UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA

GERÊNCIA DE PESQUISA E PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL - CPGEI

NILTON BARBOSA ARMSTRONG JÚNIOR

PROPOSTA E IMPLEMENTAÇÃO DE UM ALGORITMO PARALELO DE BUSCA EXAUSTIVA PARA O PROBLEMA DO DOBRAMENTO DE PROTEÍNAS COM UM MODELO DISCRETO

DISSERTAÇÃO DE MESTRADO

CURITIBA

NOVEMBRO DE 2008.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

DISSERTAÇÃO

apresentada à UTFPR

como requisito parcial

para a obtenção do grau de

MESTRE EM CIÊNCIAS

por

NILTON BARBOSA ARMSTRONG JÚNIOR

PROPOSTA E IMPLEMENTAÇÃO DE UM ALGORITMO PARALELO DE BUSCA EXAUSTIVA PARA O PROBLEMA DO DOBRAMENTO DE PROTEÍNAS COM UM MODELO DISCRETO

Banca Examinadora:

Presidente e Orientador:

Prof. Dr. Heitor Silvério Lopes — UTFPR

Examinadores:

Prof. Dr. Ernesto Luis Malta Rodrigues — ICSP/FESP

Prof. Dr. Luis Carlos Erpen de Bona — UFPR

Prof. Dr. Carlos Raimundo Erig Lima — UTFPR

Curitiba, Novembro de 2008.

Nilton Barbosa Armstrong Junior

Proposta e Implementação de um Algoritmo Paralelo de Busca Exaustiva Para o Problema do Dobramento de Proteínas com um Modelo Discreto

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do grau de "Mestre em Ciências" – Área de Concentração: Informática Industrial.

Orientador: Prof. Dr. Heitor Silvério Lopes.
Co-Orientador: Prof. Dr. Carlos R. E. Lima.

Curitiba
2008

Ficha catalográfica elaborada pela Biblioteca da UTFPR – Campus Curitiba

A735p Armstrong Júnior, Nilton Barbosa
Proposta e implementação de um algoritmo paralelo de busca exaustiva para o problema do dobramento de proteínas com um modelo discreto / Nilton Barbosa Armstrong Júnior. Curitiba, UTFPR, 2008
XXII, 166 f. : il. ; 30 cm

Orientador: Prof. Dr. Heitor Silvério Lopes
Co-orientador: Prof. Dr. Carlos Raimundo Erig Lima
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2008
Bibliografia: f. 155 – 166

1. Sistemas de computação. 2. Biologia molecular. 3. Bioinformática. I. Lopes, Heitor Silvério, orient. II. Lima, Carlos Raimundo Erig Lima, co-orient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. IV Título.

CDD: 621.3

“I cannot give you the formula for success, but I can give you the formula for failure which is: Try to please everybody.”

- H.B.Swope

AGRADECIMENTOS

Eu gostaria de agradecer a Deus por ter me concedido mais essa conquista em minha vida.

Gostaria de agradecer especialmente à minha esposa, Patrícia, pelo amor, compreensão, carinho e apoio incondicionais, principalmente durante as fases mais difíceis deste trabalho que, apesar de não terem sido poucas, foram todas vencidas graças ao seu apoio.

Aos meus pais, Ana Izabel e Nilton, cujo apoio, incentivo, conduta e exemplo, além de serem sempre fontes de inspiração, foram decisivos nesta fase da minha vida.

Ao meu orientador, Heitor, e ao meu co-orientador, Carlos, pelo suporte e dedicação que me ajudaram a atingir os objetivos deste trabalho.

Ao meu chefe, Milton, pelo enorme apoio e ajuda no desenvolvimento do projeto e na utilização da infraestrutura da Divisão de Inteligência Artificial, onde eu trabalho, no Tecpar, fundamentais para a conclusão do trabalho.

Aos meus colegas de setor, Bruno, Geraldo e Júlio, pelo apoio, pelas inúmeras conversas e debates sobre o projeto e acima de tudo pela diversão do nosso dia-a-dia.

Ao professores da banca, Ernesto e Luís, cujas contribuições ajudaram a melhorar este trabalho.

E a todas as outras pessoas que contribuíram direta ou indiretamente para o desenvolvimento deste trabalho e que porventura não foram mencionadas.

SUMÁRIO

LISTA DE FIGURAS ..... **xv**
LISTA DE TABELAS ..... **xix**
LISTA DE ABREVIATURAS E SIGLAS ..... **xxi**
RESUMO ..... **xxiii**
ABSTRACT ..... **xxiv**
1. INTRODUÇÃO **1**
1.1. MOTIVAÇÃO 1
1.2. OBJETIVOS 3
1.3. ORGANIZAÇÃO DA DISSERTAÇÃO 3
2. FUNDAMENTAÇÃO TEÓRICA **5**
2.1. AMINOÁCIDOS 5
2.1.1. Classificação 7
2.1.2. Solubilidade em meio aquoso 7
2.2. PROTEÍNAS 8
2.2.1. Síntese protéica 9
2.2.2. Classificação 12
2.2.2.1. Por número de aminoácidos 12
2.2.2.2. Por número de cadeias 12
2.2.2.3. Pela forma 13
2.2.2.4. Pela estrutura 13
2.3. DOBRAMENTO DAS PROTEÍNAS 18
2.3.1. Processo 18
2.4. PROBLEMA DA PREDIÇÃO DA ESTRUTURA TERCIÁRIA 21
2.4.1. Introdução 21
2.4.2. Paradoxo de Levinthal 22
2.4.3. Teoria da hipótese termodinâmica 23
2.4.4. Modelos de representação de polipeptídeos 26
2.4.4.1. Modelo analítico 27
2.4.4.2. Modelo discreto: modelo 2DHP 28
2.4.4.3. Outras abordagens de modelos discretos 31
2.5. MÉTODOS DE BUSCA 32
2.5.1. Introdução 32
2.5.2. Representação do espaço de estados em forma de árvore 36

2.5.3. Algoritmos de busca sem informação 38
2.5.3.1. Busca em extensão 38
2.5.3.2. Busca de custo uniforme 39
2.5.3.3. Busca em profundidade 39
2.5.3.4. Busca em profundidade limitada 40
2.5.3.5. Busca de aprofundamento iterativo em profundidade 40
2.5.4. Busca heurística 41
2.5.4.1. Busca Gulosa 42
2.5.4.2. Busca A* 43
2.6. PROCESSAMENTO PARALELO 44
2.6.1. Introdução 44
2.6.2. Algoritmos paralelos 46
2.6.2.1. Pseudo-paralelismo 46
2.6.2.2. *Pipeline* 47
2.6.2.3. Paralelismo real 49
2.6.3. Arquiteturas paralelas 50
2.6.3.1. Multiprocessadores 50
2.6.3.2. Cluster Beowulf 51
2.6.3.3. Computação em Grade 53
2.6.4. Indicadores de desempenho em sistemas paralelos 54
2.6.4.1. Fator de aceleração 55
2.6.4.2. Eficiência de execução 56
2.6.4.3. Custo 57
2.6.4.4. Taxa de ocupação 57
2.6.4.5. Taxa de participação 58
2.6.4.6. Taxa de comunicação 59
2.6.5. Limites de desempenho: Lei de Amdahl e Lei de Gustafson 59
2.6.6. Biblioteca MPI 63
2.7. TRABALHOS CORRELATOS 65
3. METODOLOGIA 67
3.1. INTRODUÇÃO 67
3.2. REPRESENTAÇÃO DO MODELO 2DHP EM ÁRVORE 67
3.3. CARACTERIZAÇÃO DO PROBLEMA 70
3.4. ANÁLISE DO ESPAÇO DE BUSCA 71

3.4.1. Determinação do espaço de busca válido 74
3.4.2. Determinação do espaço de busca promissor 81
3.4.2.1. Potencial de Dobramento Futuro 83
3.4.3. Resumo das regras de corte do espaço de busca....91
3.5. ALGORITMO DESENVOLVIDO 92
3.5.1. Algoritmo A* modificado....92
3.5.2. Ordenação Linear do Polipeptídeo....99
3.6. PARALELIZACAO EM *CLUSTER*....102
3.6.1. Divisão de tarefas....102
3.6.2. Balanceamento de processamento 105
3.6.3. Intercomunicação de resultados....109
3.6.4. Interação mestre-escravo 110
4. RESULTADOS....115
4.1. TESTES DE ACURÁCIA 119
4.2. TESTES DE DESEMPENHO – *CLUSTER* HOMOGÊNEO. 123
4.3. TESTES DE DESEMPENHO – *CLUSTER* HETEROGÊNEO....134
5. DISCUSSÃO E CONCLUSÕES....137
5.1. ANÁLISE DOS RESULTADOS 137
5.2. CONCLUSÕES 143
5.3. TRABALHOS FUTUROS 144
ANEXO 1 - Tabela dos aminoácidos-padrão (COOPER, 2000)....147
ANEXO 2 – Exemplo de análise 149
ANEXO 3 – Exemplo de resultado de análise 153
REFERÊNCIAS....155

LISTA DE FIGURAS

Figura 1 - Estrutura básica de um α-aminoácido 6
Figura 2 - Processo de síntese protéica. Fonte: Clote e Backofen (2000). 9
Figura 3 - Reação de agregação entre dois aminoácidos. 10
Figura 4 - Estrutura de um polipeptídeo hipotético com seis aminoácidos 11
Figura 5 - Formas de descrever uma proteína pelos aminoácidos que a compõem.....11
Figura 6 - Exemplo de uma α-hélice, uma das estruturas secundárias. 15
Figura 7 - Estrutura de uma β-folha, na qual os aminoácidos se organizam em longas seqüências dispostas paralelas umas às outras (ALBERTS *et al*, 2003). 16
Figura 8 - Estrutura terciária da proteína termolisina 17
Figura 9 - Exemplo de estrutura quaternária. As duas cadeias polipeptídicas que a compõe esta molécula têm com seus terminais N e C apontados 17
Figura 10 - Formação da reentrância no processo de dobramento. 20
Figura 11 - Exemplo de modelo analítico, proposto por RICHARDSON (1981).....28
Figura 12 - Exemplo do polipeptídeo HHPPHPHHHHPHPH no modelo 2DHP. 30
Figura 13 - Exemplo de representação da interligação de cidades hipotéticas.....33
Figura 14 - Relação entre a eficiência da função heurística e seu espaço de busca. 35
Figura 15 - Exemplo de um grafo em forma de árvore.....38
Figura 16 - Algoritmo de busca gulosa.....43
Figura 17 - Processamento seqüencial das subtarefas de uma tarefa original. 45
Figura 18 - Processamento pseudo-paralelo: diversas tarefas executadas simultaneamente, sem ganho de desempenho. 47
Figura 19 - Processamento em ***pipeline***. Diversas tarefas podem ser executadas simultaneamente como se houvesse uma linha de produção. 48
Figura 20 - Paralelismo real. As tarefas são executadas individualmente mais rápido que nos demais algoritmos paralelos. 49
Figura 21 - Aplicação da Lei de Amdahl na previsão do fator de aceleração para um algoritmo 90% paralelizado. 61
Figura 22 - Aplicação da Lei de Gustafson na previsão do fator de aceleração para um algoritmo 90% paralelizado. 62
Figura 23 - Comportamento do fator de aceleração de acordo com a porção seqüencial do algoritmo, no caso hipotético da utilização, com 8 processadores. 63
Figura 24 - Fragmento do espaço de busca completo do polipeptídeo HHPH. 69

Figura 25 - Representação 2D dos dobramentos LLS e SLN do polipeptídeo HHPH. ..70
Figura 26 - Composição do espaço de busca para o problema do dobramento de proteínas. ..73
Figura 27 - Exemplo de um dobramento LLO, que é inválido pela Regra2. ..75
Figura 28 - Exemplos de dobramentos inválidos por colisão. ..76
Figura 29 - Exemplo de possibilidade de geração de iteração hidrofóbica. ..77
Figura 30 - Exemplo da duplicação de dobramentos gerada por rotação no espaço bidimensional, em uma cadeia de 3 aminoácidos. ..78
Figura 31 - Exemplo de duplicação de dobramentos por espelhamento, em uma seqüência de 4 aminoácidos. ..78
Figura 32 - Espaço de busca válido para o polipeptídeo HHPH. ..80
Figura 33 - Exemplos de dobramentos válidos e não-promissores. ..81
Figura 34 - Espaço de busca final para o polipeptídeo HHPH. ..82
Figura 35 - Levantamento dos pontos de iteração hidrofóbica. ..85
Figura 36 - Exemplo de cálculo do PDF. ..86
Figura 37 - Exemplo de anulação topológica do PDF no subdobramento NLL. ..87
Figura 38 - Subdobramento NLL, em que o do quinto aminoácido não forma iteração hidrofóbica. ..89
Figura 39 - Exemplo do subdobramento NLN, no qual o PDF permanece inalterado. ..90
Figura 40 - Exemplos de subdobramentos do polipeptídeo HPPPPH. ..90
Figura 41 - Exemplo da ordem de geração dos nós na árvore. ..93
Figura 42 - Comportamento da lista de subdobramentos para um polipeptídeo de 5 aminoácidos. ..94
Figura 43 - Exemplo do subdobramento NLLLLLLLSLLLLLLLLL, com todos os aminoácidos hidrofóbicos. ..96
Figura 44 - Comportamento do PIH e do número de iterações hidrofóbicas encontradas, para o subdobramento mostrado na Figura 43. ..96
Figura 45 - Dobramento NNNNNNLSSSLSOSSLNLNNNONNLS, com 17 iterações hidrofóbicas. ..97
Figura 46 - Exemplo da avaliação do potencial de dobramento de um dobramento completo (NNNNNNLSSSLSOSSLNLNNNONNLS) de 27 aminoácidos. ..98
Figura 47 - Demonstração da avaliação do potencial de dobramento de um dobramento completo (NNNNLSSLNNLSSSOOS). ..99
Figura 48 - Exemplo de calculo do Índice de Acúmulo de Hs. ..101

Figura 49 - Modelo de interconexão e distribuição de agentes nos computadores. 104
Figura 50 - Exemplo da divisão da análise entre cinco agentes. 105
Figura 51 - Balanceamento por meio da multisubdobramentos inicial. 108
Figura 52 - Fluxograma do programa do agente 0 (mestre). 111
Figura 53 - Fluxograma dos agentes escravos. 112
Figura 54 - Comportamento do Número de iterações hidrofóbicas para diversos polipeptídeos puramente hidrofóbicos. 122
Figura 55 - Comportamento da taxa de ocupação de acordo com o tamanho do polipeptídeo e o número de agentes. 127
Figura 56 - Comportamento do coeficiente de aceleração com os diversos tamanhos de polipeptídeo, todos puramente hidrofóbicos. 128
Figura 57 - Coeficiente angular obtido para a curva do espaço de busca. 134

LISTA DE TABELAS

Tabela 1 - Classificação de polipeptídeos quanto ao seu número de aminoácidos 12
Tabela 2 - Número de combinações e estimativa de tempo de análise de acordo com o número de aminoácidos. A estimativa baseia-se em um caso hipotético de busca exaustiva, em que cada dobramento seria processado em 1 ns.....31
Tabela 3 - Levantamento dos possíveis cenários para solução.....33
Tabela 4 - Funções básicas da biblioteca MPI2.....64
Tabela 5 - Resumo das regras de corte do espaço de busca.....92
Tabela 6 - Grupo1: Lista de polipeptídeos mistos utilizados na análise.....117
Tabela 7 - Grupo2 : Lista de polipeptídeos puramente hidrofóbicos utilizados.....118
Tabela 8 – Número máximo de iterações hidrofóbicas obtidos para o grupo 1 por diversos trabalhos.....120
Tabela 9 - Número de iterações hidrofóbicas para polipeptídeos puramente hidrofóbicos. 121
Tabela 10 - Tempos de análise para os polipeptídeos do grupo 1. 123
Tabela 11 - Tempos de análise para os polipeptídeos do grupo 2. 125
Tabela 12 - Desempenho, no grupo 2, da execução com 52 agentes em relação à seqüencial.....126
Tabela 13 - Tamanho do espaço de estados obtido para o grupo 2. 130
Tabela 14 - Número de dobramentos obtidos para os polipeptídeos do grupo 2.....132
Tabela 15 - Eficiência do balanceamento, para os polipeptídeos do grupo 2.....133
Tabela 16 - Teste de desempenho, para o grupo 2, na execução com uma agente, 61 agentes homogêneos e 68 agentes heterogêneos. 135

LISTA DE ABREVIATURAS E SIGLAS

2DHP – *Bidimensional Hydrophobic-polar model* – Modelo hidrofóbico polar bidimensional.

3DHP - *Tridimensional Hydrophobic-Polar model* - Modelo hidrofóbico polar tridimensional.

AMD - *Advanced Micro Devices* - Fabricante de processadores digitais.

ANSI - *American National Standards Institute*.

C – Linguagem de programação de computadores.

CGE - *Charged Graph Embedding* .

CLP – Circuito Lógico Programável.

DNA - *DeoxyriboNucleic Acid* – Ácido desoxiribonucléico.

HPNX – Modelo de representação de polipeptídeos.

HPSCM - *Hydrophobic-Polar Side Chain Model* – Modelo hidrofóbico-polar de cadeia lateral

HP-TSSC - *Hydrophobic-Polar Tangent Spheres Side Chain Model* – Modelo hidrofóbico-polar de cadeia lateral por esferas tangentes.

IAH - Índice de Acúmulo de aminoácidos Hidrofóbicos.

IBM - *International Business Machines* – Fabricante de computadores.

LPE - L*attice Polymer Embedding*.

MPI – *Message Passing Interface*.

NASA - *National Aeronautics and Space Administration* – Agência aeroespacial norte-americana.

PDF – Potencial de Dobramento Futuro.

PH - *Perturbed Homopolymer* – Modelo de representação de polipeptídeos.

PIH – Previsão de Iterações Hidrofóbicas.

RAM – *Random Access Memory* – Memória de acesso randômico.

RNA – *Ribonucleic Acid* – Ácido ribonucléico.

RNAm - RNA mensageiro.

RESUMO

Este trabalho propõe uma abordagem paralela de análise de um dos problemas mais complexos da bioinformática: o problema do Dobramento de Proteínas. As proteínas são compostos químicos que estão presentes em todos os seres vivos. À medida que estas proteínas vão sendo sintetizadas elas vão se dobrando até assumir uma forma enovelada, conhecida como conformação nativa. Ela tem relação direta com a funcionalidade que ela tem no organismo a ela relacionado. O processo de dobramento protéico é intensamente estudado tanto do ponto de vista biológico quanto computacional. Vários modelos computacionais, discretos e contínuos foram propostos para representar o dobramento de proteínas. Este trabalho utiliza o modelo Hidrofóbico-Polar bi-dimensional (2DHP). Foi proposto um algoritmo de busca exaustiva paralela capaz de encontrar todas as ocorrências da conformação nativa de uma proteína em um tempo aceitável, para proteínas de até cerca de 50 aminoácidos. A proposta baseia-se no algoritmo A* modificado, implementando parâmetros topológicos de avaliação capaz de propor significativos cortes no espaço de busca. Foi utilizada uma arquitetura de *cluster* testes do algoritmo. Técnicas de gerenciamento de processamento paralelo foram elaboradas para balanceamento de carga, tendo sido uma taxa de aceleração superlinear. Testes de acurácia do algoritmo foram realizados, comparando seus resultados com outras abordagens. Também foram realizados testes de desempenho e escalabilidade. Os resultados indicaram que a abordagem proposta é uma alternativa viável para a análise do dobramento de proteínas com o modelo 2DHP, podendo ser extensível para outros modelos.

ABSTRACT

PROPOSAL AND IMPLEMENTATION OF A PARALLEL EXHAUSTIVE SEARCH ALGORITHM FOR THE PROTEIN FOLDING PROBLEM USING A DISCRETE MODEL

This work suggests a parallel analysis approach to one of the most complex problems in bioinformatics: the Protein Folding problem. The proteins are chemical compounds present in every organism. As soon as they are being synthesized they begin to fold over itself, assuming a coiled form, known as native conformation. This conformation has direct relation to the protein's functionality within the organism. The folding process is intensively studied in the biology as well as in the computational field. There are many discrete and continuous computational models that represent the protein folding. This work uses the bidimensional Hydrophobic-Polar model (2DHP) to propose a parallel exhaustive search algorithm. It is capable of finding every occurrence of the protein's native conformation, in a reasonable time, for polypeptides of up to 50 amino acids of length. This proposal is based on a modified A* algorithm, implementing topological parameters of evaluation, capable of making significant reductions on the search space. A *cluster* architecture was used to implement the algorithm. Some parallel processing management techniques were developed and a superlinear acceleration rate has been achieved. Some accuracy tests were carried out and the results were compared with other algorithms. Performance and scalability tests were also accomplished. The results indicated that the proposed approach is a viable alternative to the 2DHP protein folding analysis, being able to be extended to other models.

KEYWORDS

Protein folding, Exhaustive Search, Parallel Systems, Bioinformatics, *Cluster* Computing.

CAPÍTULO 1

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

As proteínas são estruturas vitais e presentes em absolutamente todos os seres vivos. Tais estruturas são componentes que desempenham funções, tanto intra quanto extra celulares. Estas funções variam desde as mais básicas (como as estruturas físicas das células, por exemplo, a membrana citoplasmática), passando por funções (o transporte de outras substâncias, por exemplo, a hemoglobina que carrega o oxigênio até as células), até mesmo funções regulatórias mais complexas, por exemplo, as enzimas e os hormônios dos organismos.

Tais estruturas são compostas por estruturas químicas ainda menores, chamados de aminoácidos, que são abundantes na natureza. Apesar de tal disponibilidade, apenas 20 destes são capazes de compor as proteínas. A proteína, portanto, é um encadeamento de aminoácidos, cuja ordem será definida diretamente pelo DNA (*DeoxyriboNucleic Acid*) responsável pela codificação daquela seqüência. A função de cada proteína, porém, não é definida pelos aminoácidos que a compõem, mas sim pela estrutura tridimensional que a proteína assume, à medida que vai sendo sintetizada. Tal processo é denominado de Dobramento da Proteína e levará a molécula a um estado físico de posicionamento de seus átomos denominado conformação nativa, que é simplesmente a proteína, com todos os seus aminoácidos, já dobrada.

Desse modo a conformação nativa é o posicionamento assumido pelos aminoácidos de forma a permitir que uma determinada função possa ser desempenhada no organismo. Durante o funcionamento normal do organismo tal processo acontece inúmeras vezes, sempre que uma proteína precisa ser sintetizada.

Entretanto, pode acontecer que um determinado DNA ordene o encadeamento de uma seqüência de aminoácidos, mas, por algum motivo, tais aminoácidos não se posicionem conforme o esperado. Isto dá origem a um dobramento errado, resultando em uma proteína sem funcionalidade para o organismo, ou pior: com desempenho nocivo a esse organismo. Isso pode acontecer devido a vários fatores: mutações no DNA, variações de pH, temperatura, danos na célula ou deficiência de outras substâncias. Isso ocorrendo, além de o organismo não

obter a proteína de que necessitava, outra foi gerada com um potencial efeito colateral pouco previsível.

Acredita-se que diversas doenças tenham sua causa em proteínas que se dobraram de forma errada, tais como os alguns tipos de câncer, o Mal de Alzheimer, a fibrose cística, a encefalopatia espongiforme bovina (doença da vaca louca), entre outras.

O desenvolvimento de um processo ou medicamento que possa reverter ou corrigir tal anomalia é essencial para a prevenção ou a cura de tais doenças. Infelizmente, o processo que faz com que o dobramento aconteça é extremamente complexo e pouco se sabe sobre ele. Além de envolver um número enorme de átomos dos aminoácidos, pode ou não envolver estruturas internas das células. Portanto, é improvável que um medicamento consiga desdobrar uma proteína anômala e redobrá-la na forma correta. Da mesma forma, é mais difícil ainda elaborar uma substância que possa atuar no processo do dobramento para garantir que ele aconteça corretamente.

Parte da solução de tal problema é aprender mais sobre como uma determinada seqüência de aminoácidos se organiza e origina uma proteína funcional. Esse estudo é chamado de Predição de Estrutura Terciária, ou Predição do Dobramento das Proteínas. Tal problema, no entanto, envolve uma quantidade de processamento extremamente grande, seja pela quantidade de dados, seja pela quantidade de dobramentos possíveis. A utilização de métodos computacionais e modelos de representação com detalhamento pequeno é vital para garantir a viabilidade de tal estudo.

Buscando-se determinar a estrutura dobrada de uma proteína, sabendo-se apenas os aminoácidos que a compõem, escolheu-se neste trabalho o modelo de representação de proteína chamado de Modelo Bidimensional Hidrofóbico-Polar, ou simplesmente 2DHP. O método para se determinar a estrutura que representa a proteína dobrada foi a teoria da Energia Livre, a qual busca conformações que minimizem a energia necessária para tornar a estrutura tridimensional da molécula estável, ou seja, a sua conformação nativa. O método de busca escolhido foi o da Busca Exaustiva, pois se analisam todas as conformações possíveis. Portanto, a chegada aos melhores resultados não depende de uma heurística que guia o desenvolvimento do algoritmo até a solução. Tal método, porém, resulta em um problema NP-difícil (CRESCENZI, GOLDMAN, PAPADIMITRIOU *et al*, 1998). Para se resolver tal problema, diversos melhoramentos no algoritmo de busca, na representação e na paralelização do problema foram desenvolvidos, de modo que se possa potencializar a pesquisa e cada vez mais contribuir para a solução do Problema do Dobramento de Proteínas.

1.2. OBJETIVOS

Este trabalho tem por objetivo principal elaborar um algoritmo paralelo de busca exaustiva, capaz de abordar o problema do dobramento de proteínas em polipeptídeos utilizando o modelo de representação computacional bidimensional Hidrofóbico-Polar (2DHP). Os objetivos específicos do trabalho são:

- Elaborar um algoritmo de predição de estrutura terciária de proteínas, utilizando métodos de busca exaustiva inteligente que consiga, garantidamente, chegar ao melhor dobramento, segundo a teoria de Energia Livre e ter a possibilidade de listar todas a ocorrências possíveis do melhor dobramento para uma determinada proteína.
- Propor melhorias no algoritmo básico para reduzir o tempo necessário para o processamento, de maneira a permitir que todas as conformações possíveis de um dado polipeptídeo possam ser analisadas em um tempo aceitável.
- Adaptar o algoritmo seqüencial para que possa ser paralelizado de forma escalável, objetivamente em *cluster* Beowulf.
- Proporcionar uma plataforma de estudo do dobramento, de modo a poder suportar outros modelos mais fidedignos à realidade do que o modelo 2DHP.
- Realizar comparações com outras abordagens, a fim de verificar a acurácia dos resultados e o desempenho do algoritmo.

1.3. ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está organizada em 5 capítulos. O capítulo 2 faz uma revisão da literatura, explicando os conceitos básicos a respeito de proteínas, o Problema do Dobramento de Proteínas, os algoritmos de busca e o processamento paralelo, bem como os trabalhos correlatos. O capítulo 3 explica em detalhes o algoritmo desenvolvido, suas características e a sua paralelização. O capítulo 4 expõe os experimentos e os resultados obtidos. Concluindo, o capítulo 5 apresenta as discussões sobre os resultados, as conclusões e as oportunidades de trabalhos futuros.

CAPÍTULO 2

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo destina-se a apresentar a fundamentação teórica deste trabalho. Primeiramente serão mostrados os conceitos relativos à biologia. Esses conceitos são compostos dos aminoácidos, das proteínas e do decorrente problema de se prever o seu dobramento. Em seguida será mostrada a abordagem computacional adotada para tratar do problema do dobramento, sendo descrita a hipótese termodinâmica e o modelo de representação computacional utilizado. Na seqüência serão mostrados os conceitos computacionais, iniciando pelos diversos métodos de busca, descrevendo-se também o conceito de computação paralela, explicando-se os diversos aspectos de algoritmos paralelos, arquiteturas paralelas, indicadores de desempenho, limites e a biblioteca de programação utilizada. Por fim serão apresentados também os trabalhos correlatos.

2.1. AMINOÁCIDOS

Os aminoácidos são as estruturas que compõem os polipeptídeos ou proteínas (CLOTE e BACKOFEN, 2000) e outros inúmeros compostos químicos, além de serem precursores de outros compostos, como purinas e pirimidinas (BERG, TYMOCZKO e STRYER, 2002), que formarão o DNA e o RNA (*Ribonucleic Acid)*. Os aminoácidos são encontrados em todas as formas de vida (AVANCINI e FAVARETTO, 1997).

No organismo, os aminoácidos possuem diversas funções (YOUNG e YU, 1996), que variam desde neurotransmissores, como o triptofano, transportadores de nitrogênio, como a arginina ou reguladores de atividade enzimática, como a fenilalanina.

Quimicamente os aminoácidos são quaisquer moléculas que contenham os grupos funcionais amina (caracterizados pela presença do átomo nitrogênio) e o grupo carboxil (caracterizado pela presença da molécula COOH) (BERG, TYMOCZKO e STRYER, 2002), ilustrados na Figura 1. Esse grupo também é chamado de ácido carboxílico.

Na Figura 1, é possível notar que, no ácido carboxílico, há um elemento que não existe, de fato, na tabela periódica: o elemento R, que denota que aquele carbono tem a

necessidade de realizar mais uma ligação. Nessa ligação, portanto, será ligada uma molécula que fará a distinção entre os diversos ácidos carboxílicos, chamada de radical ou ainda cadeia lateral (CLOTE e BACKOFEN, 2000).

Os grupos carboxil e amina podem ser agregados (em organismos capazes) em uma única molécula dando origem a um aminoácido (COOPER, 2000). A essa reação de agregação dá-se o nome de desidratação, ou condensação.

A estrutura de qualquer aminoácido é basicamente a mesma e pode ser percebida pela estrutura mostrada na Figura 1.

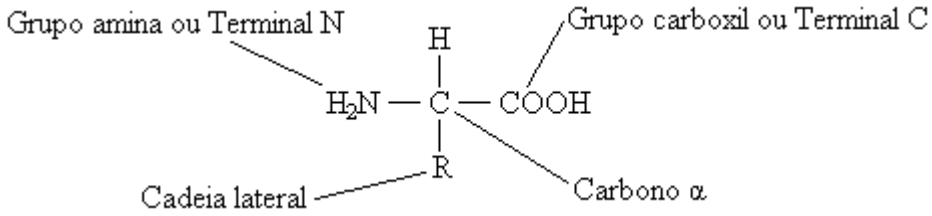

Figura 1 - Estrutura básica de um α-aminoácido

Na natureza, existem inúmeros aminoácidos, porém apenas 20 deles são proteinogênicos, ou seja, têm a capacidade de participar da composição de proteínas (COOPER, 2000). Nesses 20 aminoácidos, além dos grupos químicos funcionais que os formaram, amina e carboxil (que também são chamados respectivamente de Terminal C por conterem um carbono e Terminal N por conterem um nitrogênio), podem ser percebidas mais duas estruturas muito importantes: o carbono α e a cadeia lateral. O carbono α é dito do átomo carbono que faz a ligação, simultaneamente, entre os dois grupos químicos da molécula e a cadeia lateral (elemento R na Figura 1). Observando-se a Figura 1, pode-se perceber que o carbono α está realizando a ligação entre a amina e o carboxil. A presença desse carbono α passa a nomear os aminoácidos que o possuem de α-aminoácidos (HUNTER, 1993). A outra estrutura importante é a cadeia lateral, mostrada de forma genérica na Figura 1 pelo símbolo R. Essa cadeia lateral é, na realidade, herança da cadeia que o Terminal C carregava quando era um ácido carboxílico e não é eliminada na reação de formação do aminoácido. Como a estrutura básica, ou cadeia central, é sempre a mesma para os α-aminoácidos, a cadeia lateral é sempre responsável por definir a função que aquele aminoácido terá no organismo. Essas cadeias também definem as várias propriedades químicas do aminoácido. Isso possibilita fazer a classificação desses 20 aminoácidos. Destes, apenas 10 são produzidos pelo corpo humano, sendo denominados de não-essenciais (AVANCINI e FAVARETTO, 1997). Os outros 10 não podem ser sintetizados pelas células humanas, pois não há código genético que

ordene a síntese deles, sendo assim denominados essenciais, tendo que ser ingeridos por meio de alimentação adequada (AVANCINI e FAVARETTO, 1997).

2.1.1. Classificação

Baseando-se nas características físicas e químicas dos aminoácidos, algumas propriedades podem ser levantadas – a massa atômica, o pH e a polaridade – e podem ser utilizadas em modelos de representação.

A massa refere-se à soma das massas atômicas dos átomos que compõem a molécula. O pH, também chamado de potencial hidrogeniônico, é um índice que mede a acidez, ou seja, a neutralidade ou a alcalinidade de uma solução (BERG, TYMOCZKO e STRYER, 2002). Esta será ácida se tiver pH menor que 7; neutra se o pH for igual a 7; e básica se o pH for superior a 7. A polaridade, uma das propriedades mais exploradas nos modelos computacionais de representação de aminoácidos, refere-se à localização do centro de massa das cargas positivas (prótons) e negativas (elétrons) de uma molécula (HUNTER, 1993). Mesmo levando-se em conta que as moléculas (não os íons) têm uma carga elétrica neutra, elas podem ou não ser polarizadas. O índice considera o posicionamento tridimensional dos átomos do composto, de modo que, se o centro de massa das cargas positivas coincidir com o das cargas negativas, define-se que a molécula em questão é apolar, ou seja, não tem polaridade predominante (BERG, TYMOCZKO e STRYER, 2002).

A água tem uma disposição tridimensional que faz com que haja uma distribuição não-uniforme das cargas elétricas na molécula, fazendo-a mais eletronegativa no lado do átomo de oxigênio e mais eletropositiva do lado dos átomos de hidrogênio, portanto sendo classificada como molécula polar (BERG, TYMOCZKO e STRYER, 2002). Tal fato não acontece com o gás oxigênio cuja estrutura propicia que a molécula seja considerada apolar.

2.1.2. Solubilidade em meio aquoso

A solubilidade em água é uma característica muito relevante no estudo dos aminoácidos. Considerando-se que a solubilidade tem relação direta com a polaridade do soluto e do solvente, solventes tendem a dissolver melhor os solutos de mesma polaridade

(LODISH, BERK, MATSUDAIRA *et al*, 2000). Como os meios orgânicos em que as proteínas são sintetizadas são essencialmente meios aquosos (cerca de 70% das células é água) (HUNTER, 1993), isso significa que os aminoácidos com a mesma polaridade da água (que é polar) têm uma tendência maior de ficar em contato direto com ela. De forma similar, as moléculas apolares tendem a ser repelidas pela água, pela incapacidade de formar pontes de hidrogênio, ou seja, não são eletricamente atrativas à água e, portanto, são repelidas (LODISH, BERK, MATSUDAIRA *et al*, 2000).

De acordo com esse comportamento em relação à água, os aminoácidos são classificados em duas categorias (LODISH, BERK, MATSUDAIRA *et al*, 2000): hidrofóbicos (ou apolares), para aqueles que são repelidos pela água, e polares (ou hidrofílicos) para aqueles que conseguem interagir com o meio aquoso.

Essa característica de hidrofobicidade é de fundamental importância para a análise do dobramento pela influência que ela exerce na formação da estrutura tridimensional, como será explicado nos capítulos seguintes.

2.2. PROTEÍNAS

Proteínas são as estruturas básicas formadoras dos seres vivos (HUNTER, 1993). São, também, as moléculas orgânicas mais abundantes de todas as células, totalizando cerca de 50% do peso seco, e as principais constituintes dos músculos. O corpo humano produz em torno de 50.000 diferentes proteínas (DOBSON, 2000), dentre as quais se encontram todas as enzimas e os hormônios conhecidos (HUNTER, 1993). Assim sendo, as proteínas têm funções que variam desde o suporte mecânico das células – por exemplo, a construção das membranas e fibras musculares (actina e miosina) (MOLLER e NAIR, 1999) – até as funções de transporte de substâncias vitais – por exemplo, o oxigênio, pela hemoglobina e mioglobina (LODISH, BERK, MATSUDAIRA *et al*, 2000), ou a glicose ao interior das células, pela insulina (YOUNG e YU, 1996) – ou ainda funções mais complexas – inúmeros hormônios que agem nos organismos.

Do ponto de vista da Biologia Molecular, uma proteína é um conjunto de aminoácidos conectados entre si, na forma de uma estrutura linear, também chamada de polipeptídeo (BERG, TYMOCZKO e STRYER, 2002). Em geral, uma proteína é sintetizada dentro do

ambiente celular e sua formação é descrita pelo dogma central da Biologia Molecular (CRICK, 1958; CRICK, 1970). Cada proteína somente é produzida no organismo de acordo com a necessidade que este tem dela, e representa, portanto, a expressão de um gene do DNA (GRIFFITHS, MILLER, SUZUKI *et al*, 2000).

Nas subseções seguintes serão detalhados os processos de formação e de classificação e a organização estrutural das proteínas.

2.2.1. Síntese protéica

A síntese de uma proteína, ou um polipeptídeo, por um organismo obedece ao Dogma Central da Biologia Molecular (CRICK, 1958) e (CRICK, 1970), na qual o DNA dá origem ao RNA, que dá origem às proteínas.

Tal dogma especifica que o fluxo biológico para se obter uma proteína inicia-se no DNA, com a expressão de um gene específico. A partir desse gene, por meio do processo de transcrição de DNA (ALBERTS, JOHNSON, LEWIS *et al*, 2002), um código complementar ao DNA daquele gene é gerado dentro do núcleo da célula. Tal código é chamado de RNAm (RNA mensageiro) e, de acordo com a sua seqüência de bases nitrogenadas (GIBAS e JAMBECK, 2001), contém a informação da seqüência de aminoácidos que comporá a proteína relacionada com aquele determinado gene. Após a transcrição, o RNAm é expelido do núcleo, em direção ao citoplasma daquela célula, como mostrado na Figura 2.

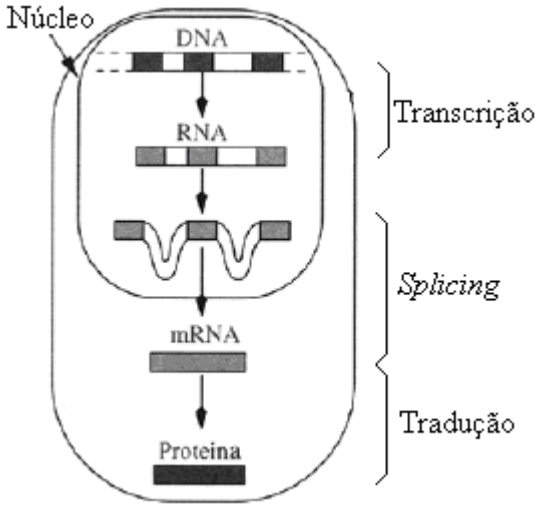

Figura 2 - Processo de síntese protéica. Fonte: Clote e Backofen (2000).

No citoplasma existe uma estrutura intracelular denominada ribossomo (GIBAS e JAMBECK, 2001). Ele é o responsável por ler o RNAm e atrair os aminoácidos correspondentes imersos no citoplasma. Efetuando-se a leitura do RNAm em grupos de três bases nitrogenadas, chamadas de códons (GIBAS e JAMBECK, 2001), o ribossomo consegue identificar qual é o aminoácido a compor a estrutura da nova proteína. À medida que essa leitura é feita, os aminoácidos são concatenados uns aos outros, de maneira a formar uma fita de aminoácidos. Tal operação de agregação de aminoácidos é também conhecida como condensação, mais especificamente como desidratação (GRIFFITHS, MILLER, SUZUKI *et al*, 2000), e pode ser observada na Figura 3.

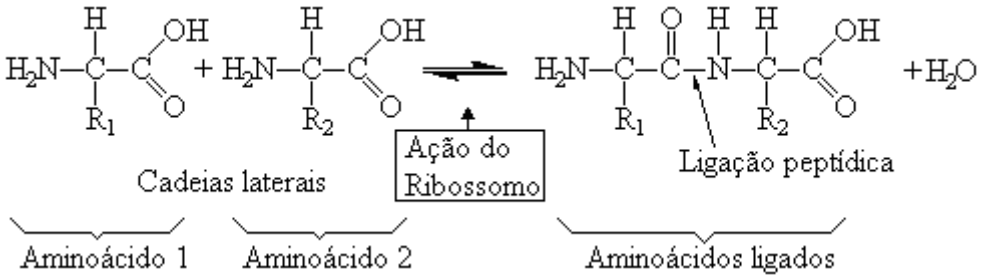

Figura 3 - Reação de agregação entre dois aminoácidos.

O processo de agregação de aminoácidos é conhecido como desidratação por liberar uma molécula de água. Pode-se perceber que os aminoácidos são basicamente os mesmos, com exceção de que, em um deles, o terminal C cedeu uma hidroxila e, no outro, o terminal N cedeu um hidrogênio. Agora, portanto, existe uma ligação entre o terminal C e o N desses aminoácidos. Tal ligação é covalente e denominada peptídica, por ser feita entre dois aminoácidos, entre os grupos carboxil de um aminoácido e o grupo amina do outro (LODISH, BERK, MATSUDAIRA *et al*, 2000). Após essa reação de condensação, os aminoácidos já não possuem mais a mesma estrutura original e, por isso, freqüentemente recebem o nome de "resíduos", pois são o que resultou como produto, além da água. O aminoácido resultante dessa conexão é então chamado de peptídeo e o conjunto deles – conectados uns aos outros formando uma cadeia – é chamado de polipeptídeo (GRIFFITHS, MILLER, SUZUKI *et al*, 2000). A síntese ocorre sempre no sentido terminal N para o terminal C: no terminal N novos aminoácidos são agregados sempre no terminal C dos aminoácidos já pertencentes à nova cadeia sendo formada. Uma característica importante é que a sua cadeia lateral não é alterada, permitindo que as propriedades químicas e funcionais do aminoácido permaneçam inalteradas (COOPER, 2000).

Tal reação de agregação não pode acontecer espontaneamente, pois depende da leitura do mRNA, por isso a necessidade de uma estrutura adicional para propiciar tal processo. Tal estrutura denomina-se ribossomo e é composto de mais de 50 proteínas e diversas cadeias de RNA ribossômico (ALBERTS, JOHNSON, LEWIS *et al,* 2002). Tais estruturas, em geral, existem aos milhões no citoplasma das células e são capazes de ler o mRNA e agregar novos aminoácidos a uma taxa de aproximadamente dois resíduos por segundo (ALBERTS, JOHNSON, LEWIS *et al,* 2002).O resultado então é uma seqüência de resíduos conectados por ligações peptídicas como mostrado na Figura 4. Nesta figura observa-se que a estrutura do polipeptídeo é basicamente uma repetição da estrutura que se vê na Figura 3. O único fator que diferencia os elementos da seqüência é a cadeia lateral que cada um carrega. Considerando-se que a estrutura do polipeptídeo (também chamada de cadeia central) é sempre a mesma, é muito mais prático referir-se ao polipeptídeo pelas cadeias laterais que nomeiam seus aminoácidos.

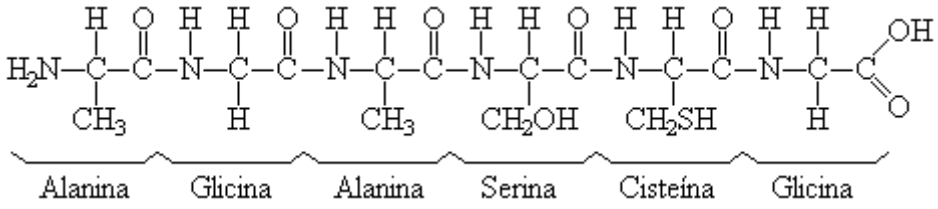

Figura 4 - Estrutura de um polipeptídeo hipotético com seis aminoácidos

Portanto, a seqüência fictícia mostrada na Figura 4 poderia ser representada por uma das formas mostradas na Figura 5. Considerando que apenas 20 aminoácidos podem participar da construção das proteínas e que não há um tamanho fixo para tais cadeias, o número de combinações possíveis é muito grande. Tomando-se, por exemplo, apenas seqüências de 100 aminoácidos, haveria a possibilidade de montar-se 20^{100} seqüências distintas. No entanto, apenas uma fração desse número é, de fato, explorada pela natureza (BALDI e BRUNAK, 2000).

Nomes por extenso:	Alanina - Glicina - Alanina - Serina - Cisteína - Glicina
Código de 3 letras:	ala-gly-ala-ser-cys-gly
Código de 1 letra:	A-G-A-S-C-G

Figura 5 - Formas de descrever uma proteína pelos aminoácidos que a compõem.

2.2.2. Classificação

Os polipeptídeos podem ser classificados de diversas formas: composição, função, número de aminoácidos, número de cadeias e forma. Entretanto, neste trabalho, somente as três últimas serão abordadas, por terem relevância direta com a pesquisa desenvolvida. A importância de se definir claramente tal classificação é basicamente para restringir e identificar o domínio de polipeptídeos que se encaixam neste trabalho.

2.2.2.1. Por número de aminoácidos

Classificam-se os polipeptídeos pelo número de aminoácidos que os compõem (AVANCINI e FAVARETTO, 1997), segundo a Tabela 1. É importante ressaltar que uma proteína é freqüentemente referida como sendo um polipeptídeo, porém o contrário nem sempre se aplica.

Tabela 1 - Classificação de polipeptídeos quanto ao seu número de aminoácidos

Quantidade de aminoácidos	**Designação**	**Exemplo**
Menor que 50	Polipeptídeo	Insulina (relativa ao processo de absorção de açúcar)
Maior	Proteína	Amyloid Beta (relativa ao Mal de Alzheimer)

2.2.2.2. Por número de cadeias

Essa maneira de classificação visa definir os polipeptídeos pelo número de cadeias de aminoácidos que compõem a proteína (LODISH, BERK, MATSUDAIRA *et al*, 2000). Basicamente dois grupos advêm dessa caracterização: as cadeias monoméricas e as multiméricas.

As monoméricas são os polipeptídeos que são já funcionais, ou seja, prontos para serem utilizados pelo organismo. Consistem de apenas uma seqüência de aminoácidos, como acontece com a proteína monomérica tetânica. Ao contrário do que acontece com as multiméricas, como a hemoglobina, que consistem de várias seqüências monoméricas agregadas. Este trabalho atém-se somente às seqüências monoméricas.

2.2.2.3. Pela forma

Esta é uma forma mais abrangente de se classificar os polipeptídeos. Ela leva em consideração a forma tridimensional assumida pela cadeia (GRIFFITHS, MILLER, SUZUKI *et al*, 2000), ou cadeias, no caso de ser um multímero, que estará diretamente relacionado à função dele. Existem basicamente duas formas de caracterização. A primeira refere-se aos polipeptídeos fibrosos, cuja função geralmente está relacionada a atributos estruturais – por exemplo, paredes celulares e o colágeno – ou funções mecânicas – por exemplo, contração muscular, como é o caso da actina e da miosina. Os polipeptídeos fibrosos, em geral, são cadeias longas e alongadas, próprias para a sua função e são insolúveis em meio aquoso, ou seja, não se combinam com a água, tendendo a se separar dela.

O outro tipo são as globulares, cuja função, em geral, não é estrutural, justamente por possuírem uma disposição espacial complexa, desempenhando outras funções como enzimas, hormônios e demais proteínas. Este tipo de polipeptídeo tem forma aproximadamente esférica e têm uma grande solubilidade em meio aquoso, ou seja, sua forma de disposição dos aminoácidos é tal que permite que sua imersão em água – por exemplo, o citoplasma – possa ocorrer sem que haja repulsão do polipeptídeo pela água. Estas são as proteínas visadas neste estudo.

2.2.2.4. Pela estrutura

A organização estrutural visa definir o nível de detalhe na forma de se representar o polipeptídeo (COOPER, 2000). Tal convenção é extremamente conveniente de acordo com a necessidade de representação. São quatro as estruturas, em que as superiores englobam os detalhes das inferiores:

a) Estrutura primária

Esta é uma estrutura que visa descrever a proteína, ou polipeptídeo, apenas pelos aminoácidos que a compõem (COOPER, 2000). De acordo com o Anexo 1 e a Figura 5, pode-se utilizar tanto o nome dos aminoácidos por extenso, o código de três letras ou o código de uma letra (COOPER, 2000), que, muitas vezes, é mais prático. A seqüência fictícia mostrada na Figura 5 é, portanto, a estrutura primária do polipeptídeo cuja estrutura química é mostrada na Figura 4. Esse é o nível mais simples de organização no âmbito do polipeptídeo e

considera apenas as ligações covalentes que ocorrem entre os aminoácidos, as ligações peptídicas. A seqüência dos aminoácidos é sempre do seu terminal N para seu terminal C (BERG, TIMOCZKO e STRYER, 2002). Naturalmente, o polipeptídeo mostrado na Figura 4, que seria A-G-A-S-C-G, seria o mesmo se fosse representado de trás para frente. Porém, a importância de se definir uma direção para a seqüência é crucial ao se analisar o dobramento dessa seqüência. É possível ter um polipeptídeo com o mesmo número e a composição de aminoácidos e ainda assim exibir uma função totalmente diferente no organismo por ele ter sido sintetizado em ordem diferente e, por isso, ter assumido um dobramento diferente (BERG, TIMOCZKO e STRYER, 2002; AVANCINI e FAVARETTO, 1997).

b) Estrutura secundária

Esta classificação visa identificar subestruturas localizadas no polipeptídeo. Como será aprofundado a seguir, as proteínas globulares tendem a se dobrar sobre si mesmas. Dentro dessas estruturas tridimensionais que englobam iterações entre todos os aminoácidos da seqüência, localmente, os aminoácidos interagem entre si, dando origem ao que se chama de estruturas secundárias. O estudo delas é importante e é base para muitas pesquisas que utilizam essas estruturas para determinar a estrutura tridimensional final dos polipeptídeos, pois o conjunto de estruturas secundárias resultará na estrutura terciária, que será discutida a seguir. Existem basicamente duas estruturas secundárias mais importantes, chamadas α-hélice (PAULING, COREY e BRANSON, 1951) e β-folha (PAULING, COREY e BRANSON, 1951; PAULING e COREY, 1951), das quais várias outras são derivadas. Tais derivações, também chamadas de *motifs* (GRIFFITHS, MILLER, SUZUKI *et al*, 2000), são padrões de estruturas freqüentemente encontradas em estruturas tridimensionais – por exemplo, hélice-*loop*-hélice, encontrada em proteínas regulatórias, que auxiliam na estrutura do DNA. Tais *motifs* também são chamados de estruturas supersecundárias (GRIFFITHS, MILLER, SUZUKI *et al*, 2000).

As α-hélices são espirais (podem ser observadas na Figura 6) formadas quando uma cadeia polipeptídica se torce em torno de si mesma, formando uma espécie de cilindro (ALBERTS, JOHNSON, LEWIS *et al*, 2002). Estas espirais ocorrem geralmente devido à proximidade de aminoácidos com grande chance de formar pontes de hidrogênio entre si (ALBERTS, JOHNSON, LEWIS *et al*, 2002) As α-hélices são formadas por composições regulares de aminoácidos hidrofóbicos e polares, formando tais pontes a cada quatro aminoácidos na cadeia (COOPER, 2000). Cada volta completa da espiral em torno de seu

eixo central se faz a cada 3,6 aminoácidos (LODISH, BERK, MATSUDAIRA *et al*, 2000). Elas são bastante observadas em proteínas de transporte e receptores (ALBERTS, JOHNSON, LEWIS *et al*, 2002).

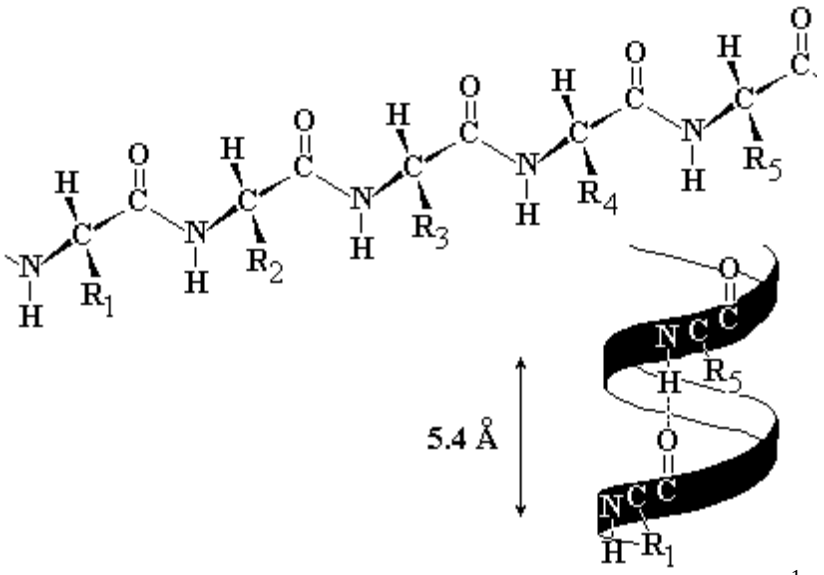

Figura 6 - Exemplo de uma α-hélice, uma das estruturas secundárias[1].

As β-folhas se originam quando duas ou mais seqüências de aminoácidos (cinco a oito) (LODISH, BERK, MATSUDAIRA *et al*, 2000) são dispostas lateralmente esticadas e paralelas, em relação umas às outras, como mostrado na Figura 7. Quando as β-folhas estão conectadas entre si no polipeptídeo, as estruturas responsáveis por tais conexões são chamadas de *turns* (LODISH, BERK, MATSUDAIRA *et al*, 2000) e são compostas de três a quatro aminoácidos. A estrutura é mais suscetível a ser atacada por meios solventes. Portanto, tem uma tendência de ser composta de aminoácidos polares, resistentes à ação solvente da água ou de um meio lipídico. As β-folhas estão presentes em diversas estruturas – por exemplo, na formação de ***binding sites*** no reforço estrutural, como acontece na seda, com a formação de β-folhas anti-paralelas (LODISH, BERK, MATSUDAIRA *et al*, 2000).

Ambas as estruturas não foram abordadas nas análises feitas neste trabalho, pois o foco ateve-se à predição da estrutura terciária a partir da primária. No entanto, a identificação de tais estruturas nos resultados deste trabalho podem potencializar a análise das proteínas e propiciar oportunidades de novos trabalhos.

[1] Figura retirada da URL: Disponível em: <http://ekhidna.biocenter.helsinki.fi/downloads/teaching/sprinH21006/proteiinianalyysi/Proteiinianalyysi-06-VI-AppendixA_files/peptide5.gif>. Acesso em: 2 ago. 2008.

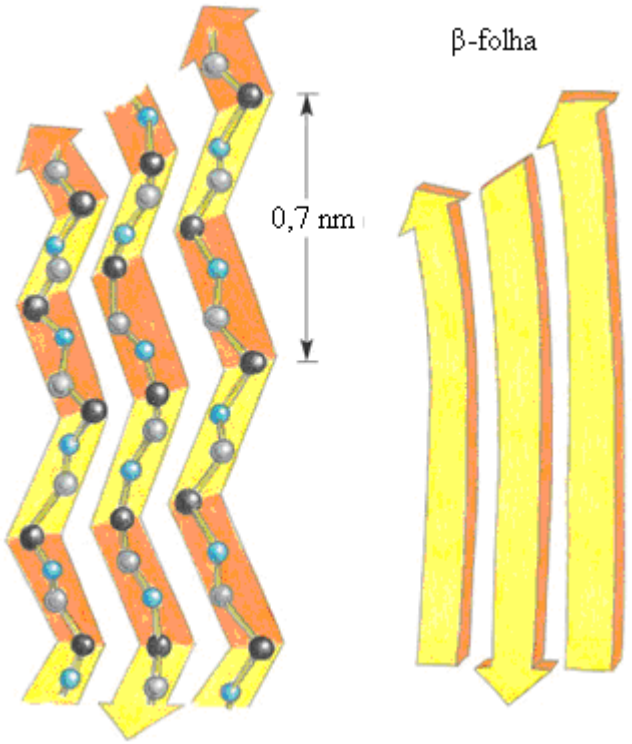

Figura 7 - Estrutura de uma β-folha, na qual os aminoácidos se organizam em longas seqüências dispostas paralelas umas às outras (ALBERTS *et al*, 2003).

c) Estrutura terciária

Esta é a estrutura mais importante para este trabalho. Ela representa as interações existentes entre as estruturas secundárias e, com isso, o dobramento tridimensional, ou seja, a forma que o polipeptídeo assume assim que é sintetizado, com exceção das proteínas fibrosas (HUNTER, 1993). Essa estrutura descreve a posição tridimensional de cada átomo. É freqüentemente referida como sinônimo da seqüência já dobrada e modela as forças não-covalentes que agem entre os aminoácidos, por exemplo, a atração hidrofóbica. Além disso, essa estrutura define a função que o polipeptídeo terá no organismo, pois está relacionada à forma que o polipeptídeo assumirá após o seu dobramento e depende diretamente da estrutura primária da cadeia e pode ser visualizada na Figura 8.

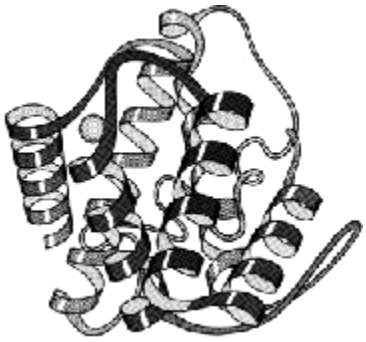

Figura 8 - Estrutura terciária da proteína termolisina [2]

d) Estrutura quaternária

Esse nível de representação é o mais complexo e representa o efeito que as estruturas terciárias têm umas sobre as outras e somente está presente em polipeptídeos multiméricos (GRIFFITHS, MILLER, SUZUKI *et al*, 2000), por destinar-se a mostrar a interação entre as diversas cadeias que compõem um polipeptídeo. A hemoglobina é um caso de proteína que tem estrutura quaternária, pois é composta de quatro cadeias independentes (AVANCINI e FAVARETTO, 1997) que interagem entre si, formando um novelo extremamente complexo. Um outro exemplo de estrutura quaternária, com duas cadeias polipeptídicas é mostrado na Figura 9.

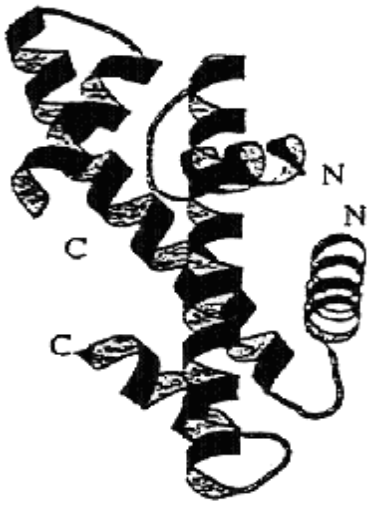

Figura 9 - Exemplo de estrutura quaternária. As duas cadeias polipeptídicas que a compõe esta molécula têm com seus terminais N e C apontados [3].

[2 e 3] Figura extraída e editada da URL: Disponível em: http://kinemage.biochem.duke.edu/~jsr/images/fig014.150.png>. Acesso em: 05 de agosto de 2008.

2.3. DOBRAMENTO DAS PROTEÍNAS

2.3.1. Processo

O processo do dobramento de proteínas é basicamente a acomodação dos átomos da proteína, que interagem uns com os outros, repelindo-os ou atraindo-os mutuamente até que todas as forças cheguem ao equilíbrio. Isso dará origem a estrutura terciária do polipeptídeo e tem relação direta com a funcionalidade da molécula (CLOTE e BACKOFEN, 2000).

No contexto de polipeptídeos, essa estrutura terciária também é conhecida como conformação nativa, e é a forma tridimensional que o polipeptídeo vai assumindo à medida que vai sendo sintetizado (LODISH, BERK, MATSUDAIRA *et al*, 2000). Essa estrutura é feita de tal maneira que evita que o polipeptídeo seja dissolvido por meios solventes – por exemplo, a água – e tenta garantir a coesão entre os átomos, evitando que outro composto químico destrua a cadeia. Essa conformação nativa é tal que o polipeptídeo globular deixa pouca área hidrofóbica em contato com o meio polar (EISENBERG, WEISS e TERWILLINGER, 1984). Para se atingir tal forma, porém, nem sempre é possível contar somente com a interação existente entre os átomos (LODISH, BERK, MATSUDAIRA *et al*, 2000). Apenas as proteínas que possuem um domínio único conseguem se dobrar sem a ajuda de nenhuma estrutura intracelular adicional (CHANDRU, DATTASHARMA e KUMAR, 2003; PEDERSON 2000). É bastante provável que muitos polipeptídeos não consigam chegar à sua forma nativa espontaneamente (SPARRER, RUTKAT e BUCHNER, 1997).

Durante a síntese protéica, existem outras estruturas intracelulares, chamadas de chaperonas (ELLIS, 1990), que são uma grande classe de proteínas e apresentam duas grandes divisões (LODISH, BERK, MATSUDAIRA *et al*, 2000). A primeira inclui as chaperonas moleculares, cuja função é basicamente auxiliar a montagem/dobramento de polipeptídeos incompletos ou desdobrados, evitando a sua degradação no meio celular ou o dobramento errado, como mostrado em PARSELL, KOWAL, SINGER (1994), em GETHING (1997), e em HARTL (1996). A segunda classe são as chaperoninas, que atuam nos polipeptídeos basicamente como facilitadores de processo de dobramento, segundo THIRUMALAI (1994), e CHAN e DILL (1996). A característica fundamental entre elas é que sua função surge apenas no processo do dobramento, sem fazer parte do polipeptídeo funcional após o processo de montagem/dobramento estar completo. As chaperonas não carregam informação para a síntese protéica, assim como o RNA. Elas são atraídas para o polipeptídeo durante a síntese deste, principalmente pelos seus resíduos hidrofóbicos (FINK, 1999), mas não influenciam

sua estrutura primária. Elas podem ou não atuar no polipeptídeo, não por fazê-lo chegar à sua conformação nativa, mas sim evitando que ele assuma uma conformação errada, seja por se agregarem a outras cadeias (FINK, 1999; ELLIS, 1990) ou mesmo por causa da própria interação entre os átomos. A utilização de tais estruturas para modificar a dinâmica do dobramento é um fator bastante complicador no estudo de tal processo, pois seu completo funcionamento não foi ainda totalmente elucidado.

Em termos práticos, pode-se verificar a estrutura terciária de uma proteína por cristalografia de raios-X ou por ressonância magnética (BERG, TYMOCZKO e STRYER, 2002) ou ainda por dispersão rotatória óptica (BAXEVANIS e OUELLETTE, 2001). Há ainda métodos que permitem acompanhar o processo de dobramento, durante a sua execução, como mostrado por UNDGAONKAR e BALDWIN (1988) e RODER, ELÖVE e ENGLANDER (1988).

Existe também o processo contrário ao dobramento, denominado “desnaturação” (COOPER, 2000). Nesse processo, os polipeptídeos saem de sua conformação nativa, podendo ou não voltar a se dobrar na sua forma funcional. As causas da desnaturação são quaisquer modificações no meio externo à molécula que possam fazer com que o equilíbrio que mantém a estrutura se perca. Alterações como mudança de temperatura, pH, radiação, teor de uréia, álcool, podem levar à desnaturação (ALONSO e DILL, 1991; ANFINSEN, HABER e WHITE, 1961). Com o retorno do ambiente às suas condições iniciais, ou seja, com a interrupção da ação do fator desnaturante, a cadeia tende a retornar a seu estado nativo (PEDERSEN, 2000; TANG, 2000). Um exemplo simples desse processo é a adição de calor à albumina ao se esquentar um ovo. A clara do ovo é composta de água e albumina. A albumina é uma proteína polar, portanto solúvel em água. Isso faz com a interação da albumina com a água permita a formação de um composto homogêneo, como, de fato, é a clara do ovo após a síntese. Quando se esquenta o ovo, provoca-se a desnaturação da albumina que, mesmo após retornar à temperatura original, não consegue mais se dobrar em sua conformação nativa. Nesse caso, a conformação agora não mais é polar, mas sim apolar, tornando-se insolúvel em água, conferindo a aparência esbranquiçada característica da clara do ovo, após seu cozimento.

O retorno do polipeptídeo à sua forma original espontaneamente só é possível em proteínas de domínio único (PONTIN e RUSSEL, 2002). Em proteínas maiores, com a presença de diversos domínios, existe uma grande chance de que o desdobramento não possa ser reversível, pois a proteína depende não só das condições do ambiente, mas também de outras organelas intracelulares (CHANDRU, DATTASHARMA e KUMAR, 2003).

Dentro da estrutura terciária ainda se pode identificar outra subdivisão, chamada de domínio (WETLAUFER, 1973). Tais domínios têm estrutura terciária própria e têm a capacidade de se dobrar sozinhos, dependendo apenas da interação entre seus átomos (WETLAUFER, 1973; PONTIN e RUSSEL, 2002). Cada polipeptídeo, ao formar sua estrutura terciária, exibe pelo menos um domínio, como a ribonuclease e a mioglobina, com apenas um domínio (COOPER, 2000). Em geral, os domínios apresentam restrições em seus tamanhos, variando de 36 a mais de 500 resíduos (SAVAGEAU, 1986). A importância fundamental deles é a definição da função no organismo daquela parte do polipeptídeo ou da cadeia inteira, no caso de esta ser pequena ou composta de apenas um domínio. Um aspecto interessante desses domínios é o aparecimento de cavidades, ou reentrâncias, na sua estrutura globular (ALBERTS, JOHNSON, LEWIS *et al*, 2002). Tais reentrâncias são os pontos em que os domínios relacionam-se com os seus ligantes, ou seja, com as substâncias com as quais eles devem interagir.

Utilizando-se como exemplo a proteína representada na Figura 8, apenas uma “fita” representa todas as suas moléculas. Na verdade, essa “fita” representa apenas a cadeia central da proteína, de certa forma desprezando as cadeias laterais. Essas cadeias, assim sendo, têm um papel fundamental na formação da disposição tridimensional que originará o domínio. As reentrâncias são pontos em que existe uma concentração de cadeias laterais, as quais juntas, propiciam o desempenho da função da proteína, ou seja, permitem que estas interajam com outros compostos químicos. Um exemplo simples de domínio, com a sua reentrância e com a disposição das cadeias laterais, pode ser visto na Figura 10.

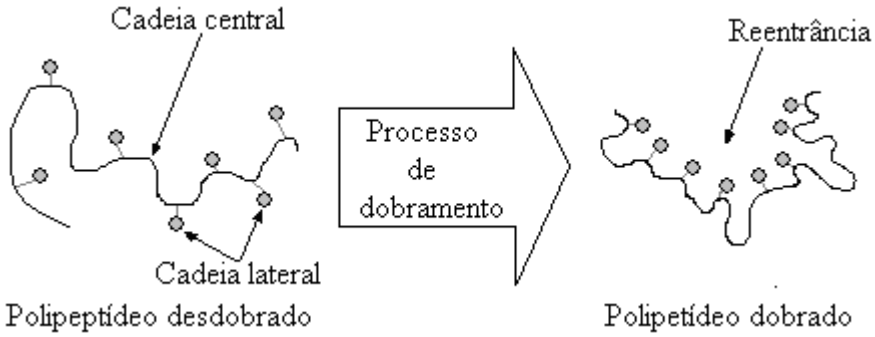

Figura 10 - Formação da reentrância no processo de dobramento.

A reentrância mostrada na Figura 10 poderia ser, hipoteticamente, uma hemoglobina. Ela seria responsável por atrair a molécula de oxigênio, que se encaixaria na reentrância, por causa da concentração e do posicionamento conveniente das cadeias laterais, que então

atrairiam o oxigênio, para fazer o seu transporte. Em geral, o aparecimento dos *binding sites* está intimamente relacionado aos domínios. Por exemplo, na proteína denominada Src, que conta com quatro domínios, dois deles, o SH2 e o SH3, têm funções regulatórias enquanto os demais têm funções catalíticas (ALBERTS, JOHNSON, LEWIS *et al*, 2002).

A incorporação da formação dos domínios no estudo de problema do dobramento de proteínas (também chamado de problema de predição de estrutura terciária) pode potencializar tal estudo. Sabendo-se de antemão com qual molécula o polipeptídeo dobrado deve interagir indicando qual a construção do dobramento que atendesse a tal necessidade, deixaria a solução do problema do dobramento razoavelmente mais simples, por limitar o espaço de busca.

A não-formação de tais estruturas pode ter duas conseqüências diretas. A primeira é a falta daquele polipeptídeo no organismo, ou seja, falha do desempenho da função original. E a segunda é um potencial efeito colateral, tal como desempenhar parcialmente a sua função, ou mesmo, uma função errada. Nesses casos, ocorre o aparecimento de diversas anomalias que, muitas vezes, manifestam-se como doenças. Exemplos dessas doenças são a doença crônica do fígado, causada pelo dobramento parcialmente errado da proteína inibidora anti-tripsina α1 (WELCH e HOWARD, 2000); a fibrose cística (THOMAS, KO e PEDERSEN, 1992) e a anemia falciforme (BERG, TIMOCZKO e STRYER, 2002); alguns tipos de câncer (PIETZSCH, 2002), a doença de Huntington (RASO e KING, 2000), o mal de Alzheimer (HASHIMOTO, ROCKENSTEIN, CREWS *et al*, 2003) e a encefalopatia espongiforme bovina, também conhecida como doença da vaca louca (THOMASSON, 2007).

O processo do dobramento de um polipeptídeo é, portanto, extremamente complexo, seja no âmbito da simples síntese protéica, seja na investigação do porquê de certas proteínas assumirem um dobramento errôneo. Ao mesmo tempo, tal estudo é crucial tanto para o entendimento do funcionamento das doenças relacionadas quanto para o projeto de substâncias que consigam corrigir a anomalia.

2.4. PROBLEMA DA PREDIÇÃO DA ESTRUTURA TERCIÁRIA

2.4.1. Introdução

O problema da predição da estrutura terciária, também conhecido como problema do dobramento de proteínas, é o segmento da ciência que se ocupa em estudar como os

polipeptídeos chegam à sua conformação tridimensional nativa (DOBSON, 2000). Em geral, como no objetivo deste trabalho, busca-se encontrar as estruturas terciárias mais prováveis de representar um polipeptídeo em seu estado nativo, partindo-se apenas da informação contida na sua estrutura primária. Ou ainda buscar pistas do comportamento assumido pela cadeia ao se dobrar, de forma a permitir que novas pesquisas incorporem novas regras que possibilitem chegar à solução de tal problema. Esse problema tem sido um dos maiores desafios para a Biologia e a Bioinformática (NICOSIA, 2004). Computacionalmente, objetiva-se aprimorar a busca pela solução do problema, contando-se com o poder de cálculo dos computadores. Em geral, como será explorado a seguir, até o modelo mais simples de representação computacional de polipeptídeos envolve a busca por um número exageradamente grande de estados que acaba por dificultar muito e, às vezes, a inviabilizar a obtenção da solução do problema (BERGER e LEIGHT, 1998). Assim sendo, o foco da pesquisa sobre o dobramento de proteínas é buscar métodos que possam representar melhor o processo de dobramento e/ou modelos que permitam a representação fidedigna da estrutura polipeptídica, sem perder a viabilidade computacional.

2.4.2. Paradoxo de Levinthal

A idéia mais básica proposta para se abordar o Problema do Dobramento de Proteínas seria a de testar todos os dobramentos possíveis de forma seqüencial. Tal paradoxo é basicamente uma questão proposta por LEVINTHAL (1968), em que se descarta a utilização da busca aleatória que, no pior caso, torna-se uma busca exaustiva, passando por todas as possíveis ocorrências do dobramento. O paradoxo estabelece que uma proteína deveria requerer um tempo exponencial, proporcional ao tamanho da estrutura primária, para atingir sua conformação nativa. Mas, o que acontece, na realidade, é que elas se dobram em tempos que variam desde frações de segundo a minutos (JAENICKE, 1987). Levinthal sugere que tal tempo exponencial seria necessário devido ao número de graus de liberdade envolvidos na questão e que os polipeptídeos teriam que fazer uma busca por todos os dobramentos possíveis até encontrar um que satisfizesse a condição de ser o dobramento nativo. Segundo, um cálculo simples poderia ser feito para calcular o tal tempo (ZWANZIG *et al*, 1992). Considerando que cada aminoácido de uma cadeia tem um número n de graus de liberdade, o número de possíveis conformações do polipeptídeo seria n^{aa-1}, onde aa é o número de aminoácidos. Este é basicamente um cálculo combinatorial de quantos elementos diferentes é possível se conseguir dado o número de elementos e o número de estados que cada elemento

pode assumir. Tomando-se, por exemplo, uma seqüência com 101 aminoácidos, e supondo que cada um deles possa ter somente três posicionamentos em relação ao aminoácido anterior, o número de conformações possíveis seria de 5×10^{47}. Mesmo que o polipeptídeo pudesse tentar aleatoriamente todas as combinações, sem repetição, a uma taxa hipotética de 10^{13} combinações por segundo, desempenho apenas conseguido por supercomputadores como o *RoadRunner* (pertencente exercito norte-americano) que processa 10^{15} flops/s, levaria 10^{27} anos para se completar tal busca. Levinthal concluiu que deveria haver, portanto, um mecanismo mais inteligente de busca, pois a busca exaustiva pura não possibilitaria a síntese protéica na ordem de segundos, como acontece nos organismos.

Tal teoria foi largamente criticada e diversos pontos foram apontados como falhos, principalmente por não levar em consideração nenhum parâmetro biológico (ZWANZIG *et al*, 1992; BROWN e KLEE, 1971; DYSON, RANCE, HOUGHTEN *et al*, 1988). Entretanto, a sua principal utilidade é levantar a questão que, de fato, não se pode simplesmente tentar gerar indefinidamente todas as conformações possíveis até se achar a melhor. Deve-se, porém, encontrar um método que utilize melhor as características já fornecidas pela estrutura do polipeptídeo, de modo a acelerar a busca da solução para o dobramento.

2.4.3. Teoria da hipótese termodinâmica

A busca da conformação nativa de uma proteína através de busca exaustiva simples é praticamente inviável, seja pela sua complexidade conceitual ou mesmo pela quantidade de dados envolvidos. Uma das formas de viabilizar tal estudo é a análise termodinâmica da molécula. Acredita-se que a conformação nativa dos polipeptídeos obedece a uma teoria conhecida como hipótese termodinâmica (ANFINSEN, 1973; DILL, 1985; WOLYNES, ONUCHIC e THIRUMALAI, 1995; DILL e CHAN, 1997; FERSHT, 1999). Tal teoria baseia-se na idéia de que os polipeptídeos tendem a assumir uma disposição tridimensional, por meio das forças que atuam entre os átomos ou entre os aminoácidos, que possibilite que a molécula tenha o menor nível de energia livre possível. Em outras palavras, em relação ao meio em que está inserida, a molécula é estável termodinamicamente. Isso significa que, sem perturbações externas, a molécula continuaria indefinidamente na mesma forma dobrada, pois já estaria estável.

Sabendo-se que os polipeptídeos são sintetizados em meio aquoso (GIBAS e JAMBECK, 2001), que eles tendem a assumir uma forma em que sua molécula esteja estável

(ELLIS, 1990), que essa estrutura tridimensional é baseada na estrutura primária e que os polipeptídeos tendem a assumir um estado de menor energia livre (ANFINSEN, 1973) muda-se a abordagem de tratamento do problema do dobramento de proteínas. Em vez de se tentar chegar ao dobramento nativo tentando reproduzir o processo intracelular, o que é extremamente difícil, a idéia passa então a tentar chegar à estrutura terciária partindo-se da estrutura primária, sem depender da reprodução do ambiente celular para reproduzir o processo de dobramento, mas sim se considerando o processo termodinâmico de interação dos átomos.

Um conceito importante na análise termodinâmica é a energia livre, também chamada de energia livre de Gibbs (KLOTZ, 1955). Essa teoria mede a espontaneidade de uma reação química acontecer. Em outras palavras, por meio da equação da energia livre, é possível definir qual será a direção da reação química, se os reagentes tendem a reagir espontaneamente e formar um produto, ou se tendem a ficar sem reação.

A equação 1 mostra a energia livre de Gibbs e representa a união conceitual da primeira e da segunda leis da termodinâmica, aplicável para reações isotérmicas (LODISH, BERK, MATSUDAIRA *et al*, 2000).

$$\Delta G = \Delta H - T\Delta S \qquad (1)$$

Onde ΔG é a variação de energia livre, ΔH é a variação da entalpia, ou da capacidade calorífica, ou ainda a energia armazenada, ela representa a energia das ligações que unem os átomos uns aos outros, T é a temperatura (que, no caso de ambiente celular, é constante) e ΔS é a variação de entropia, ou desordem dos componentes do sistema.

Se a desordem do sistema é grande, a entropia dele também o é. Por outro lado, quanto mais organizado o sistema se torna, mais a entropia diminui.

Segundo a termodinâmica química, para que uma reação possa ocorrer espontaneamente, em temperatura constante, é necessário que a variação de energia livre (ΔG) seja negativa (LODISH, BERK, MATSUDAIRA *et al*, 2000). Reduzindo-se a energia livre do conjunto produto-reagente, torna-se inviável a reversão da reação, ou o desdobramento do polipeptídeo, sem que se adicione energia de alguma forma. Isso explica o exemplo da desnaturação e redobramento da albumina, citado na seção 2.3, com o cozimento do ovo. A proteína só se desdobra quando aquecida (nesse caso) e, depois de cessar a fonte de calor, ela torna a se dobrar.

A diferença de energia livre entre uma proteína desdobrada e uma dobrada é pequena, em torno de -5 a -15 kcal/mol (LATTMAN e ROSE, 1993), quando comparada com a energia

que une a ligação peptídica (a ligação covalente entre um carbono e um nitrogênio de diferentes aminoácidos), que é de 70 kcal/mol (LODISH, BERK, MATSUDAIRA *et al*, 2000). Isso significa que a equação 1 foi satisfeita e o polipeptídeo tem pequena estabilidade em relação ao meio aquoso. Mesmo assim, o estado dobrado é melhor, em termos termodinâmicos, do que o estado desdobrado.

No processo de dobramento, a quantidade de ligações químicas se mantém praticamente constante. Os resíduos polares, quando não interagem com o meio aquoso, interagem entre si, por meio de pontes de hidrogênio. Como o tipo de ligação é o mesmo, a energia das ligações também é. Portanto, a variação de entalpia entre a proteína desdobrada e a dobrada é mínima, praticamente ínfima. O processo passa a então depender da variação da entropia. Do ponto de vista do polipeptídeo somente, se não há variação da entalpia, não poderia haver dobramento, pois, assumindo-se um estado de maior organização, seria diminuída a entropia (ΔS negativo) e, com isso, a equação 1 se tornaria positiva. Porém, deve-se analisar o conjunto soluto-solvente. A água, sem soluto, tende a assumir o seu estado de maior entropia. Ela distribui suas moléculas igualmente por todo o volume disponível, permitindo livre movimentação das moléculas e realizando pontes de hidrogênio entre elas e algum possível soluto polar, caso esteja presente. Inserindo-se um soluto hidrofóbico em meio aquoso, como o meio intracelular (GIBAS e JAMBECK, 2001), diminui-se a entropia da água (LODISH, BERK, MATSUDAIRA *et al*, 2000), ou seja, o estado de máxima entropia já não mais pode ser atingido, pois existe algum soluto insolúvel inserido. Os resíduos hidrofóbicos conseguem se combinar entre si sem problemas, mas não conseguem se ligar à água. Portanto, o meio aquoso tende a repelir o elemento causador da perturbação. No caso do dobramento, a proteína – composta tanto de aminoácidos polares quanto hidrofóbicos – mostra uma tendência de minimizar o contato dos aminoácidos hidrofóbicos com água, para se diminuir tal perturbação. O dobramento, desse modo, tende a fazer com que os resíduos hidrofóbicos fiquem dentro do novelo formado, interagindo entre si, protegidos do contato com a água pelos aminoácidos polares, que podem interagir com a água sem problemas. Como a perturbação do soluto na água é minimizada, a entropia do sistema é superior (mais próxima da entropia máxima possível da água) com a proteína dobrada do que se ela estivesse desdobrada. Lembrando-se que, durante o dobramento, a entalpia praticamente não se altera, e a entropia tende a aumentar, tornando a equação da energia livre de Gibbs negativa. Pelo mesmo motivo, a estrutura se torna estável, pois o meio aquoso simplesmente não permite a dissociação da estrutura. A estrutura nativa seria, concluindo, uma conseqüência da perturbação do soluto na água e não a causa do dobramento em si. A essa repulsão dos

aminoácidos hidrofóbicos pela água dá-se o nome de efeito hidrofóbico (LEHNINGER, NELSON e COX, 1998) que, por ajudar a aumentar a entropia do sistema, também é chamado de força entrópica.

Até os anos 80, acreditava-se que as forças eram como pontes de hidrogênio, forças iônicas, forças de Van Der Vaals e o efeito hidrofóbico (ANFINSEN e SCHERAGA, 1975; DILL, OZKAN, WEIKL *et al*, 2007). Hoje, acredita-se que o efeito hidrofóbico tem um papel muito superior às outras forças na formação do dobramento (DILL, 1999).

Como a interação entre aminoácidos hidrofóbicos é amplamente incentivada pela ação do meio aquoso, acredita-se que, maximizando o número de interações hidrofóbicas entre os aminoácidos, seja possível encontrar o menor estado de energia livre possível. Portanto, este trabalho se baseia na busca de conformações, ou estruturas terciárias, que proporcionem a maximização das interações entre aminoácidos hidrofóbicos.

2.4.4. Modelos de representação de polipeptídeos

O objetivo dos modelos de representação é possibilitar meios de processar computacionalmente dados sobre os polipeptídeos. Os modelos basicamente buscam um equilíbrio entre a quantidade de detalhes representados e a viabilidade computacional. Um modelo, para ser eficaz, deve obedecer a dois conceitos (PEDERSEN, 2000):

a) O primeiro é o de exibir semelhança estrutural com a cadeia original. Naturalmente um modelo tem que ter detalhes suficientes para representar os elementos mais básicos do polipeptídeo, sob o risco de ser um modelo falho.

b) O segundo ponto a ser observado é que o modelo deve ter semelhança comportamental com a cadeia original. Isso é necessário para chegar a resultados semelhantes ao polipeptídeo original ou, pelo menos, conseguir uma pista coerente do caminho a ser seguido.

Existem basicamente dois tipos de representação, a analítica e a discreta. A analítica representa o extremo de riqueza de detalhes e, por conseguinte, a maior dificuldade computacional. O modelo discreto trabalha basicamente reduzindo detalhes e graus de liberdade de movimentação dos elementos. A seguir ambos serão detalhados.

2.4.4.1. Modelo analítico

Este modelo é o mais completo com o qual se pode representar computacionalmente um polipeptídeo ou qualquer molécula. Ele modela todas, ou a maioria das informações relacionadas, em nível atômico ou em um nível que agrupe os átomos formando conjuntos, como no caso de se representar aminoácidos. Este modelo foi apresentado por Richardson (1981); Ngo, Marks e Karplus (1994); e Pedersen (2000), embora existam outros.

Cada átomo é representado como uma entidade no espaço tridimensional. As ligações entre cada átomo, ou cada grupo de átomos (como nos aminoácidos), têm que ser descritas. Isto é feito computando-se as informações de comprimento de ligação, tipo (covalente, iônica, van der Waals, hidrofóbica), energia da ligação, ângulo de torção ou ângulos diedrais (BERG, TYMOCZKO e STRYER, 2002) e as restrições no movimento de tais ângulos, pois há que se considerar que os átomos também ocupam lugar no espaço, limitando o movimento dos demais (RICHARDSON, 1981). Tais ângulos são denotados pelas letras gregas Φ, ψ, ω e τ e são mostrados na Figura 11, onde:

a) A letra Φ representa o ângulo de rotação da ligação covalente entre os átomos nitrogênio do grupo amina e o carbono alfa.

b) A letra ψ denota o ângulo de rotação da ligação também covalente entre os átomos carbono, do grupo carboxil, e o carbono alfa.

c) O ângulo ω é a rotação da própria ligação peptídica, ou seja, da ligação que une os átomos carbono do grupo carboxil e o átomo nitrogênio do grupo amina de aminoácidos diferentes.

d) A letra τ é o ângulo formado entre as ligações covalentes no grupo amina e carboxil com o carbono alfa, no mesmo aminoácido.

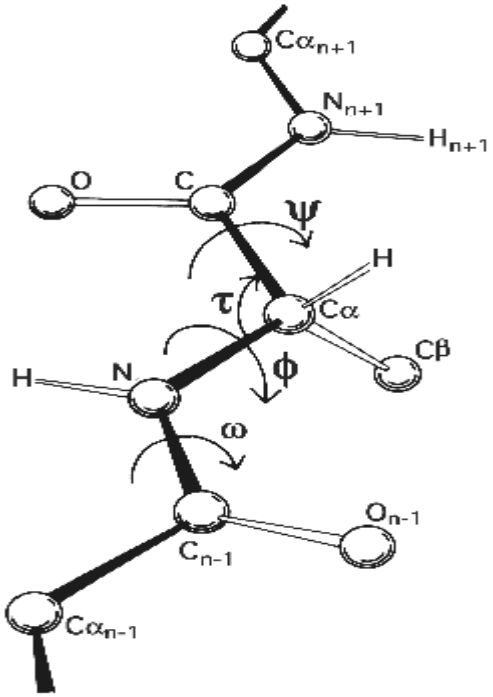

Figura 11 - Exemplo de modelo analítico, proposto por RICHARDSON (1981).

Naturalmente, esse modelo engloba uma quantidade enorme de dados a serem computados e, por considerar que pode existir flexão em diversos ângulos, a quantidade de possibilidades para os possíveis dobramentos é um número de fato gigantesco. Além disto, a sua função de energia livre ser extremamente complexa, pois tem que se levar em conta todas as variáveis representadas.

2.4.4.2. Modelo discreto: modelo 2DHP

Este modelo, também chamado de modelo bidimensional hidrofóbico-polar, foi introduzido por DILL, BROMBERG, YUE *et al*, (1995). É muito provavelmente o modelo mais simples de representação de estrutura terciária de polipeptídeos, porém é um dos mais estudados. Mesmo simplificando enormemente, o modelo ainda consegue representar algumas características comportamentais dos polipeptídeos (DILL, 1985; DILL, BROMBERG, YUE *et al*, 1995). Apesar de não ter um grau de detalhes tão alto quanto o modelo analítico, a sua confiabilidade, em relação ao modelo analítico, não é tão diferente. O modelo analítico tem que levar em consideração uma série de aproximações e convenções que sua legitimidade é colocada tão à prova quanto no modelo discreto. O 2DHP (*Bidimensional Hydrophobic-Polar*

model) é baseado em uma treliça bidimensional, que tem por objetivo comportar os aminoácidos, bem como suas ligações peptídicas. A vantagem primária desse modelo é reduzir os graus de liberdade da cadeia, reduzindo o número de computações envolvidas. No 2DHP, cada um nos 20 aminoácidos que formam as proteínas são classificados de acordo com a sua solubilidade e em meio aquoso, ou seja, são caracterizados em polares (hidrofílicos), denotados pela letra P ou hidrofóbicos (não-polares) denotados por H. O objetivo dessa caracterização é a aplicação da Hipótese Termodinâmica, em que a minimização da energia livre é o principal objetivo, e esta é conseguida maximizando-se o número de interações hidrofóbicas.

Neste modelo, as intersecções da treliça definem as possíveis localizações dos aminoácidos. Cada resíduo pode ocupar somente um lugar na matriz. As ligações peptídicas são representadas por arestas que unem os aminoácidos. Tais arestas obrigatoriamente são coincidentes com as linhas da treliça, de modo a não ser possível ter uma ligação na diagonal, pois o modelo permite somente ângulos de -90º, 0º e +90º entre seus elementos. Não é permitido que dois ou mais aminoácidos ocupem o mesmo lugar na treliça, já que nem na natureza é possível ter-se dois resíduos sobrepostos. Tal sobreposição é considerada uma violação do modelo e representa uma colisão de aminoácidos. Infelizmente não há maneira de se eliminar tais colisões da representação, o que acaba por aumentar o número de computações a serem feitas. Por outro lado, possibilita que o espaço de busca seja reduzido. A Figura 12 mostra um exemplo do modelo 2DHP, com todos os seus elementos. Outro elemento importante dessa forma de representação são as interações hidrofóbicas. Elas são representadas pela maior proximidade possível entre dois aminoácidos H na treliça. Diz-se, portanto, que, cada vez que se consegue posicionar dois aminoácidos H próximos entre si, sem que eles estejam em posições consecutivas na cadeia, ocorre uma interação hidrofóbica. Assim como mostrado na Figura 12, convencionou-se representar os aminoácidos H como círculos e os P como quadrados, e o aminoácido inicial da cadeia com um X. A treliça está em linhas pontilhadas, as ligações peptídicas em linhas cheias e as interações hidrofóbicas como elipses vazadas. Neste modelo, de acordo com a Hipótese Termodinâmica, buscam-se as conformações sem colisões que maximizem o número das interações hidrofóbicas.

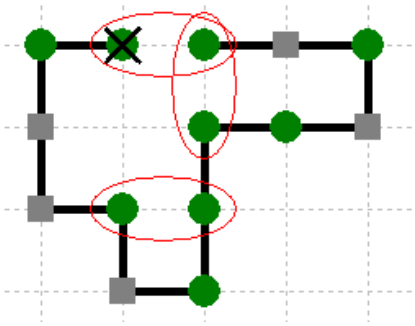

Figura 12 - Exemplo do polipeptídeo HHPPHPHHHHPHPH no modelo 2DHP.

Apesar de promover uma redução no grau de detalhes, esse modelo ainda assim está sujeito a uma considerável quantidade de estados possíveis, ou dobramentos distintos. Na verdade, é o crescimento do espaço de busca que torna o problema tão difícil de ser resolvido. A diferença do número de estados possíveis de um polipeptídeo com N aminoácidos em comparação com outro de N+1 aminoácidos é muito grande. Naturalmente, esse número é inferior ao número de estados do modelo analítico, porém ainda faz com esse busca exaustiva nesse modelo leve a um problema classificado como NP-difícil (CRESCENZI, GOLDMAN, PAPADIMITRIOU *et al*, 1998) A sua versão tridimensional, o 3DHP, que tem basicamente as mesmas regras, porém a matriz utilizada possui três dimensões, é considerado um problema NP-Completo (BERGER e LEIGHT, 1998), ou seja, razoavelmente mais complexo de ser resolvido do que o 2DHP.

A equação que representa o número total de combinações de diferentes dobramentos que podem ser gerados utilizando um polipeptídeo de tamanho genérico N será explorada no capítulo 3, e tem forma exponencial. Isso significa que, mesmo que o aumento do número de aminoácidos sob análise seja linear, o aumento de estados a serem analisados será exponencial. O problema, portanto, torna-se intratável, mesmo com poucos aminoácidos na cadeia, como mostrado na Tabela 2. Esta tabela mostra o número de estados possíveis, e uma estimativa de tempo de processamento, levando em consideração uma situação hipotética de se poder processar um estado a cada 1 nanosegundo. Essa tabela tem por objetivo mostrar um cenário otimista de uma improvável tentativa de se explorar o problema do dobramento pela abordagem da busca exaustiva pura, assim como apontado por LEVINTHAL (1968).

Para se propor soluções para esse problema, diversos artifícios foram construídos para não só reduzir o problema da explosão combinatória, mas também fazer uma exploração mais rápida e menos custosa computacionalmente, como será mostrado no Capítulo 3.

Tabela 2 - Número de combinações e estimativa de tempo de análise de acordo com o número de aminoácidos. A estimativa baseia-se em um caso hipotético de busca exaustiva, em que cada dobramento seria processado em 1 ns

Nº de aminoácidos	**Nº de dobramentos possíveis**	**Tempo estimado para explorar todo o espaço de busca**
2	4	4,00E-09 segundos
5	256	2,56E-07 segundos
10	262144	2,62E-04 segundos
15	2,68E+06	2,68E-01 segundos
20	2,74E+11	2,75E+02 segundos
25	2,81E+14	3,26E+00 dias
30	2,88E+17	9,14E+00 anos
35	2,95E+20	9,36E+03 anos
40	3,02E+23	9,58E+06 anos
45	3,09E+26	9,81E+09 anos

2.4.4.3. Outras abordagens de modelos discretos

Existem ainda outros modelos discretos, cujo grau de detalhes é um pouco maior do que o 2DHP. Entre eles, existe o 3DHP (*Tridimensional Hydrophobic-Polar model*) proposto por Dill *et al* (1995), em que se descreve um modelo muito similar ao 2DHP, com as mesmas regras, porém com uma treliça de três dimensões. Esse modelo, porém, foi provado ser de complexidade NP-Completo por Berger e Leight (1998).

O modelo LPE (L*attice Polymer Embedding*), proposto por UNGER e MOULT (1993a), é um problema NP-difícil, e é baseado em uma grade cúbica, parecido com o modelo HP. Neste modelo, cada par de aminoácidos tem um coeficiente de afinidade e o objetivo da função de energia é minimizar o produto desse coeficiente pela distância entre os aminoácidos . No modelo PH (P*erturbed Homopolymer*), proposto por SHAKHNOVICH e GUTIN (1993) e SOCCI e ONUCHIC (1994), não se considera somente o efeito hidrofóbico, mas sim a interação entre aminoácidos de mesma polaridade, H e H ou P e P.

No modelo chamado HPSCM (*Hydrophobic-Polar Side Chain Model*), proposto por BROMBERG e DILL (1994). Cada aminoácido é separado em duas entidades distintas, a

cadeia central e a lateral. A cadeia lateral é que exibe a característica da hidrofobicidade e a cadeia central se encarrega de ocupar o espaço necessário para impor limitações à acomodação das cadeias laterais.

O modelo HP-TSSC (*Hydrophobic-Polar Tangent Spheres Side Chain Model)* é baseado no modelo HP, porém não utiliza nenhum tipo de grade para o posicionamento dos aminoácidos (HART e ISTRAIL, 1997). Neste modelo os aminoácidos são dispostos na forma de um grafo tridimensional, em que a cadeia central e a lateral de cada aminoácido são transformadas em esferas de mesmo raio. O objetivo, assim como no modelo HP, é maximizar o número de iterações hidrofóbicas.

O modelo CGE (C*harged Graph Embedding)* foi proposto por FRANKEL (1993) e utiliza uma grade tridimensional e incorpora o conceito de cargas elétricas aos aminoácidos. As conformações resultantes têm o problema de poder ter suas iterações entre pares de aminoácidos cruzando ligações entre outros pares. Porém, as iterações entre os pares perdiam a validade caso a ultrapassasse uma determinada distância.

O modelo HPNX (BACKOFEN, WILL e BAUER, 1999; BORNBERG e BAUER, 1997) também é baseado na mesma grade do modelo HP, porém faz uma classificação diferente dos 20 aminoácidos. Ao invés de somente considerar os hidrofóbicos (H) e polares (P), divide os aminoácidos em hidrofóbicos (H), positivos (P), negativos (N) e neutros (X). A energia do dobramento não é calculada apenas pelas iterações hidrofóbicas, mas também pelas iterações entre os pares dos demais tipos. Em geral considera-se que iterações entre aminoácidos H e H decrescem a energia do dobramento em -4,0, as iterações entre P e P e N e N aumentam a energia livre em 1,0 e as iterações entre N e P decrescem em 1,0 a energia. O objetivo continua sendo minimizar a energia livre total, portanto quanto mais iterações HH melhor ao dobramento sem, no entanto, desprezar o valor das demais iterações.

2.5. MÉTODOS DE BUSCA

2.5.1. Introdução

Existem diversos problemas cujas soluções têm o potencial de ser encontradas por meio de métodos de busca. Há a necessidade de analisar cada problema de modo a se definir quais métodos podem solucionar o problema. Um exemplo é mostrado na Figura 13 que

hipoteticamente representa um mapa geográfico, com cidades indicadas pelas letras de A a G e as setas as estradas que interligam as cidades, sendo que o sentido das setas representa o sentido possível de se trafegar nessas estradas.

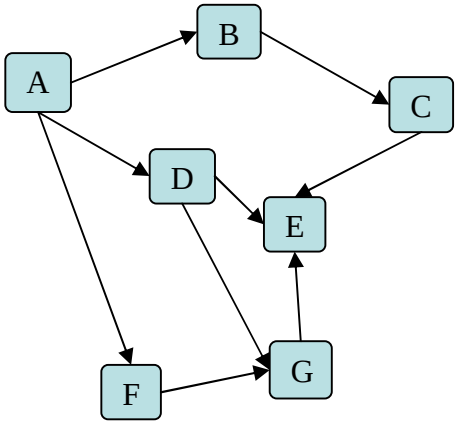

Figura 13 - Exemplo de representação da interligação de cidades hipotéticas.

Imaginando-se que o problema a ser resolvido é encontrar o melhor caminho entre as cidades A e, diversos cenários, ou candidatos à solução, podem ser elaborados. Naturalmente, o conceito de "melhor" é bastante relativo e tem que ser definido *a priori* para que se chegue à solução do problema, supondo que o melhor caminho seja, então, o caminho que passe por menos cidades apenas para incrementar o exemplo. Dessa forma, pode-se então definir os cenários possíveis que representem a solução, como mostrado na Tabela 3. Nesta tabela, todas as possibilidades, ou trajetos, de A até E estão listadas, juntamente com o número de cidades que são cobertas pelo trajeto que, nesse caso, é o parâmetro que indica o sucesso ou não do cenário candidato.

Tabela 3 - Levantamento dos possíveis cenários para solução.

Caminho	**Trajeto**	**Número de cidades**
1	A-B-C-E	4
2	A-D-E	3
3	A-D-G-E	4
4	A-F-G-E	4

Com base na Tabela 3, é fácil de identificar a única solução do problema. Claramente, o caminho número 2 é o melhor por passar pelo menor número de cidades, neste caso três cidades. Utilizando a nomenclatura mais apropriada (RUSSELL e NORVIG, 2004), as

cidades são os “estados” da busca. Os trajetos são as potenciais soluções. Esse tipo de problema é então chamado de busca em espaço de estados (LUGER, 2004).

Esse é um problema ideal para se utilizar um método de busca, pois existem diversas soluções que levam à resposta, mas apenas um subconjunto é considerado a solução para o problema. Tendo-se essa modelagem, baseada no mapa da Figura 13, é possível definir dois tipos de análise: uma busca exaustiva ou uma busca heurística.

A busca exaustiva tem por objetivo analisar absolutamente todos os cenários possíveis e com isso ser capaz de determinar se existe solução para o problema e, se sim, qual ela é. Para isso, há a necessidade de análise de todos os estados possíveis, por meio de uma função em geral extremamente simples (JONES e PEVZNER, 2004), como a lista mostrada na Tabela 1. Obviamente, este exemplo é simples, pois existem apenas sete estados e quatro potenciais soluções. Esse tipo de análise é feito quando se tem pouca informação sobre o problema. Em geral, sabe-se apenas a modelagem e o objetivo do problema. Nesse caso, seriam o mapa e o conceito do melhor caminho, respectivamente.

Portanto, para se chegar ao conjunto de respostas, é necessário analisar todos os estados possíveis do problema e relacionar todas as potenciais soluções, pois somente assim se pode afirmar com certeza que o algoritmo encontrou a solução desejada. A vantagem direta desse método é que com poucas pistas é possível chegar a absolutamente todas as respostas. A desvantagem, que em grande parte dos casos torna a sua utilização proibitiva, é que a necessidade computacional pode ser extremamente grande, pois, geralmente, o tempo para se percorrer todos os estados interconectados aumenta em uma taxa superior ao aumento do tamanho do problema. Para o exemplo citado, em que existe somente sete estados e quatro trajetos, é possível utilizar a busca exaustiva. Se fossem 1.000 estados, com milhares de caminhos entre si, provavelmente essa abordagem já não é a mais apropriada, simplesmente por requerer um tempo demasiadamente grande para chegar à solução (JONES e PEVZNER, 2004),

A outra abordagem seria a da busca heurística, ou busca com informação (RUSSEL e NORVIG, 2004). Diferentemente da busca exaustiva, essa busca necessita de um pouco mais de informações sobre o problema, ou seja, precisa de uma estratégia que utilize o conhecimento específico do problema, e não somente da sua definição (RUSSEL e NORVIG, 2004). Nesse caso, propõe-se uma função, ou heurística, simples, que seria a de avaliar, a cada estado, a distância euclidiana até as cidades vizinhas, até se alcançar o objetivo. Desse modo, partindo-se da cidade A, mede-se a distância até as cidades B, D e F, e escolhe-se qual é a mais próxima. No exemplo, a cidade D é, sem dúvida, a mais próxima. A partir da cidade D

escolhe-se novamente o menor caminho para as cidades vizinhas, porém uma delas já é o objetivo. Assim, com o uso da heurística, a solução A-D-E é encontrada rapidamente, potencialmente mais eficientemente do que com a busca exaustiva. Claro que, neste exemplo, a heurística não é perfeita. Apenas é um meio de chegar ao objetivo, portanto, a busca é mais ou menos eficiente de acordo com a qualidade da heurística utilizada.

Em geral, uma busca heurística é ineficiente quando analisa um número grande de estados, sendo que, no pior caso, ela seria reduzida a uma busca exaustiva. Por outro lado, se ela contiver exatamente as regras que resolvem o problema, ou seja, com eficiência máxima, ela analisará poucos estados, no melhor caso, apenas o número de estados necessários para se chegar à solução. Em outras palavras, sabendo-se aonde se quer chegar, a busca heurística, com certeza, levará até a solução em um tempo igual, ou melhor, que a busca exaustiva.

Na Figura 14 pode ser observada uma comparação entre o tamanho do espaço de busca e a eficiência da função heurística. O ponto A representa o ponto de menor eficiência heurística com o maior espaço de busca, ou seja, a busca exaustiva. O ponto C, pelo contrário, representa a função heurística ideal, na qual o espaço de busca é o menor possível, ou seja, a função heurística perfeita, que chega à solução imediatamente.

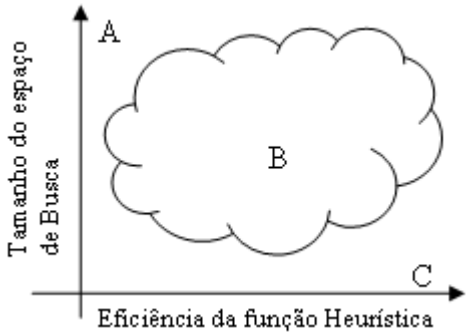

Figura 14 - Relação entre a eficiência da função heurística e seu espaço de busca.

O terceiro ponto, B, representa o meio-termo entre o pior e o melhor caso, ou seja, onde a maioria das buscas em espaço de estados estão situadas. Neste ponto existe um equilíbrio entre o conhecimento limitado modelado na função heurística e o tamanho do espaço de busca resultante

Exemplos de algoritmos de busca heurística seriam os algoritmos A* (pronuncia-se “aestrela”) ou ainda os algoritmos genéticos (JONES e PEVZNER, 2004).

Em ambas as abordagens, no exemplo apresentado, o objetivo da busca era conhecido: ir da cidade A à cidade E passando-se pelo menor número de cidades. Se o objetivo fosse menos específico ou ainda desconhecido – como em se determinar qual seriam todas as

distâncias percorridas entre todas as cidades para todas as cidades – o problema ficaria razoavelmente mais complexo.

No exemplo do dobramento de proteínas, considerando-se uma dada estrutura primária, no modelo 2DHP, explorado anteriormente neste capítulo, e querendo chegar a uma estrutura terciária que maximize o número de interações hidrofóbicas, o problema torna-se muito mais complexo. Deve-se considerar outro aspecto, além de o espaço de estados ser absurdamente grande, como será mostrado a seguir, como não se pode dizer exatamente qual é o objetivo esperado, o problema torna-se muito difícil de ser resolvido. Nesse caso, encontra-se o seguinte dilema: a busca exaustiva é extremamente demorada, mas capaz de fornecer os melhores resultados. Por outro lado, a busca heurística é potencialmente mais rápida, mas limitada ao conhecimento modelado na sua função heurística. Em outras palavras, dado um problema em que não se sabe exatamente quantificar o objetivo e nem se consegue modelar regras que levem ao resultado, é muito difícil construir uma função heurística que avalie o quão perto, ou longe, os estados analisados estão do sucesso da busca. Portanto, o uso de busca heurística, nesse tipo de problema, na qual o problema da predição de estrutura terciária se encaixa, pode trazer resultados de forma mais rápida, mas sem garantia de que eles serão, sem dúvida, a melhor resposta. Tais algoritmos de busca trazem como resposta a melhor resposta dentre o conjunto de estados que eles tiveram a chance de analisar. Porém, se esse conjunto não for o espaço completo de estados não se pode dar garantia nenhuma de que o resultado é, de fato, o melhor que se pode conseguir.

Finalmente, ambas as abordagens têm suas vantagens e desvantagens, porém, neste trabalho, pretende-se extrair o que há de melhor nas duas, em especial do algoritmo A*, de modo a conseguir elaborar um sistema de busca exaustiva aprimorado com uma heurística, para se poder fornecer garantidamente os melhores resultados em tempos razoáveis.

2.5.2. Representação do espaço de estados em forma de árvore

Existem várias formas de representar um espaço de estados. Em geral, busca-se uma forma conveniente, que possa compreender a modelagem do problema e facilitar ao algoritmo de busca e obtenção da solução. Sem dúvida, a forma escolhida para representação dos dados relativos ao problema é crucial no desempenho do algoritmo. A base para um modelo em forma de árvore é a teoria dos grafos (LUGER, 2004). Segundo essa teoria, um grafo é um conjunto de nós e um conjunto de arcos que têm a função de conectar esses nós, assim como

mostrado na Figura 15. Adicionalmente, esses arcos podem ser orientados, ou seja, permitem apenas a ligação no sentido indicado. Quando um nó não recebe nenhum arco, mas tem arcos saindo para outros, ele é considerado um nó “raiz”, e quando ele não tem nenhum arco saindo dele, é considerado um nó “folha”.

Utilizando-se a representação em grafo existe uma especialização chamada árvore. De acordo com DIESTEL (2005), um grafo de n nós é uma árvore se e somente se houver exatamente n-1 arcos.

A representação em árvore é basicamente um grafo orientado, com duas restrições (LUGER, 2004). A primeira é que só pode haver um nó raiz e a outra é que entre dois nós pode haver somente um caminho. Um exemplo é mostrado na Figura 15 onde cada nó que não é folha pode ser considerado uma sub-raiz de uma nova árvore. Isso é interessante do ponto de vista da paralelização do algoritmo, como será explorado a seguir. Uma característica importante é a existência de subárvores dentro da árvore principal, na qual qualquer nó que não seja folha é raiz de uma subárvore. Considerando-se a Figura 15, um exemplo de subárvore poderia ser a composta pelos nós 2, 4 e 5, sendo que o nó 2 seria a raiz dessa subárvore. Este conceito é importante de ser ressaltado, pois como o problema deste trabalho envolve explosão combinatória, o corte inteligente do espaço de busca trata basicamente da eliminação de subárvores. Obviamente, tal corte é mais eficiente quanto menor for a profundidade da raiz desta subárvore. Na Figura 15, se a subárvore iniciada pelo nó 2 pudesse ser eliminada, o espaço de busca seria reduzido para menos da metade. Naturalmente, o principal desafio é fazer tal corte sem eliminar informações relevantes à solução do problema.

Uma característica importante da árvore é a definição de profundidade do nó. Define-se que o nó raiz da árvore tem um profundidade igual a 1 e cada nó descendente tem a sua profundidade acrescida, de modo que, considerando a Figura 15, o nó 1 tem profundidade 1, os nós 2 e 3 profundidade 2 e os nós 4, 5 e 6 têm profundidade 3. Outro índice importante em árvores é o fator de ramificação de cada nó, que é o número de descendentes que ele gera. Na Figura 15, o nó 1 e 2 tem fator de ramificação 2, enquanto que o nó 3 tem fator igual a 1. Outra característica que pode ser incorporada a essa estrutura de árvore é o utilíssimo conceito de custo entre os nós. Retomando o exemplo do grafo representando um conjunto de cidades, o custo poderia ser a distância entre as cidades ou mesmo a dificuldade de se progredir de um nó para outro.

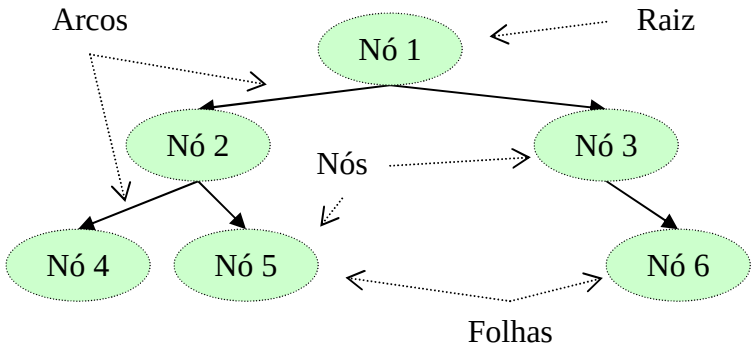

Figura 15 - Exemplo de um grafo em forma de árvore.

2.5.3. Algoritmos de busca sem informação

Considerando-se que este tipo de busca não consegue, ou não pode, incorporar nenhum tipo de conhecimento sobre o problema, qualquer otimização do espaço de busca tem que ser feita apenas sabendo-se as características do espaço de busca. Ou ainda quando não se tem todos os dados da solução. Naturalmente, se houver pistas ou mesmo uma idéia de como deve ser a solução, os métodos descritos a seguir podem ter maior eficiência. Por outro lado, se não existir tal informação, como no caso do problema do dobramento de proteínas, tais métodos perdem praticamente toda a sua vantagem. Isto acontece pois a busca por todo o espaço de busca será inevitável para se ter certeza sobre a melhor solução encontrada, ou seja, a busca seria então reduzida a uma busca exaustiva. Segundo RUSSEL e NORVIG (2004) cinco métodos de busca sem informação são sugeridos, explorados a seguir.

2.5.3.1. Busca em extensão

Na busca em extensão, que também é chamada de busca em largura, a análise inicia-se pelo nó raiz. Este é explorado, em seguida seus sucessores e os próximos sucessores e assim por diante. A característica principal é que todos os nós de um mesmo nível são explorados antes de se progredir para o próximo nível.

Este tipo de busca torna-se cada vez mais útil quanto mais raso estiver o objetivo da busca. Se um dos nós folha representar a solução, esse método não será tão eficiente pois terá que explorar a árvore toda até encontrar a solução. Isto pode, sem dúvida, acabar se tornando uma busca exaustiva. Outro problema seria a memória necessária para comportar a análise desse tipo de problema. Considerando-se o exemplo sugerido em RUSSEL e NORVIG (2004), se o problema resulta em uma árvore semelhante à da Figura 15 com um fator de

ramificação de 10 e supondo que cada nó ocupe 1000*bytes*, uma busca com profundidade 12 exigiria 10peta*bytes*. Tal aspecto seria um empecilho grave ao algoritmo, pois é muito difícil se obter um computador com tal memória. Naturalmente, pode ser que não seja necessário se manter todos os nós em memória simultaneamente. Porém, ainda assim a exploração do espaço de busca encontra outro obstáculo: a explosão combinatória. Neste mesmo exemplo, se fosse possível gerar 10000 nós por segundo, levar-se-ia 35 anos para se gerar a árvore de busca, ou seja, tornaria essa abordagem inviável. No exemplo da Figura 15 a exploração dos nós seria nesta ordem: nó 1, 2, 3, 4, 5 e 6.

2.5.3.2. Busca de custo uniforme

A busca de custo uniforme é basicamente idêntica à busca em extensão, com a diferença que sempre, expande-se o nó com caminho de menor custo. Esse tipo de busca é interessante, pois as informações necessárias a serem modeladas são a identificação do objetivo e o custo de se gerar cada descendente do nó atualmente sob análise. O resultado é a chegada ao objetivo com o caminho de menor custo, ideal para situações como aquela exemplificada na Figura 13. Pode-se descobrir, por exemplo, que o melhor caminho não é simplesmente o que tem menos nós, pois não necessariamente este tem o menor custo. Essa característica tem o potencial de fazer o algoritmo chegar à solução rapidamente, pois algum conhecimento do grafo está codificado no algoritmo. Por outro lado, se o custo de se gerar os descendentes de cada nó é sempre o mesmo, este tipo de busca será então reduzida a uma busca em extensão. Neste caso tem ainda que se contar com a desvantagem de ter que calcular a função de custo, tornando o algoritmo mais lento que a busca em extensão.

2.5.3.3. Busca em profundidade

A busca em profundidade tem um método de exploração que expande o nó com maior profundidade na árvore. Portanto, a avaliação dos nós da Figura 15 seria nesta ordem: nó 1, 2, 4, 5, 3, 6. Este método tem basicamente duas vantagens. A primeira é sobre o consumo de memória que, em relação à busca em extensão, é extremamente mais baixo, no caso do exemplo citado na seção 2.5.3.1 onde a busca em extensão necessita de 10 peta*bytes*, com este método são necessários apenas 118 k*bytes*. Ou seja, 10 bilhões de vezes menos memória. A segunda vantagem é que, pela estrutura da árvore ser sempre mais larga do que profunda, esta busca sempre é mais rápida que a busca em extensão para se chegar às folhas.

Por outro lado, há duas desvantagens. A primeira é que pode ser que, ao se encontrar nós de mesma profundidade e se fazer a opção de explorar um dentre os demais, o algoritmo leve a uma solução subótima ou ainda não resolva o problema, quando talvez outro nó naquele mesmo ponto de decisão pudesse levar rapidamente à solução. A segunda é que, no caso de não se conseguir achar logo à solução, o algoritmo pode se tornar um algoritmo de busca em extensão. Esse o algoritmo é ideal quando se deseja chegar logo às folhas ou quando se pode beneficiar, de alguma maneira, das informações adquiridas da exploração da árvore em profundidade, como para se adiantar algum conhecimento que possa servir para eliminar outros nós do espaço de busca. Por outro lado, em casos em que o grafo cresça ilimitadamente esse método não é recomendado.

2.5.3.4. Busca em profundidade limitada

A busca em profundidade limitada é basicamente a mesma busca em profundidade, porém se estabelecendo uma profundidade máxima para a exploração do grafo. Este tipo de busca é especialmente útil em árvores cujos arcos têm custo, podendo-se elaborar uma função que avalie o custo de se aprofundar na árvore levando em conta a profundidade atingida. Ou para a utilização em grafos em que possam acontecer laços, ou seja, há a possibilidade do algoritmo iterar e ficar preso dentro de conjunto de nós e, portanto, não encontrar a solução ótima.

2.5.3.5. Busca de aprofundamento iterativo em profundidade

A busca de aprofundamento iterativo em profundidade é um método híbrido, que engloba as vantagens das buscas em extensão e em profundidade, eliminando algumas desvantagens de ambas. O seu procedimento é basicamente o mesmo da busca em profundidade, explorando sempre o nó de maior profundidade. Porém, se estabelece um limite gradual para a profundidade máxima a ser explorada. Cada vez que se avança um nó até o limite, passa-se para o próximo nó cuja profundidade seja mais próxima do limite. Quando todos os nós chegarem à profundidade limite o algoritmo pára e analisa se a solução foi alcançada. Em caso afirmativo, finaliza-se o algoritmo, caso contrário aumenta-se o limite e repete-se. Deste modo a exploração é prioritariamente em profundidade e torna-se uma busca em extensão uma vez que a profundidade limite foi atingida. A vantagem é a exploração híbrida, sendo um meio-termo entre a busca em profundidade e a em extensão. A

desvantagem é que, dependendo do problema, pode ser que a utilização das técnicas anteriormente descritas sejam mais eficientes e, no pior caso, ela pode se tornar uma busca em extensão simples. Porém, se a solução tem a chance de não se situar nas folhas, ou se houver mais de uma solução, é possível que esse método traga melhores resultados em comparação com os anteriores.

2.5.4. Busca heurística

Assim como os métodos de busca sem conhecimento, o objetivo dos métodos de busca heurística é, partindo-se de um panorama inicial de um problema, chegar à solução o mais rápido possível, preferencialmente com a melhor resposta. Este é um tipo de busca um pouco diferente dos métodos de busca sem conhecimento. A principal diferença está no fato de que raramente todos nós da árvore serão analisados. O objetivo é justamente o contrário, ou seja, eliminar o máximo possível de nós que não levem à solução ou que não agreguem informações à solução do problema.

Em geral esses métodos partem dos métodos de busca sem informação e acrescentam algumas modificações para melhorar a sua eficiência a um custo que, se dimensionado corretamente, pode levar rapidamente à solução. Todos os métodos de busca heurística utilizam uma função de avaliação, em geral denotada por ***f(n)***, em que ***n*** é o nó sob análise. O objetivo de ***f(n)*** é fornecer um parâmetro numérico de avaliação do nó, fornecendo uma estimativa da sua distância até o objetivo. Por isso, em geral, tanto melhor é a avaliação quanto menor for o valor de ***f(n)***.

O objetivo, portanto, deve ser conhecido ou o algoritmo deve meios de reconhecê-lo quando o encontrar. Se não for possível identificar tal objetivo, a avaliação não será feita, pois não se sabe se está perto ou não da solução. Para tal avaliação é utilizada uma função heurística, denotada por ***h(n)***, de modo que a função de avaliação depende diretamente dela, como mostrado na equação 2. Neste caso a função avaliadora corresponde somente à função heurística. Em outros métodos, explorados a seguir, essa função será aprimorada.

$$f(n) = h(n) \tag{2}$$

Segundo RUSSEL e NORVIG (2004) e LUGER (2004), diversos métodos são sugeridos para a solução de problemas por busca heurística. Porém, neste trabalho apenas dois serão descritos.

2.5.4.1. Busca Gulosa

A busca gulosa utiliza um método de exploração da árvore baseada unicamente na função heurística, como mostrado na equação 2. O objetivo é explorar os nós cuja função ***h(n)*** indique os menores valores, ou seja, analisar sempre o nó teoricamente mais próximo da solução. Para tanto, ***h(n)*** deve ser consistente (Russel e Norvig, 2004), isto é, para um dado nó ***n*** e todo sucessor ***n'*** de ***n*** gerado por qualquer ação ***a*** (aplicação dos operadores de direção, neste caso), o custo estimado para alcançar o objetivo a partir de ***n*** não é maior que o custo de se chegar a ***n'*** somado ao custo estimado de alcançar o objetivo a partir de ***n'***. Isto torna ***h(n)*** admissível e extremamente otimista.

Este é um método de busca que assume que a função ***h(n)*** seja sempre decrescente. Cada vez que for encontrado algum nó que eleve o valor de ***h(n)*** aquele ramo da árvore terá seu desenvolvimento suspendido até que seja encontrado algum outro nó com ***h(n)*** superior. Em outras palavras, o algoritmo tenta fazer uma busca em profundidade, pois teoricamente ela se aproxima mais rápido da solução. A partir do momento que se percebe que tal aprofundamento não está mais tão próximo da solução quanto se esperava, é feita uma curta busca em extensão e novamente retorna-se à busca em profundidade. Se o problema for simples é possível que se encontre a solução muito mais rápido do que com uma busca em profundidade simples. Porém, se o problema for mais complexo existe a chance de o algoritmo executar uma busca exaustiva por todos os nós da árvore. Além disso, haverá ainda a desvantagem do cálculo de ***h(n)***, tornando-o mais demorado do que uma busca em extensão simples. O algoritmo está ilustrado na Figura 16.

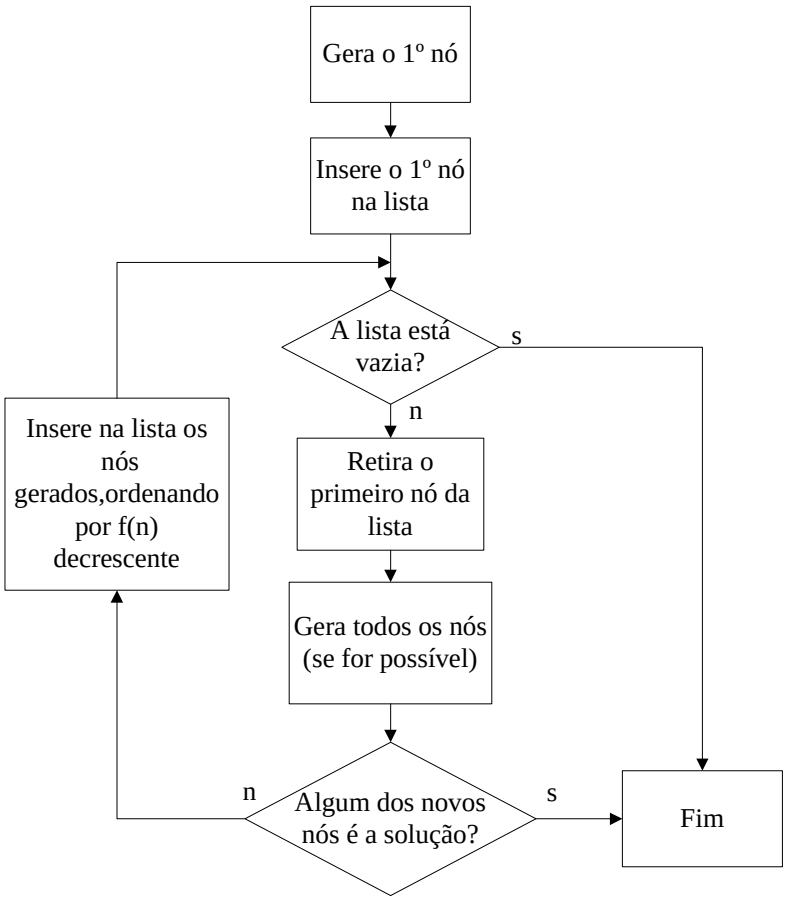

Figura 16 - Algoritmo de busca gulosa.

2.5.4.2. Busca A*

Um dos métodos de busca mais eficientes (RUSSEL e NORVIG, 2004) é o algoritmo A*. Esta abordagem tenta minimizar o custo total estimado, a cada nó, para se chegar até a solução. O esquema de funcionamento é basicamente o mesmo da busca gulosa, como mostrado na Figura 16. Porém, ao contrário do algoritmo de busca gulosa, não se depende somente do quão próximo o nó sob análise está da solução do problema, mas também do custo para se chegar da raiz até o nó sob análise. A equação 3 mostra o novo cálculo da função de avaliação ***f(n)***, na qual o termo ***g(n)*** é a representação do custo para se chegar até o nó atual.

$$f(n) = g(h) + h(n) \tag{3}$$

O uso deste parâmetro ***g(n)*** adiciona certa cautela à exploração. Ao contrário da busca gulosa, que busca somente a melhor opção local do ponto de vista da solução do problema, o A* incorpora mais conhecimento sobre a árvore. Este algoritmo entende que pode ser que o

estado que esteja aparentemente mais próximo à solução não faça parte do caminho que levará à melhor solução.

Este método é muito interessante porque é considerado completo e ótimo (RUSSEL e NORVIG, 2004), ou seja, se existir uma resposta ele a encontrará a ótima. Mesmo que a heurística escolhida (*h(n)*) não seja boa, desde que admissível, o algoritmo chegará até a resposta. Naturalmente, a qualidade da heurística, ou seja, quão melhor for a maneira de se modelar o conhecimento do problema, mais rápido será o algoritmo.

2.6. PROCESSAMENTO PARALELO

2.6.1. Introdução

O processamento paralelo é uma modalidade de computação em que diversas instruções são processadas ao mesmo tempo (DONGARRA, FOX, KENNEDY *et al*, 2002). Na verdade, isso pode ser dividido em dois conceitos básicos: as arquiteturas paralelas e os algoritmos paralelos (ROOSTA, 1999). Estes podem ser definidos como sendo algoritmos em que diversos processos ou tarefas podem ser executados simultaneamente e podem se comunicar entre si para auxiliar ou possibilitar a chegada a uma conclusão do problema. Nesse aspecto, um problema representado na forma de árvore, é especialmente adequado para ser resolvido por uma arquitetura paralela. Em geral, o termo “processo”, ou tarefa, se refere ao fragmento distinto de código que é executado por cada processador. O conjunto de tais processadores envolvidos no processamento paralelo é chamado de arquitetura paralela. Essa arquitetura pode ser composta de diversos processadores em um único computador, chamado de multiprocessador, que costuma ter uma eficiência grande com um custo elevado. Ou ainda, múltiplos computadores com apenas um processador cada, conectados por meio de uma rede de comunicação, dedicada (preferivelmente) ou não. Apesar de não terem o mesmo desempenho dos multiprocessadores, a relação custo/benefício acaba por ser mais atrativa, além de adicionar escalabilidade à estrutura. Naturalmente, abordagens mistas também existem. Entretanto, isto torna o conjunto de processadores heterogêneo, no sentido de que, tendo-se um conjunto de processadores com capacidade de processamento diferente, isso pode acarretar alguma dificuldade na maximização da utilização dos processadores, ou seja, atrapalhar o correto balanceamento e possivelmente aumentar o tráfego de mensagens na rede.

A idéia básica do processamento seqüencial é a de se executar as diversas partes do algoritmo uma após a outra, assim como mostrado na Figura 17. Nela está mostrada a divisão da tarefa original em quatro outras subtarefas, que serão tratadas pelo processador disponível. É importante notar que, no exemplo ilustrado na Figura 17, bem como nos que seguirão, o processador consegue tratar apenas de uma tarefa por vez e que cada tarefa é dimensionada para ocupar apenas um período de processamento. Outro aspecto é que as tarefas são independentes entre si, ou seja, a ordem de execução delas não importa, mas sim que todas sejam realizadas. Portanto, o processamento da tarefa da Figura 17 demoraria 4 períodos de tempo para ser completado.

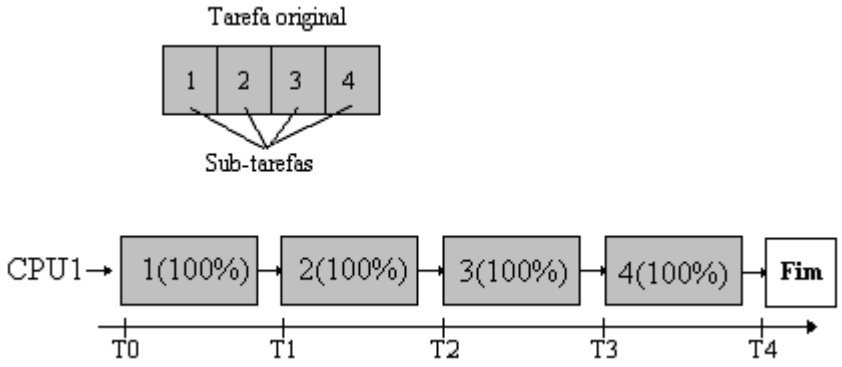

Figura 17 - Processamento seqüencial das subtarefas de uma tarefa original.

A idéia da paralelização vem de se manipular a ordem de execução das subtarefas que compõem uma tarefa, de modo que, em geral, tenta-se reduzir o tempo total de execução do algoritmo. Tal redução só é possível quando existe algum tipo de independência entre as subtarefas. Se as tarefas dependem umas das outras, ou seja, a segunda não pode iniciar antes de a primeira finalizar, torna-se inevitável o processamento seqüencial. Outro caso que a paralelização não se aplica é quando a subdivisão requer tanta comunicação entre as subtarefas que o tempo de execução acaba sendo superior que o tempo da execução seqüencial.

O objetivo, portanto, se torna identificar fragmentos de programa independentes que possam ser executados de maneira concorrente, de modo a acelerar a execução total do programa, de forma escalável, ou seja, paralelismo real. O desafio está em elaborar um algoritmo que seja independente do número de processadores utilizados, que utilize a arquitetura mais genérica possível e que, desejavelmente, consiga fornecer uma aceleração.

Primeiramente serão mostrados diversos métodos de paralelização de algoritmos, em seguida as arquiteturas paralelas e por fim as medidas de desempenho para o processamento paralelo.

2.6.2. Algoritmos paralelos

Discutindo-se sobre os algoritmos que possibilitam o paralelismo, é possível dividi-los basicamente em três aspectos, o pseudo-paralelismo, o *pipeline* e o paralelismo real que, cada um a seu modo tentam adicionar mais funcionalidade à execução seqüencial. É importante ressaltar que estes algoritmos não necessariamente existem de forma separada, é possível e bastante recomendável que, à medida que seja viável, sejam misturados os métodos a seguir apresentados.

2.6.2.1. Pseudo-paralelismo

O pseudo-paralelismo é um tipo melhorado de processamento seqüencial. Ele é classificado como paralelo, pois permite que diversos programas seqüenciais sejam executados ao mesmo tempo (ROOSTA, 1999).

O que acontece é bastante similar ao processamento seqüencial, ou seja, as 4 subtarefas levam os mesmos 4 períodos de tempo para serem executadas, um após o outro. A diferença é que todas as subtarefas são inseridas ao mesmo tempo no processador, mostrado como “CPU1” na Figura 18. Portanto a CPU pode decidir se há dependência entre os processos e selecionar a ordem de execução, baseado em prioridade ou rapidez de processamento. Caso se determine que alguma das tarefas possa ser concluída antes das outras e que a ordem não importa, seria possível adiantar resultados mesmo ao se manter o tempo total de processamento inalterado. Este tipo de paralelismo é bastante utilizado em arquiteturas multi tarefas, como sistemas operacionais, em que há a necessidade de se ter diversas subtarefas sendo executadas concorrentemente. Naturalmente esta modalidade aceita a participação de apenas um ou de vários processadores. Com o sistema de prioridades é possível, portanto, ter-se algumas subtarefas importantes sempre sendo processadas e à medida que elas são feitas mais tempo é dedicado a subtarefas menos importantes. É possível ainda se tratar da execução de um programa, ou diversos programas, que tenham interdependência, ou seja, algoritmos que dependem dos dados fornecidos por outro, ou outros programas, para poderem ser executados. Um exemplo é mostrado na Figura 18, em que uma tarefa, dividida em quatro subtarefas tem uma grande interdependência e devem ser executadas apenas por um processador.

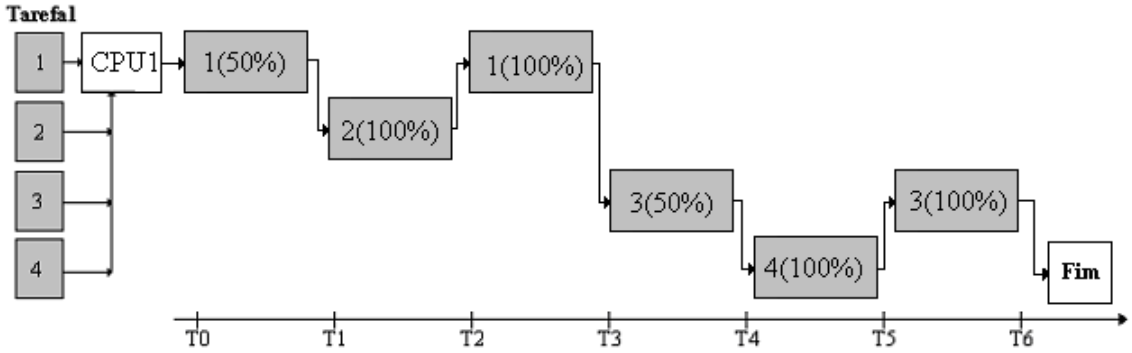

Figura 18 - Processamento pseudo-paralelo: diversas tarefas executadas simultaneamente, sem ganho de desempenho.

Neste exemplo, a subtarefa 1 tem que ser suspensa, pois fornece dados à subtarefa2 e depende do seu resultado. Portanto, as subtarefas 1 e 2 são interdependentes, bem como a 3 depende da 1 e da 4 e a 4 depende da 3. Este tipo de dependência não poderia ser resolvido desta maneira por uma arquitetura seqüencial de execução de programa. Como alternativa ao pseudo-paralelismo e, visando a melhoria no desempenho, poder-se-ia tentar dividir novamente a tarefa em subtarefas sem dependência ou utilizar um outro método de paralelismo.

2.6.2.2. *Pipeline*

O paralelismo do tipo ***pipeline*** é baseado no conceito de linha de produção, ou seja, diversas unidades de processamento especializadas em um tipo de tarefa, trabalhando fundamentalmente com o conceito de interdependência entre si, como mostrado na Figura 19 (BUYYA, 1999). Este tipo de processamento é ótimo para situações em que não se consegue desfazer a dependência entre as subtarefas e quando diversas tarefas similares deverão ser executadas repetidamente. Como exemplificado na Figura 19, existem 4 tarefas, cada uma dividida em 4 subtarefas e 4 unidades de processamento disponíveis, na Figura 19 chamadas de CPUs. O número de unidades de processamento também é chamado de profundidade do ***pipeline***. Cada subtarefa foi dimensionada de acordo com os critérios de dependência do algoritmo, ou seja, cada subtarefa recebe os dados da anterior e fornece dados para a próxima mas, durante o seu processamento, dispõe de todos os dados que necessita. Outra característica é que, como cada unidade de processamento é especializada em um tipo de tarefa, cada subtarefa deve ser apropriada àquela unidade de processamento.

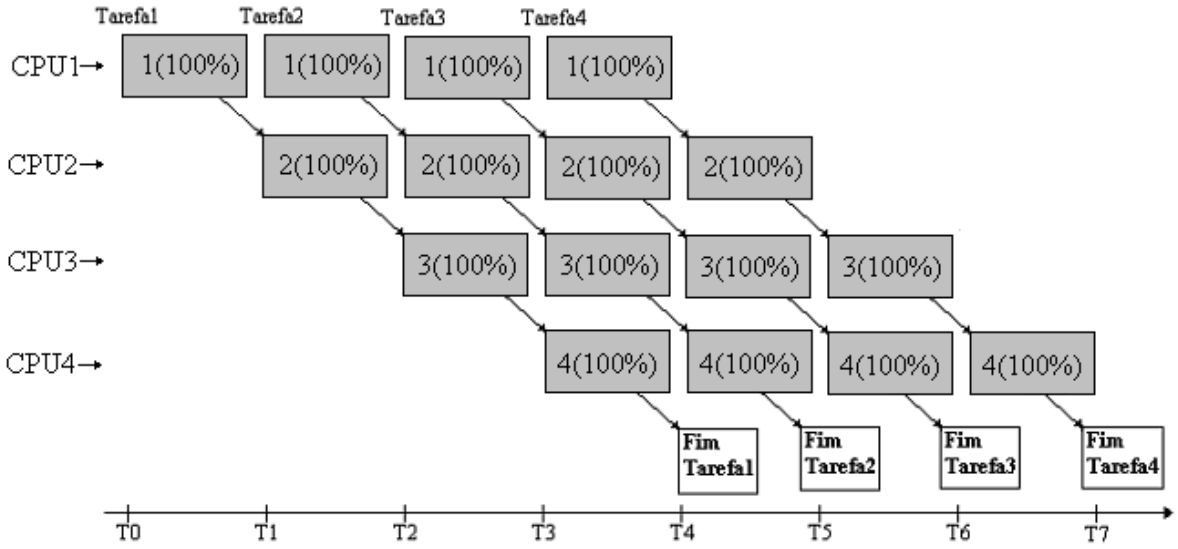

Figura 19 - Processamento em *pipeline*. Diversas tarefas podem ser executadas simultaneamente como se houvesse uma linha de produção.

Analisando-se a Figura 19, pode-se perceber que a execução inicia-se com a subtarefa 1 sendo aplicada à CPU1, que fornece dados à subtarefa 2 que é aplicada à CPU2 e assim por diante, até que a subtarefa 4 seja terminada. Observando-se a linha temporal na parte inferior da Figura 19, percebe-se que a execução da tarefa 1 não foi mais rápida do que a execução seqüencial. Porém, este método é para tarefas repetidas. Logo após a CPU1 terminar de processar a subtarefa 1 da tarefa 1, ela já recebe a subtarefa 1 da tarefa 2. Da mesma forma acontece com as demais CPUs, que à medida que terminam de executar uma subtarefa recebem outra de outra tarefa. Portanto, no instante seguinte que a CPU4 terminar de processar a subtarefa 4 da tarefa 1, ela acabará a subtarefa 4 da tarefa 2, e assim por diante, tendo-se a conclusão de uma tarefa por instante de tempo, após a primeira ser finalizada. Considerando-se ainda o exemplo da Figura 19, em que são processadas 4 tarefas de 4 subtarefas em que cada uma leva uma unidade de tempo para ser finalizada, o tempo total de processamento é de apenas 7 unidades de tempo, ao passo que um processamento seqüencial levaria 16 unidades de tempo. Ou seja, é possível executar mais de 8 tarefas no *pipeline* com o mesmo tempo de execução de apenas 4 no sistema seqüencial. Este beneficio relativo naturalmente depende da profundidade do *pipeline* e do número esperado de repetições das tarefas e, ainda mais importante, se as unidades de processamento são ou não flexíveis a ponto de exigir ou não que as tarefas sejam sempre iguais.

Todos esses fatores tornam a sua utilização em computação um tanto restrita. Nem sempre é possível determinar a dependência correta entre as subtarefas ou mesmo conseguir que essa subdivisão seja apropriada, de forma genérica, para todas as tarefas. Um exemplo de

utilização deste tipo de arquitetura é no processamento algumas instruções em microprocessadores. Há casos que um única instrução comanda o processador a executar, internamente, mais de uma ação. Como essas ações são sempre as mesmas, a montagem de um *pipeline* é muito vantajosa. Se essas ações não acontecem internamente a um processador, mas em computadores ligados em rede, pode ser que esta não seja uma boa alternativa.

2.6.2.3. Paralelismo real

O paralelismo real é um dos tipos mais poderosos de processamento paralelo. Com ele é possível ter ganhos de desempenho consideravelmente mais altos do que nas demais arquiteturas. Por outro lado é a mais complexa de ser realizada, pois é necessário que as tarefas consigam ser subdivididas em subtarefas que tenham requisitos de intercomunicação baixos o suficiente para viabilizar a arquitetura. A Figura 20 mostra um exemplo de processamento em paralelismo real.

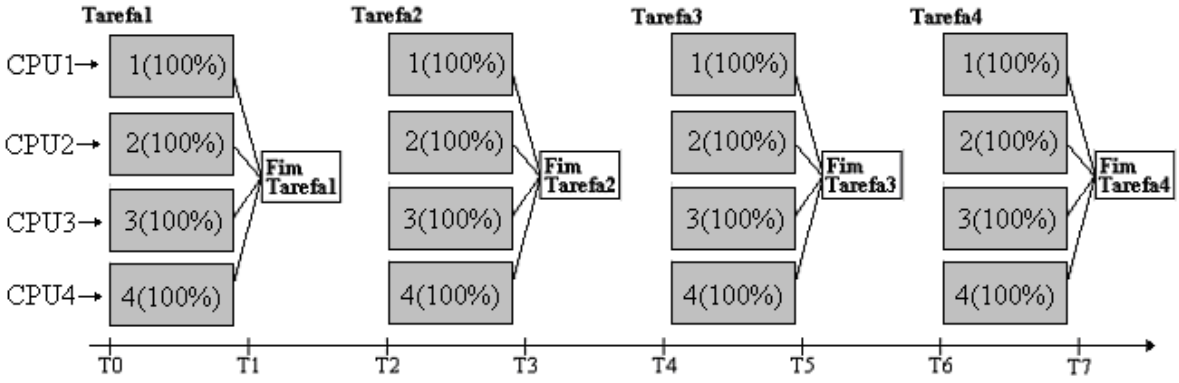

Figura 20 - Paralelismo real. As tarefas são executadas individualmente mais rápido que nos demais algoritmos paralelos.

Nesta abordagem, cada tarefa é processada ao mesmo tempo, sendo que cada subtarefa é enviada a um processador distinto (ROOSTA, 1999; DONGARRA, FOX, KENNEDY et al, 2002). Naturalmente, o exemplo é um caso ideal, em que a tarefa original pode ser dividida totalmente, sem arcar com alguma seção seqüencial que aumente o tempo de execução de cada tarefa. Em geral, este é o caso mais realista, ou seja, paraleliza-se uma parte do algoritmo, mas existe uma parte que sempre será executada seqüencialmente. Este modelo é mais flexível do que o *pipeline*, por não exigir que as unidades de processamento, neste caso computadores ligados em rede, sejam necessariamente especializados em uma função, aumentando a adaptabilidade para atender diversas tarefas diferentes.

É importante ressaltar que o tempo de execução de 4 tarefas com 4 subtarefas cada, em 4 CPUs foi exatamente o mesmo do exemplo do *pipeline* da Figura 19. Porém, cada tarefa foi individualmente mais rápida em sua execução, ou seja, se por algum motivo houvesse dependência entre as tarefas 1 e 2, este modelo conseguiria processar sem perdas no desempenho, ao passo que o *pipeline* teria o mesmo desempenho de um sistema seqüencial.

2.6.3. Arquiteturas paralelas

As arquiteturas paralelas são as formas em que elementos computacionais são interligados para permitir a execução de algoritmos paralelos. Tais elementos computacionais podem ser desde processadores até mesmo conjuntos grandes de computadores distribuídos. Nesta seção serão explorados os três tipos básicos de arquiteturas paralelas: o multiprocessamento, o processamento em *cluster* e finalmente as grades computacionais.

2.6.3.1. Multiprocessadores

Os multiprocessadores são sistemas paralelos em que se conta com mais de um processador por computador. Dentro desta categoria, podem existir diversos processadores separados fisicamente no mesmo computador, ou diversos núcleos dentro do mesmo processador. Existem diversas abordagens disponíveis comercialmente deste último tipo de processador. As mais comuns são as que encapsulam, em uma única pastilha de silício, dois ou quatro núcleos de processadores, compartilhando os diversos recursos antes disponíveis apenas para um único processador. Isto naturalmente tem a desvantagem de poder causar concorrência no acesso a alguns recursos como memória ou dispositivos de comunicação.

Há ainda a abordagem da colocação de vários processadores no mesmo computador, cada um com recursos próprios, de modo a maximizar a autonomia dos processadores, evitando ao máximo a concorrência de recursos. Esta é uma das abordagens mais eficientes pois os processadores são fisicamente independentes, cada um podendo ter a sua própria memória e com um meio de comunicação entre eles, podendo este ser tanto um canal de comunicação quanto uma memória compartilhada. Em geral este tipo de processamento é implementado em CLP (circuitos lógicos programáveis), como acontece com o supercomputador *BlueGene* da IBM.

Este tipo de paralelismo é, sem dúvida, o que pode fornecer o maior desempenho, pois a velocidade de comunicação entre os processadores dificilmente é equiparada por qualquer outro meio de comunicação externo ao computador. Por outro, lado este tipo de computador é razoavelmente mais caro do que alguma outra abordagem como o *cluster* ou a grade, pois nestes dois tipos é possível utilizar computadores de uso genérico disponíveis comercialmente, ao invés de uma arquitetura específica para um dado problema.

2.6.3.2. Cluster Beowulf

A base para a computação paralela é a utilização de uma arquitetura que seja capaz de implementar pelo menos um dos tipos de paralelismo, já descritos neste capítulo. Como neste trabalho busca-se o paralelismo real, é importante a utilização de uma arquitetura que seja capaz de suportá-lo. Nesta categoria, existem basicamente duas abordagens, os sistemas multiprocessados e os de processamento distribuído.

Os sistemas multiprocessados, ou supercomputadores, são computadores que possuem mais de um processador, podendo ou não compartilhar uma memória de trabalho. Este tipo de máquina é extremamente eficiente, pois se consegue ter uma alta eficiência de comunicação entre elementos processadores. Como eles estão conectados no mesmo barramento de dados, a comunicação entre eles tende a ser extremamente mais rápida do que a comunicação dependente de métodos externos de comunicação, como as conexões *ethernet* ou por comunicação serial. Em relação à memória do sistema, o sistema multiprocessado tem outra grande vantagem, que é a capacidade de ter uma memória compartilhada, ao mesmo tempo em que cada processador pode ter a sua própria memória de trabalho. A grande vantagem é poder ter uma forma rápida de compartilhar dados entre os processadores, reduzindo o tempo de comunicação ao tempo de acesso à memória, naturalmente assumindo a potencial desvantagem do acesso concorrente à memória. Mesmo assim, o processamento individual é pouco afetado, pois cada um tem a sua própria memória de trabalho. Tais sistemas são indicados para aplicações que necessitam de grande poder de processamento aliado à necessidade de troca de muitos dados entre os processadores. Apesar da sua inegável vantagem no desempenho, uma de suas desvantagens é o custo que, em geral, é extremamente alto.

Existe, porém, um tipo arquitetura que agrupa diversos computadores, não tantos quanto pela Internet, interligados por meio de uma rede lógica dedicada somente à

comunicação entre estas máquinas. A esta arquitetura é dado o nome de *cluster* (JORDAN, 1999; PFISTER, 1997; BADER e PENNINGTON, 2001; STERLING, 2000). Existem várias classes de *clusters*, dependendo das máquinas que o compõem. Os *clusters*, geralmente, são homogêneos, ou seja, utilizam computadores com mesmo poder computacional, pois isto facilita o balancemento de carga entre eles e a previsão e a avaliação de desempenho.

Há um caso particular chamado de *cluster* Beowulf, em que os computadores utilizados são computadores pessoais, de baixo custo e conectam-se entre si por meio de rede lógica *ethernet* convencional. O tipo de interconexão é, em geral, o modelo de interconexão completo (ROOSTA, 1999), que representa a forma mais simples de comunicação: ligam-se os computadores de modo que todos tenham visibilidade de todos. Este modelo é, sem dúvida, o mais flexível, pois permite a implementação de quaisquer outros modelos de interconexão, com a utilização de um *switch*. Pode-se implementar não só as regras de comunicação (no caso da utilização de outros modelos) como também isolar o tráfego entre os computadores participantes da comunicação.

Este tipo de arquitetura de computadores foi inventado em 1994 pelos cientistas da Agência Aeroespacial Norte-Americana (NASA) Thomas Sterling e Donald Becker. Esta é uma opção vantajosa, pois tem uma ótima relação custo/benefício.

É importante ressaltar que apenas a ligação dos computadores em uma rede dedicada não o torna, de fato, um *cluster*. O termo *cluster* está mais relacionado à capacidade de coordenar-se o processamento em um conjunto de máquinas e para isto é necessário o uso de um suporte de *software*. Existem diversas bibliotecas, ou modelos de programação que auxiliam ou implementam essa idéia, como a MPI e a OpenMP. A escolhida para ser utilizada neste trabalho é a MPI (*Message Passing Interface*) (SNIR, OTTO, HUSS-LEDERMAN *et al*, 1997), mais especificamente, a implementação do Argonne Labs, denominada MPICH (GROPP, LUSK, DOSS e SKJELLUM, 1996; GROPP, LUSK e SKJELLUM, 1999), na sua segunda versão. Esta biblioteca cria um ambiente virtual que une as máquinas e define um conjunto de funções que possibilitam e facilitam a comunicação entre os diversos computadores e incentiva a utilização de um processo-mestre comandando todo o conjunto restante de processos. Ela utiliza o conceito de processos independente do conjunto de processadores. Isto é feito de forma a deixar flexível, para o desenvolvedor, a organização do algoritmo. Desta forma, é perfeitamente possível ter um sistema com um número n de processadores executando m instâncias do algoritmo. Assim, se cada computador possui apenas um processador, n e m em geral são iguais, mas se estiverem disponíveis quatro processadores por computador pode-se ter $m = 4*n$, sempre com o objetivo de obtenção de

um processo por processador. É possível também executar um número de processos maior do que o número de processadores, assumindo-se a potencial desvantagem da ocorrência do pseudo-paralelismo, ou *multi-threading*, pois mais de um processo concorrerá para a utilização do mesmo processador. O importante é ressaltar que fica a cargo do desenvolvedor definir o número de processos e também a forma que o processamento será dividido entre os processos, ou seja, a forma que acontecerá o paralelismo, pois a biblioteca não faz nenhuma divisão automática de execução.

Esta biblioteca implementa um dos paradigmas de programação paralela, chamada passagem de mensagens (ROOSTA, 1999), em que a comunicação entre os processadores acontece por meio de mensagens enviadas uns aos outros, de forma restrita ou aberta a todos os que estiverem conectados. Esta comunicação se contrapõe à outra forma de comunicação, que é a chamada de compartilhamento de memória (ROOSTA, 1999), em que se utiliza uma memória compartilhada para se fazer uma espécie de caixa de mensagens, na qual cada processador coloca suas mensagens e tem que fazer verificações periódicas para ver se há alguma mensagem endereçada para si. Este modelo é mais indicado para arquiteturas multiprocessadas, sendo o modelo de passagem de mensagens mais apropriado para os modelos de *grid* e *cluster*.

2.6.3.3. Computação em Grade

A outra opção são os sistemas de processamento distribuído (FOSTER e KESSELMAN, 1999; BERMAN, HEY e FOX, 2003) em uma rede IP (*Internet Protocol*), em que os processadores não pertencem ao mesmo computador e, geralmente, não existe forma de ter memória compartilhada entre eles. Nessa categoria, pode-se ter um número de computadores extremamente grande, interconectados pela Internet ou por uma rede local *ethernet*, assumindo-se a desvantagem da dependência do tráfego de Internet, que pode gerar grande latência na comunicação. Sistemas distribuídos baseados em Internet, também chamados de grades ou g*rids*, em geral, não são indicados para aplicações que requeiram muita rapidez na comunicação. No entanto, são sistemas muito flexíveis, no sentido de que podem utilizar computadores pessoais comuns em conjunto com computadores dedicados, *workstations* e multiprocessadores. Obviamente essa flexibilidade traz, com certeza quase absoluta, a desvantagem da heterogeinização (utilização de computadores com diferentes capacidades de processamento) dos computadores, o que dificulta a previsão e a avaliação do desempenho do conjunto.

A computação em grade é um tipo de arquitetura paralela extremamente flexível. Ele tem uma composição muito parecida com o *cluster*. Porém, os computadores que o compõe não precisam estar próximos fisicamente, ou pelo menos não dispõe de um meio dedicado para intercomunicação. A idéia da computação em grade é justamente o contrário. O requisito é ter disponível um meio de comunicação entre os computadores, para que se possa fazer *cluster*s de centenas ou milhares de computadores. Para tanto, um meio amplamente disponível de comunicação é necessário, pois é razoavelmente complexo e caro ter uma rede dedicada com centenas de computadores. Portanto, geralmente a internet é utilizada para conectar tais computadores. Justamente neste aspecto está uma das duas principais diferenças entre os *cluster*s e as grades. Por se utilizar a internet não se pode depender de agilidade na comunicação, pois existe sempre o potencial de os dados não serem entregues tão rápido quanto seria necessário ou quanto seria possível com uma rede dedicada. Este tipo de computação paralela é bastante apropriada para a implementação de sistemas oportunistas. Estes sistemas aproveitam o tempo ocioso das máquinas, possibilitando que se tenha mais computadores disponíveis para realizar as tarefas, porém de forma não dedicada. Cada computador então é utilizado quando o seu usuário não o está utilizando, não atrapalhando, portanto, o uso normal do computador, ao mesmo tempo em que permite a sua utilização na grade.

A outra diferença entre os dois modelos é o aspecto da homogeneidade. Em uma grade, há a grande probabilidade de os computadores serem diferentes, com capacidades de processamento ou de comunicação diferentes. Portanto, comunicação entre os componentes da grade não pode ser tão importante para a execução do algoritmo. A escolha do problema a ser paralelizado tem que ser muito mais criteriosa para uma grade do que um *cluster* porque a independência entre subtarefas deve ser alta, desejavelmente absoluta, limitando os problemas que podem utilizar essa estrutura.

Por outro lado, as grades têm um custo razoavelmente menor que os *cluster*s pois não requerem que a estrutura seja dedicada, podendo aproveitar uma rede já existente e mantendo a flexibilidade necessária para atender outras demandas, como redes corporativas.

2.6.4. Indicadores de desempenho em sistemas paralelos

O objetivo das métricas, ou indicadores de desempenho, é estabelecer parâmetros de comparação de desempenho dos algoritmos. No caso específico dos algoritmos paralelizados, não são suficientes os parâmetros como a complexidade de tempo ou espaço que são

freqüentemente utilizados como métricas em sistemas seqüenciais. A simples contagem de operações elementares, que seria feita em sistemas seqüenciais não é possível em sistemas paralelos, pois isso dependeria de quantos processadores estão sendo utilizados e de como a execução está distribuída. Portanto, outras medidas são necessárias.

Segundo CHAUDHURI (1992) e ROOSTA (1999), alguns parâmetros devem ser considerados na avaliação de desempenho de sistemas paralelizados. Cinco medidas são sugeridas: fator de aceleração, eficiência de execução, custo, utilização e taxa de execução. As três primeiras são as mais importantes e, de fato, foram utilizadas neste trabalho. Elas representam índices que têm o potencial de indicar o desempenho do sistema com base em medidas simples, ao passo que as duas últimas, não foram implementadas por representarem uma certa redundância nas medições realizadas. Em vez delas, três outros índices foram utilizados: taxa de ocupação, taxa de participação e taxa de comunicação. Estes últimos e os demais serão descritos a seguir. É importante ressaltar que as medidas descritas a seguir referem-se exclusivamente a sistemas homogêneos, ou seja, sistemas compostos por computadores com capacidade de processamento igual, ou praticamente igual. A situação ideal é utilizar computadores idênticos no *cluster*.

2.6.4.1. Fator de aceleração

O fator de aceleração é um parâmetro que tem por objetivo medir o quanto um algoritmo paralelizado conseguiu ser executado mais rápido do que a sua versão seqüencial. Naturalmente, esse parâmetro é altamente dependente dos computadores que estão sendo utilizados. O ideal é que sejam utilizados, na versão paralelizada, computadores iguais ao utilizado na versão seqüencial. De preferência e, se possível, o algoritmo deve ser o mesmo. Em algumas implementações, como no caso deste trabalho, o algoritmo paralelizado pôde ser generalizado para possibilitar a execução em N processadores, sendo que N pode ser desde 1 até 255 (segundo a limitação da biblioteca MPI utilizada).

O fator aceleração é simplesmente a divisão entre o tempo de execução seqüencial (Ts) pelo tempo de execução paralelizado (Tp), descrita na equação 4. O comportamento esperado para esse fator é estar dentro dos limites teóricos de 1 a p, em que p é o número de processadores utilizados.

$$Sp = \frac{Ts}{Tp} \tag{4}$$

Um fator *Sp* igual a 1 significaria que a implementação paralela obteve o mesmo tempo de execução que a versão seqüencial. Por outro lado, um fator igual a *p* significaria que a aceleração conseguida é linear (ROOSTA, 1999). Isto significa que o benefício de utilizar 2 processadores seria reduzir pela metade o tempo de execução, ou seja, utilizando-se 8 processadores, o tempo seria 8 vezes menor. Fatores inferiores a 1 significam implementações totalmente impróprias para a arquitetura paralela, pois ficam mais lentas mesmo com mais processadores.

É possível, no entanto, obter fatores de aceleração superiores a *p*. Tal fenômeno é chamado de aceleração superlinear ou anomalia de aceleração. Isto é possível acontecer somente quando o algoritmo seqüencial não é a solução ótima para a solução do problema ou quando as características do *hardware* propiciam um ambiente melhor que o seqüencial (ROOSTA, 1999). Há ainda outra situação em que isto é possível, que se encaixa na primeira razão. Seria quando o algoritmo pode se beneficiar, no momento presente, de dados que serão gerados em um momento futuro, como pode acontecer na exploração de alguns problemas que envolvem explosões combinatórias.

2.6.4.2. Eficiência de execução

A eficiência de execução tem por objetivo medir a fração de tempo em que um dado elemento de processamento é, de fato, utilizado no processamento (ROOSTA, 1999). A equação 5 mostra cálculo da eficiência de execução, que é simplesmente o fator de aceleração dividido pelo número de processadores.

$$Ep = \frac{Sp}{p} = \frac{Ts}{p.Tp} \tag{5}$$

De forma ideal, a utilização deveria ser 1, ou seja, todos os processadores do sistema utilizariam 100% do seu tempo para executar a computação relativa ao problema. Porém, existem diversos outros processos tais como comunicação, alocação de memória, controle de processos, etc., cujos tempos de execução não podem ser desprezados, e levam a uma diminuição o tempo que cada processador tem disponível para se dedicar ao algoritmo em execução. Portanto, esse índice deve estar sempre entre 0 (eficiência mínima) e 1 (eficiência máxima) (ROOSTA, 1999). Segundo CHAUDHURI (1992), a eficiência não pode ultrapassar 1.

2.6.4.3. Custo

O índice custo basicamente reflete a soma do tempo que cada processador utilizou para a solução do problema, de forma a ser possível comparar o tempo de execução paralelizada com o tempo de execução seqüencial. Em outras palavras, seria o equivalente a ter uma execução seqüencial realizada por um processador com o poder computacional igual à soma do desempenho dos processadores utilizados na execução paralela. A equação 6 mostra a equação para o cálculo do custo, em que p é o número de processadores e Tp é o tempo da execução paralelizada.

$$Cp = p.Tp \tag{6}$$

2.6.4.4. Taxa de ocupação

O fator de eficiência, descrito anteriormente, representa o desempenho do sistema, que é o tempo de execução do algoritmo. Portanto, é um ótimo indicativo conjunto do desempenho do algoritmo paralelizado. No entanto, a sua interpretação deve estar relacionada ao processamento paralelo como um todo. Isto torna inviável a avaliação dos processadores individualmente. Obviamente, o fator de eficiência cai quando um ou mais processadores têm um desempenho ruim, e vice-versa. Porém, neste trabalho, optou-se também por utilizar um índice mais específico, que seja capaz de elucidar, de fato, o tempo em que cada processador está fazendo o processamento do algoritmo, contando com os tempos de comunicação, excluído qualquer possível tempo de espera por designação de tarefa, também chamado de tempo inativo ou latência. Portanto, este índice, chamado de $Tocup$, é basicamente definido pela equação 7, que utiliza as variáveis Ti, por exemplo, o tempo total que cada processador utilizou para fazer processamento, e Tp sendo o tempo total da execução do algoritmo.

$$Tocup(\%) = \frac{Ti * 100}{Tp} \tag{7}$$

Esse índice deve estar sempre na faixa de 0% a 100%, de modo que 0% significa que o processador não teve absolutamente nenhuma participação no processamento ficando provavelmente em estado de latência o período todo do processamento. Por outro lado, ele nunca chegará a 100%, exceto se houver apenas um processador realizando o algoritmo. Isto acontece, pois existe um tempo necessário para a comunicação utilizado pelo processador responsável pela criação, pela divisão dos processos, pelo balanceamento e pelo recolhimento dos resultados. Apesar de este tempo ser pequeno, matematicamente impede que o tempo dos

demais processadores iguale-se ao tempo total de processamento. Isto pode indicar ainda uma possível falha em um nos processadores, como um travamento ou um defeito no *hardware*, além de ser uma ótima medida da eficiência do balanceamento de carga entre os processadores.

2.6.4.5. Taxa de participação

A taxa de participação tem por objetivo medir qual foi a parcela do problema original que foi efetivamente alocada e processada por cada processador. A definição desse índice é simplesmente a porcentagem correspondente do número de processos executados em cada processador em relação ao número total de processos, como mostrado na equação 8. Nesta equação, *Nprocessos* é o número de processos que o processador executou e *Nprocessostotal* é o número total de processos gerados, incluindo os possíveis processos gerados por balanceamento de carga.

$$Tpart(\%) = \frac{Nprocessos * 100}{Nprocessostotal} \tag{8}$$

Este índice fornece uma noção muito interessante da dinâmica da solução do problema. Analisando este índice sozinho, pode-se perceber quais processadores tiveram maior participação no processamento. Por conseqüência, uma baixa participação pode indicar algum gargalo na comunicação. Se analisada juntamente com a taxa de ocupação, pode fornecer pistas de falha no processador ou problema no balanceamento, caso ambos os índices estejam baixos. Se a taxa de ocupação estiver alta e a participação baixa, pode significar que as tarefas escolhidas para o processador estão excessivamente trabalhosas, o que pode acarretar baixa na taxa de ocupação de outros processadores. Ao contrário, se *Tocup* estiver baixa e a *Tpart* estiver alta, o processador está sendo subutilizado, com muitos processos poucos trabalhosos, o que pode acarretar tráfego excessivo na rede. Os limites desse índice são 0% e 100%, mas o objetivo da avaliação desse índice não é estar alto, o que, em geral, nunca deve acontecer. Porém, desde que a taxa de ocupação esteja alta em todos os processadores e estes sejam iguais, a taxa de participação deve ter pouca variabilidade entre os processadores.

2.6.4.6. Taxa de comunicação

A taxa de comunicação tem por objetivo medir o número total de comunicações, tanto enviadas quanto recebidas, que foram realizadas durante o processamento. Esse índice analisado isoladamente pode dizer pouco sobre o comportamento do sistema, pois é complicado descobrir a quantidade exata de bytes transferidos em cada comunicação, uma vez que, além do conteúdo que se deseja enviar, há os demais bytes que a própria biblioteca de comunicação envia. Analisando-se conjuntamente com a taxa de ocupação e a taxa de participação, descobre-se se a comunicação está representando um gargalo no desempenho do sistema. Em diversos sistemas é comum que a comunicação seja um motivo de degradação de desempenho. Porém, isto nem sempre é regra e depende diretamente da necessidade de troca de informação do algoritmo. Em geral, como conseqüência dos gargalos, as taxas de comunicação e de participação, em relação aos demais processadores, apresentam-se altas enquanto que a taxa de ocupação permanece mais baixa. Naturalmente, se o fluxo de comunicação for excessivo a ponto de gerar um gargalo no processador que distribui os processos e isso for gerado não apenas por um processador, mas pela comunicação de todo o conjunto, todos terão uma taxa de ocupação baixa, de participação semelhante e de comunicação alta, em relação ao que seria considerado normal, o que é bastante relativo. Nesses casos, poderia ser percebida uma taxa de ocupação anormalmente alta no processo-mestre que, em geral, não realiza processamento significativo a ponto de ser comparado com os processos que realizam o algoritmo.

2.6.5. Limites de desempenho: Lei de Amdahl e Lei de Gustafson

Um aspecto muito importante no estudo do processamento paralelo é a previsão dos limites teóricos de desempenho que podem ser conseguidos com a implementação do sistema. A importância deles é para se determinar a eficiência do algoritmo e da arquitetura paralela, para que, em conjunto com os indicadores de desempenho já citados, estabeleça-se onde se pode aperfeiçoar a implementação, de modo a melhorar o desempenho do sistema. De acordo com CHAUDHURI (1992) e ROOSTA (1999), diversas características contribuem para o desempenho do processamento, dentre outras:

- velocidade dos processadores;
- número de acessos concorrentes à memória;

- tamanho da memória;
- capacidade de endereçamento indireto;
- manuseio de blocos condicionais de código.

Além desses, existem duas leis (ROOSTA, 1999) que são capazes de definir, no âmbito teórico, os limites de desempenho que o sistema deverá ter, que são as lei de Amdahl e de Gustafson.

A Lei de Amdahl (AMDAHL, 1967) estabelece um limite importante no que diz respeito ao desempenho máximo que se espera com a utilização de um sistema paralelizado. Segundo essa lei, o tempo de execução paralelizada de um sistema utilizando p processadores não pode ser inferior a p vezes o tempo de execução seqüencial. Isso significa o tempo de execução esperado para um sistema paralelizado seria, no máximo, o tempo de execução seqüencial dividido pelo número de processadores. Isto é expresso na equação 9, em que R é o fator de aceleração (descrito a seguir) máximo que pode ser conseguido pela arquitetura paralela. A variável S é a porção seqüencial do algoritmo e P é a porção paralelizada, sendo uma complementar à outra. A forma de representação de S e P é na forma de porcentagem, de modo que a soma das duas variáveis tem que ser 1. A variável n é o número de processadores utilizados.

$$R = \frac{1}{S + \frac{P}{n}} \tag{9}$$

Com o objetivo de saber qual é o maior benefício possível por meio da paralelização, bastaria aplicar a fórmula em um algoritmo totalmente paralelizável. Naturalmente, é bem pouco provável que isto aconteça. Porém é importante esta consideração para se determinar o desempenho máximo da arquitetura paralela. Em tal algoritmo, a porção seqüencial S seria igual a zero e, conseqüentemente, a porção paralelizável seria 1. Substituindo na equação 9, e fazendo as simplificações necessárias, resultará em $R=n$. Portanto, o maior benefício que pode ser conseguido com tal arquitetura, em que não existe programa seqüencial, ou ele é desprezível, seria o número de processadores utilizados.

Essa lei, porém, para ser válida, assume algumas condições. Uma delas é que não deve haver vantagens de infra-estrutura na versão paralelizada e que o código seqüencial deve ter sido projetado para a arquitetura monoprocessada, bem como a implementação do algoritmo deve ser a melhor possível. Isso significa que, na prática, não é possível alcançar um fator de aceleração perto de n, pois com o aumento de processadores o fator de aceleração tende a se

estabilizar, como mostrado na Figura 21, onde se observa a previsão do fator de aceleração para um algoritmo cuja porção paralelizada é de 90%.

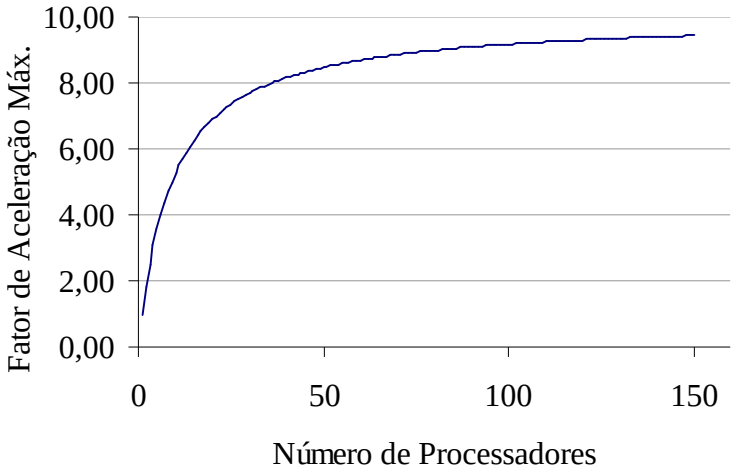

Figura 21 - Aplicação da Lei de Amdahl na previsão do fator de aceleração para um algoritmo 90% paralelizado.

Isto acontece pois a lei sugere que a parte seqüencial do código tende a crescer junto com o aumento do problema, ou seja, o número de processadores tem que ser limitado. Por outro lado, isso não significa que seja impossível obter uma aceleração substancial, perto de *n*, mas sim que a adaptação do algoritmo para a versão paralelizada deve procurar explorar alguma vantagem intrínseca à arquitetura, de modo a acelerar o algoritmo além do esperado por essa lei.

Porém, nem todos os algoritmos têm a sua parte seqüencial crescendo junto da parte paralelizada. Segundo Gustafson (1988), é possível descrever o fator de aceleração máximo para uma dada arquitetura e seu algoritmo de forma diferente à proposta por Amdahl, como mostrado na equação 10, em que *n* é o número de processadores e *S* é a porção seqüencial do algoritmo.

$$R(n) = n - S(n-1) \tag{10}$$

Considerando-se que, à medida que o problema aumenta apenas a parte paralelizável aumenta, outra dinâmica pode ser equacionada. O benefício proporcionado pela paralelização pode então ser visto como algo que aumenta de acordo com o número de processadores que estão sendo utilizados. Essa lei considera, assim como a Lei de Amdahl, que a parte seqüencial *S* permanece com a sua porcentagem inalterada de acordo com o aumento do número de processadores que participam do processamento.

Porém, ao contrário da Lei de Amdahl, a Lei de Gustafson considera que o ônus gerado pela parte seqüencial do algoritmo não aumenta com o número de processadores. Isso significa que se o algoritmo puder ser executado de forma a minimizar, e preferivelmente eliminar, o tempo de execução seqüencial do maior número de processadores possível, o fator de aceleração pode ser incrementado de forma substancial, de forma a não atingir estagnação, como acontece com a lei de Amdahl. A Figura 22mostra o progresso do fator de aceleração máximo com o aumento do número de processadores, para o caso de um algoritmo 90% paralelizado. Naturalmente, para que a Lei de Gustafson seja válida é necessário que o problema possa ser dividido igualmente entre os processadores, para que não haja concorrência o que, certamente, modificaria radicalmente a curva.

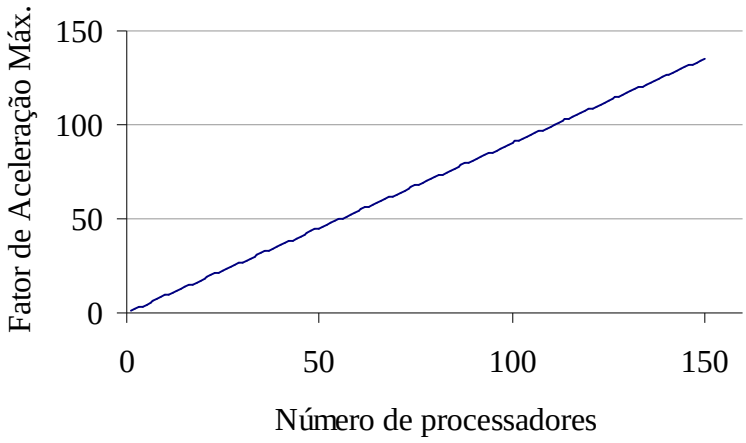

Figura 22 - Aplicação da Lei de Gustafson na previsão do fator de aceleração para um algoritmo 90% paralelizado.

É importante perceber dois aspectos das duas leis:

a) O primeiro é que, em ambos os casos, a aceleração é sublinear, ou seja, a parte seqüencial contribui para não se atingir a aceleração linear, o que ainda levanta a necessidade de explorar-se a implementação paralelizada a ponto de se obter benefícios que sejam intrínsecos à arquitetura paralela.

b) O segundo aspecto é a importância de minimizar a porção de algoritmo seqüencial e de concentrar a parte que não puder ser eliminada no menor número de processadores, preferivelmente em apenas um.

A Figura 23 mostra a dinâmica assumida pelo fator de aceleração tanto em sistemas nos quais a parte seqüencial é executada por vários processadores (Lei de Amdahl) quanto naqueles onde é executada em apenas um processador (Lei de Gustafson). Percebe-se que,

reduzindo-se o número de processadores que executam a porção seqüencial, obtém-se um benefício muito superior, considerando a mesma porção de código seqüencial.

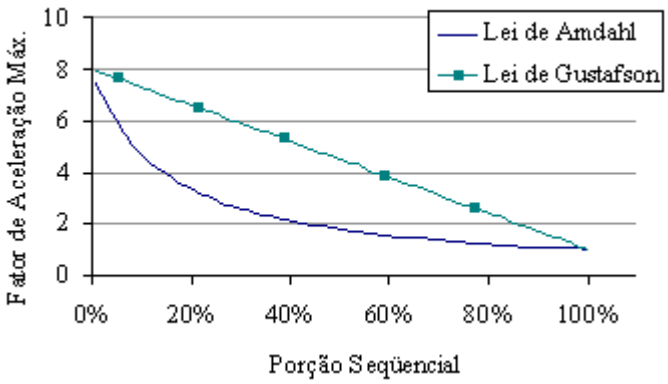

Figura 23 - Comportamento do fator de aceleração de acordo com a porção seqüencial do algoritmo, no caso hipotético da utilização, com 8 processadores.

2.6.6. Biblioteca MPI

Para implementação da computação paralela existem diversas bibliotecas destinadas a estabelecer padrões para facilitar a interligação lógica dos computadores. Dentre elas estão as abordagens baseadas em memória compartilhada como a PThreads (MUELLER, 1993) a OpenMP (CHANDRA, DAGUM, KOHR *et al*, 2001), e as baseadas em troca de mensagens como o MPI (MPI FORUM, 1994; MPI FORUM, 1998)e o PVM (GEIST, BEGUELIN, JIANG et al, 1994).

As abordagens baseadas em memória compartilhada são interessantes do ponto de vista que elas implementam uma área em comum a todos os agentes participantes do *cluster*. Nesta área os agentes trocam informações que permitem o trabalho conjunto.

A outra abordagem, baseada em troca de mensagens, estabelece um modelo de programação que, ao invés de formar uma área comum de troca de informações, se ocupa em estabelecer um padrão de troca de mensagens. Dentre elas, a MPI foi escolhido, mais especificamente a implementação chamada MPICH (GROPP, LUSK , DOSS *et al*, 1996), na sua segunda versão. Sua escolha aconteceu por causa da facilidade de utilização da biblioteca.

O padrão da biblioteca MPI é independente de linguagem de programação. O MPICH cria um ambiente que facilita a implementação da troca de mensagens. Basicamente, a biblioteca executa um programa em cada agente participante do *cluster*. Esse programa é denominado *daemon*. Assim, todos os agentes passam a fazer parte do mesmo conjunto de

agentes, chamado comunicador. Dentro do comunicador, na sua inicialização, o MPICH atribui a cada agente um número seqüencial, denominado *rank*. Em geral o agente *rank* 0 é utilizado como agente mestre, assim como foi utilizado neste trabalho. Quando se executa um programa paralelo com o MPI, o mesmo programa é executado em todos os agentes. Dentro do programa deve haver uma distinção sobre o que cada agente deve fazer. No caso deste trabalho, em que o agente 0 é o mestre, o agente *rank* 0 executa uma subrotina diferente dos demais agentes. Porém isso é flexível. Neste trabalho foram definidos dois grupos de trabalho: o agente 0 e os demais agentes, ou seja o mestre e os escravos. Porém, é possível fazer a divisão que melhor se adapte ao problema tratado.

A biblioteca possui mais de 150 funções, que incluem comunicação ponto a ponto, comunicação do tipo *broadcast* e diversos outros tipos de funções que automatizam tarefas usuais utilizadas na computação paralela. Porém com apenas 6 funções é possível se construir qualquer programa paralelo. A Tabela 4 mostra tais funções.

Tabela 4 - Funções básicas da biblioteca MPI2

Função	**Efeito**
MPI_Init	Inicializa o ambiente MPI, que permitira com que os agentes se comuniquem, ou seja, constrói o comunicador. Tem que ser chamada por todos os agentes
MPI_Comm_size	Descobre quantos agentes estão participando do comunicador
MPI_Comm_rank	Descobre qual é o *rank* do agente. Todos os agentes têm que executar essa função para saber o que devem fazer.
MPI_Send	Envia uma mensagem ao um destinatário especifico ou a todos os agentes. Essa função é do tipo bloqueante, ou seja, interrompe a execução do programa até ter certeza que a mensagem foi recebida.
MPI_Recv	Recebe uma mensagem de um remetente especifico ou de qualquer agente do comunicador. Essa função é do tipo bloqueante, ou seja, interrompe a execução do programa até ter certeza que alguma mensagem seja enviada.
MPI_Finalize	Finaliza o comunicador. Tem que ser chamada por todos os agentes.

2.7. TRABALHOS CORRELATOS

Existem diversas abordagens para a solução do dobramento de proteínas, além da proposta neste trabalho, cada uma abordando o problema a partir de um método computacional diferente, conforme a lista a seguir:

- Busca exaustiva (CHAN e DILL, 1991; YUE e DILL, 1993);
- Colônia de formigas (SHMYGELSKA, HERMANDEZ E HOOS, 2002; SHMYGELSKA, E HOOS, 2003);
- Redes Neurais (RABOW e SHCERAGA, 1993; HOLBROOK, MUSKAL e KIM, 1993; DUBCHAK, HOLBROOK e KIM, 1993);
- Redes *fuzzy* (DAUGHERTY, 1993);
- Têmpera simulada (*Simulated annealing)* (COVEL, 1994; HANSMANN e OKAMOTO, 1994; HIROYASU, MIKI e OGURA *et al*, 2003);
- Métodos estatísticos (OSGUTHORPE, 2000; SIMON, FISER e TUSNADY, 2001);
- Autômatos celulares (OSTROVSKY, CROOKS, SMITH *et al*, 2001);
- Algoritmos meméticos (KRASNOGOR, BLACKBURNE, HIRST *et al*, 2002);
- Algoritmos genéticos (UNGER e MOULT, 1993; KRASNOGOR, PELTA, LOPEZ *et al*, 1998; SCAPIN e LOPES, 2005).
- Algoritmo Monte Carlo puro e híbridos: (EYRICH, STANDLEY e FRIESNER, 1999; LIANG e WONG, 2001; NAKAMURA, SASAKI e SASAI, 2001, LI, KLIMOV e THIRUMALAI, 2002; LEONHARD, PRAUSNITZ e RADKE, 2003; YESYLEVSKY e DEMCHEMKO, 2004).

CAPÍTULO 3

3. METODOLOGIA

3.1. INTRODUÇÃO

Este capítulo tem o objetivo de descrever o sistema de busca exaustiva paralelizado, baseado em uma abordagem modificada do algoritmo A*. A implementação resultante foi o sistema utilizado como base para este trabalho. Primeiramente, será apresentada a forma que o espaço de busca foi representado, seguindo-se pela caracterização do problema, pela análise do espaço de busca e pelo algoritmo desenvolvido.

3.2. REPRESENTAÇÃO DO MODELO 2DHP EM ÁRVORE

Para aplicar algum algoritmo de busca ao problema do dobramento de proteínas, é necessário fazer uma representação conveniente do espaço de busca, ou seja, deve-se representar o problema de forma que permita que o algoritmo seja explorado ao máximo.

A forma escolhida para se representar o problema do dobramento é a de um grafo orientado em forma de árvore, seguindo a mesma filosofia exposta no Capítulo 2.

O objetivo desta forma de árvore é facilitar a utilização de um algoritmo de busca e, principalmente, garantir que, à medida que os nós vão sendo gerados, haja a certeza de que cada nó seja único. Esta garantia é necessária para não se ter subárvores com mesmo conteúdo. Como será mostrado a seguir, o problema do dobramento de proteínas envolve uma explosão combinatória no seu espaço de estados. Portanto, a não geração de subárvores idênticas é essencial para o sucesso da busca, pois se evita perder tempo realizando análises repetidas.

Na árvore utilizada neste trabalho não se representa no espaço de busca a estrutura primária do polipeptídeo, pois ela é sempre a mesma durante toda a análise. O que muda é o espaço de busca**.** Os nós da árvore representam os operadores de dobramento. Estes operadores são responsáveis por definir o dobramento em si. Neste trabalho, foi escolhido um conjunto de operadores cardinais absolutos, ou seja, N (norte), S (sul), L (leste) e O (oeste). Eles são considerados absolutos, pois sozinhos conseguem definir a orientação de um aminoácido em relação ao anterior. Pode-se, portanto, dizer que um aminoácido de número ***n***

na cadeia está ao norte do aminoácido de número *n*-1, sem necessitar conhecer toda a cadeia anterior a ele. Basta saber o posicionamento cartesiano do aminoácido *n*-1 para se definir as coordenadas do aminoácido *n*. Naturalmente, o conhecimento das coordenadas dos demais aminoácidos da cadeia é fundamental para a detecção de colisões. Porém para o sistema de orientação conhecer somente o aminoácido anterior já é o suficiente. A alternativa seria um sistema relativo, em que deveriam ser utilizados operadores do tipo “frente”, “direita” e “esquerda”. Porém, sempre se deveria conhecer todas os operadores anteriores ao aminoácido que se quer representar. Matematicamente, não há diferença entre os sistemas de operadores absoluto e relativo, pois ambos representam o mesmo espaço de busca. Há, contudo, vantagens ou desvantagens que estão relacionadas à implementação do algoritmo de busca na árvore. Daí é possível fazer com que um método se sobressaia em relação ao outro, por explorar as suas características no algoritmo de busca. Escolheu-se o método de operadores absolutos pela facilidade de se implementar os cortes do espaço de busca, que serão discutidos na seção 3.4.

A árvore tem o objetivo de ser genérica, ou seja, ela desenvolve-se de forma igual para qualquer polipeptídeo. O que deve mudar é a profundidade da árvore, aumentando para polipeptídeos maiores. Na árvore, não são representados os aminoácidos em si, pois a estrutura primária é sempre a mesma durante a análise. Cada estado da árvore contém um conjunto único de operadores de direção. Este conjunto de operadores também pode ser chamado de subdobramento, pois representa um dobramento com um número de aminoácidos menor do que o polipeptídeo sob análise, desde que não esteja em um nó folha, pois daí seria um dobramento completo. Os arcos que unem os estados representam a descendência entre os nós. Ou seja, nós de profundidade maior necessariamente contêm o subdobramento que lhe deu origem, acrescido de mais um operador. Desta forma, cada ramo completo da árvore representa os passos que foram tomados para se gerar um dado dobramento, sendo que as folhas sempre têm o dobramento com número de operadores suficientes para comportar todos os aminoácidos do polipeptídeo. Em cada nó da árvore, são aplicados todos os operadores de direção. Como os operadores representam todos os movimentos possíveis dentro do modelo 2DHP, garante-se que todos os dobramentos possíveis serão gerados.

A Figura 24 mostra um exemplo de um fragmento do espaço de busca para um polipeptídeo de 4 aminoácidos fictício de cuja estrutura primária é HHPH. O espaço de busca completo não poderia ser mostrado na figura, pois, apesar de não ser grande (64 dobramentos), sua representação ficaria ilegível. Diversos nós de profundidade 3 não foram desenvolvidos pela questão do espaço disponível. Porém, na representação computacional,

idealmente todos são explorados até a profundidade que represente o número de ligações peptídicas da estrutura primária do polipeptídeo sob análise. Neste caso, para o polipeptídeo HHPH, existem 3 ligações peptídicas para interconectá-los.

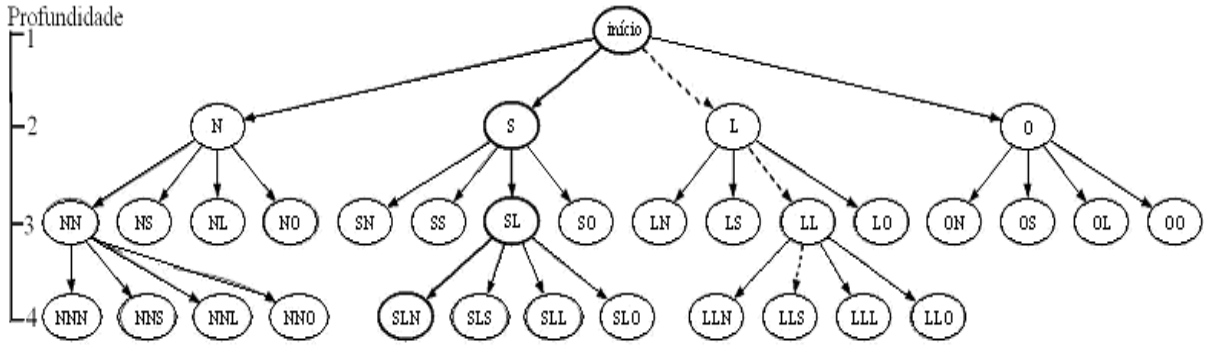

Figura 24 - Fragmento do espaço de busca completo do polipeptídeo HHPH.

Para seguir os passos que levaram a um dado dobramento, basta seguir da raiz até uma folha. Seguindo o caminho tracejado na árvore da Figura 24, pode-se perceber que a partir da raiz “início”, considerando-se o aminoácido inicial fixo, o segundo aminoácido está a leste. O terceiro aminoácido está situado a leste do segundo. Finalmente o quarto aminoácido está ao sul do terceiro. Tal sucessão completa de operadores é representada sempre no nó folha. Ao final do caminho tracejado, encontra-se o nó que representa o dobramento, ou seja, LLS. É importante perceber a descendência entre os nós. Todos os nós de profundidade 3, como o LL, estão contidos no estado subseqüente, ou seja, de profundidade 4, neste caso, o LLS. A relevância deste conceito está relacionada a própria maneira que o algoritmo deste trabalho faz a exploração da árvore, iniciando-se pela raiz, gerando os seus descendentes e assim por diante, com todos os nós, até chegar às folhas.

Outro aspecto importante a ser visualizado na Figura 24 é o dobramento SLN, marcado em negrito. Ele é uma das ocorrências do melhor dobramento (neste caso, há apenas uma iteração hidrofóbica) para o polipeptídeo HHPH. A representação gráfica no plano bidimensional do mesmo dobramento pode ser vista na Figura 25. Nesta figura, o aminoácido inicial de cada cadeia é marcado com um “X”, os aminoácidos hidrofóbicos com círculos e os polares com quadrados.

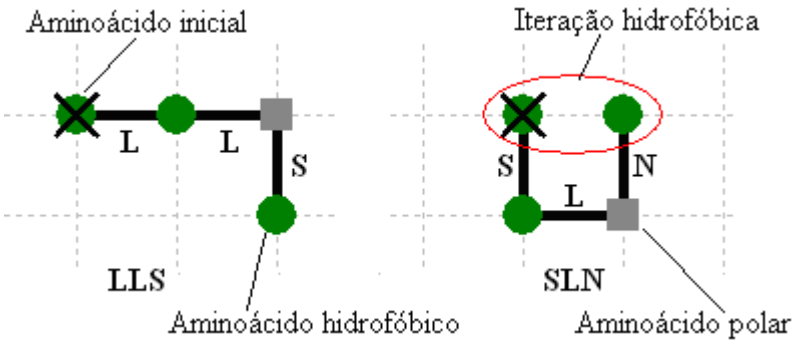

Figura 25 - Representação 2D dos dobramentos LLS e SLN do polipeptídeo HHPH.

3.3. CARACTERIZAÇÃO DO PROBLEMA

Antes de iniciar a explicação do algoritmo desenvolvido, é importante definir a forma matemática como o problema é visto, com o objetivo de estimar o tempo requerido para se executar uma análise do espaço de busca referido. Como a predição do dobramento é um problema de complexidade de tempo elevada é necessário quantificar o tempo por meio de uma equação, de modo a vislumbrar onde é possível trabalhar de modo a reduzir o tempo total de análise. A principal proposta é desenvolver a fórmula mostrada na equação 11, em que o termo $n_{estados}$ corresponde ao número de estados efetivamente presentes na análise, o termo $n_{instruções\_por\_estado}$, o número de instruções necessárias para se processar um estado e o termo $t_{instrução}$ significa o tempo que cada instrução leva para ser processada. Portanto, o tempo total corresponderia simplesmente à multiplicação dos termos e teria a mesma unidade de medida do termo $t_{instrução}$.

$$t_{total} = n_{estados} . n_{instruções\_por\_estado} . t_{instrução} \quad (11)$$

A interpretação destes termos não depende estritamente do seu valor matemático, pois o objetivo não é exatamente descobrir quanto de tempo uma determinada arquitetura leva para executar uma instrução. O que se busca com esta equação é principalmente o significado que cada um dos seus termos tem e qual a influência dele no desempenho do algoritmo. Assim, torna-se mais fácil abordar cada um dos aspectos do algoritmo e descobrir como pode ser feita a sua otimização. Portanto, o termo $n_{estados}$ está relacionado com o tamanho do espaço de estados que o problema envolve. Como será mostrado a seguir, o tamanho desse espaço, mesmo para cadeias pequenas, é grande e cresce de forma muito rápida se comparado ao aumento de aminoácidos da cadeia. Isso significa que o tamanho do espaço de busca para um polipeptídeo de n aminoácidos é razoavelmente maior que o espaço para uma cadeia de n-1

aminoácidos. Este espaço de estados é inteiramente independente da estrutura primária do polipeptídeo em questão.

Um dos avanços deste trabalho foi fazer o espaço ser dependente da estrutura primária, porém este aspecto será abordado na seção 3.4.2.1. De qualquer forma, este é o parâmetro que será utilizado para mensurar as reduções no espaço de busca. O termo $n_{instruções\_por\_estado}$ tem o significado de mensurar a qualidade da implementação, ou seja, ele indica o benefício relativo conseguido pela implementação atual. A interpretação disto é que quanto melhor for a implementação, menos instruções por estado deverão ser executadas. Finalmente, o parâmetro $t_{instrução}$ representa o desempenho da máquina em que se executa o código. Se algum tipo de paralelismo for utilizado, este tempo pode de ser reduzido, tendo-se como resultado a emulação de um processador com o desempenho de diversos processadores.

Inicialmente, será discutida a questão do número de estados, com a descrição da adequação do modelo 2DHP com a forma de árvore. Em seguida, a questão do número de instruções por ciclo com a explicação da implementação do algoritmo A* modificado. Por último, será abordada a questão do tempo por instrução, com a descrição da implementação paralelizada do algoritmo desenvolvido.

3.4. ANÁLISE DO ESPAÇO DE BUSCA

Tendo-se estabelecido o modelo de representação e a ferramenta matemática de quantificação do problema (equação 11), é importante definir-se a quantidade de estados que serão processados, para que se possa fazer uma previsão do tempo estimado para se varrer todo o espaço de busca. Como o algoritmo de análise de dobramentos deste trabalho é basicamente um algoritmo de busca exaustiva, quanto menos estados houver para serem analisados melhor será o desempenho do sistema. Por meio da análise do espaço de busca é possível identificar regiões que podem ser eliminadas na análise.

Considerando-se que quatro operadores de direção foram definidos neste modelo, é possível então elaborar uma equação para se encontrar o tamanho do espaço de busca de acordo com o tamanho do polipeptídeo.

A Equação 12 foi elaborada de acordo com o cálculo de permutação com repetição. O objetivo é encontrar o número de permutações com repetição possíveis com quatro operadores aplicados a n_{amino}*-1* aminoácidos, visto que o primeiro aminoácido da cadeia está fixo em seu lugar. Esta equação, justamente por descrever matematicamente todos os dobramentos

possíveis para um polipeptídeo, engloba os dobramentos que não devem ser incluídos na análise. Tal equação resultará em um espaço de busca que cresce muito rapidamente com um aumento linear do número de aminoácidos na cadeia.

$$n_{estados} = \sum_{i=1}^{i=n_{a\min o}} 4^{i-1} \qquad (12)$$

Nesta equação, $n_{estados}$ representa o número máximo de estados da árvore e n_{amino} é o número de aminoácidos do polipeptídeo.

É importante ressaltar a diferença entre o tamanho do espaço de busca do problema do dobramento e o número de dobramentos possíveis. O espaço de busca inclui todos os nós que podem ser gerados pela árvore. O número de dobramentos possíveis é o número de nós que podem ocupar a profundidade igual ao número de aminoácidos da seqüência. Em outras palavras, se a seqüência tem quatro aminoácidos, o número máximo de dobramentos que pode ser conseguido é o número de nós folha da árvore, ou seja, nós com profundidade 4. A equação 13 mostra como chegar a esse número, em que $n_{dobramentos}$ é o número máximo de dobramentos que podem ser gerados.

$$n_{dobramento\ s} = 4^{n_{a\min o}-1} \qquad (13)$$

Tanto o número de estados quanto o número de dobramentos calculados pelas equações muito raramente, e preferencialmente nunca, serão atingidos pelas análises do sistema desenvolvido neste trabalho, pois serão propostos cortes no espaço de busca para viabilizar tal análise. Porém, tais equações servem para se ter uma idéia do tamanho original do problema.

Estudando-se o espaço de busca completo, é possível perceber algumas características relativas ao problema tratado, algumas inerentes ao modelo adotado. A principal delas é que é possível definir o espaço de busca em algumas categorias de acordo com as características do próprio modelo, como mostrado na Figura 26, que seriam os dobramentos inválidos, os duplicados e os válidos. O espaço de busca válido ainda pode ser dividido entre dobramentos promissores e não-promissores. Finalmente, os dobramentos promissores ainda podem ser divididos em subótimos e ótimos.

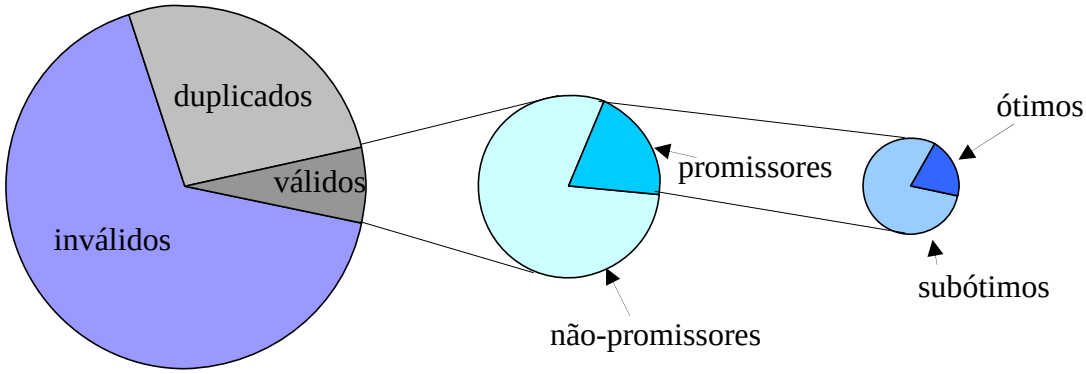

Figura 26 - Composição do espaço de busca para o problema do dobramento de proteínas.

Os dobramentos inválidos são aqueles que possuem colisões, ou seja, violam a regra do modelo 2DHP. Os duplicados são aqueles que ocorrem mais de uma vez, ou rotacionados no plano ou espelhados. Os dobramentos válidos seriam o conjunto que possui dobramentos livres de colisão e sem nenhuma duplicidade. Porém somente o fato de serem válidos não os caracteriza como, de fato, solução do problema. Dentro dos válidos, há ainda os dobramentos promissores e os não-promissores. Os não-promissores são aqueles dobramentos que não tem chance de ser ocorrências do melhor dobramento, como dobramentos com poucas ou nenhuma iteração hidrofóbica. O grupo dos promissores são aqueles que ou chegam perto do resultado ótimo, chamados de subótimos e os ótimos que são aqueles que de fato são ocorrências do melhor dobramento.

As categorias promissores e não-promissores são um pouco mais difíceis de classificar, pois se tratam de dobramentos válidos para o modelo 2DHP, entretanto podem ou não levar à solução ótima. Esta determinação, como será explicada no decorrer do capítulo, tenta separar, entre os dobramentos válidos, os que têm potencial de chegar a ser dobramentos ótimos e os que definitivamente não tem. Como não se sabe exatamente quantas iterações hidrofóbicas terão os melhores dobramentos, é feita uma avaliação baseada no potencial de gerar contatos futuros. Essa avaliação não é capaz de afirmar se um determinado ramo da árvore levará até uma solução ótima, mas consegue dizer com certeza quando não vale mais a pena desenvolver um dado ramo da árvore, pois ele já não mais tem potencial de gerar dobramentos ótimos. Tal avaliação será explicada em detalhes na seqüência deste capítulo.

Portanto, é neste espaço de busca promissor que efetivamente a busca exaustiva ocorrerá. Mesmo após a determinação dos ramos mais promissores, ainda há outras duas subdivisões, como mostrado na Figura 26: os dobramentos subótimos e os ótimos. Este último refere-se aos dobramentos que, de fato, não têm o seu número de iterações hidrofóbicas maior

do que qualquer outro dobramento, ou seja, eles são o resultado da busca. No entanto, eles ainda tendem a ser o grupo minoritário dentro dos promissores. Isto acontece justamente por causa da função que avalia o quão promissor é o dobramento. Se a função fosse ideal, o conjunto dos promissores seria composto apenas dos dobramentos ótimos. Como não se sabe de antemão as regras que gerarão os dobramentos ótimos e para que a função avaliadora seja confiável, ela tem que ser projetada de forma a eliminar os subdobramentos não promissores sem eliminar os promissores. Aceita-se, portanto, o possível revés de se classificar como promissores alguns subdobramentos não promissores, ou seja, falsos positivos. A desvantagem disso é a de gerar uma divisão no espaço dos dobramentos promissores, pois além dos ótimos, um conjunto de subótimos é gerado. Este conjunto representa ramos da árvore que são válidos e foram considerados promissores, mas que acabaram por não levar ao resultado ótimo. Sem se saber as regras exatas do processo de dobramento, a faixa dos subótimos não pode ser eliminada, sem o risco de se eliminar com ela dobramentos verdadeiros positivos. Até porque, se essas regras fossem claras, o espaço todo seria reduzido apenas aos dobramentos ótimos.

3.4.1. Determinação do espaço de busca válido

Na Figura 26, pode ser observado o espaço de busca completo para o dobramento de proteínas no modelo 2DHP. A maior categoria dentro dele é a de dobramentos inválidos que, em alguns casos, correspondem a mais de 99% do espaço de busca, como pôde ser detectado experimentalmente neste trabalho. Infelizmente, 1% pode não significar um espaço de busca pequeno, pois o problema trata de um número de estados realmente muito grande. Os dobramentos são classificados como inválidos quando contêm algum tipo de violação do modelo utilizado.

A violação que ocorre no modelo 2DHP é a sobreposição de aminoácidos na grade. O modelo especifica que, a cada ponto da grade, somente um aminoácido pode ocupar a posição. Estas sobreposições ocorrem sob a forma de cruzamentos que a cadeia realiza sobre ela mesma. O estudo sobre os dobramentos inválidos mostrou que, quanto mais cedo na árvore uma violação é detectada mais estados são eliminados da busca, pois uma violação no início de uma subárvore invalida toda a subárvore. Observando-se a árvore, isto se torna óbvio, pois quanto menor a profundidade da árvore, menos estados já foram analisados. Outra conclusão, nesse estudo do espaço de busca, mostrou que a absoluta maioria das violações é ocasionada por aminoácidos dobrando-se sobre outros, com intervalo de duas ligações

peptídicas. Tal violação, por ser extremamente comum no estudo, foi denominada de violação de Regra2, ou simplesmente Regra2, e pode ser vista na Figura 27 em que está mostrado o dobramento inválido LLO.

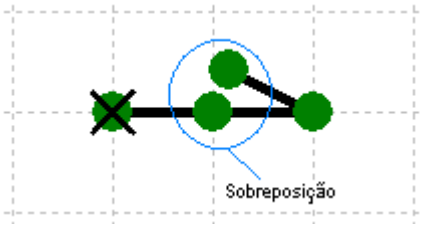

Figura 27 - Exemplo de um dobramento LLO, que é inválido pela Regra2.

Pela Regra2 chega-se a uma regra simples, mas extremamente útil, que será utilizada mais adiante, que é jamais permitir a ocorrência consecutiva dos operadores S e N e dos operadores L e O, pois eles gerariam uma violação da Regra2. Outra característica muito importante do modelo e que foi percebida por meio do estudo da Regra2, é que somente pode haver colisão entre aminoácidos onde há um número ímpar de aminoácidos entre eles, ou um número par de ligações peptídicas. Ou seja, o 1º aminoácido somente pode colidir com o 3º, com o 5º, com o 7º, com o 9º e assim por diante. Da mesma forma o 2º apenas colide com o 4º, com o 6º, com o 8º e etc. Na Regra2, existem somente duas ligações peptídicas entre os aminoácidos envolvidos na violação. Da mesma forma, elaborando-se uma Regra4, ela representaria as violações realizadas por resíduos distantes de quatro ligações peptídicas, ou três aminoácidos. Analogamente poderia ser definida a Regra6 ou a Regra8. Este comportamento pode então ser generalizado para a forma da RegraN, que representa todas as violações possíveis de ocorrerem no modelo 2DHP. A Figura 28 mostra dois exemplos de dobramentos com colisão. Em um deles, há uma colisão entre o 3º e o 7º aminoácidos (Regra 4)e no outro colisão entre o 4º e o 10º aminoácidos (Regra 6).

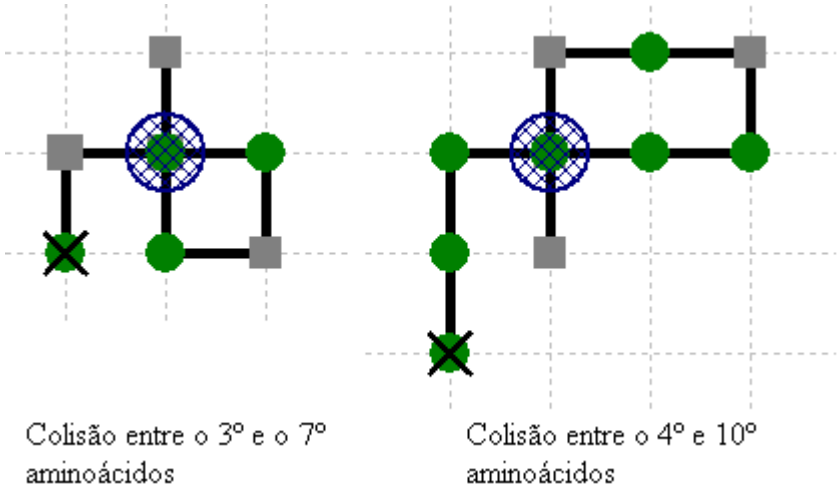

Figura 28 - Exemplos de dobramentos inválidos por colisão.

Estas colisões, quando colocadas no modelo de árvore, indicam que não há motivo para continuar desenvolvendo tais ramos. Isto acontece porque, como os subdobramentos subseqüentes herdam as características dos nós que os geraram, qualquer outro subdobramento de um nó com uma colisão conterá a mesma colisão. Portanto um ramo da árvore que contenha uma colisão deve ser cortado da árvore. Este corte representa uma redução expressiva no espaço de busca. Quanto mais cedo é detectada uma colisão, maior é o corte. Portanto, a redução obtida pela Regra2 é razoavelmente maior que pela Regra4 e assim por diante, pois quanto menor estiver o desenvolvimento do ramo, menos nós terão sido desenvolvidos. Porém, é imprescindível a implementação da RegraN, para se garantir que não haja nenhum dobramento inválido sendo analisado.

A característica mais interessante no modelo 2DHP que leva à RegraN, é a de que só é possível ter violações com aminoácidos espaçados com número par de ligações peptídicas na cadeia, ou seja, aminoácidos cujos índices tenham paridade igual. Na Figura 28, os aminoácidos que provocam a colisão são sempre ou ambos pares ou ambos ímpares. A conseqüência disto é que só é possível ter interações hidrofóbicas entre aminoácidos cujos índices tenham paridade diferente. É importante observar que um espaçamento mínimo de três ligações peptídicas tem que ser respeitado para a geração de iterações hidrofóbicas e um espaçamento de duas ligações para a formação de colisões. Portanto, não é possível ter iteração hidrofóbica entre aminoácidos H que ocupem, por exemplo, as posições 0 e 1, por duas razões. A primeira é que não se pode ter ligação peptídica entre os aminoácidos formadores de iteração hidrofóbica, como já descrito anteriormente. A outra é que simplesmente não há espaço para se dobrar a seqüência. Isso é mostrado na Figura 25, no dobramento SLN, no qual existe uma iteração hidrofóbica entre os aminoácidos de índice 1 e

4, um par e outro ímpar. Isso reduz bastante o esforço computacional do algoritmo de análise dos estados. A Figura 29 mostra as possibilidades de iteração hidrofóbica para um polipeptídeo hipotético HHPHHPHP. É importante perceber que só é possível dobrar a seqüência de modo que o primeiro, o quarto e o sexto aminoácidos ou o segundo e o quinto tenham iteração entre si. Qualquer outra combinação entre eles não é possível, pois a seqüência não conseguiria se dobrar.

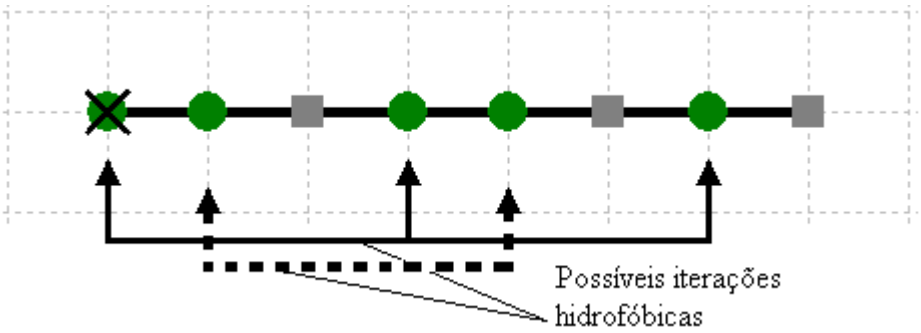

Figura 29 - Exemplo de possibilidade de geração de iteração hidrofóbica.

A parcela do espaço de busca total denominada como inválida é o resultado de todos os dobramentos que se encaixam na RegraN e que deverão ser eliminados durante a análise.

Outra parcela do espaço de busca, já considerando a eliminação proporcionada pela RegraN, seriam os dobramentos duplicados, ou seja, dobramentos que se repetem no espaço de busca. Tais dobramentos, por já terem sido representados anteriormente por outros dobramentos, não representam adição de conhecimento à análise e devem ser eliminados o mais rápido possível, a fim de evitar desperdício de esforço computacional.

Esta categoria chamada “duplicados” pode ser dividida em duas subcategorias. A primeira e maior seria a dos dobramentos idênticos, apenas rotacionados no espaço bidimensional, como mostrado na Figura 30.

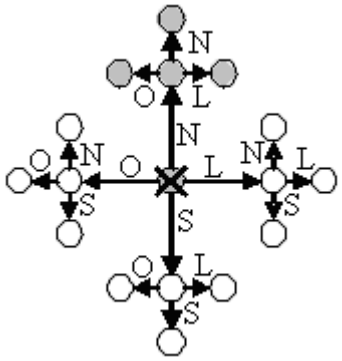

Figura 30 - Exemplo da duplicação de dobramentos gerada por rotação no espaço bidimensional, em uma cadeia de 3 aminoácidos.

Como pode ser visto na Figura 30, estaria então mostrado, fora de escala para melhor entendimento, o espaço de busca para uma cadeia de três aminoácidos. Pode-se perceber que existem vários dobramentos idênticos, rotacionados no espaço. Por exemplo, o dobramento NN é o mesmo que o dobramento OO ou o LL ou SS. Da mesma forma, o dobramento NL é o mesmo que o LS, o SO e o ON. Dessa maneira, se fossem eliminados os dobramentos duplicados por rotação, o espaço de estados seria reduzido de doze dobramentos para apenas três. Se isso for aplicado em seqüências maiores, o benefício torna-se cada vez mais atrativo. Tal ganho é sempre de 75%, pois o que se faz é eliminar todos os dobramentos que não se iniciam por N, pois teriam sempre um equivalente rotacionado no espaço. O resultado seria então apenas seqüências iniciadas por N, destacadas na Figura 30, eliminando 75% do espaço de busca.

A outra subcategoria dos dobramentos duplicados é a denominada “dobramentos espelhados”. Tal parcela já conta com a redução de espaço proporcionada pela eliminação de dobramentos rotacionados no espaço e diz respeito a dobramentos que continuam ainda assim a se apresentarem como duplicados, mesmo na subárvore restante mostrada na Figura 31.

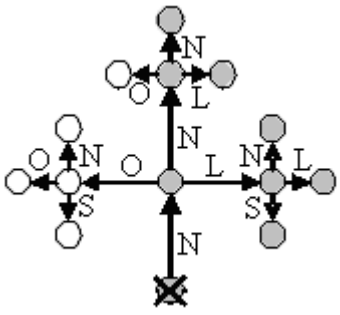

Figura 31 - Exemplo de duplicação de dobramentos por espelhamento, em uma seqüência de 4 aminoácidos.

Nesta figura, mesmo com a redução de 75% do espaço de busca continuam existindo dobramentos duplicados. Tal evento ocorre devido ao fato de naturalmente ocorrerem espelhamentos dentro do espaço de busca restante. Na Figura 31, os dobramentos NLS e NOS, NLN e NON, NLL e NOO e também por NNL e NNO são exatamente iguais, e não precisam todos ser analisados. Ou seja, dos 9 dobramentos mostrados esquematicamente na Figura 31, 4 dobramentos são repetições. Tais ocorrências devem, portanto, ser eliminadas da busca simplesmente por não agregar mais nenhuma informação que já não pudesse ser extraída dos demais dobramentos. A regra adotada então para a geração dos dobramentos é de nunca gerar um dobramento que contenha operadores Oeste precedidos somente por operadores Norte. Desta forma evita-se que haja um dobramento que seja espelhado em torno no eixo vertical, como acontece na Figura 31. Portanto, neste caso de uma seqüência de 4 aminoácidos, o espaço de busca foi cortado em 45% do espaço de busca que já havia sido cortado em 75% pela eliminação de dobramentos duplicados. A conseqüência é que, aplicando-se os cortes no espaço de busca a um polipeptídeo de 4 aminoácidos, o espaço é reduzido de 85 nós, com 64 dobramentos possíveis, para apenas 8 nós, com 5 dobramentos possíveis. Ou seja, apenas 9,41% do espaço original do número de nós, com 7,81% dos dobramentos possíveis, de fato são analisados. Este valor ainda é grande, pois, como será mostrado no capítulo 4, essa redução chega a um bilionésimo do espaço de busca original. A Figura 32 mostra o espaço de busca válido para o polipeptídeo HHPH, salientando a única ocorrência do melhor dobramento em negrito (NLS), com uma iteração hidrofóbica.

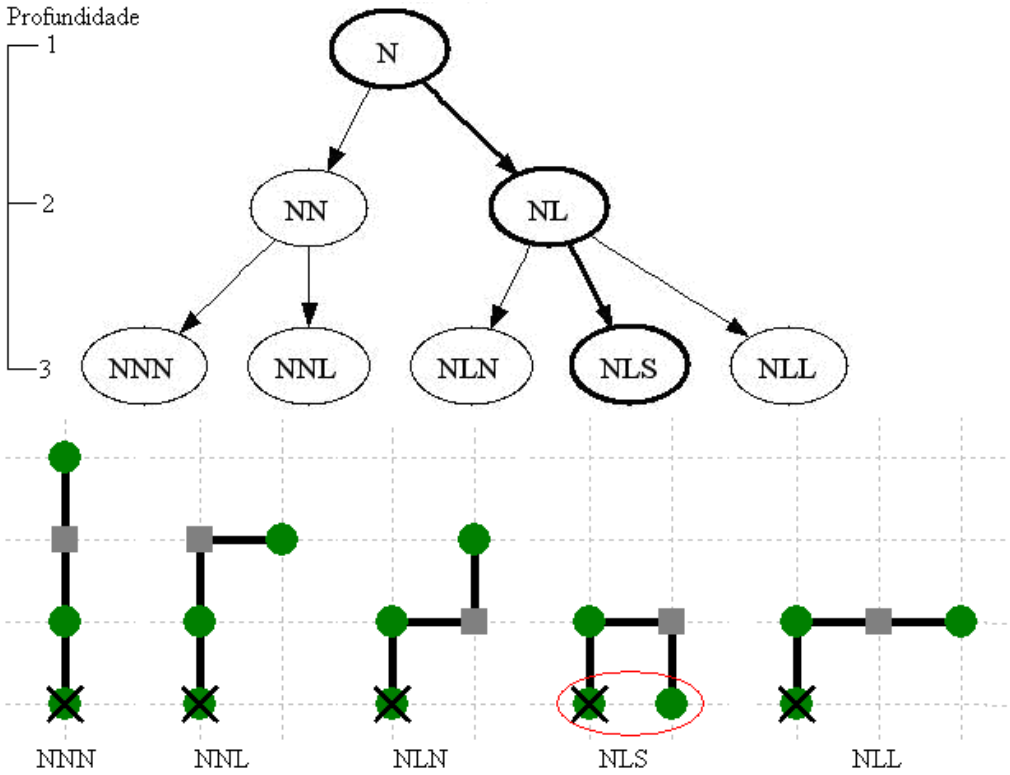

Figura 32 - Espaço de busca válido para o polipeptídeo HHPH.

Pode-se perceber na Figura 32 que o corte de dobramentos inválidos eliminou a ocorrência obrigatória de todos os operadores de direção em todos os nós e também eliminaram todas subárvores que não começam por N ou que têm O precedido somente por operadores N. Com isso, também é eliminado o nó nomeado como “início” na Figura 24, pois seu único descendente passa a ser o N, que assume o papel de raiz da árvore. Outro aspecto é que o primeiro aminoácido da seqüência é sempre considerado fixo. Todos os outros aminoácidos são dispostos no modelo 2DHP, considerando o posicionamento fixo do primeiro aminoácido. Portanto, o primeiro nó, que sempre é N, representa a informação de que o segundo aminoácido está ao norte do primeiro, pois a posição deste nunca muda. O conceito de profundidade da arvore agora também passa a representar o número de operadores de direção de cada dobramento e, conseqüentemente, o número de aminoácidos daquele dobramento menos 1. Por exemplo, na Figura 32, o dobramento completo NNL está em um nó de profundidade 3 na árvore, tem 3 operadores de direção e representa um polipeptídeo de 4 aminoácidos.

É importante notar que o melhor dobramento (NLS) também pode ser visto na Figura 25, porém rotacionado no espaço bidimensional. O dobramento SLN, da Figura 25, é exatamente o mesmo que o dobramento NLS, o que enfatiza a importância dos cortes já executados, eliminando dobramentos duplicados.

3.4.2. Determinação do espaço de busca promissor

Analisando-se a Figura 32, tem-se o correspondente espaço de busca válido da Figura 26, ou seja, todos os dobramentos inválidos ou duplicados já foram eliminados da análise. Sobrou somente o conjunto de dobramentos que podem levar ao conjunto dos dobramentos ótimos. Esse conjunto é dividido, então, entre os dobramentos promissores e os não-promissores, pois tem dobramentos que simplesmente não podem nunca ser ocorrências do melhor dobramento, mas ao mesmo tempo não contém nenhuma violação e nem são duplicações. Dois exemplos são mostrados na Figura 33, com o polipeptídeo fictício HPHHPP. Naturalmente, o dobramento LLLLLL mostrado não está em conformidade com a eliminação de duplicidade explicada anteriormente. Tal dobramento foi utilizado pela sua equivalência ao dobramento NNNNNN (que respeita a redução no espaço de busca) e ao menor espaço que ele ocupa.

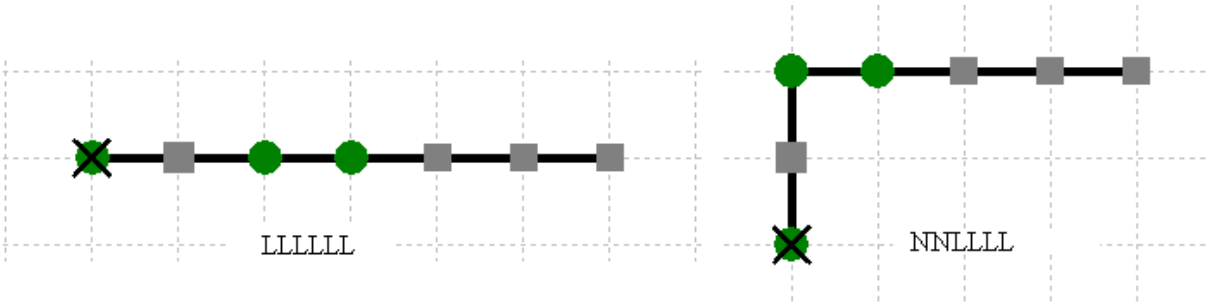

Figura 33 - Exemplos de dobramentos válidos e não-promissores.

Em ambos os casos, não existe nenhuma razão para não classificá-los como dobramentos válidos. Porém nenhum deles teria a mínima chance de ser candidato a uma ocorrência do melhor dobramento. Considerando que o polipeptídeo em questão, HPHHPPP só permite uma iteração hidrofóbica (verificado experimentalmente), que seria feita entre o primeiro e o quarto aminoácidos, qualquer ocorrência do melhor dobramento deve, obrigatoriamente, conter esses resíduos próximos um do outro. Em ambos os casos, a partir da colocação do 3º aminoácido já seria possível perceber que topologicamente não há como se colocar o 1º e o 4º aminoácidos juntos. Isso significa que qualquer outro desenvolvimento feito nesses ramos da árvore simplesmente não levará a nenhum resultado útil nenhum, pois a única chance que havia de se formar iteração hidrofóbica não resultou na tal iteração. Portanto, identificando-se essa situação a partir do 3º aminoácido, seria evitado todo o desenvolvimento da subárvore LL e da NN, pois garantidamente nenhuma das duas pode levar a alguma ocorrência do melhor dobramento. Em polipeptídeos maiores, isso, sem dúvida, poderia fazer a diferença entre uma busca durar anos ou alguns segundos.

Aplicando-se uma função de classificação no espaço válido da Figura 32, seria possível dividir o espaço restante, identificando o nó com o subdobramento NN como não-promissor, pois, de fato, nenhum nó descendente dele levará a solução. Isto provocaria o corte da subárvore iniciando em NN. A Figura 34 mostra como seria o espaço de busca resultante para o polipeptídeo de quatro aminoácidos HHPH mostrado na Figura 24. Finalmente, tem-se o espaço de busca que, verdadeiramente, será explorado pelo algoritmo proposto neste trabalho. A partir da árvore mostrada na Figura 24, que tinha 85 nós com 64 dobramentos possíveis, a redução fez com que a árvore tenha apenas cinco nós com três dobramentos possíveis. Ou seja, redução de 94,11% no número de nós e 95,31% no número de dobramentos possíveis.

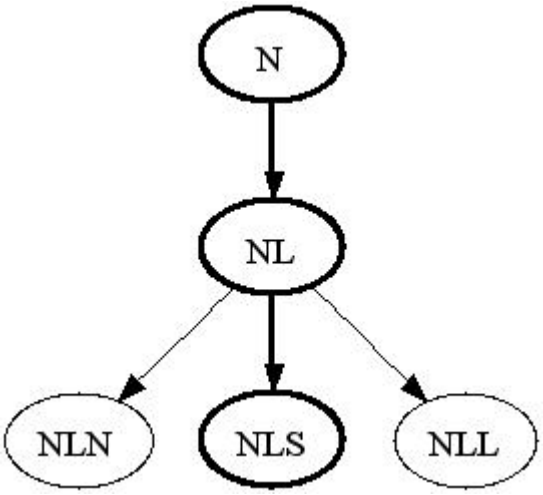

Figura 34 - Espaço de busca final para o polipeptídeo HHPH.

É importante observar na Figura 32 que dos 3 dobramentos possíveis apenas um é ótimo (NLS), ou seja, que atingiu o maior número possível de iterações hidrofóbicas. Os outros dois dobramentos, NLN e NLL, não conseguiram exibir o mesmo número de iterações hidrofóbicas que o NLS e também não são inválidos nem duplicados nem foram classificados como não-promissores. Ou seja, eles são a parcela do espaço promissor classificado como subótimos na Figura 26. Eles justamente representam a falha na função que classifica como promissor ou não e são casos falso-positivos de dobramentos promissores. Se essa função fosse mais eficiente, a Figura 32 mostraria apenas o dobramento NLS e a árvore que o gera. Porém, há que se avaliar se não é melhor incluir alguns falso-positivos na análise a correr o risco de excluir verdadeiros-positivos. Naturalmente, essa função hipotética aplicada ao polipeptídeo HHPH é apenas um exemplo de como a classificação de promissor ou não-promissor funcionará.

Para se conseguir essa redução adicional no espaço de busca foi elaborada, neste trabalho, a função avaliadora. Ela foi chamada de PDF (Potencial de Dobramento Futuro) e será explorada na próxima seção e a forma como o PDF aplica-se no algoritmo será explorada na seção 3.4.2.1.

3.4.2.1. Potencial de Dobramento Futuro

A estratégia para se calcular o espaço de busca promissor não é determinar, durante o desenvolvimento da árvore, quais são os subdobramentos que têm mais chance de levar a solução. Esta seria feita em uma abordagem heurística tradicional, porque simplesmente não há meio de saber com certeza absoluta qual vai ser o resultado. Logo, não se pode dizer o quão perto um subdobramento está da solução sem já ter explorado todo o espaço de busca e identificado quais de fato são os melhores dobramentos. Por outro lado, pela análise topológica é possível dizer o potencial que um dado subdobramento terá de gerar novas iterações hidrofóbicas.

É importante relembrar que o algoritmo deste trabalho desenvolve a árvore sempre gerando os descendentes de um dado nó, ou seja, se o nó continha o subdobramento NNLN, o próximo nó gerado conterá NNLN mais algum outro operador de direção, até atingir o tamanho do polipeptídeo sob análise, menos 1.

A função avaliadora tem como objetivo, portanto, quantificar a capacidade que uma dada proteína tem de gerar contatos hidrofóbicos, com base no seu dobramento parcial, obtido até o momento da análise e da sua estrutura primária. À medida que a estrutura da árvore vai crescendo, ou seja, à medida que os dobramentos parciais vão se tornando cada vez maiores, eles vão sendo avaliados pela função avaliadora. Esta função pode ser basicamente resumida a um quantificador de potencial de dobramento, denominado Potencial de Dobramento Futuro (PDF).

O índice PDF consiste de um vetor de números inteiros que estima o potencial que um dado subdobramento tem de gerar mais contatos hidrofóbicos. Ele considera a estrutura primária do polipeptídeo, fazendo com que o índice não se altere conforme o dobramento se desenvolve. Obviamente o potencial de dobramento varia de acordo com o subdobramento. Esta métrica tem como objetivo fornecer uma forma determinística e superestimada de se eliminar do espaço de busca nós que não levarão a dobramentos promissores, ou seja, os dobramentos com baixo ou nenhum potencial de gerar dobramentos ótimos.

No início da árvore, quando os subdobramentos ainda estão pequenos, o PDF indicado é alto, ao passo que, à medida que a árvore cresce sempre se verá um decréscimo no seu valor. Isso reproduz o processo de dobramento na natureza, sem considerar quaisquer desdobramentos que eventualmente possam ocorrer.

Quando a proteína está desdobrada, ela tem seu potencial máximo de dobramento, ou seja, ela pode assumir qualquer conformação, o que corresponde ao alto valor inicial do vetor PDF. Conforme o polipeptídeo vai se dobrando, o seu potencial de dobramento vai decrescendo naturalmente, pois os aminoácidos já dobrados estão fisicamente atados uns aos outros. Ou seja, um novo aminoácido adicionado na cadeia não tem o poder de modificar um subdobramento já formado, sem a ajuda de forças externas. Portanto, se um aminoácido teria, por exemplo, o potencial de ligar-se a, no máximo, outros dois aminoácidos gerando dois contatos hidrofóbicos, mas um dado dobramento o fez ficar próximo de outros dois aminoácidos polares ou simplesmente longe de qualquer outro aminoácido, seu potencial de dobramento foi desperdiçado, reduzindo o potencial de dobramento da proteína como um todo. O cálculo do PDF é apresentado na equação 14.

$$PDF_i = \sum_{i=i_p}^{i=n_{a\,\min o}} nIH_i - nRT \qquad (14)$$

A equação 14 pode ser traduzida como a soma das iterações hidrofóbicas (*nIH*) a partir da profundidade do próximo aminoácido a ser adicionado na seqüência (índice i_p) até o último aminoácido da estrutura primária (índice n_{amino}) menos o número de restrições topológicas (variável *nRT*). Em que n_{amino} é o número de aminoácidos do polipeptídeo. O aminoácido inicial da seqüência tem índice 1 o último deve ter índice n_{amino}. O somatório depende basicamente da estrutura primária do polipeptídeo e o quanto dela já foi utilizada. A variável *nRT* representa uma correção no somatório, pois serve para diminuir a previsão do número de iterações hidrofóbicas de acordo com que o dobramento permite. Portanto, a parte do somatório do PDF depende exclusivamente da sua estrutura primária e é calculado somente no início da análise. A parte da variável *nRT* é recalculada para todos os subdobramentos, pois fornecerá a correção necessária para potencial de dobramento, de acordo com o formato bidimensional assumido pelo subdobramento.

Em outras palavras, o PDF é sempre calculado desconsiderando o último aminoácido adicionado, bem como todo o conjunto de aminoácidos colocados antes dele. Exemplificando, supondo-se que o polipeptídeo sob análise tenha 20 aminoácidos. O algoritmo está

desenvolvendo o seu espaço de busca, fazendo os cortes necessários. Supondo-se que o algoritmo está no primeiro nó da árvore, ou seja, analisando a colocação do segundo aminoácido. O cálculo do PDF será aplicado do aminoácido de índice i = 2 ao aminoácido de índice i = 20. Da mesma forma, se o algoritmo já estivesse em um ponto mais avançado da árvore, digamos, progredindo da profundidade 14 para a 15, o nó a ser adicionado na árvore conteria 15 operadores de direção. Portanto, o PDF seria o somatório das iterações hidrofóbicas dos aminoácidos de índices 16 a 20.

Portanto, a base do cálculo do PDF é o levantamento da possibilidade de geração de iteração hidrofóbica que cada aminoácido da seqüência tem, de acordo com a posição que ele ocupa na mesma. Na Figura 35, os pequenos triângulos perto de cada aminoácido hidrofóbico indicam os pontos em que podem existir iteração hidrofóbicas.

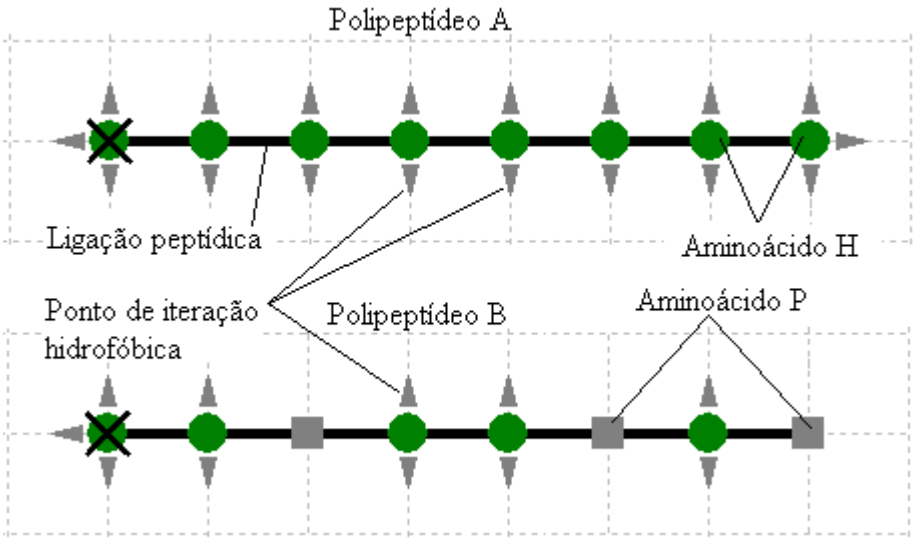

Figura 35 - Levantamento dos pontos de iteração hidrofóbica.

Como cada ponto de iteração hidrofóbica representa um local no modelo 2DHP que pode gerar uma iteração hidrofóbica, somente aminoácidos hidrofóbicos têm tais pontos. Na Figura 35 está mostrada outra característica importante, conseqüência da grade bidimensional utilizada no modelo 2DHP. Apenas os aminoácidos H que estão nas pontas da seqüência podem fazer três iterações hidrofóbicas. Todos os aminoácidos situados entre as pontas podem somente se aproximar de outros dois aminoácidos, tendo a chance de fazer somente duas iterações hidrofóbicas cada um.

O PDF então faz a soma dos pontos de iteração hidrofóbica, do final da seqüência para o começo, como mostrado na Figura 36. O resultado é montado em forma de um vetor que

não muda (pois depende da estrutura primária) e que será utilizado durante a análise. O fator de correção *nRT*, parte da equação 14, ainda não está mostrado e será discutido a seguir.

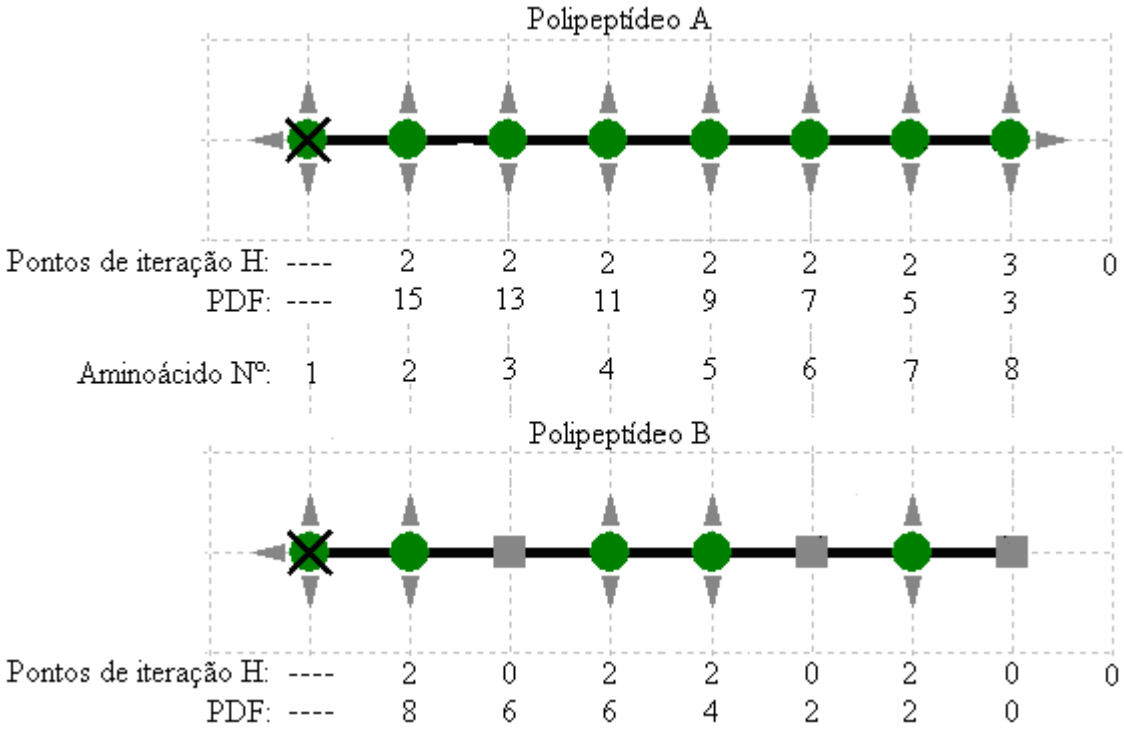

Figura 36 - Exemplo de cálculo do PDF.

Portanto, tomando-se, por exemplo, a seqüência B mostrada na Figura 36, e desenvolvendo a sua árvore o PDF do seu primeiro estado, a raiz da árvore, será 8. Isto acontece porque o primeiro aminoácido já está fixo, no início da árvore. O valor de 8 significa que, considerando a disposição atual dos aminoácidos já colocados, existe ainda o potencial de gerar 8 iterações hidrofóbicas, considerando o potencial individual de cada aminoácido. Da mesma forma, para esta mesma seqüência B, se a árvore fosse expandida até a profundidade 3, ou seja, com subdobramentos de 3 operadores de direção e, portanto, 4 aminoácidos já colocados, o PDF para a colocação dos próximos aminoácidos seria 4, de acordo com a Figura 36. Isto se deve ao fato de ainda haver 2 aminoácidos H, dos 4 que ainda faltam ser colocados, cada um com 2 pontos de iteração.

É importante notar que, na Figura 36, o PDF para o primeiro aminoácido de cada seqüência não existe. Na verdade, não faz diferença o seu valor, pois para a colocação do primeiro aminoácido, o PDF não pode ser consultado, pois é a raiz da árvore. Naturalmente, se a análise for feita em um polipeptídeo composto inteiramente por aminoácidos P o PDF para qualquer aminoácido será zero, o que será detectado pelo algoritmo. A última posição do vetor PDF é, conceitualmente, sempre zero, pois o potencial de se gerar novas iterações hidrofóbicas após todos os aminoácidos já terem suas posições fixas é, sem dúvida, nulo.

Uma característica que deve ser notada é que, na Figura 36 o PDF mostrado é razoavelmente superior ao número de iterações hidrofóbicas possíveis para cada seqüência. Experimentalmente, neste trabalho, foi determinado que a seqüência A faz 3 iterações hidrofóbicas e a B apenas 2. Por esse motivo, o cálculo do PDF é superestimado, ou seja, faz uma previsão acima do valor real. É este erro na previsão que justamente levará ao aparecimento do conjunto dos dobramentos subótimos da Figura 26. Na verdade, não é um erro, pois se baseia em características reais do modelo 2DHP e seu potencial de realizar iterações hidrofóbicas. A razão de ele ser tão mais alto que o número real de iterações para o cada polipeptídeo é que no seu cálculo, que depende somente da estrutura primária, não se leva em consideração o aspecto topológico do dobramento já realizado. Um exemplo deste efeito é mostrado na Figura 37.

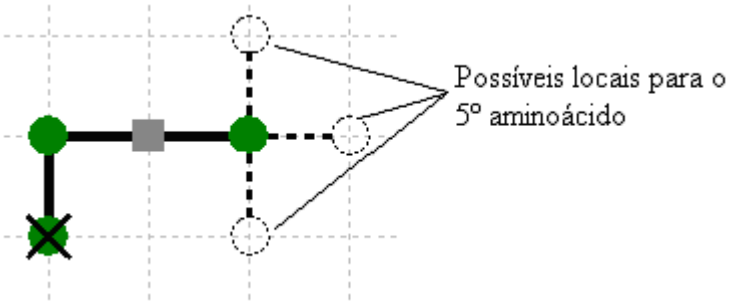

Figura 37 - Exemplo de anulação topológica do PDF no subdobramento NLL.

De acordo com a Figura 29, o quinto aminoácido somente poderia fazer iteração hidrofóbica com o segundo aminoácido. Porém, considerando-se o cenário da Figura 37, que mostra o subdobramento NLL, não há nenhum lugar em que se possa colocar o quinto aminoácido que faça com que ele gere uma iteração hidrofóbica com o segundo aminoácido. Portanto, o PDF para a colocação do quinto aminoácido não poderia ser mais 4 como mostrado na Figura 36, mas, sim, 2, pois o potencial de gerar iterações deste aminoácido foi desperdiçado.

A solução então foi implementar um fator de correção, denominado ***nRT***, ou número de restrições topológicas, conforme mostrado na equação 14. O objetivo é calcular um índice que possa incluir no PDF a informação relevante do subdobramento em análise. Ao passo que o vetor do PDF é calculado no início da análise e permanece sempre igual, o fator ***nRT*** tem que ser calculado para todo novo nó adicionado na árvore, ou seja, cada operador de direção adicionado na árvore gera um novo cálculo do ***nRT***. Isto torna o PDF mais fidedigno ao modelo, pois não faz sentido computacional algum considerar que um subdobramento totalmente esticado, por exemplo, NNNNNNN, que não gerará sequer uma iteração

hidrofóbica em qualquer polipeptídeo, tenha o mesmo PDF de outro subdobramento que poderá levar a um dos dobramentos ótimos.

Idealmente o cálculo do índice ***nRT*** deveria responder a seguinte pergunta: "Considerando-se o posicionamento dos aminoácidos que já fazem parte de um dado subdobramento, como tal subdobramento afeta o potencial de gerar iterações hidrofóbicas do aminoácidos que ainda restam para serem colocados na seqüência?". A resposta ideal seria um tanto complexa computacionalmente. Considerando-se a Figura 37, no caso da colocação do quinto aminoácido, teria que ser feito um algoritmo que pudesse determinar se é possível colocar o quinto aminoácido perto do segundo, dispondo de apenas 1 ligação peptídica, pois as outras 2 do intervalo de 3 ligações entre o segundo e o quinto já estão colocadas e fixas no subdobramento. Em outras palavras, seria determinar se existe um caminho na grade 2DHP que possibilite a colocação do quinto aminoácido perto do segundo, já considerando os demais aminoácidos. Imediatamente após isso, teria que se calcular se há um caminho na grade para que sétimo aminoácido (de acordo com a Figura 29) chegue perto do quarto ou o do primeiro aminoácido, ou ambos. Esse problema poderia resolvido por um algoritmo de busca que possa determinar caminho em grafos, como o A*, já descrito neste trabalho, a um determinado custo computacional. Porém, isto teria que ser feito em todos os estados da árvore, para cada um dos aminoácidos que ainda não foram adicionados. Ou seja, adicionando tal carga computacional (um algoritmo de busca funcionando a cada iteração do algoritmo de busca principal) certamente reduziria o conjunto dos dobramentos não-promissores, potencialmente eliminando-os. Porém, considerando que o espaço de busca é enorme, o processamento poderia não ser feito em tempo viável.

Para se implementar o índice ***nRT*** sem ter que recorrer a métodos de busca de caminho em grafo, foi elaborado o método de viabilidade topológica. Este método define o parâmetro ***nRT*** como uma condição para a inclusão ou não dos pontos de iteração hidrofóbica no cálculo do PDF daquele aminoácido. Por exemplo, na Figura 37, deveria excluir do cálculo do PDF os pontos de iteração do quinto aminoácido, considerando apenas o do sétimo aminoácido, baixando o PDF de 4 para 2.

O índice é então calculado por meio da condição mostrada na equação 15,

$$dm(a_i, a_u) - d(a_u, a_f) < 2 \qquad (15)$$

em que a_i é o aminoácido inicial da suposta iteração hidrofóbica, a_u é o último aminoácido do subdobramento e a_f é o aminoácido final da iteração hidrofóbica. O termo $dm(a_i,a_u)$ é a distância *Manhattan* entre a_i e a_u e $d(a_i,a_f)$ é a distância em número de ligações peptídicas entre a_u e a_f. A distância *Manhattan* é o número de arestas da grade 2D necessárias para se chegar de um lugar a outro. Os pontos de iteração hidrofóbica de um dado aminoácido a_f somente são utilizados no PDF a condição da equação 15 for verdadeira. Em outras palavras, como o vetor PDF já estava calculado desde o inicio da análise, se a condição da equação 15 não for verdadeira o valor correspondente ao número de pontos de iteração daquele aminoácido é subtraído do PDF, como mostrado na equação 14.

Para o caso do exemplo mostrado da Figura 37, com o subdobramento NLL, o cálculo de *nRT* aconteceria conforme se segue. Quando chega o momento da colocação do quinto aminoácido terá de ser verificada a viabilidade topológica da realização de iteração hidrofóbica dos aminoácidos H que ainda faltam ser inseridos no subdobramento, no caso o quinto e o sétimo aminoácidos. Primeiramente determinam-se as coordenadas cartesianas dos aminoácidos já colocados, assim como ilustrado na Figura 38.

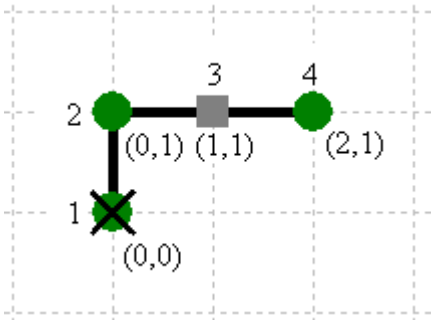

Figura 38 - Subdobramento NLL, em que o do quinto aminoácido não forma iteração hidrofóbica.

Identificando-se as variáveis da equação 15, para o quinto aminoácido os valores de a_i = 2, pois é o único aminoácido H que pode ter iteração com o quinto, a_u = 4 pois o último aminoácido do subdobramento é o de índice 4 e a_f = 5, que é o índice do aminoácido que se deseja adicionar. As distâncias seriam então: $dm(a_ia_u)$ = 3 e $d(a_ia_f)$ = 1. Aplicando-se na equação 15, descobre-se que a condição não se prova, ou seja, 3 − 1 = 2 e a condição teria que ser menor que 2. Portanto, isto significa que não há meios de o aminoácido 5 formar uma iteração hidrofóbica com o aminoácido 2. Seu potencial de dobramento deve ser subtraído da previsão inicial feita no cálculo do vetor PDF, baixando seu valor previsto de 4 para 2.

Se o subdobramento fosse NLN, como mostrado na Figura 39, ao invés de NLL da Figura 38, o resultado seria outro. Considerando que as variáveis são as mesmas e recalculando-se as distâncias, tem-se que $dm(a_i a_u) = 2$ e $d(a_i a_f) = 1$. Portanto, a condição da equação 15 está provada, pois $2 - 1 < 2$. Isto significa que existe o potencial de se gerar iteração entre os aminoácidos 2 e 5 e que o subdobramento atual não o eliminou. Este processo é repetido para todos os demais aminoácidos hidrofóbicos que serão colocados a seguir, podendo levar o PDF para zero, a partir do valor que foi calculado inicialmente, caso não haja mais potencial de contato.

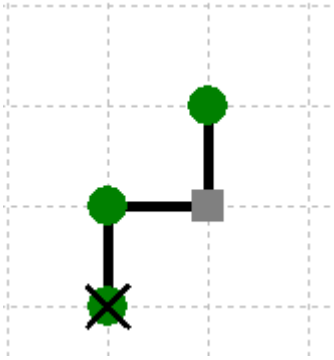

Figura 39 - Exemplo do subdobramento NLN, no qual o PDF permanece inalterado.

O cálculo da viabilidade topológica, porém, é simples demais para conseguir determinar esta viabilidade com 100% de precisão. Existem alguns casos em que a equação 15 não consegue determinar se existe ou não potencial de gerar novas iterações hidrofóbicas. Se for considerado um polipeptídeo hipotético HPPPPH, que permite somente uma iteração hidrofóbica, um exemplo desta falha pode ser evidenciado. A Figura 40 mostra dois subdobramentos que fazem parte do espaço de busca deste polipeptídeo.

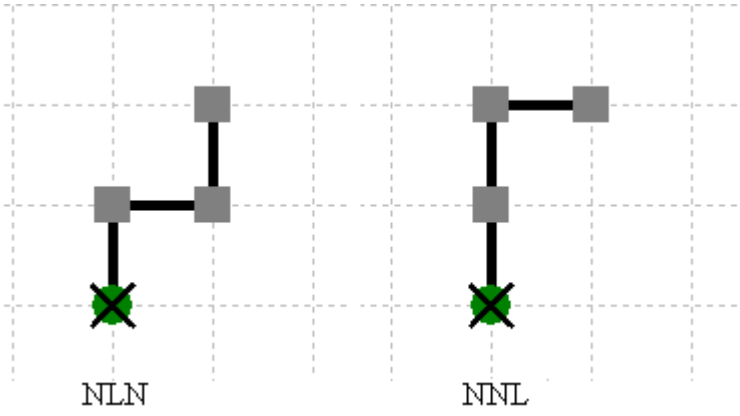

Figura 40 - Exemplos de subdobramentos do polipeptídeo HPPPPH.

Na Figura 40, são mostrados dois subdobramentos, o NLN e o NNL. Este levará a uma ocorrência do melhor dobramento (NNLSS), enquanto que aquele, NLN, não tem chance

alguma de fazer com que o 6º aminoácido venha a formar uma iteração hidrofóbica. Aplicando-se a equação 15 para se descobrir o PDF para a colocação do 5º aminoácido percebe-se que ela resulta igual para os dois, pois todos os parâmetros da equação 15 são iguais. O resultado é que ambos serão classificados como promissores, quando na verdade o NLN não é promissor. Se ambos forem desenvolvidos até o tamanho do polipeptídeo em análise, qualquer descendente do NLN não será uma ocorrência do melhor dobramento, ao passo que pelo menos um descendente do NNL (o NNLSS), será uma ocorrência do melhor dobramento.

Este tipo de anomalia acontece por falha na função avaliadora de viabilidade topológica. Os descendentes do NLN serão os dobramentos classificados como subótimos, dentro do espaço de busca promissor, na Figura 26, pois foi considerado promissor e acabou não resultando em uma ocorrência do melhor dobramento.

A utilização do PDF no algoritmo é que fará dinamicamente o corte dos subdobramentos não-promissores. A sua utilização no algoritmo será explicado na próxima seção.

3.4.3. Resumo das regras de corte do espaço de busca

A Tabela 5 apresenta de forma resumida os cortes realizados no espaço de busca, na ordem que eles acontecem. A coluna “Aplicação” se refere ao espaço de busca em que a regra é aplicada. Os espaços de busca referenciados são os mesmos da Figura 26. A ordem é extremamente importante pois os resultados são completamente diferentes quando se aplica uma regra ao espaço de busca errado.

A coluna “Regra” identifica o procedimento realizado para se fazer o corte. Por último, a coluna “Efeito” mostra o que acontece com o espaço de busca indicado após a aplicação da regra.

É importante notar que as duas regras de corte de duplicação são aplicadas ao mesmo tempo, no espaço de busca completo. O resultado seria um espaço de busca composto pelos dobramentos válidos e os inválidos. A eliminação de colisões remove os dobramentos inválidos. O espaço de busca resultante é o que contem os dobramentos válidos. Por ultimo, a aplicação do PDF no algoritmo de busca separará os dobramentos promissores dos não-promissores.

Tabela 5 - Resumo das regras de corte do espaço de busca

Aplicação	Regra	Efeito
Espaço de busca completo	Eliminação dos dobramentos que não se iniciam com o operador N (norte) Eliminação dos dobramentos que contém O (oeste) precedidos de N (norte	Corte dos dobramentos duplicados
Espaço de busca completo, livre de duplicações	Eliminação de todos os dobramentos com colisão	Corte dos dobramentos inválidos, gerando o espaço de busca válido
Espaço de busca válido	Aplicação do PDF no algoritmo	Corte dos dobramentos não-promissores

3.5. ALGORITMO DESENVOLVIDO

3.5.1. Algoritmo A* modificado

O algoritmo desenvolvido é uma modificação do algoritmo A*, já descrita na seção 2.5.4. Ou seja, cada nó da árvore, ou subdobramento, é expandido nos seus descendentes e assim sucessivamente, de modo que todos os nós tenham a chance de gerar descendentes. Esta geração, como já mencionado anteriormente, acontece progressivamente, iniciando-se pela raiz da árvore, o nó N, incorporando os cortes necessários, desenvolvendo os subdobramentos, até se chegar ao tamanho desejado do polipeptídeo sob análise. A sistemática do algoritmo segue o fluxograma apresentado na Figura 16, na seção 2.5.4.2. A forma de explorar o espaço de busca é mostrada na Figura 41, na forma do desenvolvimento da árvore de estados, com a seqüência de geração dos nós, ainda sem a aplicação do PDF. Pode-se perceber que a exploração tende, sem dúvida, a ser uma busca em extensão, como descrito na seção 2.5.3.1.

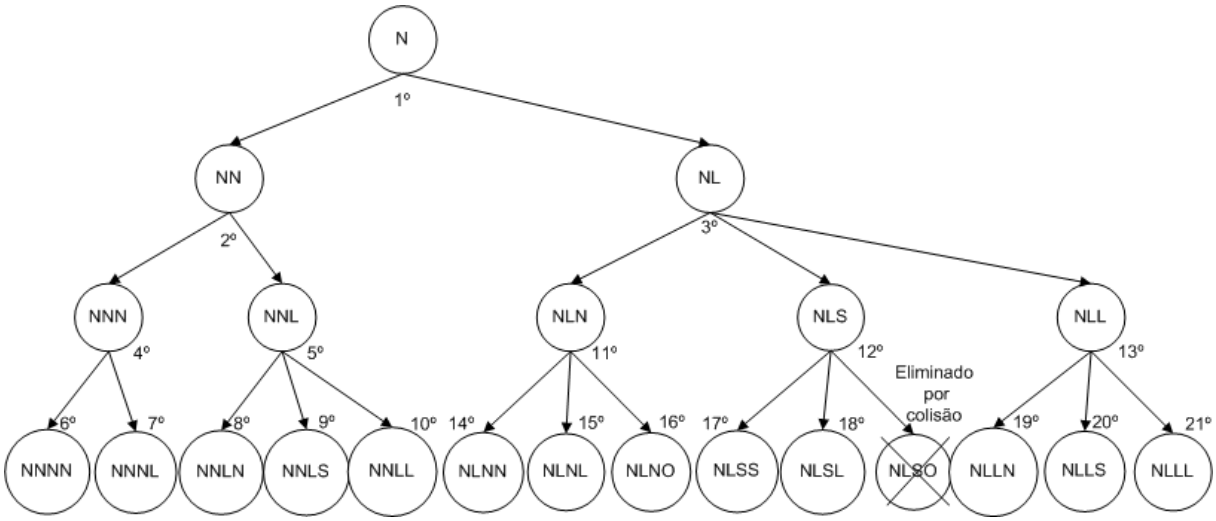

Figura 41 - Exemplo da ordem de geração dos nós na árvore.

Na Figura 41 também está mostrada eliminação do nó NLSO, pois ele contém uma colisão entre o primeiro e o quinto aminoácido.

O algoritmo se inicia analisando o nó N. A partir dele, gera-se todos os seus descendentes que não violem as regras de redução de espaço de busca descritas na seção 3.4. Então, a partir do nó N são gerados apenas os nós NN e NL. Portanto, o nó N, que estava na lista de subdobramentos a serem analisados é retirado da lista, gerando-se seus descendentes e estes inseridos na lista. Esta inserção é feita na ordem que os subdobramentos são gerados. Mesmo com o uso do PDF, que será descrito a seguir, qualquer tipo de ordenação feita nesta lista não traz vantagem ao desempenho do algoritmo. Possivelmente há algum beneficio na ordenação dos subdobramentos, porém, por se tratar de uma busca exaustiva, não faz muita diferença a ordem que eles são inseridos, pois todos serão analisados.

A lista do algoritmo é preenchida de forma a sempre conter os descendentes do nó sob análise, que é sempre o nó que ocupava a primeira posição na lista. O comportamento da lista é mostrado na Figura 42, sem a aplicação do PDF.

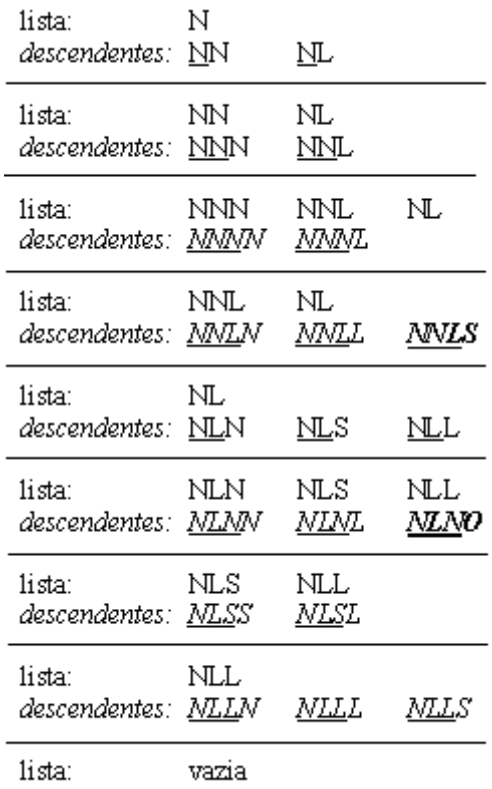

Figura 42 - Comportamento da lista de subdobramentos para um polipeptídeo de 5 aminoácidos.

Na Figura 42 também estão mostradas algumas características da lista. Nos descendentes gerados, a herança do nó que os gerou está sublinhada, como no subdobramento NLLN, que foi gerado a partir do nó NLL. Outra característica é que apenas os nós que possuem o número requerido de operadores de direção, neste caso 4, estão marcados em itálico e dentre eles, os ótimos estão em negrito.

É importante notar, na Figura 42, que os subdobramentos vão crescendo até chegar ao tamanho de 4 operadores de direção, ou seja, suficientes para comportar cinco aminoácidos, o tamanho do polipeptídeo sob análise. Quando um descendente de um subdobramento atinge o tamanho requerido, ele é avaliado para se saber se é ótimo ou não. Tais nós, que já são dobramentos completos nunca são re-inseridos na lista ou tem seus descendentes gerados, pois não faz sentido ter um dobramento de um polipeptídeo de cinco aminoácidos com mais de quatro operadores de direção. Isto faz com que a lista se esvazie gradualmente. À medida que os dobramentos completos vão sendo encontrados eles vão sendo armazenados em uma lista auxiliar, chamada de lista dos melhores dobramentos, de modo que ela contenha sempre as ocorrências dos dobramentos com maior número de iterações hidrofóbicas. Cada vez que um dobramento exibe um número de iterações hidrofóbicas maior do que o que já estava na

lista dos melhores dobramentos, toda a lista é descartada e este novo dobramento é inserido, somente aceitando dobramentos com número igual de iterações hidrofóbicas.

Este processo de geração de nós se repete até que lista esteja vazia, representando que todo o espaço de busca já foi explorado.

O algoritmo descrito até agora ainda não tem a principal característica dos algoritmos de busca com informação, como o busca gulosa e o A*, que seria uma função de avaliação dos subdobramentos para se encontrar os dobramentos promissores. Para se fazer tal seleção o uso PDF, é necessário.

A idéia da utilização do PDF é utilizá-lo na definição do conceito de PIH (Previsão de Iterações Hidrofóbicas). A geração de descendentes baseia-se no fato de que só vale a pena analisar um determinado subdobramento se o PIH do subdobramento indicar que ainda há potencial de gerar novos dobramentos interessantes. Portanto, os descendentes de cada nó somente são gerados se o PIH para o tal nó, somado ao número de iterações hidrofóbicas já obtidos pelo subdobramento representado por aquele nó, for igual ou superior ao maior número de contatos hidrofóbicos já obtido em toda a análise daquela proteína. Ou seja, se a soma das iterações hidrofóbicas já feitas com as que se ainda tem potencial de serem feitos, para aquele subdobramento, for menor do que o melhor resultado conseguido até o momento, o tal subdobramento não é promissor. Portanto, não adianta investir tempo analisando aquele dobramento, nem nenhum de seus descendentes, pois, de fato, não há mais potencial de igualar ou superar o melhor resultado obtido até agora. As equações 16 e 17 mostram esta condição.

$$PIH = PDF + nIHatual \tag{16}$$

$$PIH \geq Maior(nIH) \tag{17}$$

Nestas equações o PDF é o potencial de dobramento futuro, descrito na seção 3.4, ***nIHatual*** é o número de iterações hidrofóbicas apresentadas pelo subdobramento atual e ***Maior(nIH)*** é o maior número de iterações hidrofóbicas já encontrado durante toda a análise, ou seja, o número de iterações dos dobramentos da lista de melhores dobramentos.

Uma conseqüência importante da representação no modelo 2DHP e, provavelmente em todos os demais modelos, é que seu número de iterações hidrofóbicos tende a crescer ao se aumentar a árvore (pois se tem cada vez mais aminoácidos disponíveis para se formar iterações) e o potencial de dobramento tende a decrescer (pois há cada vez menos aminoácidos para serem adicionados). O resultado é uma dinâmica semelhante àquela mostrada na Figura 44, que mostra as curvas do número de iterações hidrofóbicas encontradas durante a análise e o comportamento do potencial de dobramento para o ramo da árvore que

resultará no subdobramento NLLLLLLLSLLLLLLLLLL, de um polipeptídeo hipotético de 27 aminoácidos, todos hidrofóbicos, como mostrado na Figura 43.

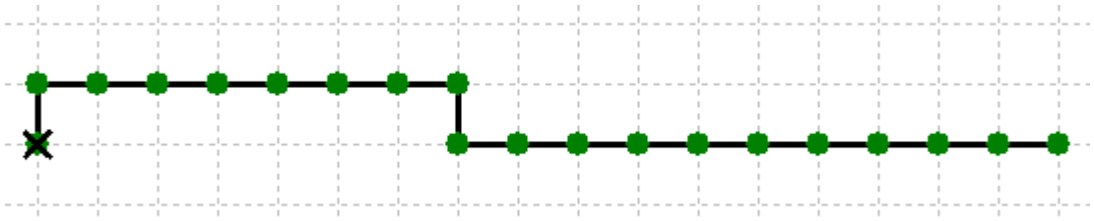

Figura 43 - Exemplo do subdobramento NLLLLLLLSLLLLLLLLLL, com todos os aminoácidos hidrofóbicos.

Essa curva foi obtida acompanhando a geração de cada subdobramento desse polipeptídeo e registrando o PIH a cada nova geração. Cada ramo da arvore tem um comportamento diferente, esse exemplo se aplica somente ao subdobramento citado.

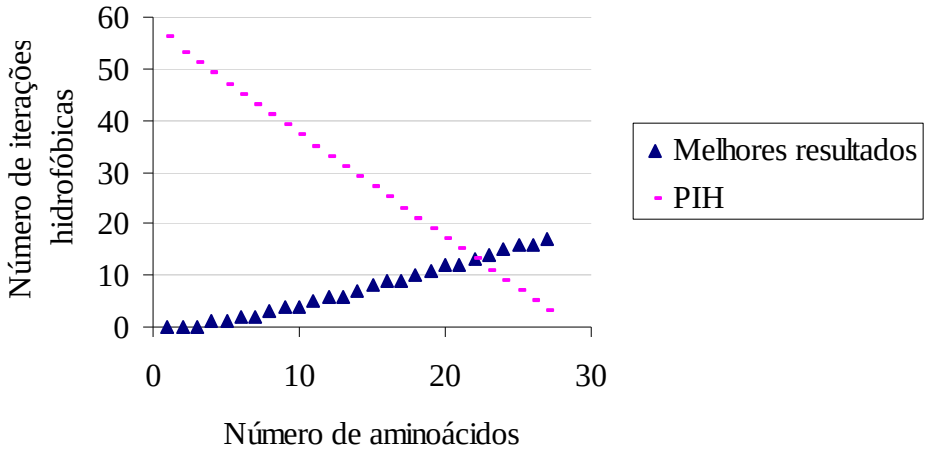

Figura 44 - Comportamento do PIH e do número de iterações hidrofóbicas encontradas, para o subdobramento mostrado na Figura 43.

É importante notar na Figura 44 o cruzamento das curvas do PIH e dos melhores resultados obtidos na análise. Ela indica o ponto de corte no qual se decide quando um subdobramento não mais será analisado, por ser não promissor.

De acordo com a equação 16, não se deve expandir o subdobramento Figura 43, cuja curva do PIH está mostrada na Figura 44. Considerando que esse subdobramento não gera nenhuma iteração hidrofóbica e supondo que o melhor resultado já conseguido por outros ramos da árvore é 17, nenhum descendente deste subdobramento é capaz de fazer com que tal marca seja igualada ou superada. Eliminando-se da análise o subdobramento da Figura 43, enquanto possui apenas 20 aminoácidos, evita-se que toda a sua subárvore seja desenvolvida

até atingir 27 aminoácidos para então se descobrir que nenhuma ocorrência do melhor dobramento resultou dela. Esse corte é muito dinâmico e extremamente difícil de ser quantificado. Ao contrário dos cortes mostrados na seção 3.4, não há uma redução padrão, ela depende tanto da estrutura primária quanto dos diversos subdobramentos do espaço de busca.

Por outro lado, analisando-se uma das ocorrências do melhor dobramento para a mesma proteína totalmente hidrofóbica, (NNNNNNLSSSLSOSSLNLNNNONNLS, com 17 iterações hidrofóbicas), o resultado é bastante diferente. Este dobramento é mostrado na Figura 45 e a Figura 46 mostra o seu gráfico.

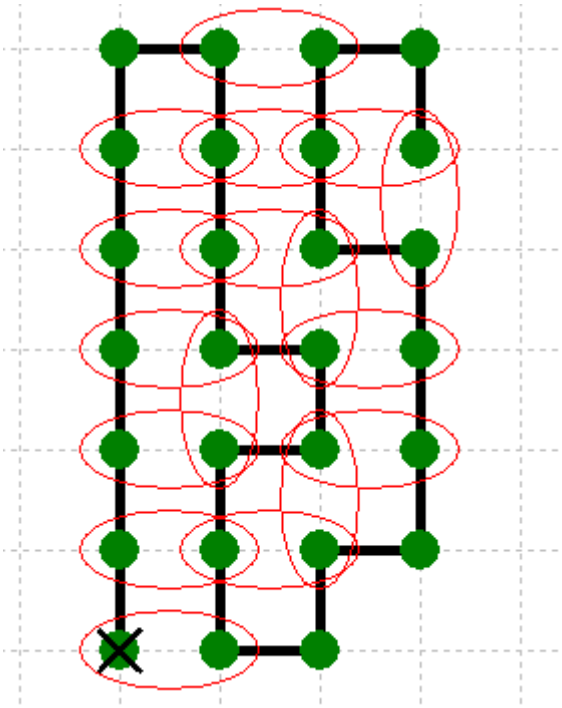

Figura 45 - Dobramento NNNNNNLSSSLSOSSLNLNNNONNLS, com 17 iterações hidrofóbicas.

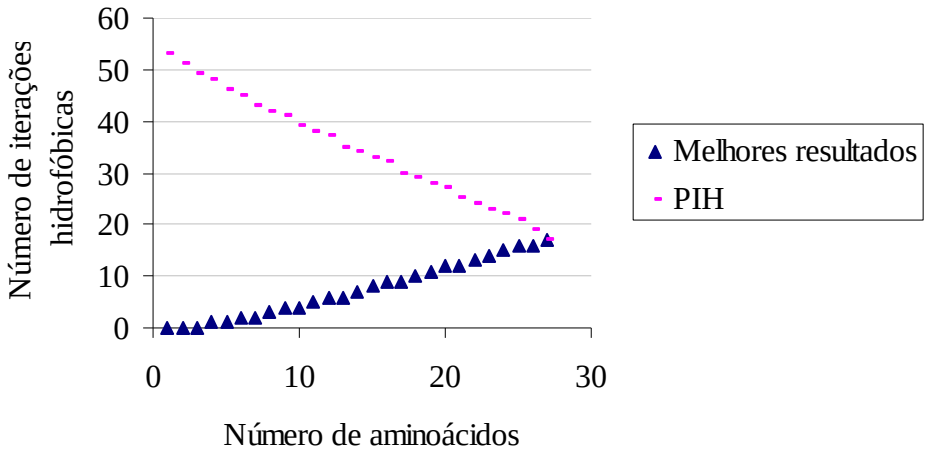

Figura 46 - Exemplo da avaliação do potencial de dobramento de um dobramento completo (NNNNNNLSSSLSOSSLNLNNNONNLS) de 27 aminoácidos.

Pode-se observar na Figura 46 dois aspectos importantes: as duas curvas também se cruzam e o cruzamento ocorre quando a proteína atinge seu tamanho máximo, de 27 aminoácidos. O segundo aspecto deve-se ao fato de a função que calcula o PDF tê-lo estimado como promissor durante todo o seu desenvolvimento, permitindo que ele progredisse até se tornar um dobramento completo, de 27 aminoácidos. O primeiro aspecto é o mais importante, pois as curvas se cruzam somente quando o subdobramento chega a ter 27 aminoácidos. Isto significa que este dobramento é uma ocorrência do melhor dobramento, pois o potencial mínimo de dobramento atingido é o máximo encontrado em toda a análise.

De forma análoga pode se perceber o mesmo comportamento no gráfico mostrado na Figura 47, que representa uma das ocorrências do melhor dobramento (NNNNLSSLNNLSSSOOS) para o polipeptídeo HHHHHHHHHHHHHHHHHHPPPPPPPPP, também com 27 aminoácidos. Assim como na Figura 46, as curvas se cruzam, porém, neste caso, elas permanecem juntas e niveladas, desde quando a proteína tem 18 aminoácidos de tamanho até que ela chegue a ter todos os seus 27 aminoácidos. Isto se deve, além do fato desta ser uma ocorrência do melhor dobramento, também ao fato de que de por causa da proteína ter, a partir do seu aminoácido nº18, somente aminoácidos polares, não apresentando potencial adicional algum de gerar novas iterações hidrofóbicas. Porém, o PIH também não cruza a curva dos melhores resultados. O PDF do subdobramento cai a zero, pois não há mais aminoácidos H a serem adicionados, mas o subdobramento já contava com 10 iterações hidrofóbicas, impedindo que o PIH tenha uma valor inferior a 10.

O significado disto é que qualquer outro dobramento válido feito após a colocação do 18º aminoácido não trará nenhuma iteração nova ao dobramento, pois o PIH sempre será igual ao melhor resultado já obtido.

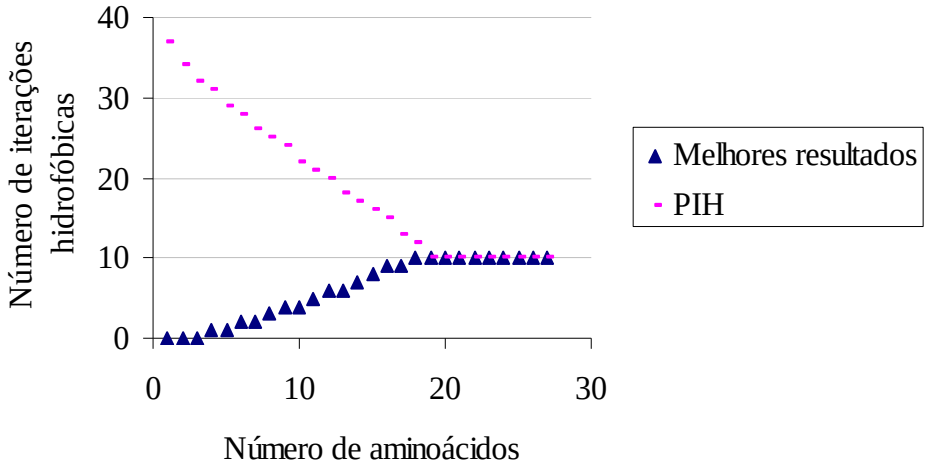

Figura 47 - Demonstração da avaliação do potencial de dobramento de um dobramento completo (NNNNLSSLNNLSSSOOS).

Portanto, quanto mais cedo a curva do PIH cortar a curva dos melhores resultados menos subdobramentos inúteis serão analisados, diminuindo o número de estados da árvore, potencialmente reduzindo o tempo de análise. Quanto melhor for a função que define o PDF, mais íngreme será a queda do PIH e mais rápido o processamento.

3.5.2. Ordenação Linear do Polipeptídeo

Como foi mostrado na seção 3.5.1, o comportamento da curva PIH é sempre decrescente de acordo com o aumento de aminoácidos colocados em um dobramento. Foi mostrado também a importância da curva do PIH cair o mais rapidamente possível, a fim de eliminar a maior parte do espaço de busca o mais cedo possível na análise. Portanto foi notado, durante o desenvolvimento deste trabalho, que alguns polipeptídeos tinham um desempenho razoavelmente mais rápido do que outros, mesmo tendo o mesmo tamanho e estruturas primárias parecidas. Um exemplo extremo pode ser mostrado com os dois polipeptídeos hipotéticos de 20 aminoácidos: PPPPPPPPPPHHHHHHHHHH e o HHHHHHHHHHPPPPPPPPPP. Ambos têm o mesmo número de aminoácidos e a mesma distribuição de aminoácidos, porém em ordem inversa. Experimentalmente foi determinado que o primeiro aminoácido demora razoavelmente mais que o segundo para ser processado e

fornecem exatamente os mesmos resultados. Isto se deve a um fato simples. No segundo, o PDF chega a zero quando os subdobramentos estão com 10 aminoácidos de comprimento, pois a partir daí todos os aminoácidos são polares, ou seja, não adicionam potencial de gerar novas iterações hidrofóbicas. Para o primeiro, o PDF somente chegará a zero quando os subdobramentos chegarem ao último aminoácido. Ou seja, na primeira seqüência o PDF se manterá alto e constante do 1º ao 10º aminoácido, caindo somente após o 10º aminoácido. Isto significa que no segundo polipeptídeo o PIH convergirá para seu valor final (o melhor resultado) mais cedo do que no primeiro. Portanto, para a segunda seqüência, a parcela do espaço de busca eliminada por não ser promissora é bem maior.

Experimentalmente foi levantado que, para a primeira seqüência o número de dobramentos promissores é de 7437051 enquanto que para a segunda é de apenas 1298673. O tempo de análise da primeira, em um computador com processador Intel *Pentium* 4 (tecnologia *HyperThreading*), com freqüência de *clock* de 3GHz, com 1GB de memória RAM, foi de 30,9 segundos, enquanto que da segunda foi de 3,05 segundos, no mesmo computador. É importante ressaltar que ambas as análises obtiveram os mesmos resultados. Em outras palavras o número de dobramentos promissores diminuiu 5,72 vezes e o tempo de análise diminuiu 10,13 vezes, apenas por se analisar o polipeptídeo de forma diferente.

Isto significa que a ordenação do polipeptídeo tem uma influência enorme no desempenho da análise. Para se automatizar a solução deste problema foi criado um índice nomeado IAH (Índice de Acúmulo de aminoácidos Hidrofóbicos). Este índice determina se a estrutura primária deve ou não ser invertida, ou seja, ordenada de trás para frente, para ser analisada.

O objetivo do IAH é ter seqüências com maior número de aminoácidos H perto da origem. Este índice diminui seu valor numérico quando há um maior acúmulo de aminoácidos H para a esquerda da estrutura primária. Portanto, comparando-se as duas estruturas primárias citadas anteriormente, a HHHHHHHHHHPPPPPPPPPP teria um IAH inferior, e portanto melhor, que a PPPPPPPPPPHHHHHHHHHH.

O cálculo do IAH é simples e leva em conta a posição que o aminoácido ocupa na estrutura primária, não importando se o número de aminoácidos é par ou ímpar. É feita a soma dos índices dos aminoácidos H da seqüência. Desta forma, como o índice dos aminoácidos é 1 no primeiro aminoácido (que está mais à esquerda), quanto mais aminoácidos H estiverem para o lado esquerdo menor será o IAH. A Figura 48 mostra um exemplo utilizando as duas seqüências de 20 aminoácidos. O valor absoluto do IAH de uma seqüência não representa nada ao processamento, ele tem obrigatoriamente que ser

comparado com o IAH da mesma seqüência invertida, para que se tenha um parâmetro de comparação.

Naturalmente, se a seqüência for invertida, as ocorrências do melhor dobramento no resultado são automaticamente ajustadas para refletir os dobramentos da seqüência original.

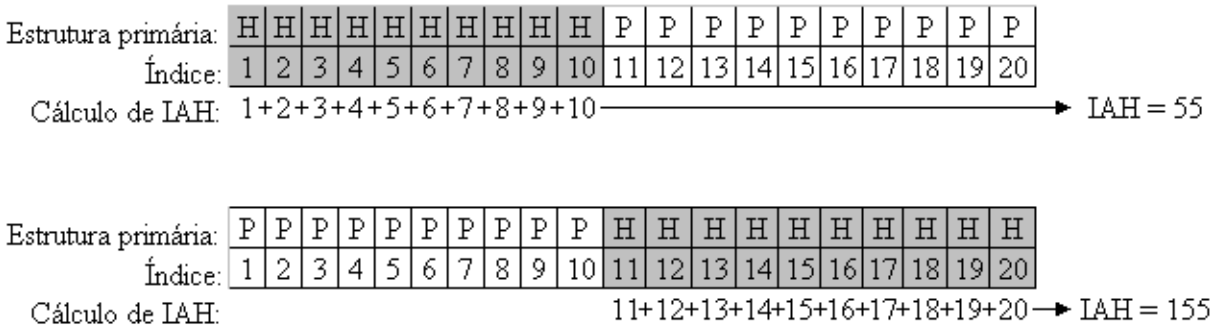

Figura 48 - Exemplo de calculo do Índice de Acúmulo de Hs.

3.6. PARALELIZACAO EM *CLUSTER*

Neste trabalho foi optou-se por elaborar uma solução paralelizada para o algoritmo. Esta solução consiste basicamente do algoritmo A* modificado mostrado na seção 3.5.1 aplicado na estrutura física de ***cluster***, como mostrado na seção 2.6.3.2, com o objetivo de se conseguir o paralelismo real, descrito na seção 2.6.2.3. O meio utilizado para essa utilização foi a linguagem de programação ANSI C, sobre o sistema operacional Linux, distribuição Fedora 4.5, utilizando a biblioteca de passagem de mensagens MPICH2.

3.6.1. Divisão de tarefas

O algoritmo paralelizado foi desenvolvido seguindo-se basicamente a metodologia proposta por ROOSTA (1999). Nesta abordagem primeiramente se elabora um algoritmo seqüencial e depois se analisa de que forma este pode ser paralelizado. Desta forma, o algoritmo deste trabalho foi elaborado primeiramente seqüencial, segundo a descrição na seção 3.5.

A biblioteca MPICH2 foi utilizada, que é uma implementação do padrão MPI, por implementar um mecanismo de troca de mensagens de fácil uso. Portanto, a abordagem paralelizada depende basicamente de troca de mensagens entre os computadores componentes do processamento. Como cada processador pode conter mais de um processador, como mostrado na seção 2.6.3.1, cada processador do sistema será denominado de agente. Cada um destes agentes será integrado de forma virtual por uma plataforma, que de fato realiza a troca das mensagens, desvinculando o desenvolvedor de se preocupar como a comunicação é realizada. Além do alto desempenho do MPI, uma grande vantagem é a possibilidade de se fazer ***clusters*** de uma forma extremamente flexível pois, tendo um meio comum de comunicação, como uma rede dedicada, o MPI (***Message Passing Interface***) se encarrega do tráfego das mensagens. Cada agente deve estar executando um programa denominado ***daemon***, que faz com que cada agente saiba com quem está conectado, permitindo o envio e o recebimento de mensagens. A este conjunto de agentes conectados dá-se o nome de comunicador (ou ***communicator***, termo original (SNIR, OTTO, HUSS-LEDERMAN,1999)).

O mecanismo do comunicador, além de fazer com que diversos agentes se comuniquem, implementa execução paralelizada um programa próprio para esse tipo de estrutura. Quando se executa um programa utilizando o MPI, ele envia uma cópia do programa a cada um dos agentes, que são identificados com um número seqüencial,

denominados de *ranks*. Cada agente é individualizado dentro de um comunicador por um *rank* diferente. Portanto, a princípio, todos os agentes executam o mesmo algoritmo. Como não há um mecanismo de gerência do comunicador, o programa paralelizado tem que implementar qual agente vai executar qual parte da tarefa, bem como quem se comunica com quem.

Outro aspecto importante da biblioteca MPICH é que não existe nenhum mecanismo que faça a divisão do problema automaticamente, o desenvolvedor é responsável por decidir quantos processos serão executados em cada agente. Isto deixa flexível o aspecto da arquitetura do algoritmo, podendo ser implementada uma estrutura *pipeline*, um pseudo-paralelismo ou ainda o paralelismo real.

Naturalmente, como neste trabalho se objetiva o processamento real, foi feita a divisão de processos igual ao número de agentes mais um. Este processo adicional é para adequar o comunicador à estrutura escolhida de gerenciamento de tarefas, sendo ela a mestre-escravo. Nesta estrutura, todos os agentes não fazem nenhuma ação sem que o processo mestre dê o comando. O processo mestre é o responsável por todo o gerenciamento da divisão correta de tarefas aos demais agentes, a coleta das respostas e a montagem do relatório. Porém, este processo mestre tem um tempo de execução bastante pequeno, pois ele manda as ordens aos agentes e espera pelas respostas.

Como pôde ser verificado experimentalmente, este processo mestre não acarreta uma carga de processamento significativa ao agente que o executa. Portanto, um dos agentes, além de executar o algoritmo de análise de dobramentos, executa também o processo mestre. Então a arquitetura de interconexão escolhida para ser implementada no *cluster* será a estrela, que cabe exatamente no modelo mestre-escravo. Basicamente, a arquitetura estrela exige que cada agente se comunique somente com um agente central, neste caso o agente que conterá o processo mestre. Como a arquitetura do *cluster* fornece um meio para que todos os agentes possam se comunicar com todos, a forma de interconexão pode ser definida simplesmente pelo fluxo das mensagens entre cada agente. A Figura 49 mostra um exemplo da interconexão em estrela e da distribuição dos agentes nos computadores.

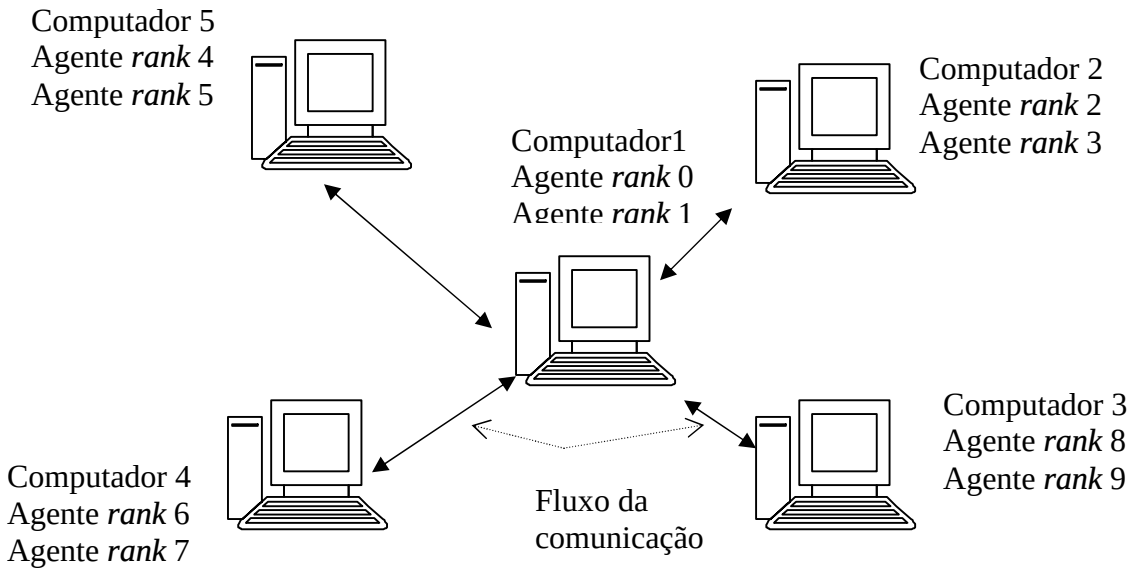

Figura 49 - Modelo de interconexão e distribuição de agentes nos computadores.

É importante lembrar que foram utilizados dois *clusters*: um para a fase de desenvolvimento e outro para os testes. No *cluster* utilizado nos testes, cada computador possui dois processadores (tecnologia *dual core*), portanto é possível colocar mais de um agente por processador sem degradar o processamento. Esta tecnologia não implementa dois processadores fisicamente separados, como descrito na seção 2.6.3.1, mas sim dois núcleos de processador dentro do mesmo *chip*, compartilhando recursos, como memória RAM (*Random Access Memory)*. No *cluster* de desenvolvimento os processadores são da tecnologia *hyperthreading* na qual existe somente um núcleo de processamento, sendo que o segundo é emulado, não existe fisicamente como na tecnologia *dual-core*. Portanto, não se deve esperar o mesmo ganho de desempenho entre essas tecnologias. Assim, nos dois *clusters* foi considerado que cada computador tinha dois processadores, porém apenas o *cluster* de testes de fato possui dois processadores por computador.

A forma de paralelização foi a de separação da árvore em subárvores e estas entre os agentes. Como a estrutura da árvore permite que cada nó gere uma subárvore totalmente independente e inédita em relação ao resto da árvore, basta distribuir aos agentes o número de processos correspondentes. A Figura 50 mostra a maneira como é feita esta divisão entre os agentes, com base em uma árvore genérica. O agente 0 é responsável por fazer o desenvolvimento inicial da árvore. A árvore é desenvolvida até que haja um número de nós na sua lista suficiente para suprir todos os agentes. Tal número é denominado o número de subdobramentos inicial. O agente 0 faz este desenvolvimento explorando a árvore em extensão, ou seja, procura fazer com que haja sempre nós da mesma profundidade. Quando

um número determinado de nós é atingido o agente 0 passa a fazer a distribuição deles aos demais agentes.

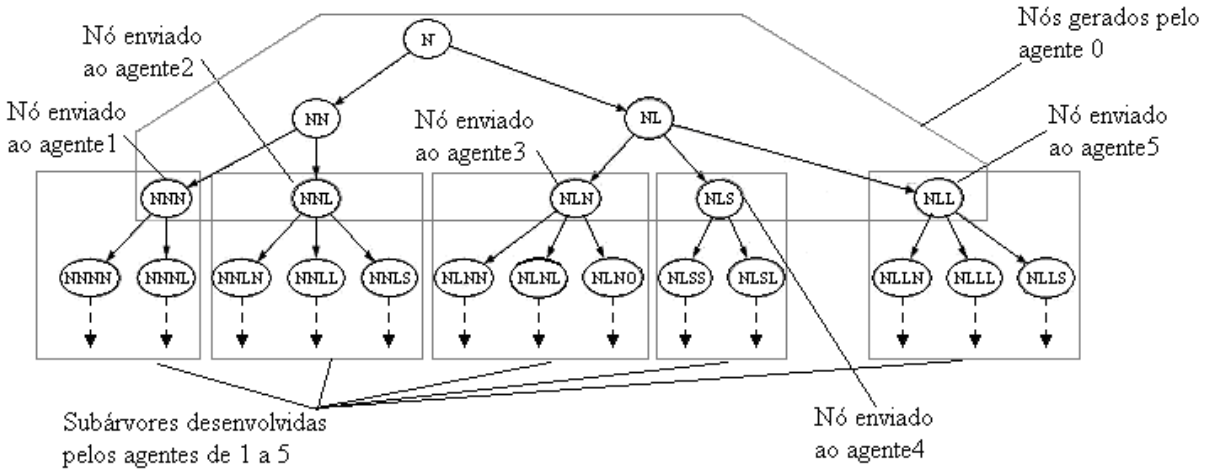

Figura 50 - Exemplo da divisão da análise entre cinco agentes.

No exemplo da Figura 50 o agente 0 deveria gerar tarefas para cinco agentes. Portanto ele iniciou com o subdobramento inicial N, gerou seus descendentes e assim por diante, até que a contagem de subdobramentos ultrapassasse o número de agentes, de modo a nenhum agente ficar ocioso. A árvore da Figura 50 supõe um número de subdobramentos iniciais igual ao número de agentes disponíveis. Nem sempre a divisão exata é possível. Por exemplo, se na Figura 50 houvesse apenas quatro agentes disponíveis, da mesma maneira seriam gerados cinco subdobramentos, pois o nó NL gera três subdobramentos. Então, o subdobramento NLL ficaria aguardando o seu processamento na lista dentro do agente 0. Assim que algum agente acabasse seu processamento o subdobramento NLL seria então distribuído.

3.6.2. Balanceamento de processamento

A abordagem adotada para divisão de tarefas, como citada na seção 3.6.1, explora bem a característica do paralelismo intrínseca à árvore. Porém, não corresponde ao esperado no quesito de balanceamento de processamento entre os agentes. Experimentalmente foi determinado que a taxa de ocupação (conforme seção 2.6.4) desta abordagem era um tanto baixa, na casa dos 60% a 70%, nos melhores casos. Isso significa que, de todo o tempo do processamento, 30% a 40% do tempo os agentes ociosos. Analisando a estrutura do algoritmo, tal comportamento foi considerado normal em um primeiro momento, pois se

percebeu que é possível e muito comum os subdobramentos iniciais darem origem a subárvores com número diferentes de nós. Teoricamente, ao se expandir subdobramentos de mesma profundidade as subárvores teriam o mesmo tamanho. Porém, devido aos diversos cortes propostos neste trabalho, principalmente pelo PIH, as subárvores não mais tem a obrigatoriedade de ter o mesmo tamanho. Portanto, é possível que cada agente tenha um tempo de processamento diferente. O balanceamento de carga é imprescindível nesse caso, pois assim tenta-se minimizar o tempo ocioso de cada agente, acelerando o processamento.

Assim, a abordagem foi modificada para se implementar o balanceamento. Tentou-se, então, fazer com que cada um dos agentes soubesse qual agente está ocioso e assim solicitar-lhe a divisão do seu processamento. O ideal para se conseguir isto seria uma memória compartilhada, na qual todos os agentes pudessem informar se estão ou não ociosos e consultar quem precisa de ajuda. O problema, porém, é que, como não se implementa o conceito de memória compartilhada na arquitetura MPI, apenas de troca de mensagens, é necessário que haja um agente responsável por gerenciar esta tarefa.

O agente 0 passou a saber sempre qual agente está ocioso, de modo a poder ordenar a alguém que esteja ocupado para dividir o processamento. Porém, para se dar tal ordem seria necessário que o agente 0 soubesse exatamente qual agente possui o maior número de subdobramentos a serem processados, o que, sem dúvida, geraria um tráfego imenso de comunicação.

Em vez disso, cada agente avisa o agente 0 quando estiver ocioso. Ao contrário de o agente 0 controlar qual agente deve ajudar qual, periodicamente, em intervalos de tempo aleatórios, cada agente pergunta ao agente 0 se há alguém ocioso que possa fornecer ajuda. Dessa forma, o agente 0 não precisa saber exatamente quantos subdobramentos existem na lista de cada agente. Ele simplesmente espera algum agente pedir ajuda. Se ninguém houver informado que está ocioso, o agente 0 responde que não há ninguém para ajudar e que então o agente tem que continuar fazendo seu processamento sozinho. Se, por outro lado, o agente 0 souber que alguém está ocioso, ele responde ao agente que pediu ajuda para informar qual é o seu menor subdobramento. Foi escolhido o menor subdobramento porque ele está, obviamente, em uma profundidade menor na árvore, logo, ele tem o maior potencial de gerar subárvore grande, deixando o agente que estava ocioso ocupado e diminuindo o processamento total do agente que pediu ajuda.

Experimentalmente, este procedimento conseguiu elevar a taxa de ocupação para valores entre 90% a 95%. Porém, ainda havia alguma ociosidade nos agentes. As taxas de comunicação não se apresentavam tão altas para indicar se havia tráfego intenso entre o

agente 0 e os demais agentes para justificar 5% de tempo ocioso. No caso de processamentos muito longos, com dias de duração, essa ociosidade poderia chegar a algumas horas.

Foi detectado então que tal abordagem para o balanceamento, apesar de eficiente, continha uma falha, que era a causa desta latência nos agentes. Ocorriam situações em que todos os agentes pediam ajuda e logo depois algum agente avisava que estava ocioso. É importante lembrar que a periodicidade com a qual os agentes pedem ajuda é diferente para cada agente, sendo aleatória dentro de uma faixa de valores. No caso deste trabalho, um valor definido empiricamente, foi de cada agente pedir por ajuda em algum ponto entre 5000 e 10000 nós processados, sendo eles denominados limites inferior e superior de balanceamento, respectivamente. Eventualmente, poderia acontecer de um agente ficar um tempo próximo a 5000 ciclos ocioso, pois teria que esperar que algum outro agente completasse o número de ciclos necessário para requisitar ajuda novamente.

A solução seria então reduzir os valores absolutos dos limites de balanceamento ou, ainda, reduzir a diferença entre eles. Porém, abaixando-se tais valores aumenta muito o tráfego na rede, ou seja, a ociosidade dos agentes aumenta. Isso acontece pois, havendo muitos agentes, muitos tendem a comunicar-se ou mais freqüentemente (ao reduzir-se o valor absoluto dos limites), ou muito próximos uns dos outros (ao reduzir-se a diferença entre os limites), ou ambos, o que acaba por ser ainda pior. Naturalmente, não se pode tentar diminuir o tráfego na rede tentando fazer o contrário, aumentando o valor dos limites ou a diferença entre eles, pois aumenta chance de algum agente ficar ocioso esperando por alguém pedir ajuda.

A abordagem que solucionou esse problema foi a de manter os limites de balanceamento, a fim de garantir um tráfego moderado na rede, mas tentar diminuir a necessidade de os agentes ociosos esperarem por outro agente pedir ajuda. A solução foi fazer com que o agente 0 sempre tenha alguma tarefa a ser distribuída a quem ficar ocioso, chamada de multisubdobramentos inicial. Para tanto, o agente 0 não mais desenvolve a árvore até que seu número de nós atinja o número de agentes disponíveis. Ele explora a árvore até que ela tenha um número de subdobramentos bastante superior ao número de agentes, ou seja, multisubdobramentos inicial. Neste trabalho, foi estabelecido um número padrão de 1500 subdobramentos. Um exemplo é mostrado na Figura 51, em que são utilizados 4 agentes para processar a árvore. Naturalmente, a árvore não está desenvolvida até os seus 1500 nós, mas apenas até 13 nós para ilustrar o funcionamento. É importante notar que uma vez que o agente 0 atinge o número de nós pretendido a árvore não se desenvolve além. Portanto, o agente 0

passa a não mais ter uma árvore, mas sim uma lista de subdobramentos, representada na Figura 51 pelos nós dentro do retângulo.

Dessa forma, havendo 5 agentes como no exemplo da Figura 50, apenas 5 subdobramentos são distribuídos inicialmente, mas um número muito superior ainda aguarda a distribuição, como na Figura 51. À medida que os agentes terminam de processar esses subdobramentos, eles continuam informando ao agente 0 que estão ociosos. Imediatamente, o agente 0 verifica se há algum subdobramento da sua lista inicial esperando para ser processado e prontamente informa ao agente qual a nova tarefa que tem. Os demais agentes continuam pedindo periodicamente ajuda, de modo que, se o agente 0 não tiver mais subdobramentos a serem distribuídos, um eventual agente ocioso espera alguém pedir ajuda no esquema de balanceamento que já estava sendo feito.

Essa nova abordagem conseguiu reduzir bastante o tempo de ociosidade, sem aumentar a taxa de comunicação, elevando a taxa de ocupação para valores quase sempre acima dos 98%.

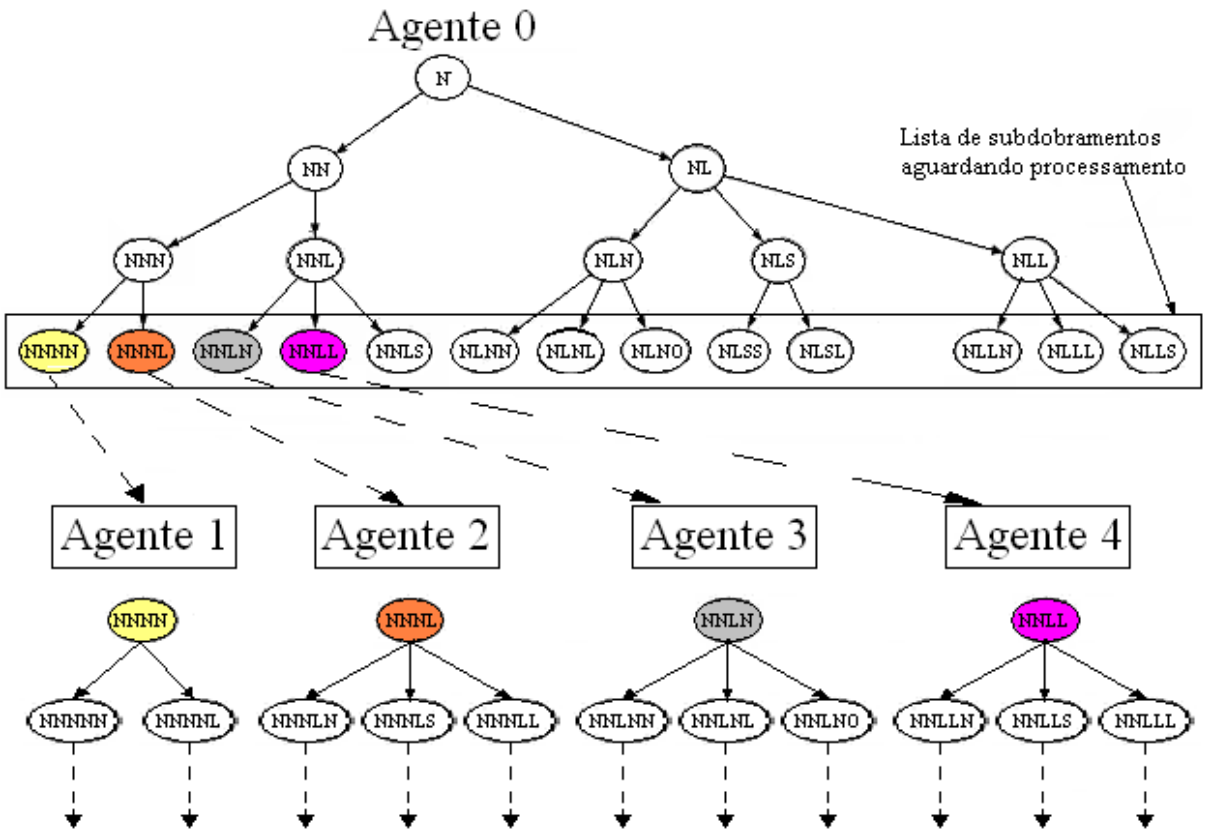

Figura 51 - Balanceamento por meio da multisubdobramentos inicial.

3.6.3. Intercomunicação de resultados

Como foi descrito na seção 3.5.1, a curva de PIH é decrescente, mas sempre quantifica um potencial de dobramento bastante superior ao número de iterações hidrofóbicas exibidas pelos subdobramentos. A eliminação dos subdobramentos não-promissores somente ocorre quando a curva do PIH cruza com a curva dos melhores resultados. Como mostrado na seção 3.5.1 e 3.5.2, tenta-se fazer com que a curva do PIH decresça o mais rápido possível, antecipando tal cruzamento, a fim de que se aumente número de subdobramentos não-promissores eliminados. A rapidez com que a curva PIH cai é fruto exclusivamente da eficiência do corte na árvore.

A outra maneira de conseguir-se melhorar a eliminação de subdobramentos não-promissores é tentando fazer com que a curva de melhores resultados cresça mais rapidamente. O ideal seria que essa curva não crescesse, mas, sim, já começasse no seu valor final, sendo uma reta constante até o fim da análise. Isso provocaria uma eliminação potencialmente maior do espaço de busca. Infelizmente essa redução não é muito grande, pois a taxa de decrescimento da curva PIH é muito maior do que a taxa de crescimento da curva dos melhores resultados, portanto, o ponto de cruzamento não se desloca tanto. De qualquer forma, porém, existe um deslocamento possível e, se ele puder ser alcançado, haverá alguma redução no espaço de busca.

Como não se sabe de antemão qual é o maior número de iterações hidrofóbicas para um dado polipeptídeo (melhor resultado), não se pode tornar a curva dos melhores dobramentos uma reta constante. Entretanto, como cada agente da análise processa uma subárvore diferente, é bastante provável que um agente chegue ao resultado antes dos demais. Se esse resultado puder ser compartilhado com os demais agentes, haverá um aumento na chance global de diminuição do espaço de busca.

Como explicado na seção 3.6.2, cada agente comunica-se periodicamente com o agente 0 ou para pedir ajuda ou para informar que acabou seu processamento. A solução adotada foi que, como o formato das mensagens trocadas inclui um código e um corpo da mensagem e o corpo, de fato, não estava sendo utilizado, agora, quando o agente comunica-se com o agente 0, informa o seu melhor resultado. Na resposta, o agente 0 informa o melhor resultado que ele conhece. Desta forma, os dois agentes envolvidos na comunicação atualizam os seus melhores resultados, potencialmente aumentando o desempenho dos agentes. Esta é uma forma de os agentes compartilharem informações, ou seja, um agente acaba por antecipar um resultado que provavelmente o outro agente também encontraria, mais cedo ou mais tarde.

Por se antecipar essa descoberta, é possível que haja uma eliminação mais acentuada do espaço de busca.

3.6.4. Interação mestre-escravo

Esta seção tem por objetivo mostrar como é feita a comunicação entre o agente 0 (mestre) e os demais agentes (escravos). Serão mostrados dois diagramas: na Figura 52, o fluxograma do programa executado no agente 0 (mestre) e, na Figura 53, o fluxograma executado nos demais agentes (escravos).

O processamento no agente 0, na Figura 52, inicia-se validando o polipeptídeo, para se assegurar que não há erros na estrutura primária, como algum aminoácido fora do padrão hidrofóbico-polar. Em seguida, ele faz a divisão da árvore no número de subdobramentos definido, neste caso 1500 nós, e insere-os na lista que o agente 0 mantém dos subdobramentos que ainda não foram distribuídos. Enquanto a lista não estiver vazia, o agente 0 procura, no seu registro, se há algum agente ocioso. Se houver, retira um subdobramento da sua lista e envia-o para o agente. Se não houver agente livre, o agente 0 fica aguardando algum outro agente enviar uma mensagem ou informar que já finalizou seu processamento. Se algum agente avisar que acabou o seu processamento, o agente 0 envia outra tarefa, caso haja alguma em seu poder. Se algum agente requisitar ajuda, o agente 0 procura algum agente ocioso. Se não houver nenhum, ele avisa ao agente que requisitou ajuda que nenhum agente poderá ajudá-lo. Caso contrário, requisita o subdobramento com maior chance de resultar na maior árvore e envia-o ao agente ocioso. Esse processo repete-se até que não haja nenhum subdobramento a ser processado e todos os agentes estejam ociosos. Se a lista estiver vazia e nenhum agente estiver ocupado, faz a requisição dos resultados, monta o relatório, finaliza os agentes e finaliza o processamento como um todo.

Os demais agentes apresentam um funcionamento um pouco diferente. Eles basicamente esperam pelo agente 0 ordenar o que fazer. Essas ordens podem ser de requisitar os resultados obtidos, finalizar o processamento ou, ainda, pode ser uma tarefa (um subdobramento) a ser processado. Caso seja um subdobramento, o agente procede com a sua análise, até chegar ao fim dela, ou chegar ao momento de perguntar ao agente 0 se há algum outro agente disponível para ajudar (balanceamento). Se houver algum agente disponível, o agente que requisitou ajuda envia ao agente 0 seu menor subdobramento. Foi escolhido o menor subdobramento pois é aquele que tem maior chance de gerar uma subárvore maior, ou seja, deixar o agente ocioso ocupado por mais tempo. Se não houver quem o ajude, ele

simplesmente volta a processar sua tarefa, até que acabe ou chegue novamente o momento de fazer o balanceamento.

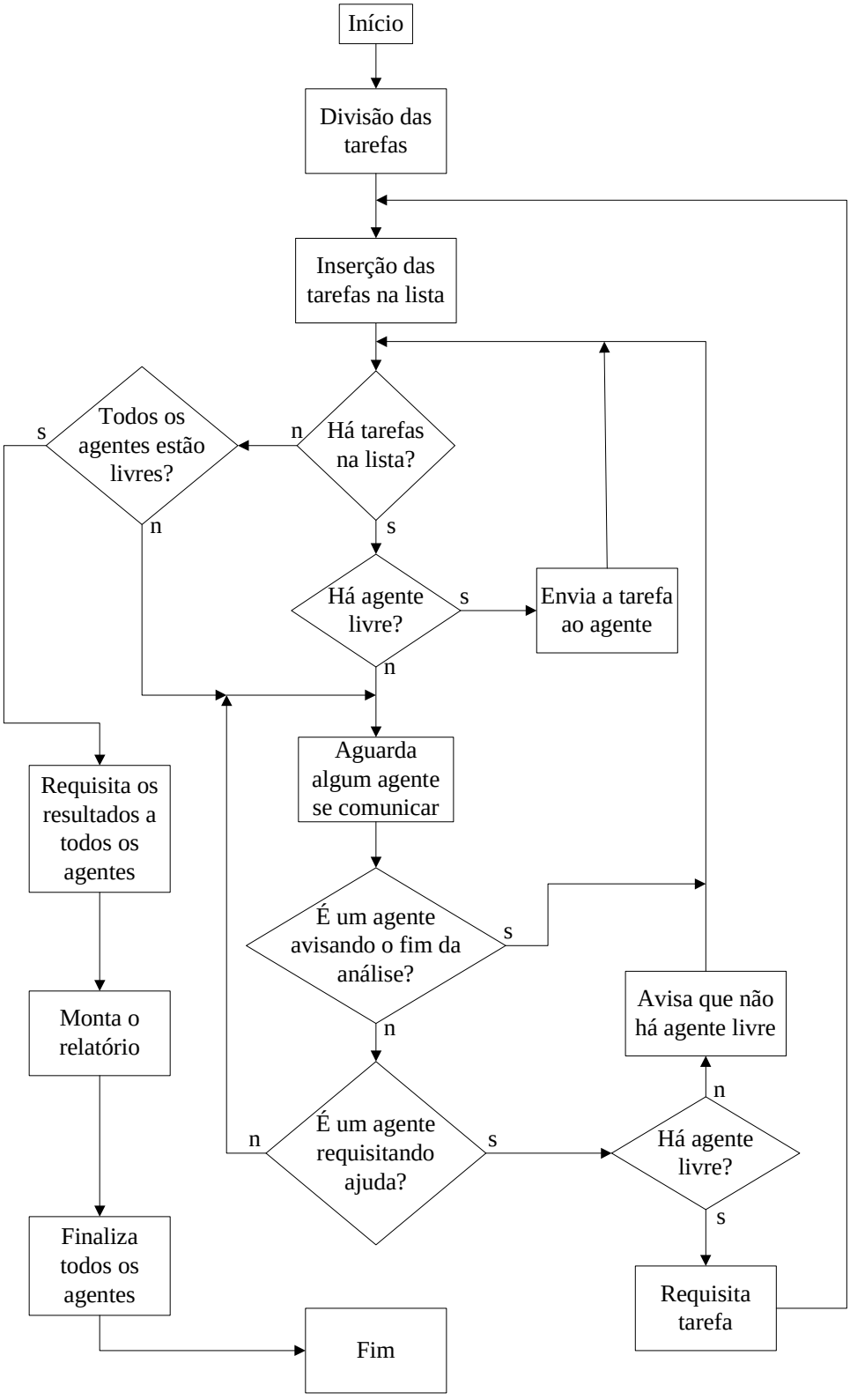

Figura 52 - Fluxograma do programa do agente 0 (mestre).

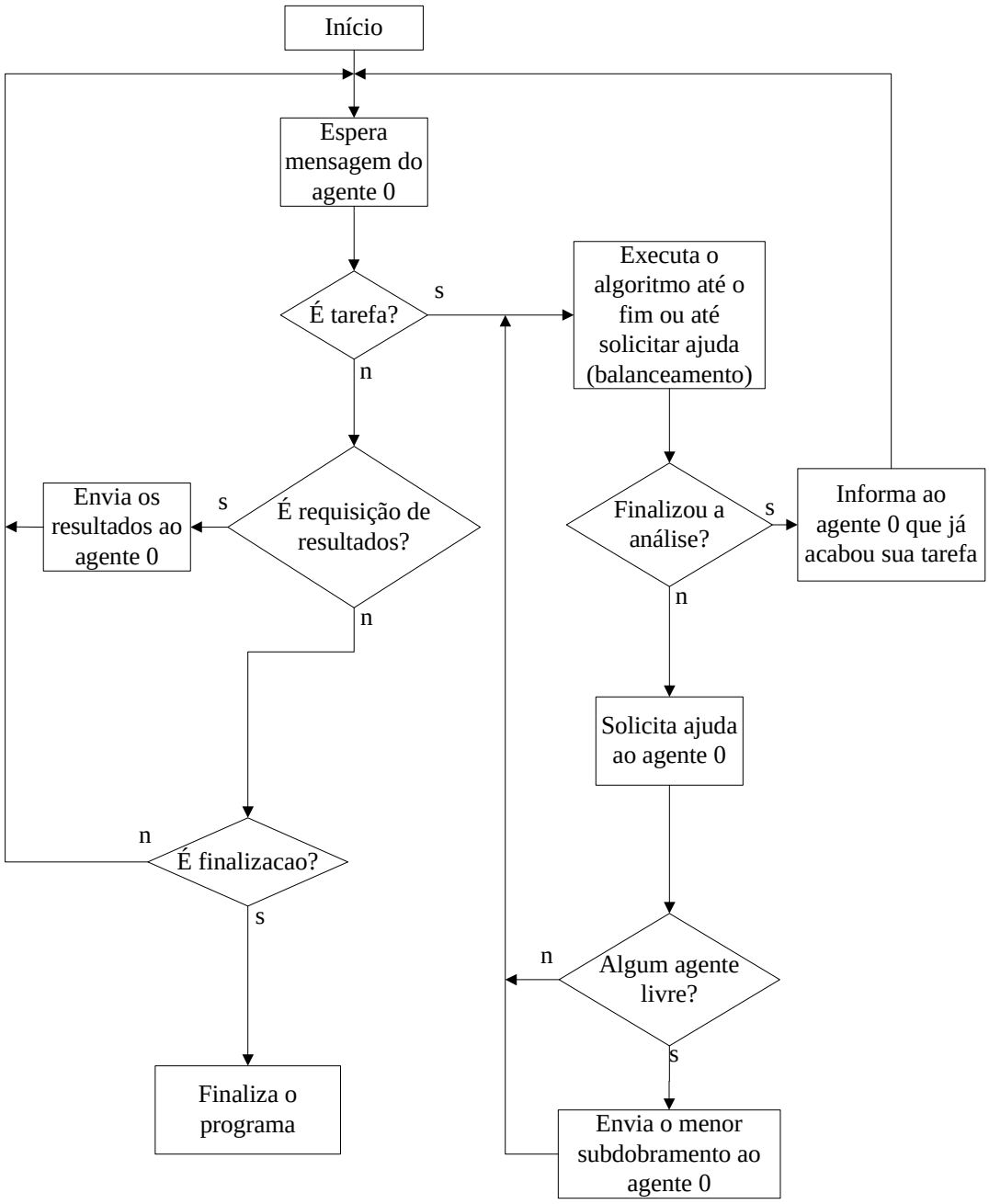

Figura 53 - Fluxograma dos agentes escravos.

CAPÍTULO 4

4. RESULTADOS

Neste capítulo serão apresentados os resultados obtidos a partir da execução do *software* elaborado neste trabalho. Os testes executados foram divididos em três partes. A primeira parte, chamada de testes de acurácia, destina-se a verificar se o algoritmo é capaz de reproduzir os resultados já encontrados por outras abordagens de análise do problema do dobramento de proteínas. O objetivo é definir se o *software* é ou não confiável para ser utilizado como um sistema de busca exaustiva da conformação nativa de polipeptídeos.

A segunda parte, chamada de testes de desempenho, tem por objetivo fazer medições relativas ao desempenho do algoritmo. São abordados aspectos como o tempo de busca, , taxa de ocupação dos agentes, escalabilidade (decréscimo do tempo de análise ao se adicionar mais agentes), coeficiente angular da curva de tempo de análise, tamanho do espaço de busca e coeficiente de aceleração.

A terceira parte é composta pelos experimentos em *cluster* heterogêneo, ou seja, testes que objetivam analisar qual é o comportamento do algoritmo em uma situação em que os computadores não são todos exatamente iguais, como quando acontece nos demais testes. Esses testes servem principalmente como preparação para uma futura implementação baseada em internet, ou seja, migrar a estrutura atual de *cluster* para uma grade de computadores, como descrito na seção 2.6.3.3.

Os testes de acurácia foram realizados em um *cluster* de 8 computadores com processador Intel *Pentium* 4 (tecnologia *HyperThreading*), com freqüência de *clock* de 3GHz, com 1GB de memória RAM. Neste *cluster* sempre foram utilizados 17 agentes, dois por computador, sendo um deles sempre o agente-mestre (chamado de agente 0, ao longo deste trabalho), que faz o gerenciamento dos demais agentes. De acordo com a arquitetura desses processadores, o arquivo executável utilizado é de 32 bits.

Os testes de desempenho foram realizados em um *cluster* de 26 computadores, cada um com um processador Intel Core 2-Duo modelo E6750, com freqüência de 2,667 GHz, com 1GB de memória RAM. Neste *cluster* sempre foram utilizados 53 agentes, dois por computador, mantendo o esquema mestre-escravo, em que um dos agentes é sempre o mestre (agente 0). De acordo com a arquitetura desses processadores, o arquivo executável utilizado é de 64 bits.

Nos testes de *cluster* heterogêneo foram utilizadas dois c*lusters* distintos. O primeiro é composto de 61 agentes homogêneos, ou seja, exatamente iguais. A configuração dos computadores é a mesma dos testes de desempenho, com a exceção de que eles são em número diferente. Nestes testes foram utilizados 30 computadores, ao invés dos 26 utilizados no teste de desempenho, mantendo-se o esquema de mestre-escravo, no qual o agente 0 é o mestre, com outros 60 agentes para o processamento. Para o segundo *cluster* outros computadores foram adicionados, para torná-lo homogêneo. Foram agregados sete computadores idênticos, com processador AMD Athlon XP 2400+, com 512 MB de memória RAM. Sendo estes sete adicionados aos outros 30 já existentes o *cluster*, agora heterogêneo por ser composto de computadores diferentes, totaliza 37 computadores. Neste novo *cluster* foram alocados 68 agentes: dois agentes em cada processador *dual-core* (60 agentes), um em cada processador *single-core* (7 agentes) e mais um agente mestre em um dos *dual-core*.

Os dois *clusters* foram criados para este teste porque não foi possível aproveitar os resultados obtidos nos testes de desempenho, pois eles foram obtidos com um executável de 64 bits. A estrutura do MPICH2 somente permite que o mesmo programa seja executado em todos computadores. A estrutura atual é composta de computadores de 64 bits (os Intel Core 2-Duo) e 32 bits (os AMD Athlon XP 2400+). Portanto, não foi possível realizar, com o *cluster* heterogêneo, testes com o executável de 64 bits, pois ele é impróprio para os computadores de 32 bits. Para poder realizar os testes foi utilizado o programa compilado em 32 bits em todos os computadores do *cluster* heterogêneo. Com isto assume-se uma inevitável perda de desempenho nos processadores de 64 bits, pois, executando-se um aplicativo de 32 bits em uma arquitetura de 64 bits certamente não se explora todo o potencial desses processadores, como será mostrado nos testes.

Os testes, portanto, tiveram que ser reelaborados, pois não é possível se comparar desempenho de programas pertencentes a arquiteturas diferentes. Então nenhum dos resultados obtidos nos testes com o *cluster* heterogêneo podem ser comparados diretamente com o *cluster* homogêneo do testes anterior. Para este teste foram elaborados testes de desempenho com um agente, com 60 agentes homogêneos e com 67 agentes heterogêneos, todos na arquitetura de 32 bits, para se possibilitar a comparação.

Dois conjuntos de polipeptídeos foram utilizados para a análise, ambos tiveram seus polipeptídeos renomeados para facilitar a visualização. O primeiro, mostrado na Tabela 6, compreende diversos polipeptídeos (ISTRAIL e WALENZ, 1995)utilizados em outros trabalhos. O segundo, mostrado na Tabela 7, contém polipeptídeos fictícios, puramente hidrofóbicos. Apesar de estes terem uma chance mínima de existirem na natureza, eles

representam o pior caso na análise, ou seja, a mais trabalhosa, e por esta razão foram utilizados.

Tabela 6 - Grupo1: Lista de polipeptídeos mistos utilizados na análise.

Seqüência	**Polipeptídeo**	**Nº de aminoácidos**
M18A	HHPPPPPHHPPPHPPPHP	18
M18B	HPHPHHHPPPHHHHPPHH	18
M18C	PHPPHPHHHPHHPHHHHH	18
M20A	HHHPPHPHPHPPHPHPHPPH	20
M20B	HPHPPHHPHPPHPHHPPHPH	20
M24	HHPPHPPHPPHPPHPPHPPHPPHH	24
M25	PPHPPHHPPPPHHPPPPHHPPPPHH	25
M36	PPPHHPPHHPPPPPHHHHHHHHPPHHPPPPHHPPHPP	36
M48	PPHPPHHPPHHPPPPPHHHHHHHHHHPPPPPPHHPPHHPP HPPHHHHH	48

Tabela 7 - Grupo2 : Lista de polipeptídeos puramente hidrofóbicos utilizados.

Seqüência	**Polipeptídeo**	**Nº de aminoácidos**
H15	HHHHHHHHHHHHHHH	15
H16	HHHHHHHHHHHHHHHH	16
H17	HHHHHHHHHHHHHHHHH	17
H18	HHHHHHHHHHHHHHHHHH	18
H19	HHHHHHHHHHHHHHHHHHH	19
H20	HHHHHHHHHHHHHHHHHHHH	20
H21	HHHHHHHHHHHHHHHHHHHHH	21
H22	HHHHHHHHHHHHHHHHHHHHHH	22
H23	HHHHHHHHHHHHHHHHHHHHHHH	23
H24	HHHHHHHHHHHHHHHHHHHHHHHH	24
H25	HHHHHHHHHHHHHHHHHHHHHHHHH	25
H26	HHHHHHHHHHHHHHHHHHHHHHHHHH	26
H27	HHHHHHHHHHHHHHHHHHHHHHHHHHH	27
H28	HHHHHHHHHHHHHHHHHHHHHHHHHHHH	28
H29	HHHHHHHHHHHHHHHHHHHHHHHHHHHHH	29
H30	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	30
H31	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	31
H32	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	32
H33	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	33
H34	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	34
H35	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	35
H36	HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH	36

4.1. TESTES DE ACURÁCIA

Primeiramente, foram feitos os testes de acurácia, que têm o objetivo de verificar se a abordagem elaborada não acusa resultados errados. A forma de se fazer essa verificação de acurácia do algoritmo é a da comparação com os resultados obtidos por outras abordagens, não necessariamente de busca exaustiva.

O primeiro teste, mostrado na Tabela 8, foi executado com base nos polipeptídeos do grupo 1 e comparado a outras abordagens. A primeiro (coluna A) é um algoritmo de *Ant-Colony*, publicado por HU, ZHANG, XIAO (2008). A segundo (coluna B) é um algoritmo elaborado por TOMA e TOMA (1996), que consiste em uma abordagem Monte-Carlo modificada. O terceiro e o quarto (colunas C e D, respectivamente) foram elaborados por UNGER e MOULT (1993b). A abordagem C é um algoritmo Monte-Carlo e a D é um algoritmo genético. A quinta abordagem (coluna E), proposta por KÖNIG e DANDEKAR (1999) e a sexta (coluna F), apresentada por SCAPIN e LOPES (2005), são também algoritmos genéticos, aplicados ao problema do dobramento de proteínas. O resultado utilizado para a comparação com este trabalho é o maior número de iterações hidrofóbicas que cada abordagem conseguiu encontrar para cada polipeptídeo do grupo 1. A coluna "Número de dobramentos" apresenta o número de ocorrências do melhor dobramento encontradas neste trabalho. Ela está mostrada apenas para referência pois não foi encontrado na literatura nenhum outro trabalho que fizesse essa medição.

Tabela 8 – Número máximo de iterações hidrofóbicas obtidos para o grupo 1 por diversos trabalhos.

Polipeptídeo	**A**	**B**	**C**	**D**	**E**	**F**	**Este trabalho**	**Nº de dobramentos**
M18A	4	--	--	--	--	--	4	1
M18B	8	--	--	--	--	--	8	1
M18C	9	--	--	--	--	--	9	1
M20A	10	--	--	--	--	--	10	1
M20B	9	9	9	9	9	9	9	2
M24	--	9	9	9	--	9	9	19
M25	--	8	8	8	--	8	8	16
M36	--	14	13	14	14	14	14	192
M48	--	22	20	23	23	23	23	492

Em seguida foram feitos experimentos com os polipeptídeos do grupo 2, como está mostrado na Tabela 9. A comparação foi realizada com o trabalho realizado em ARMSTRONG JUNIOR, LOPES e LIMA (2006), mostrado na coluna A. Tal trabalho também propunha um algoritmo de busca exaustiva, porém sem incorporar todos os cortes no espaço de busca feitos neste trabalho.

Tabela 9 - Número de iterações hidrofóbicas para polipeptídeos puramente hidrofóbicos.

Polipeptídeo	**Tamanho**	**A**	**Este trabalho**
H15	15	8	8
H16	16	9	9
H17	17	9	9
H18	18	10	10
H19	19	11	11
H20	20	12	12
H21	21	12	12
H22	22	13	13
H23	23	14	14
H24	24	15	15
H25	25	16	16
H26	26	16	16
H27	27	17	17
H28	28	--	18
H29	29	--	19
H30	30	--	20
H31	31	--	20
H32	32	--	21
H33	33	--	22
H34	34	--	23
H35	35	--	24
H36	36	--	25

Os resultados das análises realizadas no grupo 2 levaram ao levantamento de uma curva de comportamento do número de iterações hidrofóbicas em relação ao número de aminoácidos de um polipeptídeo totalmente hidrofóbico. Essa curva é mostrada na Figura 54.

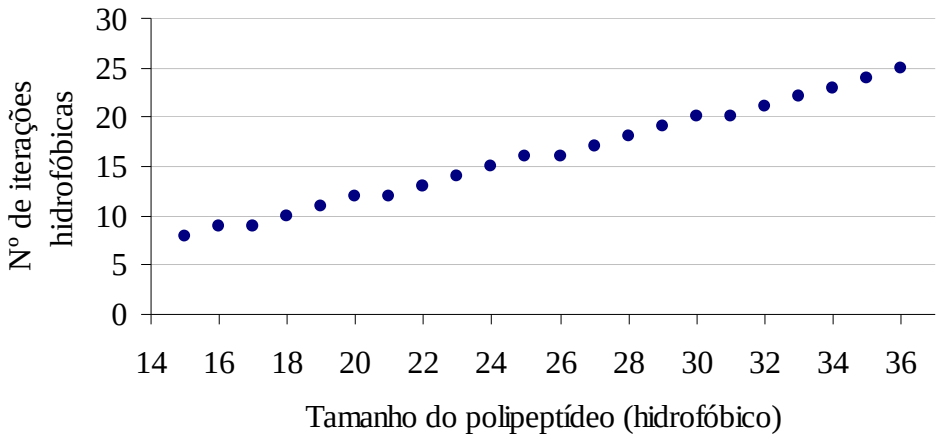

Figura 54 - Comportamento do Número de iterações hidrofóbicas para diversos polipeptídeos puramente hidrofóbicos.

Com base nessa curva foi elaborada a equação 18, como objetivo de fazer a predição do número máximo de iterações hidrofóbicas, dado o tamanho de um polipeptídeo totalmente hidrofóbico. O método utilizado foi de regressão polinomial de 2ª ordem, com um coeficiente de determinação (R^2) de 0,9962, que significa que a equação 18 tem 99,71% de fidelidade aos dados da Figura 54. Nesta equação, *nIH* é o número de iterações hidrofóbicas e *x* representa o número de aminoácidos em um polipeptídeo totalmente hidrofóbico.

$$nIH = 0{,}8007x - 4{,}3264 \qquad (18)$$

Nesta equação, *nIH* é o número de iterações hidrofóbicas e *x* representa o número de aminoácidos em um polipeptídeo totalmente hidrofóbico. A equação 18 foi elaborada com os dados de todas as seqüências do grupo 2.

Um exemplo da análise em 17 agentes do polipeptídeo PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP, de 36 aminoácidos, pode ser visto no Anexo 2 e do seu resultado no Anexo 3. No fim deste anexo estão mostradas somente algumas ocorrências do melhor dobramento, a título de exemplo, e não todas as 7078 ocorrências.

4.2. TESTES DE DESEMPENHO – *CLUSTER* HOMOGÊNEO.

O primeiro teste de desempenho, mostrado na Tabela 10, mostra os tempos de análise obtidos pela aplicação do algoritmo no grupo 1 de polipeptídeos. Estes testes foram realizados executando-se o algoritmo com 53 agentes e comparando-os com os resultados da execução do algoritmo seqüencial, elaborado para ser executado com apenas um agente utilizando os mesmos cortes da versão paralelizada. A Tabela 10 também mostra o levantamento da eficiência do algoritmo na análise do grupo 1, em relação à execução seqüencial e a taxa de ocupação média dos agentes, conforme exposto na seção 2.6.4.

Tabela 10 - Tempos de análise para os polipeptídeos do grupo 1.

	Tempo de análise (segundos)			
Polipeptídeo	**Seqüencial**	**53 agentes**	**Eficiência**	**Taxa de Ocupação média**
M18A	0,1	0,7	0,00	0,1%
M18B	0,8	1,3	0,01	0,4%
M18C	2,0	0,7	0,06	0,2%
M20A	1,2	0,7	0,03	0,5%
M20B	0,2	1,1	0,00	0,5%
M24	19,2	1,6	0,23	6,5%
M25	65,4	1,3	0,94	1,0%
M36	134,0	3,9	0,67	38,1%
M48	93932,67	1578,9	1,14	97,9%

O próximo teste é idêntico ao primeiro, porém agora para seqüências do grupo 2 e é mostrado na Tabela 11. Novamente é feita uma comparação com o algoritmo seqüencial (utilizando os mesmos cortes da versão paralelizada) e também são mostrados os tempos teóricos de análise para cada seqüência se não houvesse nenhum corte do espaço de busca. Naturalmente este tempo teórico é fictício e calculado como se 53 agentes analisassem o espaço de busca total. Este tempo é obtido dividindo-se o tempo da análise dos 53 agentes pelo número de estados promissores, mostrado na Tabela 13. Com isto se descobre o tempo gasto em cada estado analisado. Então se multiplica este tempo pelo número de estados do espaço total, também mostrado na Tabela 13, obtendo-se, assim, o tempo que o *cluster* de 53 agentes levaria para analisar o espaço de busca total, sem cortes.

Tabela 11 - Tempos de análise para os polipeptídeos do grupo 2.

	Tempo de análise (segundos)		
Proteína	**Total (teórico)**	**Seqüencial**	**53 agentes**
H15	1,36E+04	0,4	1,3
H16	2,54E+04	0,4	1,2
H17	4,11E+04	0,4	1,4
H18	6,85E+04	1,5	1,1
H19	1,87E+05	2,5	1,4
H20	4,06E+05	2,8	1,4
H21	6,71E+05	7,2	1,8
H22	1,81E+06	17,2	2,2
H23	4,90E+06	29,4	2,6
H24	1,42E+07	56,2	3,4
H25	4,29E+07	107,4	4,7
H26	9,48E+07	368,5	7,9
H27	3,74E+08	680,3	13,8
H28	1,54E+09	1271,3	25,4
H29	6,15E+09	2325,5	45,1
H30	2,51E+10	4299,5	81,5
H31	1,06E+11	13672,4	265,9
H32	2,48E+11	24909,4	488,1
H33	1,03E+12	47322,2	898,9
H34	4,29E+12	87288,2	1648,0
H35	1,78E+13	161321,3	3020,6
H36	7,36E+13	302344,5	5523,9

A Tabela 12 relaciona o fator de aceleração e a eficiência na análise dos polipeptídeos do grupo 2 pelo do *cluster* de 53 agentes. Tais valores são calculados com base nas medidas da Tabela 11. Como parâmetro comparativo também é apresentado a taxa de ocupação média dos agentes. O fator de aceleração e a eficiência da execução são calculados conforme descrito na seção 2.6.4.

Tabela 12 - Desempenho, no grupo 2, da execução com 52 agentes em relação à seqüencial.

Polipeptídeo	**Fator de aceleração**	**Eficiência da execução**	**Taxa de ocupação média**
H15	0,31	0,01	0,2%
H16	0,33	0,01	0,3%
H17	0,30	0,01	0,6%
H18	1,33	0,03	1,3%
H19	1,78	0,03	2,0%
H20	1,95	0,04	3,7%
H21	4,11	0,08	9,0%
H22	8,01	0,15	13,9%
H23	11,10	0,21	22,0%
H24	16,41	0,32	31,8%
H25	23,07	0,44	44,6%
H26	46,89	0,90	83,6%
H27	49,15	0,95	88,9%
H28	50,10	0,96	90,5%
H29	51,59	0,99	94,5%
H30	52,77	1,01	96,6%
H31	51,42	0,99	97,6%
H32	51,04	0,98	97,8%
H33	52,64	1,01	97,9%
H34	52,97	1,02	98,0%
H35	53,41	1,03	98,1%
H36	54,73	1,05	98,2%

No aspecto da taxa de ocupação é importante levantar como ela se comporta de acordo com o número de agentes utilizados no *cluster*. Não é interessante ter muitos agentes trabalhando com uma taxa de ocupação pequena, pois a aceleração não é ótima e, portanto, a relação custo-benefício passa a ser desvantajosa. A Figura 55 mostra a taxa de ocupação medida no *cluster*, utilizando dois conjuntos de agentes, sendo eles de 31 e 53 agentes, para todas as seqüências do grupo 2. O objetivo é determinar o comportamento da taxa de ocupação com a variação do tamanho de um polipeptídeo totalmente hidrofóbico, de modo a não se utilizar mais agentes do que o necessário, otimizando os recursos. Estes polipeptídeos representam o pior caso de análise, com o maior espaço de busca.

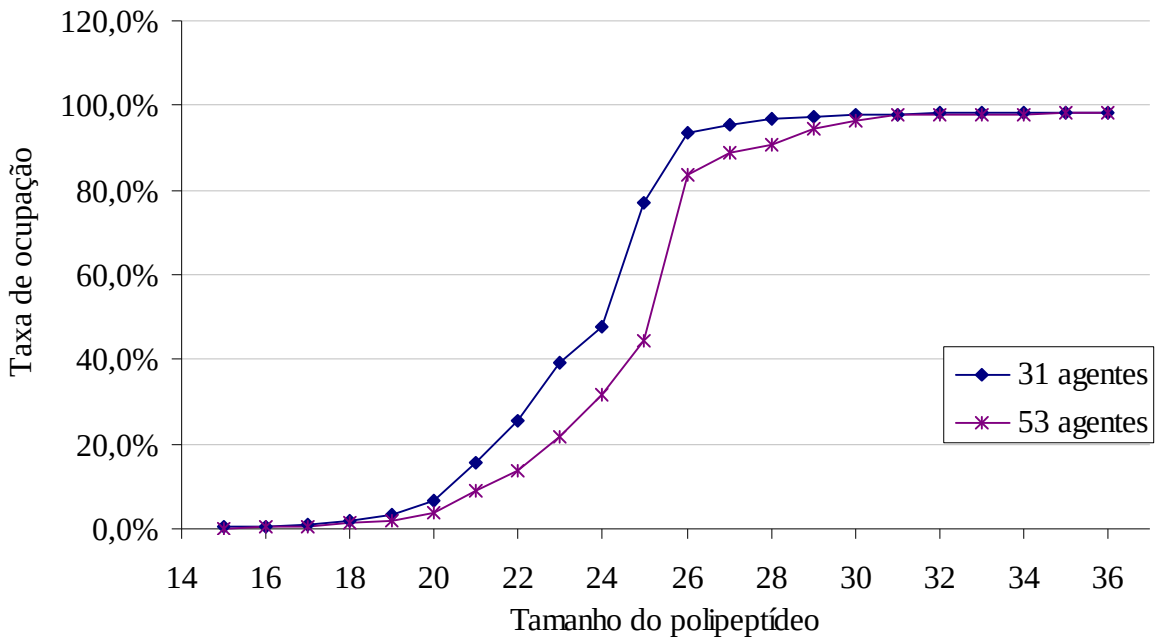

Figura 55 - Comportamento da taxa de ocupação de acordo com o tamanho do polipeptídeo e o número de agentes.

Na Figura 56 é mostrado o comportamento do coeficiente de aceleração de dois grupos de agentes, um com 31 e outro com 53 agentes, para os polipeptídeos do grupo 2. No eixo “x” está mostrado apenas o tamanho de cada polipeptídeo do grupo 2. O objetivo é identificar se o tamanho dos polipeptídeos causou perda de aceleração, pelo aumento do tamanho do espaço de busca.

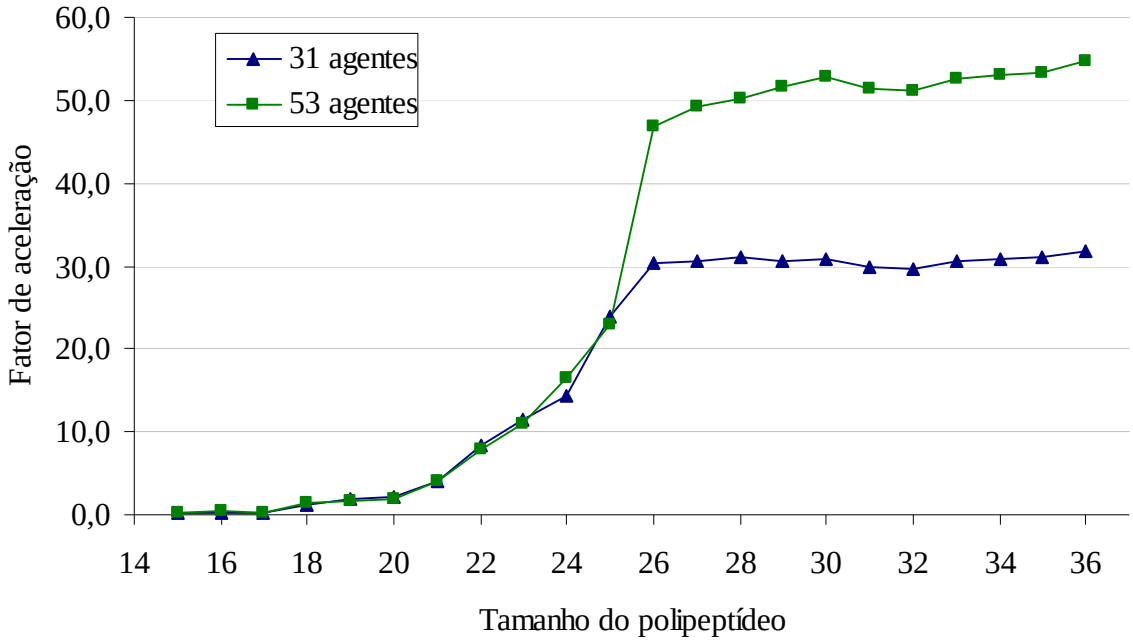

Figura 56 - Comportamento do coeficiente de aceleração com os diversos tamanhos de polipeptídeo, todos puramente hidrofóbicos.

O próximo teste é a medição do tamanho do espaço de busca. A medição é dividida em duas partes e foi realizada somente com os polipeptídeos do grupo 2, pois para eles é possível o cálculo do tamanho teórico do espaço de busca, de acordo com a equação 12, mostrada na seção 3.4. A primeira parte, mostrada na Tabela 13, mostra o total de estados possíveis calculado para cada polipeptídeo do grupo 2 (espaço total), juntamente com o número de estados classificados como válidos e os estados promissores, ambos medidos experimentalmente. A coluna “Redução PDF” indica o percentual representado pelo espaço promissor em relação ao espaço total. É importante lembrar que o espaço promissor é o que é, de fato, analisado pelo algoritmo.

Tabela 13 - Tamanho do espaço de estados obtido para o grupo 2.

Polipeptídeo	Número de estados possíveis			Redução PDF
	Espaço total	Espaço Válido	Espaço Promissor	
H15	3,36E+08	7,35E+04	3,16E+04	0,0094139576%
H16	1,34E+09	1,41E+05	6,13E+04	0,0045667589%
H17	5,37E+09	4,18E+05	1,83E+05	0,0034041144%
H18	2,15E+10	7,94E+05	3,49E+05	0,0016266387%
H19	8,59E+10	1,46E+06	6,52E+05	0,0007585087%
H20	3,44E+11	2,67E+06	1,20E+06	0,0003480984%
H21	1,37E+12	7,97E+06	3,59E+06	0,0002610071%
H22	5,50E+12	1,44E+07	6,52E+06	0,0001185708%
H23	2,20E+13	2,61E+07	1,19E+07	0,0000540124%
H24	8,80E+13	4,63E+07	2,12E+07	0,0000240894%
H25	3,52E+14	8,31E+07	3,82E+07	0,0000108437%
H26	1,41E+15	2,53E+08	1,17E+08	0,0000082871%
H27	5,63E+15	4,50E+08	2,08E+08	0,0000037019%
H28	2,25E+16	8,01E+08	3,72E+08	0,0000016505%
H29	9,01E+16	1,42E+09	6,61E+08	0,0000007335%
H30	3,60E+17	2,51E+09	1,17E+09	0,0000003250%
H31	1,44E+18	7,73E+09	3,62E+09	0,0000002512%
H32	5,76E+18	2,41E+10	1,14E+10	0,0000001970%
H33	2,31E+19	4,26E+10	2,01E+10	0,0000000870%
H34	9,22E+19	6,39E+09	3,55E+10	0,0000000384%
H35	3,69E+20	6,38E+10	6,27E+10	0,0000000170%
H36	1,48E+21	6,38E+10	1,11E+11	0,0000000075%

A Tabela 14 mostra a conseqüência direta da redução evidenciada na Tabela 13, e é a segunda parte da medição do espaço de busca. Esta tabela apresenta o total calculado de dobramentos possíveis para o número de estados do espaço total (mostrado na Tabela 13), juntamente com o número de dobramentos subótimos e ótimos, ambos medidos experimentalmente. A coluna “Redução PDF” mostra o número de dobramentos ótimos, em percentual, em relação número de dobramentos possíveis total, calculado conforme a equação 13 descrita na seção 3.4. Esta coluna representa também é uma medida de eficiência da função PDF, em conjunto com os resultados da Tabela 13. A coluna de dobramentos ótimos é, de fato, quantas ocorrências foram encontradas do dobramento com a maior contagem de iterações hidrofóbicas. O registro de cada ocorrência também foi feito e armazenado, para cada análise. A coluna de subótimos é a quantidade de dobramentos que foram classificados como promissores, mas que não resultaram em dobramentos ótimos.

Tabela 14 - Número de dobramentos obtidos para os polipeptídeos do grupo 2.

	Número de dobramentos			
Polipeptídeo	**Total**	**Subótimos**	**Ótimos**	**Redução PDF**
H15	2,68E+08	11480	142	0,0000528991222382%
H16	1,07E+09	17768	69	0,0000064261257648%
H17	4,29E+09	56493	1890	0,0000440049916506%
H18	1,72E+10	85640	1673	0,0000097381416708%
H19	6,87E+10	125388	544	0,0000007916241884%
H20	2,75E+11	169393	503	0,0000001829903340%
H21	1,10E+12	607396	11226	0,0000010209987522%
H22	4,40E+12	848238	11584	0,0000002633896656%
H23	1,76E+13	1,19E+06	4577	0,0000000260172328%
H24	7,04E+13	1,59E+06	3997	0,0000000056800786%
H25	2,81E+14	2,20E+06	1081	0,0000000003840483%
H26	1,13E+15	8,43E+06	100750	0,0000000089483976%
H27	4,50E+15	1,16E+07	52594	0,0000000011678214%
H28	1,80E+16	1,53E+07	45238	0,0000000002511213%
H29	7,21E+16	2,09E+07	16294	0,0000000000226125%
H30	2,88E+17	2,63E+07	13498	0,0000000000046831%
H31	1,15E+18	1,16E+08	636771	0,0000000000552311%
H32	4,61E+18	2,03E+08	656376	0,0000000000142329%
H33	1,84E+19	2,55E+08	306498	0,0000000000016615%
H34	7,38E+19	3,38E+08	265502	0,0000000000003598%
H35	2,95E+20	4,12E+08	105265	0,0000000000000357%
H36	1,18E+21	4,12E+08	57337	0,0000000000000049%

A Tabela 15 mostra a medição de eficiência do método de balanceamento utilizado, conforme o método elaborado descrito na seção 3.6.2. Novamente é abordada a taxa de ocupação nos agentes do *cluster* como medida de sucesso, pois o desejável é elevar esta taxa média o mais próximo possível de 100%.

Tabela 15 - Eficiência do balanceamento, para os polipeptídeos do grupo 2.

Proteína	**Taxa de ocupação média (53 agentes)**	
	Sem balanceamento	**Com balanceamento**
H15	0,2%	0,2%
H16	0,4%	0,3%
H17	0,5%	0,6%
H18	1,8%	1,3%
H19	2,1%	2,0%
H20	3,3%	3,7%
H21	8,0%	9,0%
H22	12,4%	13,9%
H23	21,2%	22,0%
H24	29,8%	31,8%
H25	41,6%	44,6%
H26	47,9%	83,6%
H27	51,2%	88,9%
H28	57,4%	90,5%
H29	58,1%	94,5%
H30	59,3%	96,6%
H31	60,4%	97,6%
H32	63,0%	97,8%
H33	64,1%	97,9%
H34	64,2%	98,0%
H35	64,9%	98,1%
H36	64,9%	98,2%

A Figura 57 mostra o comportamento do coeficiente angular da curva do aumento do número de nós para as análises, calculada com base nos dados da Tabela 13. O objetivo desta curva é medir o quanto cresce o tamanho do espaço de busca de acordo com o aumento do número de aminoácidos nos polipeptídeos. O aumento acelerado do espaço de busca total é a causa do problema do dobramento de proteínas ser classificado como NP-difícil. Naturalmente este teste foi feito apenas para os polipeptídeos do grupo 2, pois se desejava obter o pior caso. Os dados no eixo “y” está em escala logarítmica.

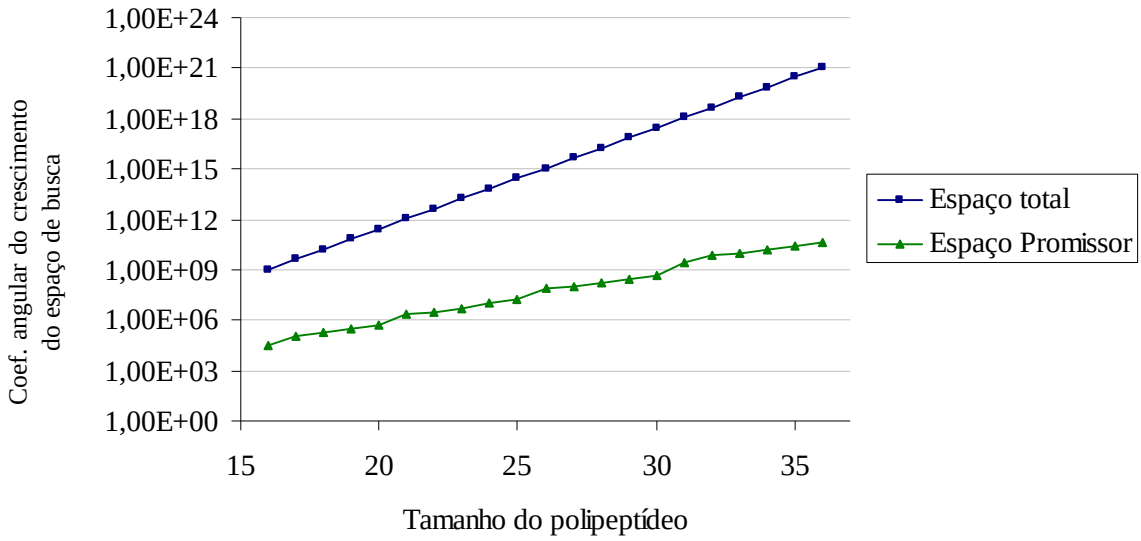

Figura 57 - Coeficiente angular obtido para a curva do espaço de busca.

4.3. TESTES DE DESEMPENHO – *CLUSTER* HETEROGÊNEO

O objetivo destes testes foi analisar o comportamento do sistema de busca quando executado em um *cluster* heterogêneo. O conjunto de polipeptídeo utilizados foi o grupo 2. Os únicos parâmetros medidos neste testes foram os tempos de análise, o fator de aceleração e a taxa de ocupação com um agente, com 60 agentes homogêneos e com 67 agentes heterogêneos. Na Tabela 16 são mostrados os tempos de execução do algoritmo, o fator de aceleração de cada implementação e a taxa de ocupação média dos *clusters* homogêneo e heterogêneo.

Tabela 16 - Teste de desempenho, para o grupo 2, na execução com uma agente, 61 agentes homogêneos e 68 agentes heterogêneos.

Proteína	Tempo de análise			Aceleração		Taxa de ocupação média	
	1 agente	61 agentes	68 agentes	61 agentes	68 agentes	61 agentes	68 agentes
H15	0,52	1,50	1,33	0,35	0,39	0,22%	0,24%
H16	0,53	1,28	1,53	0,42	0,35	0,38%	0,35%
H17	0,45	1,36	1,47	0,33	0,30	0,80%	0,74%
H18	2,04	1,31	1,48	1,55	1,38	1,52%	1,22%
H19	2,94	1,44	1,59	2,04	1,85	2,56%	2,19%
H20	3,72	1,63	1,76	2,29	2,11	4,32%	3,82%
H21	10,03	2,12	2,11	4,74	4,75	10,31%	9,72%
H22	23,54	2,55	2,71	9,23	8,68	16,34%	14,31%
H23	35,00	3,19	3,12	10,97	11,24	25,52%	24,25%
H24	66,00	4,03	3,90	16,36	16,94	37,53%	36,15%
H25	126,00	5,74	5,62	21,97	22,42	50,55%	47,60%
H26	422,00	10,68	10,19	39,50	41,42	87,47%	84,57%
H27	788,00	19,41	18,02	40,61	43,72	90,09%	90,32%
H28	1457,00	34,35	32,67	42,42	44,59	95,45%	92,91%
H29	2708,00	63,20	59,52	42,85	45,50	96,81%	95,31%
H30	4993,00	116,29	107,51	42,94	46,44	97,47%	97,53%
H31	9311,28	381,72	353,34	24,39	26,35	98,42%	98,39%
H32	30447,89	705,11	650,15	43,18	46,83	98,68%	98,50%
H33	56024,12	1307,26	1196,27	42,86	46,83	98,64%	98,70%
H34	103084,39	2385,72	2210,26	43,21	46,64	98,80%	98,84%
H35	189675,27	4413,59	4066,88	42,98	46,64	98,82%	98,88%
H36	349002,50	8076,87	7430,19	43,21	46,97	98,85%	98,98%

CAPÍTULO 5

5. DISCUSSÃO E CONCLUSÕES

5.1. ANÁLISE DOS RESULTADOS

Nesta seção apresentadas as análises dos resultados apresentados no capítulo 4. Estas análises serão feitas seguindo a ordem em que os resultados foram mostrados, no Capítulo 4.

Para o teste de acurácia, mostrado na Tabela 8, todas as seis abordagens escolhidas para se fazer tal comparação são algoritmos heurísticos. Todas elas foram aplicadas em alguns, ou todos, os polipeptídeos do grupo 1, como pode ser visto na Tabela 6. Para algumas seqüências do grupo 1, mais especificamente para os polipeptídeos M18A, M18B, M18C e M20A a comparação foi feita apenas com o algoritmo de HU, ZHANG, XIAOLI (2008), que coincidiram totalmente. O mesmo aconteceu com os polipeptídeos M20B, M24 e M25, cujas contagens de iterações hidrofóbicas coincidiram com todas as abordagens que realizaram esta medição nestes polipeptídeos. Para os polipeptídeos M36 e M48 a contagem deste trabalho foi a mesma da maioria das abordagens. As exceções feitas são para os trabalhos de TOMA e TOMA (1996) e o algoritmo Monte-Carlo de UNGER e MOULT (1993). Este apresentou um resultado diferente nas seqüências M36 e M48 e aquele mostrou-se diferente na seqüência M48. Em ambas as disparidades o *software* deste trabalho apresentou contagem de iterações hidrofóbicas superior. A comprovação de que os resultados obtidos são, de fato, ótimas advém do fato de que, além das contagens de iterações hidrofóbicas, há também o registro de todas as ocorrências dos dobramentos daquelas seqüências. Assim que a análise acaba todos os resultados registrados são verificados para se certificar que, de fato, contém o número de iterações hidrofóbicas indicados pela análise. Portanto, não há dúvida de que os resultados obtidos neste trabalho são, de fato, são melhores que os das abordagens de TOMA e TOMA (1996) e de UNGER e MOULT (1993), para os polipeptídeos M36 e M48.

Para o segundo teste, mostrado na Tabela 9, todos os experimentos obtiveram resultados idênticos. Para as seqüências H28 até a H36 não foi possível estabelecer comparação, pois nenhum outro trabalho, dentre os pesquisados, exibiu tais análises. Adicionalmente, os resultados obtidos para os polipeptídeos de H28 a H36 tiveram suas ocorrências verificadas em relação ao número de iterações hidrofóbicas indicadas na Tabela 9. Um aspecto interessante disto é que existe um comportamento para o número de iterações

hidrofóbicas, como mostrado na Figura 54. É possível, por meio da equação 18, estimar um limite superior de iterações hidrofóbicas sabendo-se o número de aminoácidos em um polipeptídeo puramente hidrofóbico. O levantamento desta equação, cujo coeficiente de determinação é alto, pode servir como base para outros estudos de definição do maior número de iterações hidrofóbicas para qualquer polipeptídeo.

Tendo-se, portanto, obtido a validação necessária para considerar o algoritmo confiável, iniciam-se os testes de desempenho. O primeiro teste, mostrado na Tabela 10, com relação ao tempo de análise para os polipeptídeos M18A, M18B e M20B, a abordagem seqüencial foi levemente mais rápida do que a abordagem paralelizada. Porém, estes resultados não são expressivos, sendo necessário observar a coluna da taxa de ocupação, mostrada na mesma tabela. Observa-se que a taxa de ocupação dos agentes para as seqüências de M18A até M25 são baixíssimas. Por este motivo os resultados exibidos pela análise destas seqüências não pode ser comparada com os tempos da versão seqüencial. A taxa de ocupação fica bastante baixa nestas análises porque o espaço de busca resultante de cada polipeptídeo é muito pequeno para conseguir ocupar os 53 agentes. Por outro lado, o único agente da versão seqüencial fica sempre totalmente ocupado, pois é somente ele o responsável pela análise.

Apesar desta característica da taxa de ocupação, as análises dos polipeptídeos de M24 a M36 obtiveram tempos bastante inferiores com a versão paralelizada. O último resultado, na seqüência M48, obteve uma taxa de ocupação bem alta, com mais de 97%, com eficiência maior que 1,0. O reflexo disto é a aceleração da análise em relação ao algoritmo seqüencial da ordem de 59,49 vezes mais rápido, ou seja, foi obtida uma aceleração superlinear. Este tipo de aceleração contribui muito para a diminuição do tempo de busca, como mostrado na 2.6.4.1. Conseguir este tipo de aceleração é extremamente importante pois, como mostrado na seção 2.6.5, tanto a Lei de Amdahl quanto a Lei de Gustafson não prevêem, normalmente, a ocorrência da aceleração superlinear. Ela só é possível quando o algoritmo seqüencial é subótimo ou existe alguma outra vantagem intrínseca à implementação paralelizada. No caso deste trabalho, a ocorrência da aceleração superlinear deve-se à intercomunicação de resultados, mostrada na seção 3.6.3, pois ela permite que a implementação paralelizada traga benefícios que a versão seqüencial não é capaz de oferecer. Na verdade a aceleração superlinear somente foi conseguida após a implementação da intercomunicação de resultados. Por outro lado, sem que a taxa de ocupação estivesse próxima de 100% a intercomunicação seria ineficiente, como acontece com os outros polipeptídeos da Tabela 10 e da Tabela 11 com fatores de aceleração sublineares. Não foi possível fazer a comparação de tempo de

execução com as abordagens heurísticas do grupo 1, por não se ter os respectivos valores de tempo que cada implementação necessitou para executar a busca.

Com relação ao levantamento da eficiência do algoritmo na análise do grupo 1, em relação à execução seqüencial, também mostrado na Tabela 10, está mostrado que, dos polipeptídeos M18A ao M20B, os valores são bastante inexpressivos, pois ocorre baixa ocupação dos agentes. A partir do polipeptídeo M24 a eficiência começa a aumentar. É importante lembrar que se a eficiência for maior do que 1,0 foi atingida a aceleração superlinear. Se este fator cai significa que a aceleração obtida foi pequena ou que houve um ou mais agentes que, apesar de terem participado em certo ponto da análise, não forneceram a mesma contribuição dos demais. No caso deste trabalho, das seqüências M18A até a M20B tanto a taxa de ocupação quanto a eficiência estiveram muito baixos, indicando falta de demanda para ocupar todos os processadores. Na seqüência M24 a aceleração estava alta, mas ainda com baixa eficiência. Isto significa que, mesmo não havendo participação dos agentes em 100% da sua capacidade o algoritmo já foi capaz de determinar os melhores dobramentos, em um tempo razoavelmente inferior à da abordagem seqüencial. Na análise dos polipeptídeos de M25 e M36 houve taxas de ocupação maiores, visto que, como contém mais aminoácidos, o espaço de busca aumenta. O ápice ocorre na seqüência M48, em que a taxa de ocupação é próxima da máxima.

No teste de desempenho para o grupo 2, mostrado na Tabela 11, as primeiras seqüências, de H15 a H20, têm um espaço de busca relativamente pequeno. Portanto, apesar de apresentarem alguma diferença entre a execução seqüencial e a paralelizada, ela ainda é pouco significativa, pelo alto número de agentes utilizados. Ainda assim, ambas já exibem diferenças grandes em relação ao tempo original teórico. Classificar o espaço de busca destes polipeptídeos como pequeno é relativo ao número de agentes utilizados. Em termos absolutos, o espaço de busca deles é bastante grande, mas explorado por 53 agentes não gera ocupação adequada. Porém, a partir do polipeptídeo H21 a diferença entre as versões seqüencial e paralela fica mais evidente. Em relação ao tempo original teórico a diferença para a versão paralela torna-se cada vez maior. O destaque é a análise da seqüência H36 que, de acordo com a previsão original teórica, levaria algo perto de 3,5 dias (302344,5 segundos), ao passo que ela pôde ser analisada em apenas 5500 segundos (uma hora e meia, aproximadamente), fornecendo todas as ocorrências do melhor dobramento.

Com relação aos cálculos do fator de aceleração e a eficiência da execução, mostrados na Tabela 12, os polipeptídeos H15 ao H25 têm o seu fator de aceleração abaixo do número de agentes utilizados, ou seja, apresentam uma aceleração sublinear. Conseqüentemente, a

eficiência também está abaixo de 1,0 e a taxa de ocupação abaixo de 50%, o que significa que os agentes não puderam desempenhar todo o seu potencial durante a análise. A partir do momento em que a taxa de ocupação chega mais próxima dos 100%, como acontece a partir do polipeptídeo H26, a aceleração cresce e a eficiência se aproxima de 1,0. Isto significa que, a partir daí, os agentes estão de fato tendo seu potencial utilizado, chegando a quase exibir aceleração linear. O aumento da eficiência a partir da H26 se deve ao fato de este polipeptídeo ter muitas ocorrências do melhor dobramento e, conseqüentemente, ter um espaço promissor bastante grande. Nas seqüências H30 e a partir da H33, na Tabela 12, pode ser observado uma aceleração superior ao número de agentes e, portanto, uma eficiência acima de 1,0, ou seja, aceleração superlinear. Considerando-se que isto aconteceu para o pior caso, que são polipeptídeos totalmente hidrofóbicos, certamente pode-se esperar um desempenho ainda melhor para seqüências mistas, em que o espaço de busca tende a ser significativamente menor, pelas eliminações proporcionadas pelo PDF.

Como mostrado na Figura 55, a taxa de ocupação é basicamente a mesma para todos os polipeptídeos menores que 20 aminoácidos e maiores que 32. Isto significa que, para menos de 20 aminoácidos não é necessário o poder computacional de 53 agentes, muito menos de 31 agentes. Portanto, tal análise poderia ser executada com menos agentes e obter o mesmo efeito. Acima de 26 aminoácidos todos os grupos de agentes atingiram valores de taxa de ocupação próximos de 85%, o que significa um aumento significativo da demanda computacional. Naturalmente, considerando-se os dois conjuntos com taxas de ocupação altas, aquele que tiver mais agentes processará mais rápido as análises.

Considerando a curva do grupo de 31 agentes, ela praticamente estabilizou sua ocupação a partir seqüência de 26 aminoácidos. Portanto, começa-se a explorar, de fato, o potencial computacional de cada agente. Há uma hipótese que, a partir do momento que o potencial de cada agente tenha chegado ao seu máximo, a tendência natural é que o desempenho do sistema comece a decrescer, caso se aumente o tamanho do espaço de busca do problema. Isto aconteceria pois o capacidade de processamento estaria se tornando um gargalo. Tal hipótese não pôde ser comprovada, pois tal ponto não foi encontrado nesta implementação, como mostrado na Figura 56, em que não há decréscimo na aceleração exibida pelos dois *clusters*. Pode se observar que eles têm um comportamento semelhante e que ambos exibem, em grande parte do gráfico, aceleração superlinear. Pela tendência levemente crescente na aceleração exibida na Figura 56, pode-se concluir que mesmo o grupo de 31 agentes não chegou ainda ao limite da sua capacidade de processamento.

Outra característica interessante, que é conseqüência da taxa de ocupação, é que, se esta estivesse alta e o desempenho caindo, quando se espera que ele aumente, poderia significar que a comunicação estivesse se tornando um gargalo. Seria necessário verificar as taxas de comunicação e avaliar se seria ou não necessário modificar a estratégia de balanceamento. Como a taxa de ocupação e a aceleração estiveram altas na maior parte dos experimentos, a taxa de comunicação não foi mensurada como aspecto principal.

Com relação à medição do tamanho do espaço de busca, mostrada na Tabela 13 é importante lembrar que os válidos são os estados que não contém nenhuma colisão, ou seja, não violam as regras do modelo 2DHP e os promissores são os estados válidos que foram classificados como incapazes de chegar ao melhor resultado final. O objetivo desta medição é avaliar o a eficiência da função PDF. Quanto menos estados promissores melhor para a análise e mais eficiente é a função PDF. Obviamente isto considera que nenhum estado promissor deve ser eliminado erroneamente. O número de estados no espaço válido tende a ficar sempre igual, pois é uma característica do modelo 2DHP e das suas regras e, portanto, não depende do tamanho do *cluster* ou da função PDF. Como mostrado na coluna “Redução PDF”, na Tabela 13, o corte no espaço de busca total foi grande. Esta coluna expressa em valores percentuais a fração do espaço de busca original classificado como espaço promissor e depende diretamente da função PDF implementada.

Para seqüências mistas, como as do grupo 1, não foi possível levantar uma equação capaz de calcular o tamanho do espaço de busca dependente da estrutura primária do polipeptídeo, pois o PDF realiza cortes dinamicamente, muito difíceis de serem previstos com precisão. É possível, por meio de estudos adicionais, fazer um levantamento do tamanho do espaço de busca de acordo com o número e posição dos aminoácidos hidrofóbicos. Obviamente, serão necessários muitos mais polipeptídeos do que os listados no grupo 1, pois, por causa do PDF, o corte depende da quantidade e da posição dos aminoácidos hidrofóbicos na estrutura primária do polipeptídeo.

Os resultados apresentados na Tabela 14 mostram que o reflexo do corte do espaço de busca teve uma conseqüência muito positiva. Houve uma redução bastante grande entre o número de dobramentos ótimos e o total teórico. O número de dobramentos subótimos também se mostrou bastante inferior ao número total teórico de dobramentos, porém ainda bastante acima do número de dobramentos ótimos. Este grande número de dobramentos subótimos é o erro da função PDF, ou seja, ela identificou alguns subdobramentos como tendo um potencial de chegar a ser ocorrência do melhor dobramento que na realidade eles não tinham. A conseqüência é que tais ramos da árvore foram explorados até virarem

dobramentos completos, porém, sem exibir o número de iterações hidrofóbicas mostrado pelas melhores ocorrências. O ideal seria trazer o número de subótimos o mais perto de zero possível, ou seja, eliminá-los, de modo que todos os subdobramentos classificados como promissores de fato dessem origem a ocorrências do dobramento ótimo. Isso evitaria que se analisasse subdobramentos que não serão ocorrências do melhor dobramento, reduzindo ainda mais o tempo da análise.

Nos resultados mostrados na Tabela 15 a diferença de ocupação dos agentes é bastante sensível ao longo de toda a análise dessa tabela. Na análise dos polipeptídeos de H15 a H25, não houve muita modificação na ocupação. Isto se deve ao fato de que, como já abordado anteriormente, o processamento do tamanho do espaço de busca resultante não é suficiente para deixar os agentes ocupados. Eles processam muito rápido suas tarefas e passam boa parte do tempo ociosos. A partir da seqüência H26 começa a aparecer uma diferença mais significativa nas taxas de ocupação. Isto significa que a utilização do balanceamento conseguiu aumentar a taxa de ocupação dos agentes e, conseqüentemente, proporcionar maior desempenho às análises. Todos os testes mostrados anteriormente com o *cluster* utilizaram o balanceamento.

A Figura 57 mostra a característica mais importante da utilização do PDF e seu efeito de corte no espaço de busca: a redução do coeficiente angular da curva do crescimento do espaço de busca, ou seja, o quão rápido ela cresce. O corte proporcionado pelas reduções do espaço de busca conseguiu reduzir o crescimento acelerado da curva, do espaço promissor em relação ao espaço total. Porém, este crescimento ainda é exponencial. Ou seja, apesar de conseguir um corte bastante grande no espaço de busca, ele ainda continua crescendo muito rápido. Este é o principal problema do dobramento de proteínas. Quando este crescimento é rápido em geral significa que existe explosão combinatória, o que classifica como problema NP-difícil. Se o coeficiente fosse constante, paralelo ao eixo “Tamanho do polipeptídeo” significaria que o problema era linear, ou seja, o tamanho do problema cresceria linearmente com o aumento do tamanho do polipeptídeo e deixaria de ser NP-difícil. Mesmo exponencial, o crescimento foi bastante amenizado, fazendo a curva do crescimento ser muito mais próxima da curva de um problema linear do que a curva do crescimento do espaço total. Portanto, isto sugere que os cortes realizados no espaço de busca tenham feito com que com que este problema deixe de ser considerado um problema NP-difícil.

Com relação aos testes realizados com o *cluster* heterogêneo, como mostrado na Tabela 16, os resultados começam a se tornar expressivos, como nos demais testes, a partir do polipeptídeo H26, no qual a taxa de ocupação exibe um salto de cerca de 50% para valores

próximos de 90%. Até esse valor, tanto os tempos de execução quanto os fatores de aceleração se mantiveram constantes. A partir do H26 os resultados começaram a se diferenciar. Considerando o comportamento do *cluster* homogêneo mostrado na Tabela 12, seria razoável esperar uma aceleração pelo menos quase linear neste experimento. Porém, tal expectativa não se concretizou nem no *cluster* de 60 agentes e nem no de 67 agentes. Isto se deve basicamente ao fato de se utilizar um programa de 32 bits em computadores de 64 bits, ou seja, não se explora totalmente o seu potencial de processamento. Comparando-se os fatores de aceleração do *cluster* com 60 agentes com o de 67 agentes, percebe-se que o de 67 agentes, heterogêneo, conseguiu melhorar o desempenho do *cluster* de 60 agentes. Porém, esse aumento não foi perto do aumento proporcionado pela implementação de 64 bits, na qual cada agente adicionado acrescia pelo menos uma unidade no fator de aceleração. O *cluster* heterogêneo mostrou-se também mais vulnerável ao aumento do espaço de busca, sendo que para o polipeptideo H31, em que ocorre um aumento razoável do espaço de busca (como mostrado na Tabela 14) o fator de aceleração caiu consideravelmente em relação aos demais. Isso significa que, a arquitetura do algoritmo com certeza permite que exista a heterogeneidade, porém é necessário sempre se avaliar a viabilidade de tal solução, pois o benefício é razoavelmente inferior àquele proporcionado por um *cluster* homogêneo.

5.2. CONCLUSÕES

A elaboração de um método de eliminação progressivamente mais agressivo do enorme espaço de busca, utilizando cortes topológicos que eliminam os dobramentos inválidos, duplicados e os não promissores, por meio do Potencial de Dobramento Futuro (PDF), é a maior contribuição deste trabalho. Tanto os cortes proporcionados pela análise das violações no modelo 2DHP quanto o corte dinâmico provocado pelo PDF tornaram a aplicação da busca exaustiva factível para o problema do dobramento de proteínas. É importante ressaltar que todas as reduções de espaço de busca mostradas no capítulo 4 foram medidas em relação a polipeptídeos puramente hidrofóbicos, ou seja, o pior caso. Quando se trata de seqüências mistas, com aminoácidos tanto hidrofóbicos quanto polares os resultados tendem a melhorar.

Naturalmente, a redução proposta do espaço de busca ainda não representa a solução definitiva do dobramento para este modelo pois, apesar de amenizar o crescimento do espaço de busca, este ainda leva a uma busca exaustiva inviável para seqüências grandes. Mesmo

assim, o trabalho conjunto da redução com a abordagem paralelizada conseguiu trazer a curva de tempo mais perto da linearização, que seria a solução definitiva para problema do dobramento.

Outra contribuição importante, decorrente da utilização da busca exaustiva, foi a determinação garantida do melhor dobramento, aquele que contém o maior número de iterações hidrofóbicas para um dado polipeptídeo. A comparação com outras abordagens foi benéfica não só para assegurar que os resultados obtidos foram de fato fidedignos, mas também para consolidar a abordagem utilizada neste trabalho como eficaz.

A coleta de todas as ocorrências do melhor dobramento pode servir de pista para que se construa um melhor entendimento sobre o processo de dobramento no modelo 2DHP, ou mesmo para melhorar o desempenho da função PDF deste trabalho.

A adaptação do algoritmo seqüencial elaborado para a estrutura paralelizada, mais especificamente em um *cluster* se provou extremamente útil e computacionalmente poderosa. Esta paralelização foi capaz de acelerar as análises de modo substancial, além do esperado, conseguindo aceleração superlinear em alguns casos. O modelo implementado no *cluster* se provou flexível e independente ao número de agentes participantes. Durante os testes foram utilizadas configurações com 1, 31 e 53 agentes sem problema algum de desempenho ou de gerenciamento, conferindo à arquitetura a característica de escalável sendo robusto o suficiente para suportar a implementação de um *cluster* heterogêneo.

Contando com o poder de corte do espaço de busca, a garantia de determinação do melhor resultado e a flexibilidade do modelo paralelizado, o sistema proposto neste trabalho está pronto para servir de plataforma de estudo para o problema de dobramento de proteínas, baseando-se em busca exaustiva. Além, obviamente, de permitir que outros modelos mais detalhados e mais próximos às expectativas da Biologia sejam implementados. Considerando o exposto, considera-se que os objetivos pretendidos neste trabalho foram alcançados e os resultados foram bastante satisfatórios.

5.3. TRABALHOS FUTUROS

Apesar de este trabalho ter apresentado uma solução computacional promissora, principalmente da redução do espaço de busca e algoritmo de busca paralelizado, ainda há bastante campo para se implementar melhorias que possam incrementar as pesquisas na área do dobramento de proteínas.

A primeira delas é um estudo ainda mais aprofundado da função PDF que, sem dúvida, seria uma fonte poderosa de redução adicional de espaço de busca. Foi possível perceber que os melhoramentos realizados nesta função ao longo do desenvolvimento deste trabalho diminuíram o tempo de análise bem mais do que a paralelização em si pôde fazer.

A segunda melhoria seria a adaptação do modelo atual para poder suportar uma estrutura baseada em *grid*. Seria necessário que fosse alterada estratégia de balanceamento, para que se dependesse menos ainda da comunicação entre agentes, ou talvez modificar a estrutura mestre-escravo. Outro aspecto interessante ainda por ser explorado seria aprimorar a implementação heterogênea, ou seja, compostas por computadores diferentes, até para se fazer uma melhor preparação para a estrutura de *grid*.

Outra melhoria possível seria a disponibilização de um serviço de *Internet* capaz de ser utilizado por qualquer pesquisador interessado em análise de estrutura terciária de polipeptídeos. Juntamente com isto, a disponibilização, também pela *Internet* de um banco de dados contendo os resultados das análises já executadas pouparia não só trabalho de outros pesquisadores, mas também disseminaria as ocorrências do melhor dobramento de cada polipeptídeo, bem como *benchmarks* de desempenho a todos que se interessarem pesquisar sistemas paralelizados ou dobramento de proteínas.

Por fim, uma melhoria considerável deste trabalho seria a implementação de outros modelos de representação de polipeptídeos, como o modelo HPNX (BORNBERG e BAUER, 1997), que melhor descreve os aminoácidos, e também outras versões tanto 2D quanto 3D ou quaisquer outros modelos que tenham mais detalhes sobre as proteínas reais, de modo a melhor solucionar os problemas da Biologia Molecular.

ANEXO 1 - TABELA DOS AMINOÁCIDOS-PADRÃO (COOPER, 2000)

Aminoácido	Código de 3 letras	Código de 1 letra	Polaridade	Cadeia lateral	pH
Alanina	ala	A	Hidrofóbico	-CH3	Neutro
Arginina	arg	R	Polar	-(CH2)3NH-C(NH)NH2	Base forte
Asparagina	asn	N	Polar	-CH2CONH2	Neutro
Ácido aspartico	asp	D	Polar	-CH2COOH	Ácido
Cisteína	cys	C	Hidrofóbico	-CH2SH	Neutro
Ácido glutâmico	glu	E	Polar	-CH2CH2COOH	Ácido
Glutamina	gln	Q	Polar	-CH2CH2CONH2	Neutro
Glicina	gly	G	Hidrofóbico	-H	Neutro
Histidina	his	H	Polar	-CH2-C3H3N2	Base fraca
Isoleucina	ile	I	Hidrofóbico	-CH(CH3)CH2CH3	Neutro
Leucina	leu	L	Hidrofóbico	-CH2CH(CH3)2	Neutro
Lisina	lys	K	Polar	-(CH2)4NH2	Base forte
Metionina	met	M	Hidrofóbico	-CH2CH2SCH3	Neutro
Fenilalanina	phe	F	Hidrofóbico	-CH2C6H5	Neutro
Prolina	pro	P	Hidrofóbico	-CH2CH2CH2	Neutro
Serina	ser	S	Polar	-CH2OH	Neutro
Treonina	thr	T	Polar	-CH(OH)CH3	Neutro
Triptofano	trp	W	Hidrofóbico	-CH2C8H6N	Neutro
Tirosina	tyr	Y	Polar	-CH2-C6H4OH	Neutro
Valina	val	V	Hidrofóbico	-CH(CH3)2	Neutro

ANEXO 2 – EXEMPLO DE ANÁLISE

[root@Tecpar_cluster]# mpiexec -n 17 ./ProjetoDobramentoMPI PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP

Qui Out 9 09:31:03 2008 - Iniciando a analise da proteina: PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP, com 36 aminoacidos, em 16 agentes.
Qui Out 9 09:31:03 2008 - Melhor resultado inicial: 0 contatos
Qui Out 9 09:31:03 2008 - Analise dividida em 1501 partes:
Qui Out 9 09:31:03 2008 - Agente rank 2: encontrou um novo melhor resultado: 11, com 0.309000 segundos.
Qui Out 9 09:31:04 2008 - Agente rank 2: encontrou um novo melhor resultado: 12, com 0.935000 segundos.
Qui Out 9 09:31:04 2008 - Agente rank 5: encontrou um novo melhor resultado: 14, com 1.092000 segundos.
Qui Out 9 09:31:06 2008 - Agente rank 0: completados 10.06% da lista
Qui Out 9 09:31:07 2008 - Agente rank 0: completados 20.05% da lista
Qui Out 9 09:31:08 2008 - Agente rank 0: completados 30.05% da lista
Qui Out 9 09:31:10 2008 - Agente rank 0: completados 40.04% da lista
Qui Out 9 09:31:15 2008 - Agente rank 0: completados 50.03% da lista
Qui Out 9 09:31:15 2008 - Agente rank 0: completados 60.03% da lista
Qui Out 9 09:31:15 2008 - Agente rank 0: completados 70.02% da lista
Qui Out 9 09:31:16 2008 - Agente rank 0: completados 80.01% da lista
Qui Out 9 09:31:16 2008 - Agente rank 0: completados 90.01% da lista
Qui Out 9 09:31:16 2008 - Agente rank 0: completados 100.00% da lista
Qui Out 9 09:31:16 2008 - Fim da analise
Qui Out 9 09:31:16 2008 - Iniciando avaliacao e coleta dos resultados obtidos

Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 1
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 1
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 2
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 2
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 3
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 3
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 4
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 4
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 5
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 5
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 6
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 6
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 7
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 7
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 8
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 8
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 9
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 9
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 10
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 10
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 11
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 11

Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 12
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 12
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 13
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 13
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 14
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 14
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 15
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 15
Qui Out 9 09:31:16 2008 - Avaliacao do agente rank: 16
Qui Out 9 09:31:16 2008 - Requisitados os resultados do agente rank: 16
Qui Out 9 09:31:16 2008 - Gerando o relatorio
Qui Out 9 09:31:16 2008 - Finalizando os agentees
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 1 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 2 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 3 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 4 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 5 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 6 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 7 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 8 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 9 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 10 e recebida a resposta.

Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 11 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 12 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 13 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 14 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 15 e recebida a resposta.
Qui Out 9 09:31:16 2008 - Enviado comando de finalizacao ao agente rank: 16 e recebida a resposta.

Qui Out 9 09:31:16 2008 - Processamento finalizado

ANEXO 3 – EXEMPLO DE RESULTADO DE ANÁLISE

Dados:

Processador: Tecpar_cluster
Numero de Processos (MPI_WORLD_COMM): 16
Numero de Tarefas Inicial (AEstrela): 1500
Proteina: PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP
Tamanho: 36 aminoacidos
Numero de Contatos Hidrofobicos Inicial: 0
Limite de Balanceamento Inferior: 1000 ciclos
Limite de Balanceamento Superior: 10000 ciclos

Analise:

Versao PDF: 7.0 (com Intercomunicacao de resultados)
Inicio: Thu Oct 9 09:31:03 2008
Termino:Thu Oct 9 09:31:16 2008
Tempo de execucao: 12.862 segundos
Tempo para encontrar o primeiro melhor resultado: 1.092 segundos
Numero de mensagens por segundo (m/s): 1326.621
Taxa de Ocupacao Media: 98.188%
Numero de processos redistribuidos(balanceamento): 1

Estatisticas por processador:

Nome	Rank	Tocupado(s)	Tocupado(%)	Nprocs	Partic.	Resultado
cluster01	0	0.850	6.609%	0	0.000%	0
cluster01	1	12.595	97.924%	97	6.458%	14
cluster05	2	12.595	97.924%	92	6.125%	14
cluster05	3	12.606	98.010%	77	5.126%	14
cluster04	4	12.617	98.095%	95	6.325%	14
cluster04	5	12.628	98.181%	93	6.192%	14
cluster02	6	12.578	97.792%	114	7.590%	14
cluster02	7	12.602	97.979%	79	5.260%	14
cluster03	8	12.634	98.227%	78	5.193%	14
cluster03	9	12.606	98.010%	93	6.192%	14
cluster08	10	12.601	97.971%	89	5.925%	14
cluster08	11	12.598	97.947%	104	6.924%	14
cluster07	12	12.547	97.551%	138	9.188%	14
cluster07	13	12.503	97.209%	82	5.459%	14
cluster06	14	12.491	97.116%	100	6.658%	14
cluster06	15	12.471	96.960%	97	6.458%	14
cluster01	16	12.541	97.504%	74	4.927%	14

Resultados:

Melhor resultado: 14 contatos
Espaco de busca valido: 69186178 nos
Espaco de busca promissor: 30451982 nos
Numero de dobramentos promissores: 8384
Numero de dobramentos otimos registradas: 192
Numero de dobramentos otimos encontrados: 192

Ocorrencias:

NLLLNLSLNLSSSONOSONOOSLSSSLNNLSLNLL
NLLLNLSLNLSSSONOSONOOSLSSSLNNLSLNLS
NLLLNLSLNLSSSONOSONOOSLSOSLLNLSLNLL

NLLLNLSLNLSSSONOSONOOSLSOSLLNLSLNLS
NLLLNLSLLSLSOONOSONOOSLSSSLNNLSLNLL
NLLLNLSLLSLSOONOSONOOSLSSSLNNLSLNLS
NLLLNLSLLSLSOONOSONOOSLSOSLLNLSLNLL
NLLLNLSLLSLSOONOSONOOSLSOSLLNLSLNLS
NLLLNLSLLLSSOONOSONOOSLSSSLNNLSLNLL
NLLLNLSLLLSSOONOSONOOSLSSSLNNLSLNLS
NLLLNLSLLLSSOONOSONOOSLSOSLLNLSLNLL
NLLLNLSLLLSSOONOSONOOSLSOSLLNLSLNLS
NLLLNLSLLLSOSONOSONOOSLSSSLNNLSLNLL
NLLLNLSLLLSOSONOSONOOSLSSSLNNLSLNLS
NLLLNLSLLLSOSONOSONOOSLSOSLLNLSLNLL
NLLLNLSLLLSOSONOSONOOSLSOSLLNLSLNLS
NLNLSLNLSLNNNOSONOSOONLNNNLSSLNLSLL
NLNLSLNLSLNNNOSONOSOONLNNNLSSLNLSLN
NLNLSLNLSLNNNOSONOSOONLNONLLSLNLSLL
NLNLSLNLSLNNNOSONOSOONLNONLLSLNLSLN
NLNLSLNLLNLNOOSONOSOONLNNNLSSLNLSLL
NLNLSLNLLNLNOOSONOSOONLNNNLSSLNLSLN
NLNLSLNLLNLNOOSONOSOONLNONLLSLNLSLL
NLNLSLNLLNLNOOSONOSOONLNONLLSLNLSLN
NLNLSLNLLLNNOOSONOSOONLNNNLSSLNLSLL
NLNLSLNLLLNNOOSONOSOONLNNNLSSLNLSLN
NLNLSLNLLLNNOOSONOSOONLNONLLSLNLSLL
NLNLSLNLLLNNOOSONOSOONLNONLLSLNLSLN
NLNLSLNLLLNONOSONOSOONLNNNLSSLNLSLL
NLNLSLNLLLNONOSONOSOONLNNNLSSLNLSLN
NLNLSLNLLLNONOSONOSOONLNONLLSLNLSLL
NLNLSLNLLLNONOSONOSOONLNONLLSLNLSLN
NLLLSLNLSLNNNOSONOSOONLNNNLSSLNLSLL
NLLLSLNLSLNNNOSONOSOONLNNNLSSLNLSLN
NLLLSLNLSLNNNOSONOSOONLNONLLSLNLSLL
NLLLSLNLSLNNNOSONOSOONLNONLLSLNLSLN
NLLLSLNLLNLNOOSONOSOONLNNNLSSLNLSLL
NLLLSLNLLNLNOOSONOSOONLNNNLSSLNLSLN
NLLLSLNLLNLNOOSONOSOONLNONLLSLNLSLL
NLLLSLNLLNLNOOSONOSOONLNONLLSLNLSLN
NLLLSLNLLLNNOOSONOSOONLNNNLSSLNLSLL
NLLLSLNLLLNNOOSONOSOONLNNNLSSLNLSLN
NLLLSLNLLLNNOOSONOSOONLNONLLSLNLSLL
NLLLSLNLLLNNOOSONOSOONLNONLLSLNLSLN
NLLLSLNLLLNONOSONOSOONLNNNLSSLNLSLL
NLLLSLNLLLNONOSONOSOONLNNNLSSLNLSLN
NLLLSLNLLLNONOSONOSOONLNONLLSLNLSLL
NLLLSLNLLLNONOSONOSOONLNONLLSLNLSLN
NNNNLNONLNOOOSLSOSLSSONOSONNLNONLNO
NNNNLNONLNOOOSLSOSLSSONOSONNLNONLNN
NNNNLNONLNOOOSLSOSLSSONOOONLLNONLNO
NNNNLNONLNOOOSLSOSLSSONOOONLLNONLNN

REFERÊNCIAS

ALBERTS, B.; JOHNSON, A.; LEWIS, J.; RAFF, M.,; ROBERTS, K.; WALTER, P. M**olecular Biology of The Cell,** 4th ed. New York: Garland Science, 2002.

ALBERTS, B.; BRAY D.; HOPKIN, K.; JOHNSON, A.; LEWIS, J.; RAFF M.; ROBERTS K.; WALTER P. **Essential Cell Biology**, 2nd ed. New York: Garland Science, 2003.

ALONSO, D.; DILL, K. Solvent denaturation and stabilization of globular proteins. **Biochemistry**, v. 30, p. 5974-5985, 1991.

AMDAHL, G. M. Validity of the single processor approach to achieving large scale computing capabilities. In: **American Federation of Information Processing Societies (AFIPS) Spring Joint Computer Conference**, v. 30, p. 483-485, 1967.

ANFINSEN, C. B. Principles that govern the folding of protein chains. **Science**, v. 181, p. 223-230, 1973.

ANFINSEN, C.; HABER, E.; WHITE, F. The kinetics of the formation of native ribonuclease during oxidation of the reduced polypeptide domain. **Proceedings of the National Academy of Science of the USA**, v. 47, p. 1309-1314, 1961.

ANFINSEN, C. B.; SCHERAGA, H. A. Experimental and theoretical aspects of protein folding. **Advances in Protein Chemistry**, v. 29, p. 205-300, 1975.

ARMSTRONG JUNIOR, N. B.; LOPES, H. S.; LIMA, C. E. Reconfigurable computing for accelerating protein folding simulations. In: **Lecture Notes on Computer Science (LNCS)**, v. 4419, p. 314-325, 2007.

ARMSTRONG JUNIOR, N. B.; LOPES, H. S.; LIMA. Preliminary steps towards protein folding prediction using reconfigurable computing, International conference on reconfigurable computing and FPGAS, In: **International Conference on ReConFigurable Computing and FPGAs (ReConfig)**, 2006.

AVANCINI, E.; FAVARETTO, J. **Biologia**: **Uma Abordagem Evolutiva e Ecológica.** 1ª ed. São Paulo: Moderna, 1997.

BACKOFEN, R.; WILL, S.; BAUER, E. **Bioinformatics**, v. 15, p. 234-242, 1999.

BADER, D. A.; PENNINGTON, R. Cluster Computing: Applications, **The International Journal of High Performance Computing**, v. 15, p. 181-185, 2001.

BALDI, P.; BRUNAK, S. **Bioinformatics: The Machine Learning Approach.** 4th ed. Massachusetts: MIT Press, 2000.

BAXEVANIS, A.; OUELLETTE, B. **Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins**, 3rd ed., New York: Wiley Interscience, 2001.

BERG, J.; TYMOCZKO, J.; STRYER, L. **Biochemistry**, 5th ed. New York: Freeman, 2002.

BERGER, B.; LEIGHT, T. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. **Journal of Computational Biology**, v. 5, p. 27-40, 1998.

BERMAN, F.; HEY, A.; FOX, G. C. **Grid Computing: Making The Global Infrastructure a Reality**. 1st ed. New York: Wiley, 2003.

BORNBERG, S.; BAUER, E. Chain growth algorithms for HP-type lattice proteins, In: **Proceedings of the 1st Annual International Conference on Computational Molecular Biology**, p. 47-55, 1997.

BROMBERG, S.; DILL, K. A. Side-chain entropy and packing in proteins. **Protein Science**, v. 3, p. 997-1009, 1994.

BROOKS, C. L. Chemical physics of protein folding. **Proceedings of National Academy of Science of USA**, v. 95, p. 11037-11038, 1998.

BROWN, J.; KLEE, W. Helix-coil: transition of the isolated amino terminus. **Biochemistry**, v. 10, p. 470-476, 1971.

BUYYA, R. **High Performance Cluster Computing: Programming and Applications**, 1st ed. New Jersey: Prentice Hall, v. 2, 1999.

CHAN, H. S.; DILL, K. A. “Sequence space soup” of proteins and copolymers. **Journal of Chemical Physics**, v. 95, p. 3775-3787, 1991.

CHAN, H.; DILL, K. A simple model of chaperonin-mediated protein folding. **Proteins: Structure, Function and Genetics**, v. 24, p. 345-351, 1996.

CHANDRA, R. , DAGUM, L. , KOHR, D., MENON, R., MAYDAN, D., MCDONALD, J.; **Parallel Programming in OpenMP**, 5th ed. Morgan Kaufmann Publishers: San Francisco, 2001.

CHANDRU, V.; DATTASHARMA, A.; KUMAR. V. The algorithmics of folding proteins on lattices. **Discrete Applied Mathematics,** v. 127, 2003.

CHAUDHURI, P. **Parallel Algorithms: Design and Analysis**, 1st ed. Brunswick: Prentice Hall, 1992.

CLOTE, P.; BACKOFEN, R. **Computational Molecular Biology**: **An Introduction**, 1st ed. Chichester: Wiley Interscience, 2000.

COOPER, G. **The Cell: A Molecular Approach**. 2nd ed. Massachussetts: Sinauer Associates, 2000.

COVEL, D. G. Lattice model simulations of polypeptide chain folding, **Journal of Molecular Biology**, v. 235, p. 1035–1043, 1994.

CRESCENZI, P.; GOLDMAN, D.; PAPADIMITRIOU, C.; PICCOLBONI A.; YANNAKAKIS, M. On the complexity of protein folding. **Journal of Computational Biology**, v. 5, p. 423-466, 1998.

CRICK, F. On protein synthesis. **Symposia of the Society of Experimental Biology**, 12th ed. p. 138-163, 1958.

CRICK, F. Central dogma of molecular biology, **Nature**, v. 227, p. 561-563, 1970.

DAUGHERITY, W. C. A neural-fuzzy system for the protein folding problem. In: **Proceedings of The 3rd International Workshop on Industrial Fuzzy Control & Intelligent Systems**, p. 47–49, 1993.

DIESTEL, R. **Graph Theory,** 3rd ed. New York: Springer-Verlag Heidelberg, 2005.

DILL, K. A. Theory for the folding and stability of globular proteins. **Biochemistry**, v. 24, p. 1501-1509, 1985.

DILL, K. A. Polymer principles and protein folding. **Protein Science**, v. 8, p. 1166-1180, 1999.

DILL, K. A.; BROMBERG, S.; YUE, K.; FIEBIG, K. M.; YEE, D. P.; THOMAS, P. D.; CHAN, H. S. Principles of protein folding: a perspective from simple exact models. **Protein Science**, v. 4, p. 561-602, 1995.

DILL, K. A.; CHAN, H. S. From Levinthal to pathways to funnels. **Nature Structural Biology**, v. 4, p. 10-19, 1997.

DILL, K. A.; OZKAN, S. B.; WEIKL T. R.; CHODERA, J. D.; Voelz V. A. The protein folding problem: when will it be solved? **Structural Biology**, v. 17, p. 342–346, 2007.

DOBSON, C. The nature and significance of protein folding. In: PAIN R. **Mechanisms of Protein Folding**, 2nd ed. Oxford: Oxford University Press, 2000.

DONGARRA, J.; FOX G.; KENNEDY, K.; TORCZON, L.; GROPP, W.; FOSTER, I.; WHITE, A. **The Sourcebook of Parallel Computing**, 1st ed. New York: Morgan Kaufmann, 2002.

DUBCHAK, I.; HOLBROOK, S. R.; KIM S., SUNG-HOU K. Comparison of two variations of neural network approach to the prediction of protein folding pattern, In: **Proceedings of the International Conference on Intelligence Systems in Molecular Biology**, p. 118-126, 1993.

DYSON, H.J.; RANCE, M.; HOUGHTEN, R.A.; LERNER, R.A.; WRIGHT, P.E. Folding of immunogenic peptide fragments of proteins in water solution. **Journal of Molecular Biology**, v. 201, p. 161-200, 1988.

EASTWOOD, M. P.; HARDIN, C.; LUTHEY-SCHULTEN, Z.; WOLYNES, P. G. Evaluating protein structure-prediction schemes using energy landscape theory. **IBM Journal of Research and Development**, v. 45, p. 56-70, 2001.

EISENBERG, D.; WEISS, R.; TERWILLINGER, T. The hydrophobic moment detects periodicity in protein hydrophobicity. **Biophysics**, v. 81, p. 140-144, 1984.

ELLIS, R. The molecular chaperone concept. **Seminary of Cellular Biology**, v. 1, p. 1-9, 1990.

EYRICH, V. A.; STANDLEY, D. M.; FRIESNER, R. A. Prediction of protein tertiary structure to low resolution: performance for a large and structurally diverse test set**.** **Journal of Molecular Biology**, v. 288, p. 725–742, 1999.

FERSHT, A. R. **Structure and Mechanism in Protein Science: a Guide to Enzyme Catalysis and Protein Folding,** 1st ed. New York: Freeman, 1999.

FINK, A. Chaperone-mediated protein folding. **Physiological Reviews**, v. 79, 1999.

FOSTER, I.; KESSELMAN C. **The Grid: Blueprint for a New Computing Infrastructure**, 1st ed. San Francisco: Morgan Kaufmann Publishers, 1999.

FRAENKEL, A. S. Complexity of protein folding. **Bulletin of Mathematical Biology**. v. 55, p. 1199-1210, 1993.

GEIST, A.; BEGUELIN, A. ; DONDARRA, J., JIANG, W.; MANCHEK, B.; SUNDERAM, V. **PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Network Parallel Computing**, 1st ed. Cambridge: MIT Press, 1994.

GETHING, M. **Guidebook to Molecular Chaperones and Protein-Folding Catalysts**, 1st ed. Oxford: Oxford University Press, 1997.

GIBAS, C.; JAMBECK, P. **Developing Bioinformatics Computer Skills**, 1st ed. Sebastopol: O'Reilly & Associates, 2001.

GRIFFITHS, A. J. F.; MILLER, J. H.; SUZUKI, D. T.; LEWONTIN, R. C.; GELBART, W. M. **An Introduction to Genetic Analysis**, 7th ed. New York: Freeman, 2000.

GROPP, W.; LUSK, E.; DOSS, N.; SKJELLUM, A. A high performance, portable implementation of the MPI Message-Passing Interface standard. **Parallel Computing**, v. 22, p. 789–828, 1996.

GROPP, W.; LUSK, E.; SKJELLUM, A. **Using MPI: Portable Parallel Programming with the Message Passing Interface**, 2nd ed. Cambridge: MIT Press, 1999.

GRUEBELE, M. Protein folding: the free energy surface current opinion. **Structural Biology**, v. 12, p. 161-168, 2002.

GUSTAFSON, J. Reevaluating Amdahl's law. **Communications of the ACM**, v. 31, p. 532-533, 1988.

HANSMANN, U. H. E.; OKAMOTO, Y. Comparative study of multicanonical and simulated annealing algorithms in the protein folding problem. **Physica A**, v. 212, p. 415–437, 1994.

HART, W.E., ISTRAIL, S. Lattice and off-lattice side chain models of protein folding. **Journal of Computational Biology**. v. 4, p. 241-259, 1997.

HARTL, F. Molecular chaperones in cellular protein folding. **Nature**. v. 381, p. 571-580, 1996.

HASHIMOTO, M.; ROCKENSTEIN, E.; CREWS, L.; MASLIAH, E. Role of protein aggregation in mitochondrial dysfunction and neurodegeneration in Alzheimer's and Parkinson's diseases. **Neuromolecular Medicine**, v. 4, p. 21-36, 2003.

HIROYASU, T.; MIKI, M.; OGURA, S.; AOI, K.; YOSHIDA, T.; OKAMOTO, Y. DONGARRA, J. Energy minimization of protein tertiary structure by parallel simulated annealing using genetic crossover. **Transactions on Advanced Computing Systems**, v. 44, p: 11–27, 2003.

HOLBROOK, S. R.; MUSKAL, S. M.; KIM S. Predicting protein structural features with artificial neural networks. In: HUNTER L., **Artificial Intelligence and Molecular Biology**, 1st ed., New York: AAAI Press, p. 162-194, 1993.

HU, X.; ZHANG, J.; XIAO, J.; LI, Y. Protein folding in hydrophobic-polar lattice model: a flexible ant-colony optimization approach. **Protein and Peptide Letters**, v. 15, p. 469-477, 2008.

HUNTER, L. **Artificial Intelligence and Molecular Biology**, 1st ed**.** Boston: AAAI Press, 1993.

ISTRAIL S.; WALENZ B. Folding proteins fast random samples. **Science Magazine**, v. 269, p. 1821, 1995.

JAENICKE, R. Protein folding and protein association. **Progress in Biophisics and Molecular Biology**, v. 49, p. 117-237, 1987.

JONES, N.; PEVZNER, P. **An Introduction to Bioinformatics Algorithms,** 1st ed. Cambridge: MIT Press, 2004.

JORDAN, H. F. **Fundamentals of Parallel Processing**. 2nd ed. New York: Prentice Hall, 1999.

KLOTZ, I. M. **Chemical Thermodynamics: Basic Theory and Methods**. 3rd ed. New York: Prentice-Hall, 1955.

KÖNIG, R.; DANDEKAR, T. Improving genetic algorithms for protein folding simulations by systematic crossover. **Biosystems**, v. 50, p. 17-25, 1999.

KRASNOGOR, N., PELTA, D., LOPEZ, P. E. M., CANAL E. Genetic algorithm for the protein folding problem: a critical view. In: **Proceedings of Engineering of Intelligent Systems**, v. 2, p. 353-360, 1998.

KRASNOGOR, N., BLACKBURNE, B. P., HIRST, J. D., BURKE, E. K. Multimeme algorithms for protein structure prediction. In: **Proceedings of Parallel Problem Solving From Nature, Lecture Notes in Computer Science**, v. 2439, p. 769 – 778, 2002.

LATTMAN, E. E.; ROSE, G. D. Protein folding: what's the question? **Proceedings of the National Academy of Science of USA**, v. 90, p. 439-441, 1993.

LEHNINGER, A. L; NELSON, D. L.; COX, M. M. **Principles of Biochemistry**, 3rd ed. New York: Worth Publishers, 1998.

LEONHARD, K.; PRAUSNITZ, J. M.; RADKE, C. J. 3D-lattice Monte Carlo simulations of model proteins: side effects on folding thermodynamics and kinetics. **Biophysical Chemistry**, v. 106, p. 81–89, 2003.

LESK, A. **Introduction to Bioinformatics**. 2nd ed. New York: Oxford University Press, 2002.

LEVINTHAL, C. Are there pathways for protein folding ? **Journal de Chimie Physique**, v. 65, p. 44-45, 1968.

LI, M. S.; KLIMOV, D. K.; THIRUMALAI, D. Folding in lattice models with side chains. **Computer Physics Communications**, v. 147, p. 625–628, 2002.

LIANG, F.; WONG, W. H. Evolutionary Monte Carlo for protein folding simulations. **Journal of Chemical Physics**, v. 115, p. 3374–3380, 2001.

LODISH, H.; BERK , A.; MATSUDAIRA, P.; KAISER, C. A.; KRIEGER, M.; SCOTT, M. P.; ZIPURSKY, L.; DARNELL, J. **Molecular Cell Biology**. 4th. ed. New York: Freeman, 2000.

LUGER, G. **Inteligência Artificial: Estruturas e Estratégias Para a Solução de Problemas Complexos**, 4ª ed. Porto Alegre: Bookmann, 2004.

MITCHELL, M. **An Introduction to Genetic Algorithms**, 1st ed. New York: MIT Press, 1996.

MESSAGE PASSING INTERFACE FORUM; MPI: A message-passing interface standard. **International Journal of Supercomputer Applications**, v. 8, p. 165–414, 1994.

MESSAGE PASSING INTERFACE FORUM; MPI2: A message passing interface standard. **International Journal of High Performance Computing Applications**, v. 12, p. 1–299, 1998.

MOLLER, N.; NAIR, K. Regulation of muscle mass and function: effects of aging and hormones. **Protein and Aminoacids**, v. 1, p. 121-136, 1999.

MOUNT, D. **Bioinformatics: Sequence and Genome Analysis**, 1st ed. New York: Cold Spring Harbor Laboratory Press, 2001.

MUELLER, F.; A library implementation of POSIX threads under UNIX, In: **Proceedings of the Advanced Computing Systems Association (USENIX) Conference**, San Diego, 1993.

NAKAMURA, H. K.; SASAKI, T. N.; SASAI, M. Strange kinetics and complex energy landscapes in a lattice model of protein folding. **Chemical Physics Letters**, v. 347, p. 247–254, 2001.

NGO, J. T.; MARKS, J.; KARPLUS, M. Computational complexity, protein structure prediction and the Levinthal paradox. In: MERZ JUNIOR, K.; LeGRAND, S. **The Protein Folding Problem and Tertiary Structure Prediction**. 1st ed. Boston: Birkhäuser, 1994.

NICOSIA, G. **Immune Algorithms for Optimization And Protein Structure Prediction**. PhD. Thesis, University of Catania, Italy, 2004.

OSGUTHORPE, D. J. Ab initio protein folding. **Current Opinion in Structural Biology**, v. 10, p. 146–152, 2000.

OSTROVSKY, B., CROOKS, G., SMITH, M. A., BAR-YAM, Y. Cellular automata for polymer simulation with application to polymer melts and polymer collapse including implications for protein folding. **Parallel Computing**, v. 27, p. 613-641, 2001.

PARSELL, D.A.; KOWAL, A.S.; SINGER, M.A.; LINDQUIST, S. Protein disaggregation mediated by heat-shock protein Hsp104. **Nature,** v. 372, p. 475-478, 1994.

PAULING, L.; COREY, R. Configurations of polypeptide chains with favored orientations of the polypeptide around single bonds: two pleated sheets. **Proceedings of the National Academy of Science of the USA**, v. 37, p. 729-740, 1951.

PAULING, L.; COREY, R.; BRANSON, H. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. **Proceedings of the National Academy of Science of the USA**, v. 37, p. 205-211, 1951.

PEDERSEN, C. **Algorithms in Computational Biology,** Ph.D thesis, Department of Computer Science, University of Aarhus, Denmark, 2000.

PFISTER, G. **In Search of Clusters**. 2nd ed. New Jersey: Prentice Hall, 1997.

PIETZSCH, J. Protein folding diseases. **Horizon Symposia**, v. 1, p: 45-54, 2002.

PONTIN, C.; RUSSELL, R. The natural history of protein domains. **Annual Review of Biophysics and Biomolecular Structure**. v. 31, p. 45-71, 2002.

RABOW, A. A.; SCHERAGA, H. A. Lattice neural network minimization: application of neural network optimization for locating the global-minimum conformations of proteins. **Journal of Molecular Biology**, v. 232, p. 1157–1168, 1993.

RASO, S. W.; KING, J. Protein Folding and human disease. In: PAIN, R. **Mechanisms of Protein Folding**. 2nd ed. NewYork: Oxford University Press, 2000.

RICHARDSON, J. S. The anatomy and taxonomy of protein structure and advances. **Protein Chemistry**, v. 34, p. 167-339, 1981.

RODER, H.; ELÖVE, G.; ENGLANDER, S. Structural characterization of folding intermediates in cytochrome C by H-exchange abelling and proton NMR. **Nature**, v. 335. p. 700-704, 1988.

ROOSTA, S. H. **Parallel Processing And Parallel Algorithms: Theory And Computation**. 1st ed. New York: Springer-Verlag, 1999.

RUMERHART, D. **Parallel Distributed Processing**., 10th ed., Cambridge: MIT Press, 1992.

RUSSEL, S. J.; NORVIG, P. **Inteligência Artificial**. 2ª ed. Rio de Janeiro: Elsevier, 2004.

SAVAGEAU, M. Proteins of escherichia coli come in sizes that are multiples of 14 kDa: Domain concepts and evolutionary implications. **Proceedings of the National Academy of Sciences**, v. 83, p. 1198-1202, 1986.

SCAPIN, M. P.; LOPES, H. S. Protein structure prediction using an enhanced genetic algorithm for the 2D HP model. In: **Proceedings of 3rd Brazilian Workshop on Bioinformatics**, Brasilia, CD-ROM, 2005.

SHAKHNOVICH, E. I.; GUTIN, A. M. Engineering of stable and fast-folding sequences of model proteins. **Proceedings of the National Academy of Science of the USA**, v. 90, p. 7195-7199, 1993.

SHMYGELSKA, A.; HERNÁNDEZ, R. A.; HOOS, H. H. An ant colony optimization algorithm for the 2D HP protein folding problem. In: **Proceedings of the 3rd**

International Workshop (ANTS), Lecture Notes in Computer Science, v. 2463, p. 40–55, 2002.

SHMYGELSKA, A.; HOOS, H. H. An improved ant colony optimization algorithm for the 2D HP protein folding problem. In**: Proceedings of the 16th Canadian Conference on Artificial Intelligence**, v. 16, p.35-43, 2003.

SIMON, I.; FISER, A.; TUSNÁDY, G. E. Predicting protein conformation by statistical methods. **Biochimica et Biophysica Acta**, v. 1549, p. 123–136, 2001.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.; DONGARRA, J. **MPI: The Complete Reference**. 2nd. ed. Cambridge: MIT Press, 1997.

SOCCI, N. D.; ONUCHIC, J. N. Folding kinetics of protein-like heteropolymers. **Journal of Chemical Physics**, v. 101, p. 1519-1528, 1994.

SPARRER, H.; RUTKAT, K.; BUCHNER, J. Catalysis of protein folding by symmetric chaperone complexes. **Biochemistry**, v. 94, p. 1096-1100, 1997.

STERLING, T. An introduction to PC clusters for high performance computing. In: BAKER M., **Cluster Computing White Paper**, New York: Oxford University Press, 2000.

TANG, C. Simple models of the protein folding problem. **Physica A**, v. 288, p. 31-48, 2000.

THIRUMALAI, D. **Statistical Mechanics, Protein Structure and Protein-Substrate Interactions**. 1st ed. NewYork: Doniach, 1994.

THOMAS, P. J.; KO, Y. H.; PEDERSEN, P. L. Altered protein folding may be the molecular basis of most cases of cystic fibrosis. **FEBS Letters**, v. 312, p. 7-9, 1992.

THOMASSON, W. A. Unravelling the mistery of protein folding, **Breakthroughs in Bioscience, Federation of American Societies for Experimental Biology**. Disponível em: http://opa.faseb.org/pdf/protfold.pdf. Acesso em: 05 de junho de 2008.

TOMA, L.; TOMA, S. Contact interactions method: a new algorithm for protein folding simulations. **Protein Science**, v. 5, p147-153, 1996.

UDGAONKAR, J.; BALDWIN, R. NMR evidence for an early framework intermediate on the folding pathway of ribonuclease A. **Nature**, n. 335. p. 694-699, 1988.

UNGER, R.; MOULT, J. Finding the lowest free energy conformation of a protein is a NP-hard problem: proof and implications. **Bulletin of Mathematical Biology**, n. 55, p. 1183-1198, 1993a.

UNGER, R.; MOULT, J. A genetic algorithm for three dimensional protein folding simulations. In: **Proceedings of the 5th Annual International Conference on Genetic Algorithms**, p. 581-588, 1993b.

WELCH, W. J.; HOWARD, M. Antagonists to the rescue. **The Journal of Clinical Investigation**, v. 105, 2000.

WETLAUFER, D. Nucleation, rapid folding, and globular intrachain regions in proteins. **Proceedings of the National Academy of Sciences of the USA**, v. 70, p. 697-701, 1973.

WOLYNES, P. G.; ONUCHIC, J. N.; THIRUMALAI; D. Navigating the folding routes, **Science**, v. 267, p. 1619-1620, 1995.

YESYLEVSKYY, S. O.; DEMCHENKO, A. P. Towards realistic description of collective motions in the lattice protein folding models. **Biophysical Chemistry**, v. 109, p. 17–40, 2004.

YUE, K., DILL K. A. Sequence-structure relationships in proteins and copolymers. **Physical Review**, v. 48, p 2267-2278, 1993.

YOUNG, V.; YU, Y. Protein and aminoacid metabolism. In: **Nutrition and Metabolism in the Surgical Patient**, 2nd ed., Boston: Little Brown and Company, p. 159-200, 1996.

ZWANZIG, R.; SZABO, A.; BAGCHI, B. Levinthal's paradox. **Proceedings of the National Academy of Sciences of the USA**, v. 89, p. 20-22, 1992

RESUMO:

Este trabalho propõe uma abordagem paralela de análise de um dos problemas mais complexos da bioinformática: o problema do Dobramento de Proteínas. As proteínas são compostos químicos que estão presentes em todos os seres vivos. À medida que estas proteínas vão sendo sintetizadas elas vão se dobrando até assumir uma forma enovelada, conhecida como conformação nativa. Ela tem relação direta com a funcionalidade que ela tem no organismo a ela relacionado. O processo de dobramento protéico é intensamente estudado tanto do ponto de vista biológico quanto computacional. Vários modelos computacionais, discretos e contínuos foram propostos para representar o dobramento de proteínas. Este trabalho utiliza o modelo Hidrofóbico-Polar bi-dimensional (2DHP). Foi proposto um algoritmo de busca exaustiva paralela capaz de encontrar todas as ocorrências da conformação nativa de uma proteína em um tempo aceitável, para proteínas de até cerca de 50 aminoácidos. A proposta baseia-se no algoritmo A* modificado, implementando parâmetros topológicos de avaliação capaz de propor significativos cortes no espaço de busca. Foi utilizada uma arquitetura de *cluster* testes do algoritmo. Técnicas de gerenciamento de processamento paralelo foram elaboradas para balanceamento de carga, tendo sido uma taxa de aceleração superlinear. Testes de acurácia do algoritmo foram realizados, comparando seus resultados com outras abordagens. Também foram realizados testes de desempenho e escalabilidade. Os resultados indicaram que a abordagem proposta é uma alternativa viável para a análise do dobramento de proteínas com o modelo 2DHP, podendo ser extensível para outros modelos.

PALAVRAS-CHAVE

Dobramento de proteínas, Busca exaustiva, Sistemas Paralelizados, Bioinformática, *Cluster*.

ÁREA/SUBÁREA DE CONHECIMENTO

1.03.04.00-2 Descrição: Sistemas de Computação

2.08.04.00-8 Descrição: Biologia Molecular

2008

Nº: 488

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo