duas estruturas.

O cálculo do maior escore (4.5) entre duas estruturas P_i, P_j não é caro sob o ponto de vista computacional, pois precisamos determinar apenas nb_{ij} bijeções consecutivas entre os pontos dessas estruturas e selecionar a bijeção que maximiza o valor $gf(P_i, P_j)$. Vale ressaltar que o cálculo do verdadeiro escore *Structal* normalizado s_{ij} entre duas proteínas $A_i, A_j \in C_{\text{prot}}$ requer uma estratégia de programação dinâmica cujo número de operações cresce quadraticamente com o número de átomos das proteínas envolvidas [70, 72].

Em nosso novo modelo para a construção de mapas de proteínas vamos ajustar os escores *gap-free* (4.5) aos verdadeiros escores *Structal* normalizados s_{ij}. Para que este modelo se enquadre na filosofia *LOVO* apresentada anteriormente, devemos proceder da seguinte forma:

(1) Para cada par fixo de estruturas P_i e P_j em $\mathbb{R}^3$, encontramos a bijeção consecutiva sem intervalos que faz com que $gf(P_i, P_j)$ esteja o mais próximo possível do verdadeiro escore s_{ij} entre A_i e A_j.

(2) Fixadas as bijeções encontradas em (1), determinamos $P_{\text{prot}} = \{P_1, P_2, \ldots, P_n\} \subset \mathbb{R}^3$ resolvendo o problema

$$\begin{aligned} \min \quad & \sum_{i,j \ \in \ I_{\text{prot}}} (gf(P_i, P_j) - s_{ij})^2, \\ \text{s. a} \quad & P_i \subset \mathbb{R}^3, \ i = 1, 2, \ldots, n. \end{aligned} \tag{4.6}$$

O problema de minimização que acabamos de descrever é um problema do tipo (4.1) cujas variáveis são as coordenadas dos pontos que constituem as estruturas P_i. Portanto, o mapa que buscamos para o conjunto de proteínas C_{prot} será o conjunto $P_{\text{prot}} = \{P_1, P_2, \ldots, P_n\}$, solução do problema (4.6). Os experimentos numéricos que apresentamos a seguir foram realizados em um computador Apple com as seguintes especificações: sistema operacional MAC OS X 10.5, processador Intel Core 2 Duo com 2.4GHz e 2GB de memória.

4.2.1 Experimentos numéricos

Primeira bateria de testes

Nos experimentos que seguem construímos alguns mapas com conjuntos de proteínas retiradas do endereço `www.ime.unicamp.br/~martinez/lovoalign`. Os conjuntos que figuram nestes experimentos também foram utilizados para validar os modelos apresentados no Capítulo 3. Utilizamos *GENCAN* na resolução dos problemas (4.6) mas não empregamos nenhuma estratégia *multistart* para a obtenção dos resultados, pois os tempos de execução destes experimentos foram significativamente maiores que os testes realizados no capítulo anterior. Os resultados estão indicados na Tabela 4.1. Nos cinco primeiros experimentos (testes com 10, 25, 50, 75 e 100 proteínas), a quantidade de pontos utilizada na representação de cada proteína foi determinada pela fórmula (4.3) com $N_0 = 10$. Já no último experimento (teste com 200 proteínas) adotamos $N_0 = 50$. O número de variáveis de cada problema resolvido é dado pela soma da quantidade de pontos presentes em cada estrutura P_i do mapa multiplicada pela dimensão do espaço $\mathbb{R}^3$, ou seja,

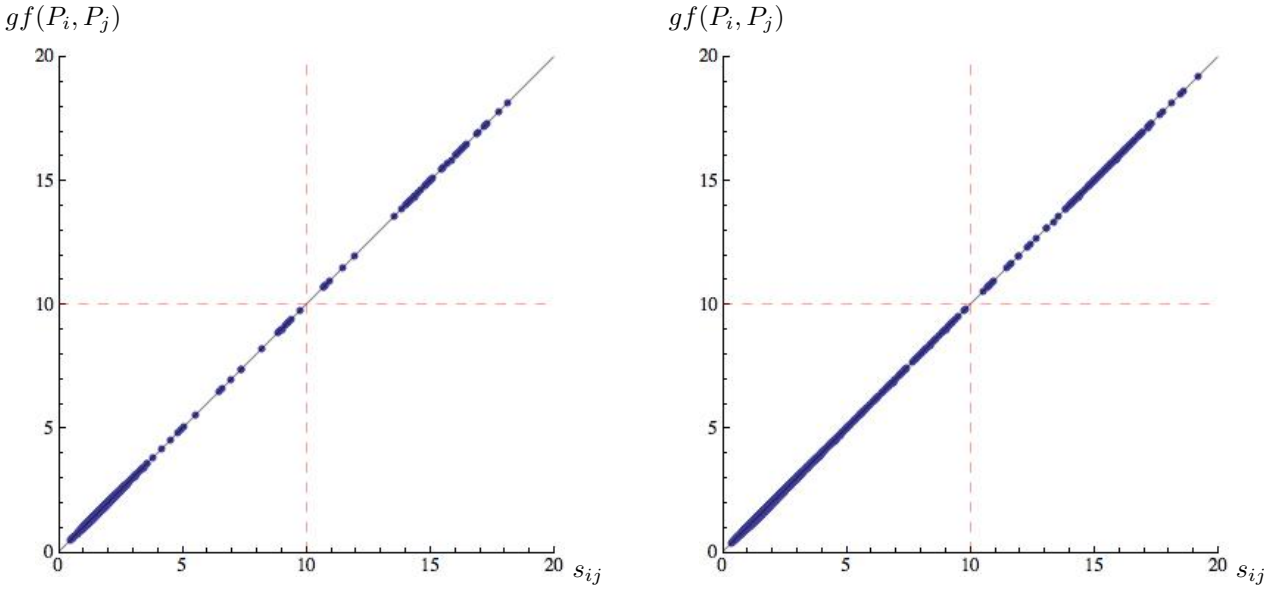
$$n_{\text{var}} = 3 \times \sum_{i=1}^{n} n_{p_i}.$$

As siglas *nprot*, n_{var}, *niter*, *avalf*, $f(x^*)$ e $t(s)$, presentes na Tabela 4.1 indicam, respectivamente, a quantidade de proteínas no mapa, o número total de variáveis do problema resolvido, o número total de iterações, o número total de avaliações de função, o valor final de função objetivo e o tempo de execução de *GENCAN* medido em segundos.

nprot	n_{var}	niter	avalf	$f(x^*)$	$t(s)$
10	1233	141	583	5.346547E−09	1.970210E+00
25	2751	229	836	1.067875E−07	3.588943E+01
50	5238	381	1863	2.361406E−07	2.216286E+03
75	8583	750	4080	1.221408E−07	1.248927E+04
100	12060	697	4003	3.250229E−07	2.719007E+04
200	4833	974	6340	9.887642E+03	4.441393E+04

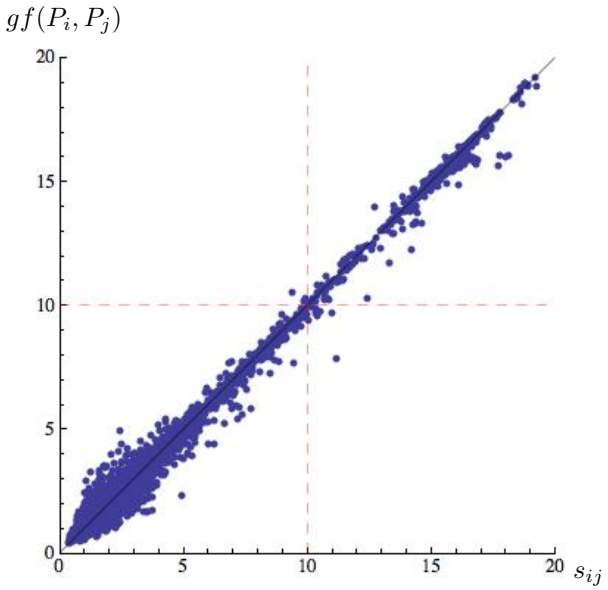
Tabela 4.1: Mapas via formulação *LOVO*: primeira bateria de testes.

Podemos notar pela tabela que os ajustes obtidos nos testes com mapas de 10 a 100 proteínas foram excelentes. Observe na coluna $f(x^*)$ os valores finais de função obtidos por *GENCAN* nestes experimentos. As Figuras 4.3 a) e 4.3 b) ilustram, respectivamente, os ajustes nos mapas com 50 e 100 proteínas. No eixo x de cada gráfico estão os valores verdadeiros dos escores *Structal* normalizados e, no eixo y estão os escores *gap-free* obtidos numericamente pela resolução do problema (4.6). No experimento realizado com o mapa de 200 proteínas, apesar do valor final de função não ser tão pequeno, consideramos os ajustes obtidos bastante razoáveis. Notemos a distribuição dos pontos deste último teste na Figura 4.4.



a) 50 proteínas: 1225 escores. b) 100 proteínas: 4950 escores.

Figura 4.3: Mapas com 50 e 100 proteínas: primeira bateria de testes.



200 proteínas: 19 900 escores.

Figura 4.4: Mapa com 200 proteínas: primeira bateria de testes.

Nos experimentos anteriores o critério de parada de *GENCAN* foi o da norma do gradiente menor que 10^{-4}. Pudemos observar também um aumento considerável no número de iterações, avaliações de função e tempo de execução despendidos pela rotina. Com efeito, exigimos mais do modelo pedindo que todos os escores *Structal* normalizados fossem ajustados efetivamente.

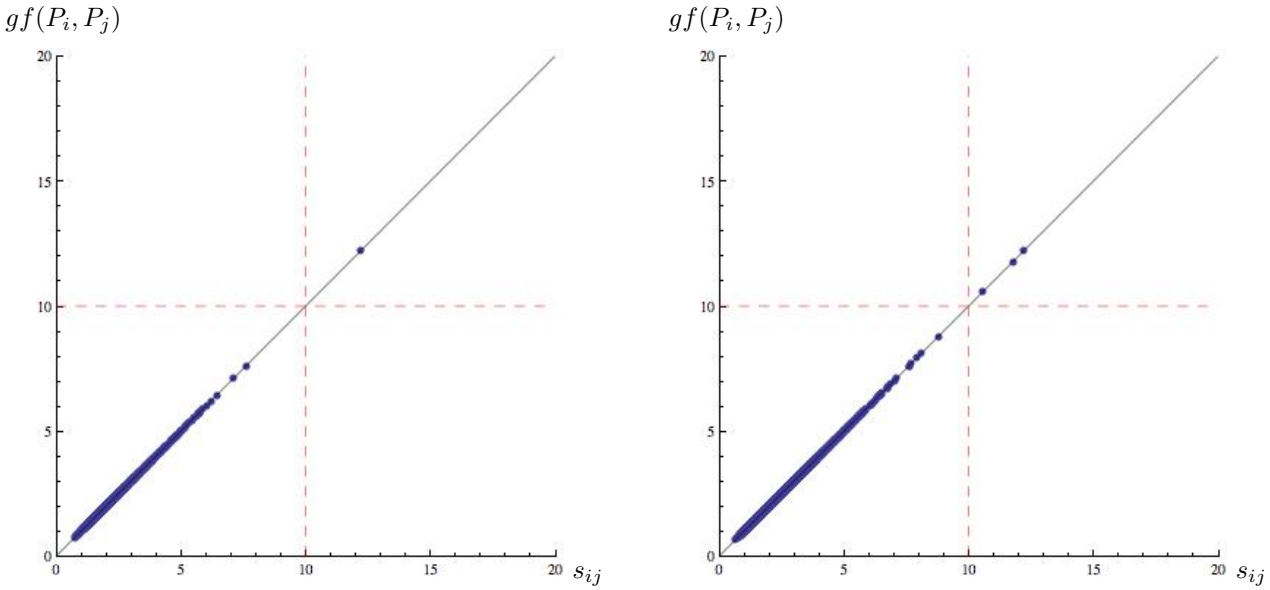
Segunda bateria de testes

Os resultados computacionais apresentados a seguir são referentes a conjuntos de proteínas retirados de `http://zhanglab.ccmb.med.umich.edu/TM-align` [96]. Segundo este endereço eletrônico, estas proteínas foram extraídas do *Protein Data Bank* e possuem menos de 30 % de sequências semelhantes de aminoácidos. Realizamos testes com 10, 25, 50, 75 e 100 proteínas e utilizamos *GENCAN* na resolução dos problemas (4.6) sem aplicar estratégias *multistart* para a obtenção dos resultados. O número de pontos presentes em cada representação de proteína foi determinado pela fórmula (4.3) com $N_0 = 10$. Os resultados obtidos nos experimentos estão indicados na Tabela 4.2. Da mesma forma que na Tabela 4.1, a coluna *nprot* indica a quantidade de proteínas presentes em cada mapa, a coluna n_{var} traz o número de variáveis de cada problema e as colunas *niter*, *avalf*, $f(x^*)$ e $t(s)$ são referentes ao desempenho de *GENCAN*.

nprot	n_{var}	niter	avalf	$f(x^*)$	$t(s)$
10	738	31	81	4.373831E−10	1.200999E+00
25	2034	70	267	7.475684E−09	4.841950E+02
50	4302	147	965	1.174324E−08	1.310406E+04
75	6639	167	917	2.051146E−08	2.584911E+04
100	8418	182	1045	1.403336E−07	3.909715E+04

Tabela 4.2: Mapas via formulação *LOVO*: segunda bateria de testes.

Podemos observar novamente a boa qualidade dos ajustes obtidos. Como podemos notar, os valores finais de função atingidos por *GENCAN* são bastante razoáveis. Nestes testes a rotina também parou com a norma do gradiente menor que a tolerância 10^{-4}. Assim como na primeira rodada de experimentos, fizemos gráficos do tipo escore verdadeiro × escore *gap-free* para ilustrar os resultados numéricos. A Figura 4.5 a) ilustra os ajustes obtidos com o mapa de 50 proteínas. Apenas um ponto nesta figura possui abscissa maior que 10. Isto indica que apenas um par de proteínas possui alguma semelhança estrutural significativa segundo nossa classificação. A Figura 4.5 b) mostra o resultado obtido com o mapa de 100 proteínas. Neste caso vemos que apenas três pontos possuem abscissa maior que 10. Apresentaremos a seguir mais uma rodada de experimentos envolvendo *globinas* retiradas do *Protein Data Bank*.



a) 50 proteínas: 1225 escores. b) 100 proteínas: 4950 escores.

Figura 4.5: Mapas com 50 e 100 proteínas: segunda bateria de testes.

Terceira bateria de testes

Nesta terceira rodada de experimentos realizamos em 19/02/2011 uma busca pelo termo *globins* (globinas) no *Protein Data Bank* e pedimos que o banco de dados excluísse dos resultados as proteínas com até 90 % de semelhança em suas sequências de aminoácidos. Como resultado dessa pesquisa, obtivemos 98 estruturas que foram comparadas pela rotina *LOVOALIGN* e utilizadas na construção de um mapa. O número de pontos para representar cada proteína foi determinado pela fórmula (4.3) com $N_0 = 40$. A rotina *GENCAN* foi novamente utilizada na resolução do problema (4.6), partindo de um único ponto inicial gerado aleatoriamente. Os resultados obtidos ao final deste experimento estão indicados na Tabela 4.3, cujas legendas das colunas têm o mesmo significado usado nas tabelas dos testes anteriores. Neste experimento o critério de parada de *GENCAN* também foi o da norma do gradiente menor que 10^{-4}. Notemos na tabela que a rotina de otimização realiza um número considerável de iterações e avaliações de função. No entanto, o ajuste obtido é razoável. Podemos comprovar este fato observando o gráfico ilustrado na Figura 4.6. Este gráfico foi construído da mesma forma que os outros mostrados neste capítulo.

nprot	n_{var}	niter	avalf	$f(x^*)$	$t(s)$
98	3708	2348	18717	3.830398E+02	4.414949E+05

Tabela 4.3: Mapa com 98 globinas: terceira bateria de testes.

Aplicaremos a seguir a mesma formulação *LOVO* para resolver o problema da inclusão

de novas proteínas em mapas. Lembremos que neste problema precisamos selecionar no conjunto total de proteínas representadas um subconjunto pequeno de proteínas. Faremos isso através do algoritmo de clusterização e busca proposto no Capítulo 3. Apresentaremos o modelo de inclusão de proteínas que segue a filosofia *LOVO*, realizaremos alguns experimentos numéricos e, por fim, discutiremos os resultados obtidos.

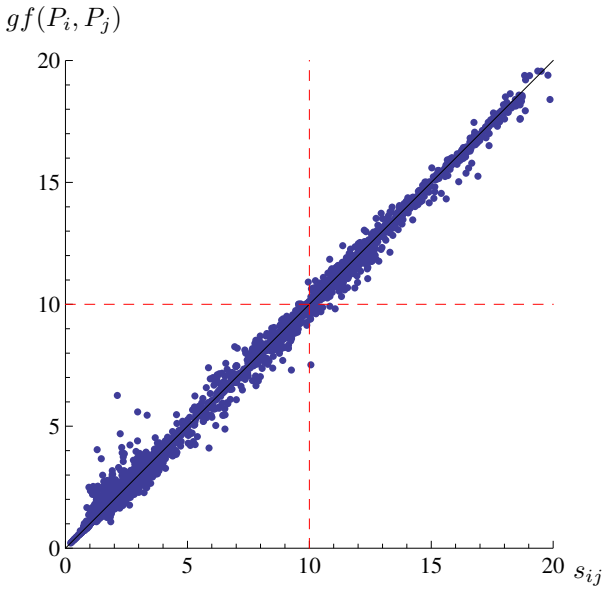


Figura 4.6: Mapa com 98 globinas: terceira bateria de testes.

4.3 Modelo *LOVO* para inclusão de proteínas

Nesta seção retomamos o problema da inclusão de proteínas novas em mapas. Seguimos a mesma linha de raciocínio desenvolvida na Seção 3.2.1 do Capítulo 3, mas agora nos baseamos na teoria *LOVO* para formular e resolver os problemas. Sejam $C_{\text{prot}} = \{A_1, A_2, \ldots, A_n\}$ um conjunto de n proteínas comparadas duas a duas e $P_{\text{prot}} = \{P_1, P_2, \ldots, P_n\} \subset \mathbb{R}^3$ uma representação para C_{prot} obtida através da resolução do problema (4.6). Seja $A \notin C_{\text{prot}}$ uma proteína nova e considere P_A uma representação para A. Nossa meta é determinar P_A em P_{prot} de tal forma que P_A fique próxima das representações P_i que correspondem a proteínas semelhantes. Os passos para cumprir essa tarefa são os seguintes:

(1) Selecionar um pequeno subconjunto $\tilde{P} \subset P_{\text{prot}}$.

(2) Comparar, via *LOVOALIGN*, a estrutura 3D de A com as estruturas 3D das proteínas que correspondem aos elementos de $\tilde{P}$ e obter os escores *Structal* normalizados s_{iA} para $i = 1, 2, \ldots, |\tilde{P}|$.

(3) Obter a representação P_A para a proteína A através da resolução do problema

$$\begin{aligned} \min \quad & \sum_{i=1}^{|\tilde{P}|} (\ gf(P_i, P_A) - s_{iA} \ )^2, \\ \text{s. a} \quad & P_A \subset \mathbb{R}^3, \end{aligned} \tag{4.7}$$

onde $gf(P_i, P_A)$ é o escore *gap-free* (4.5) entre as estruturas P_i e P_A.

Definido desta maneira, o problema (4.7) também é do tipo *LOVO* e suas variáveis são as coordenadas dos pontos que constituem o conjunto P_A. Após determinarmos a localização da proteína nova no mapa, podemos estimar os escores *Structal* normalizados entre A e as proteínas que não foram selecionadas para compor $\tilde{P}$ através do seguinte cálculo:

$$s_{jA} \approx gf(P_j, P_A), \text{ onde } P_j \in P_{\text{prot}}/\tilde{P}. \tag{4.8}$$

Os cálculos (4.8) são baratos sob o ponto de vista computacional, pois não envolvem nenhum tipo de alinhamento entre as estruturas que estão sendo comparadas. Basta determinarmos, para cada par de estruturas (P_j, P_A), a bijeção consecutiva que maximiza o valor de $gf(P_j, P_A)$. A seguir, realizaremos experimentos numéricos para validar a estratégia apresentada acima e, para compararmos os resultados obtidos, utilizaremos os mesmos conjuntos de proteínas empregados nos testes da seção 3.2.1 do Capítulo 3.

4.3.1 Experimentos para validação do modelo

Os experimentos para validação do modelo foram realizados com os mapas de 100 e 200 proteínas, construídos na primeira bateria de testes da seção 4.2. Selecionamos do endereço `www.ime.unicamp.br/~martinez/lovoalign` cinco proteínas que não foram representadas nestes mapas (as mesmas utilizadas nos testes da seção 3.2.1 do Capítulo 3). Cada teste de inclusão foi realizado isoladamente. Utilizamos o algoritmo de clusterização e busca para escolher em cada experimento o subconjunto $\tilde{P}$ de representações de proteínas. Então, usando essas representações, resolvemos o problema (4.7). Os resultados numéricos obtidos por *GENCAN* estão indicados na Tabela 4.4. Nesta tabela, a coluna A contém as siglas das proteínas incluídas nos mapas. A coluna $|\tilde{P}|$ indica o número de proteínas representadas no mapa que foram selecionadas pelo algoritmo de clusterização e busca. A coluna *nvar* mostra o número de variáveis de cada problema. Lembremos que este número é determinado pela fórmula (4.3). Esta fórmula foi utilizada com $N_0 = 10$ nos testes com o mapa de 100 proteínas e $N_0 = 50$ nos testes com o mapa de 200 proteínas. Por fim, as colunas *niter*, *avalf*, $t(s)$ e $f(x^*)$ são referentes ao desempenho da rotina *GENCAN*. O critério de parada em todos os testes foi o da norma do gradiente menor que a tolerância 10^{-4}. Como podemos perceber na Tabela 4.4, a rotina *GENCAN* resolveu todos os problemas rapidamente e encontrou valores finais de função objetivo bastante razoáveis.

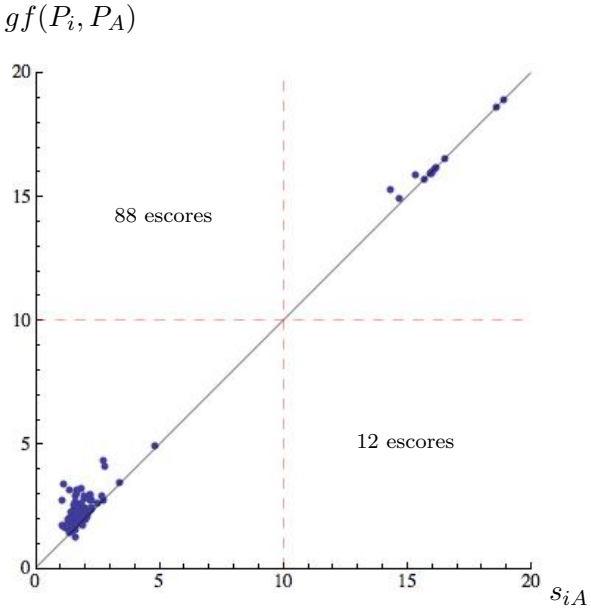
Mapa com 100 proteínas						
A	$\lvert\tilde{P}\rvert$	nvar	niter	avalf	$t(s)$	$f(x^*)$
`1h4w`	12	72	148	568	1.078657	1.504719E−10
`1fiw`	12	78	163	618	1.336360	1.075625E−10
`2cst`	2	126	131	418	2.253739	8.814859E−08

Mapa com 200 proteínas						
A	$\lvert\tilde{P}\rvert$	nvar	niter	avalf	$t(s)$	$f(x^*)$
`1npm`	20	15	39	103	1.604618	4.448825E−01
`1trn`	20	15	26	91	1.607781	7.935456E−01

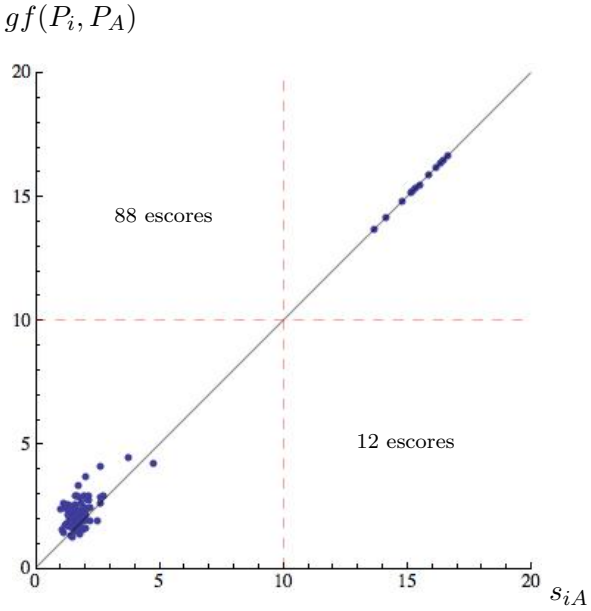
Tabela 4.4: Inclusão de proteínas em mapas via formulação *LOVO*.

As Figuras 4.7 a) e 4.7 b) ilustram, respectivamente, os resultados obtidos nos testes de inclusão das proteínas `1h4w` e `1fiw` no mapa com 100 proteínas. No eixo x de cada gráfico estão os verdadeiros escores *Structal* normalizados e, no eixo y, estão as estimativas para esses escores que foram calculadas de acordo com a fórmula (4.5). A reta vertical $s_{iA} = 10$ separa cada gráfico em duas regiões: a quantidade de escores maiores que 10 está indicada no canto inferior direito e a quantidade de escores que não ultrapassa 10 está indicada no canto superior esquerdo. Ilustramos nas Figuras 4.8 a) e 4.8 b) os resultados obtidos nos testes com as proteínas `1npm` e `1trn`, respectivamente. A reta vertical $s_{iA} = 10$ divide os gráficos em duas regiões e a quantidade de pontos em cada região está indicada nas figuras. Estes gráficos nos revelam que as proteínas novas foram posicionadas razoavelmente nos mapas, pois elas ficam próximas de proteínas semelhantes e afastadas de proteínas menos parecidas.

Neste capítulo, ao empregarmos a filosofia *LOVO* tanto na construção de mapas de proteínas quanto na classificação de proteínas novas, obtivemos resultados numéricos muito semelhantes aos que conseguimos através dos mapas construídos em esferas de $\mathbb{R}^m$. Por exemplo, ao comparar as Figuras 3.9 e 3.10 do Capítulo 3 com as Figuras 4.7 e 4.8 deste capítulo, podemos notar que nas figuras do Capítulo 3 os pares ordenados correspondentes aos escores menores que $s_{\text{fix}} = 10$ ficam mais concentrados em torno da reta $y = x$. Com efeito, nestes experimentos aumentamos consideravelmente a dimensão do espaço de vetores. No entanto, ao representarmos cada proteína por um conjunto de pontos em $\mathbb{R}^3$ e ajustarmos a esses conjuntos uma fórmula semelhante ao escore *Structal*, obtivemos resultados satisfatórios sem elevar muito o número de variáveis dos problemas resolvidos.

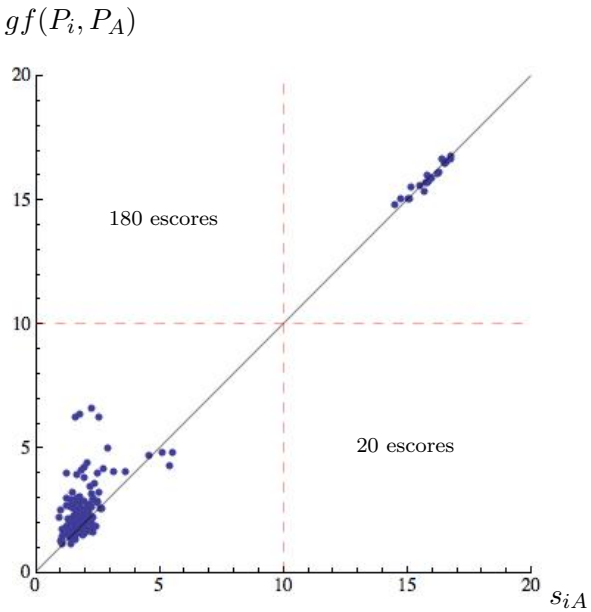


a) `1h4w` e $|\tilde{P}| = 12$

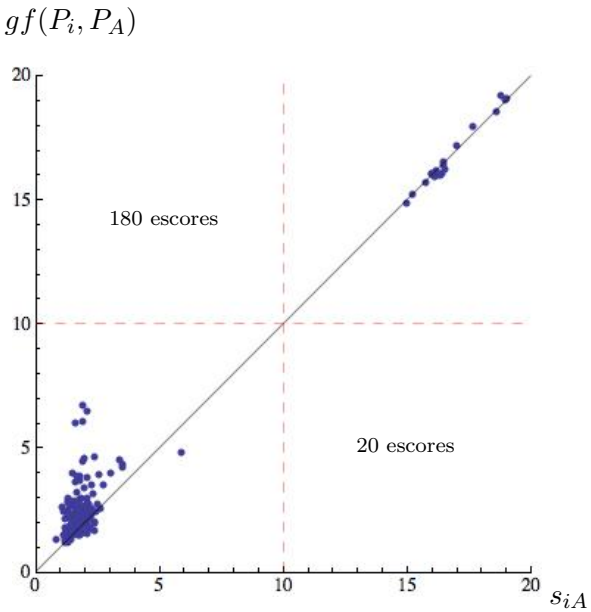


b) `1fiw` e $|\tilde{P}| = 12$

Figura 4.7: Inclusões em um mapa com 100 proteínas: formulação *LOVO*.



a) `1npm` e $|\tilde{P}| = 20$



a) `1trn` e $|\tilde{P}| = 20$

Figura 4.8: Inclusões em um mapa com 200 proteínas: formulação *LOVO*.

Capítulo 5

Conclusões

Nosso objetivo neste trabalho foi propor algumas maneiras de reconstruir e classificar estruturas tridimensionais de proteínas utilizando métodos de otimização contínua. Em particular, empregamos na resolução dos problemas a rotina *GENCAN* que integra o pacote de otimização numérica *ALGENCAN* disponível gratuitamente em `www.ime.usp.br/~egbirgin/tango`.

Começamos o trabalho estudando o Problema da Geometria de Distâncias Moleculares. Neste problema, tentamos recuperar a estrutura 3D de uma proteína admitindo conhecidas apenas as distâncias entre seus pares de átomos. Através de experimentos numéricos, conseguimos recuperar a estrutura original de várias proteínas utilizando somente um subconjunto de distâncias intra-átomos. Em alguns testes, observamos que a solução fornecida pela rotina *GENCAN* era a imagem espelhada da estrutura original, ou seja, as configurações original e numérica diferiam por um movimento rígido cuja matriz ortogonal era de reflexão. A explicação para este comportamento se deve ao fato de usarmos apenas informações escalares na construção dos modelos. Com efeito, na resolução dos problemas sabíamos de antemão apenas os valores da distâncias e não consideramos nenhuma informação sobre a orientação espacial das proteínas analisadas. As funções que propusemos minimizar para recuperar as estruturas das proteínas possuiam muitos minimizadores locais. Por essa razão, rodamos cada experimento diversas vezes fornecendo um ponto inicial diferente em cada rodada. A rotina *GENCAN* ora encontrava uma solução satisfatória, ora parava em um minimizador local que não nos interessava. Em trabalhos futuros, pretendemos agregar à rotina *GENCAN* uma estratégia de otimização global para facilitar a busca por minimizadores globais do problema da geometria de distâncias moleculares e acelerar o processo de recuperação das estruturas 3D das proteínas.

Após o estudo do problema da geometria de distâncias moleculares, começamos a investigar algumas maneiras de representar proteínas comparadas em espaços euclidianos. As estruturas das proteínas foram comparadas via rotina *LOVOALIGN* e o grau de semelhança entre cada par de proteínas analisado foi medido pelo escore *Structal*, definido no Capítulo 1. Utilizamos os escores *Structal* resultantes da comparação das estruturas 3D de um conjunto de n proteínas para estipular disparidades entre elas. Criamos um conjunto Δ de

números reais não negativos $\hat{d}_{ij}$ que eram inversamente proporcionais aos escores s_{ij}. Então, construímos modelos de otimização não linear em que as n proteínas eram representadas por objetos que deveriam ser alocados em um determinado espaço euclidiano de tal forma que a distância entre um par de objetos i e j estivesse tão próxima quanto possível do correspondente elemento $\hat{d}_{ij}$ do conjunto Δ.

Vimos através de um teorema uma condição necessária e suficiente para que o conjunto Δ fosse realizável por uma configuração de pontos em um espaço euclidiano. Através de exemplos pudemos perceber as dificuldades que surgiam ao tentarmos representar proteínas por pontos. Baseados nestes exemplos, decidimos investigar outras formas geométricas para representar as proteínas. Fizemos inicialmente experimentos em $\mathbb{R}^2$ onde as proteínas eram representadas por círculos cujos raios podiam variar em tamanho, mas não obtivemos resultados muito bons. Em seguida, partimos para experimentos em $\mathbb{R}^3$ onde construímos modelos em que as proteínas comparadas eram representadas por segmentos de reta e poligonais formadas por dois segmentos de comprimento fixo. Nestes modelos ajustamos somente os valores $\hat{d}_{ij}$ que correspondiam a proteínas razoavelmente semelhantes. Para isso, fixamos um valor positivo d_{fix} e ajustamos efetivamente todos os valores $\hat{d}_{ij}$ menores que d_{fix}. Nos testes computacionais que empregamos esta estratégia não conseguimos manter constante o valor do parâmetro d_{fix} quando a quantidade de proteínas envolvida era grande. Como consequência desse fato, não conseguimos ajustar efetivamente uma quantidade considerável de números $\hat{d}_{ij}$.

Na busca por outros tipos de representações, abandonamos os testes com ajustes de distâncias e passamos a ajustar diretamente os escores *Structal* normalizados. Representamos cada proteína por um vetor em $\mathbb{R}^m$ e construímos modelos que tentavam ajustar produtos escalares entre pares de vetores e escores, ou seja, nosso problema consistia em alocar vetores em um determinado espaço euclidiano de tal forma que o produto escalar entre os vetores i e j estivesse o mais próximo possível do escore s_{ij}. Mostramos, através de uma proposição, que esse problema sempre possui solução se a dimensão do espaço é igual ao número de proteínas consideradas. Propusemos três modelos para realizar experimentos numéricos. O modelo mais promissor foi aquele em que os vetores eram representados em uma esfera de $\mathbb{R}^m$. Utilizando este modelo construímos mapas exigindo que todos os escores s_{ij} fossem ajustados. Os mapas obtidos foram razoáveis e, por essa razão, os utilizamos para classificar proteínas novas.

O processo de classificação de uma proteína nova consistiu em aplicar a seguinte estratégia: dada uma proteína nova "A", escolhemos no mapa de proteínas um subconjunto de representações correspondentes às proteínas mais parecidas com "A". Utilizamos as representações escolhidas para determinar as novas coordenadas de "A" no mapa. Elaboramos um algoritmo para realizar esta tarefa. Dado o conjunto de proteínas comparadas C_{prot} e uma proteína nova $A \notin C_{\text{prot}}$, a ideia central de nosso algoritmo foi realizar buscas em C_{prot} procurando por proteínas que fossem estruturalmente semelhantes à proteína A. As comparações estruturais requeridas pelo algoritmo foram feitas pela rotina *LOVOALIGN*, no entanto, não comparamos a proteína nova com todas as proteínas de C_{prot}. Para evitar excessivas chamadas de *LOVOALIGN*, aplicamos em C_{prot} uma estratégia de clusterização para dividí-lo

em subconjuntos menores e realizamos buscas nestes subconjuntos. Aplicamos o algoritmo nos experimentos de inclusão de proteínas em mapas e constatamos que o algoritmo realizou poucas chamadas de *LOVOALIGN* para localizar em C_{prot} proteínas parecidas. Nos problemas de inclusão de proteínas em mapas na esfera, obtivemos resultados satisfatórios pois as proteínas novas ficaram razoavelmente próximas de proteínas semelhantes, além disso, a rotina *GENCAN* resolveu todos estes problemas muito rapidamente.

Para encerrar nosso trabalho, decidimos investir em um modelo que representa cada proteína por um conjunto de pontos em $\mathbb{R}^3$. Definimos como critério de semelhança entre os pares de proteínas representadas o escore *gap-free* e construímos um problema de otimização cujo objetivo era alocar os conjuntos de pontos no espaço tridimensional de tal forma que os escores *gap-free* estivessem tão próximos quanto possível dos verdadeiros escores *Structal* normalizados. Formulamos este problema como um problema de otimização do menor valor ordenado e realizamos experimentos computacionais. Os resultados foram bastante aceitáveis, pois conseguimos ajustar conjuntos de 100 e 200 proteínas comparadas em $\mathbb{R}^3$. A mesma formulação empregada na construção dos mapas foi utilizada para resolvermos os problemas de inclusão de proteínas novas. Os resultados obtidos nos testes também foram satisfatórios. Nos experimentos de inclusão de proteínas que realizamos com essa formulação, utilizamos também o algoritmo de clusterização e busca que elaboramos no Capítulo 3.

De um modo geral, para construir os mapas de proteínas em espaços euclidianos, tivemos que propor modelos com um número grande de variáveis. Esta necessidade ocorreu devido ao fato de trabalhamos com ajustes de números que não eram distâncias euclidianas. Sendo assim, era preciso aumentar a dimensão dos espaços para conseguirmos ajustes razoáveis. Apesar dos modelos apresentarem muitas variáveis, a rotina *GENCAN* resolveu grande parte dos problemas de maneira bastante satisfatória. Os mapas construídos nos Capítulos 3 e 4 serviram também para classificarmos proteínas que não estavam representadas. Nestes experimentos de inclusão de proteínas, conseguimos colocar a proteína nova junto de proteínas que eram semelhantes a ela.

Referências Bibliográficas

[1] D. K. Agrafiotis, D. Rassokhin, V. S. Lobanov, "Mutidimensional scaling and visualization of large molecular similarity tables", *Journal of Computational Chemistry*, vol. 22, pp. 408–500, 2001.

[2] A. Y. Alfakih, "On the eigenvalues of Euclidean distance matrices", *Computational & Applied Mathematics*, vol. 27, pp. 237–250, 2008.

[3] A. Y. Alfakih, A. Khandani, H. Wolkowicz, "Solving Euclidean Distance Matrix Completion Problem Via Semidefinite Programming", *Computational Optimization and Applications*, vol. 12, pp. 1–3, 1999.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, "Basic Local Alignment Search Tool", *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990.

[5] R. Andreani, E. G. Birgin, J. M. Martínez, M. L. Schuverdt, "On Augmented Lagrangian Methods with general lower-level constraints", *SIAM Journal on Optimization*, vol. 18, pp. 1286–1309, 2007.

[6] R. Andreani, E. G. Birgin, J. M. Martínez, M. L. Schuverdt, "Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification", *Mathematical Programming*, vol. 111, pp. 5 – 32, 2008.

[7] R. Andreani, J. M. Martínez, L. Martínez, F. Yano, "Continuous Optimization Methods for Structural Alignments", *Mathematical Programming*, vol. 112, pp. 93 – 124, 2008.

[8] R. Andreani, J. M. Martínez, L. Martínez, "Trust-region superposition methods for protein alignment", *IMA Journal of Numerical Analysis*, vol. 28, pp. 690–710, 2008.

[9] R. Andreani, J. M. Martínez, L. Martínez, F. S. Yano, "Low Order-Value Optimization and applications", *Journal of Global Optimization*, vol. 43, pp. 1–22, 2009.

[10] A. Andreeva, D. Howorth, J. Chandonia, S. E. Brenner, T. Hubbard, C. Chothia, A. Murzin, "Data growth and its impact on the SCOP database: new developments", *Nucleic Acids Research*, pp. 1–7, 2007.

[11] M. Bakonyi, C. R. Johnson, "The Euclidean distance matrix completion problem", *SIAM Journal on Matrix Analysis and Applications*, vol. 16, pp. 646–654, 1995.

[12] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, "The Protein Data Bank", *Nucleic Acids Research*, vol. 28, pp. 235–242, 2000.

[13] E. G. Birgin, J. M. Martínez, "Large-scale active-set box-constrained optimization method with spectral projected gradients", *Computational Optimization and Applications*, vol. 23, 2002.

[14] I. Borg, P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer, 1997, 470 p.

[15] M. Carpentier, S. Brouillet, J. Pothier, "YAKUSA: A Fast Structural Database Scanning Method", *Proteins*, vol. 61, pp. 137–151, 2005.

[16] A. Chan, "An Analysis of Pairwise Sequence Alignment Algorithm Complexities: Needleman-Wunsch, Smith-Waterman, FASTA, BLAST and Gapped BLAST", *Biochemistry – Final Project*, `http://www.biochem218.stanford.edu/Projects.html`, último acesso em 28/01/2012.

[17] R. Clason, "Finding Clusters: An Application of the Distance Concept", *The Mathematics Teacher*, vol. 83, pp. 301–307, 1990.

[18] A. L. Cuff, I. Sillitoe, T. Lewis, O. C. Redfern, R. Garrat, J. Thornton, A. Orengo, "The CATH classification revisited – architectures reviewed and new ways to characterize structural divergence in superfamilies", *Nucleic Acids Research*, vol. 37, pp. 310–314, 2009.

[19] M. L. Curtis, *Matrix Groups*, New York: Springer-Verlag, 1984, 228 p.

[20] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*, Meboo Publishing, 2005, 722 p.

[21] M. L. Davison, *Multidimensional Scaling*, John Wiley & Sons, 1983, 242 p.

[22] J. E. Dennis, R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Society for Industrial and Applied Mathematics, 1987, 394 p.

[23] J. De Leeuw, "Differentiability of Kruskal's Stress at a Local Minimum", *Psychometrika*, vol. 49, pp. 111–113, 1984.

[24] G. Di Pillo, M. Roma, *Nonconvex Optimization and its Applications: Large-Scale Nonlinear Optimization*, Springer, 2006, 298 p.

[25] Q. Dong, Z. Wu, "A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances", *Journal of Global Optimization*, vol. 22, pp. 365–375, 2002.

[26] D. H. Eberly, *3D Game Engine: A Practical Approach to Real-Time Computer Graphics*, The Morgan Kaufmann Series in Interactive 3D Technology, Springer, 2000, 561 p.

[27] C. Eckart, G. Young, "The approximation of one matrix by another of lower rank", *Psychometrika*, vol. 1, pp. 211–218, 1936.

[28] J. F. Gibrat, T. Madej, S. H. Bryant, "Surprising similarities in structure comparison", *Current Opinion in Structural Biology*, vol. 6, pp. 377–385, 1996.

[29] W. Gish, D. J. States, "Identification of protein coding regions by database similarity search", *Nature Genetics*, vol. 3, pp. 266-272, 1993.

[30] A. Globerson, S. Roweis, "Visualizing pairwise similarity via semidefinite programming", *Proceddings of The 11th International Workshop on Artificial Intelligence and Statistics, 8 p, 2007.*

[31] W. Glunt, T. L. Hayden, M. Raydan, "Molecular conformations from distance matrices", *Journal of Computational Chemistry*, vol.14, pp. 114–120, 1993.

[32] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Baltimore: The Johns Hopkins University Press, 1996, 728 p.

[33] J. C. Gower, "Euclidean Distance Geometry", *The Mathematical Scientist*, vol. 7, pp. 1–14, 1982.

[34] J. C. Gower, "Properties of Euclidean and Non-Euclidean Distance Matrices", *Linear Algebra and its Applications*, vol. 67, pp. 81–97, 1985.

[35] P. S. Gouveia, "Resolução do Problema de Alinhamento Estrutural entre Proteínas via Técnicas de Otimização Global" *Tese de Doutorado*, Departamento de Matemática Aplicada – IMECC – Unicamp, 2011, 133 p.

[36] A. Guerler, E. Knapp, "Novel protein folds and their non-sequential structural analogues", *Protein Science*, vol. 17, pp. 1374–1382, 2008.

[37] Y. Guo, J. Gao, P. W. Kwan, K. X. Hou, "Visualization of Protein Relationships Using Twin Kernel Embedding", *Proceddings of The 1st International Conference on Bioinformatics and Biomedical Engineering*, pp. 1–4, 2007.

[38] H. Hasegawa, L. Holm, "Advances and pitfalls of protein structural alignment", *Current Opinion in Structural Biology*, vol. 19, 341–348, 2009.

[39] A. Heger, S. Mallick, C. Wilton, L. Holm, "The global trace graph, a novel paradigm for searching protein sequence databases", *Bioinformatics*, vol. 23, pp. 2361–2367, 2007.

[40] D. G. Higgins, P. M. Sharp, "CLUSTAL: a package for performing multiple sequence alignment on a micro computer", *Gene*, vol. 73, pp. 237–244, 1988.

[41] L. Holm, C. Sander, "Protein Structure Comparison by Alignment of Distance Matrices", *Journal of Molecular Biology*, vol. 233, pp. 123–138, 1993.

[42] L. Holm, S. Kääriäinen, P. Rosenström, A. Schenkel, "Searching protein structure databases with DaliLite v. 3", *Bioinformatics*, vol. 24, pp. 2780–2781, 2008.

[43] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions", *Journal of the Optical Society American*, vol. 4, pp. 629–642, 1987.

[44] J. Hou, S. Jun, C. Zhang, S. Kim, "Global mapping of the protein structure space and application in structure-based inference of protein function", *PNAS - Proceedings of the National Academy of Sciences*, vol. 102, pp. 3651–3656, 2005.

[45] J. Hou, G. Sims, C. Zhang, S. Kim, "A global representation of the protein fold space", *PNAS - Proceedings of the National Academy of Sciences*, vol. 100, pp. 2386–2390, 2003.

[46] H. Huang, Z. Liang, P. M. Pardalos, "Some properties for the euclidean distance matrix and positive semidefinite matrix completion problems", *Journal of Global Optimization*, vol.25, pp. 3–21, 2003.

[47] W. Huyer, A. Neumaier, "Global optimization by multilevel coordinate search", *Journal of Global Optimization*, vol. 14, pp. 331–355, 1999.

[48] W. Kabsch, "A discussion of the solution for the best rotation to relate two sets of vectors", *Acta Crystallographica – Section A*, vol. 34, pp. 827–828, 1978.

[49] T. Kawabata, K. Nishikawa, "Protein Structure Comparison Using the Markov Transition Model of Evolution", *Proteins*, vol. 41, pp. 108–122, 2000.

[50] S. K. Kearsley, "On the orthogonal transformation used for structural comparisons", *Acta Crystallographica – Section A*, vol. 45, pp. 208–210, 1989.

[51] R. Kolodny, N. Linial, "Approximate protein structural alignment in polynomial time", *Proceedings of the National Academy of Sciences*, vol. 101, pp. 12201–12206, 2004.

[52] R. Kolodny, P. Koehl, M. Levitt, "Comprehensive evaluation of protein structure alignment methods: Scoring by geometric measures", *Journal of Molecular Biology*, vol. 346, pp. 1173–1188, 2005.

[53] E. Krissinel, K. Henrick, "Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions", *Acta Crystallographica*, pp. 2256–2268, 2004.

[54] J. B. Kruskal, "Multidimensional scaling by otimizing goodness of fit to a nonmetric hypothesis", *Psychometrika*, vol. 29, pp. 1–27, 1964.

[55] J. B. Kruskal, "Nonmetric multidimensional scaling: a numerical method", *Psychometrika*, vol. 29, pp. 115–129, 1964.

[56] J. B. Kruskal, M. Wish, *Multidimensional Scaling, Series: Quantitative Applications in the Social Sciences*, Sage Publications, vol. 11, 1977, 95 p.

[57] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, New Jersey: Princeton University Press, 2002, 400 p.

[58] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettingan, H. McWillian, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Gibson, T. J. Gibson, D. G. Higgins, "Clustal W and Clustal X version 2.0", *Bioinformatics*, vol. 23, pp. 2947–2948, 2007.

[59] C. Lavor, L. Liberti, A. Mucherino, "The iBP algorithm for the discretizable molecular distance geometry problem with interval data", *Optimization Online*, `www.optimization-online.org/DB_HTML/2011/01/2889.html`, último acesso em 28/01/2012.

[60] C. Lavor, A. Mucherino, L. Liberti, N. Maculan, "On the computation of protein backbones by using artificial backbones of hydrogens", *Journal of Global Optimization*, vol. 50, pp. 329–344, 2011.

[61] C. Lavor, A. Mucherino, B. Masson, J. Lee, "Discrete approaches for solving molecular distance geometry problems using NMR data", *International Journal of Computational Bioscience*, vol. 1, pp. 88 – 94, 2010.

[62] A. Mucherino, L. Liberti, C. Lavor, "MD-jeep: an implementation of a branch-and-prune algorithm for distance geometry problems", *Mathematical Software*, vol. 6327, pp. 186 –197, 2010.

[63] C. M. Leslin, A. Abyzov, V. A. Ilyin, "TOPOFIT-DB, a database of protein structural alignments based on the TOPOFIT method", *Nucleic Acids Research*, vol. 35, pp. 317–321, 2007.

[64] L. Liberti, C. Lavor, N. Maculan, "A branch-and-prune algorithm for the molecular distance problem", *International Transactions in Operational Research*, vol. 15, pp. 1–17, 2008.

[65] L. Liberti, C. Lavor, A. Mucherino, N. Maculan, "Molecular distance geometry methods: from continuous to discrete", *International Transactions in Operational Research*, v. 18, pp. 33–51, 2010.

[66] L. Liberti, C. Lavor, N. Maculan, F. Marinelli, "Double variable neighbourhood search with smoothing for the molecular distance geometry problem", *Journal of Global Optimization*, vol. 43, pp. 207–218, 2009

[67] R. Lima, "Descrição de Rotações: Prós e Contras na Teoria e na Prática", *Dissertação de Mestrado*, Departamento de Matemática Aplicada – IMECC – Unicamp, 2007, 97 p.

[68] D. G. Luenberger, *Linear and Nonlinear Programming*, New York: Addisson-Wesley, 2003, 516 p.

[69] J. M. Martínez, "Generalized Order-Value Optimization", Technical Report, DMA – IMECC – Unicamp, 2011.

[70] L. Martínez, R. Andreani, J. M. Martínez, "Convergent algorithms for protein structural alignment", *BMC Bioinformatics*, vol. 8, artigo 306, 2007.

[71] A. G. Murzin, S. E. Brenner, T. Hubbard, C. Chothia, "SCOP: A structural Classification of Proteins Database for the Investigation of Sequences and Structures", *Journal of Molecular Biology*, vol. 247, pp. 536–540, 1995.

[72] B. Needleman, C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *Journal of Molecular Biology*, vol. 48, pp. 443–453, 1970.

[73] A. Neumaier, "Distances, graphs and designs", *European Journal of Combinatorics*, vol. 1, pp. 163–174, 1980.

[74] J. Nocedal, S. Wright, *Numerical Optimization*, New York: Springer-Verlag, 1999, 656 p.

[75] A. Orengo, A. Michie, S. Jones, D. Jones, M. Swindells, J. Thornton, "CATH – a hierarchic classification of protein domain structures", *Structure*, vol.5, pp. 1093–1109, 1997.

[76] A. R. Ortiz, C. E. Straus, O. Olmea, "MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison", *Protein Science*, vol. 11, pp. 2606–2621, 2002.

[77] W. R. Pearson, D. J. Lipman, "Improved tools for biological sequence comparison", *Proceddings of the National Academy of Sciences*, vol. 85, pp. 2444–2448, 1998.

[78] M. J. D. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives", *Report No. DAMTP – Center for Mathematical Sciences – University of Cambridge*, 2009.

[79] A. Sacan, I. H. Toroslu, H. Ferhatosmanoglu, "Integrated Search and Alignment of Protein Structures", *Bioinformatics*, vol. 24, pp. 2872–2879, 2008.

[80] J. B. Saxe, "Embeddability of weighted graphs in k-space is strongly NP-hard", *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pp. 480–489, 1979.

[81] I. J. Schoenberg, "Remarks to Maurice Fréchet's article: *Sur la définition axiomatique d'une classe d'espace distanciés vectoriellement applicable sur l'espace de Hilbert*", *Annals of Mathematics*, vol. 36, pp. 724–732, 1935.

[82] G. E. Sims, I. Choi, S. Kim, "Protein conformational space in higher order ϕ-ψ maps", *PNAS - Proceedings of the National Academy of Sciences*, vol. 102, pp. 618–621, 2005.

[83] G. E. Sims, "Global Mapping and Multivariate Analysis of Protein and Nucleic Acid Conformations", *Tese de Doutorado*, Universidade da Califórnia, 2001, 129 p.

[84] T. F. Smith, M. S. Waterman, "Identification of Common Molecular Subsequences", *Journal of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[85] A. M. So, Y. Ye, "Theory of semidefinite programming for sensor network localization", *Mathematical Programming*, vol. 109, pp. 367-384, 2007.

[86] M. Spong, M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, 1989, 336 p.

[87] S. Subbiah, D. V. Laurents, M. Levitt, "Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core", *Current Biology*, vol. 3, pp. 141–148, 1993.

[88] J. C. Traina, A. Traina, B. Seeger, C. Faloutsos, "Slim-trees: High performance metric trees minimizing overlap between nodes", *Proceedings of the 7th International Conference on Extending Database Technology*, pp. 51–65, 2000.

[89] M. Trosset, "Distance Matrix Completion by Numerical Optimization", *Computational Optimization and Applications*, vol. 17, pp. 11–12, 2000.

[90] M. Veeramalai, Y. Ye, A. Godzik, "TOPS++FATCAT: Fast flexible structural alignment using constraints derived from TOPS+ Strings Model", *BMC Bioinformatics*, vol. 9, artigo 358, 2008.

[91] F. S. Yano, "Otimização da menor soma de valores ordenados", *Tese de Doutorado*, Departamento de Matemática Aplicada – IMECC – Unicamp, 2006, 99 p.

[92] M. P. Williamson, T. F. Havel, K. Wüthrich, "Solution conformation of proteinase inhibitor IIA from bull seminal plasma by ^1H nuclear magnetic resonance and distance geometry", *Journal of Molecular Biology*, vol. 315, pp. 182–295, 1985.

[93] A. E. Xavier, "The Hyperbolic Smoothing Clustering Method", *Pattern Recognition*, vol. 43, pp. 731–737, 2010.

[94] A. E. Xavier, V. L. Xavier, "Solving the Minimum Sum-of-Squares Clustering Problem by Hyperbolic Smoothing and Partition into Boundary and Gravitacional Regions", *Pattern Recognition*, vol. 44, pp. 70–77, 2010.

[95] A. E. Xavier, J. A. M. Brito "Modelagens Min-Max-Min para o Problema de Localização de Estações de Rádio", Pesquisa Operacional, vol. 1, pp. 295–319, 2006.

[96] Y. Zhang, J. Skolnick, "TM-align: A protein structure alignment algorithm based on TM-score", *Nucleic Acids Research*, vol. 33, pp. 2302–2309, 2005.

[97] J. Zhu, Z. Weng, "FAST: a novel protein structure alignment algorithm", *Proteins*, vol. 14, pp. 417–423, 2005.