UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
E INFORMÁTICA INDUSTRIAL

CÉSAR MANUEL VARGAS BENÍTEZ

UM ALGORITMO GENÉTICO PARALELO PARA O PROBLEMA DE DOBRAMENTO DE PROTEÍNAS UTILIZANDO O MODELO 3DHP COM CADEIA LATERAL

DISSERTAÇÃO DE MESTRADO

CURITIBA

2010

CÉSAR MANUEL VARGAS BENÍTEZ

UM ALGORITMO GENÉTICO PARALELO PARA O PROBLEMA DE DOBRAMENTO DE PROTEÍNAS UTILIZANDO O MODELO 3DHP COM CADEIA LATERAL

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

Orientador: Heitor S. Lopes

CURITIBA

2010

Para Reinaldo, Lilian, Mara, Fatima, Maria José e Karina

AGRADECIMENTOS

Gostaria de agradecer especialmente à minha familia pela educação, incentivo e amor incondicional. Com certeza não alcançaria este ponto da minha vida sem as suas orientações e infinitas palavras de motivação.

Também gostaria de agradecer à minha esposa, por seu amor, apoio e dedicação. Agradeço pela ajuda nos momentos iniciais do trabalho e compreensão em meus momentos de ausência.

Reverencio ao Prof. Dr. Heitor Silvério Lopes pela sua dedicação, paciência, motivação e orientação que me ajudaram na realização deste trabalho.

Agradeço a todos os colegas do laboratório de bioinformática da UTFPR, pelas conversas e ajuda operacional. Em especial gostaria de agradecer aos amigos Chidambaram e Wagner pelas sabias palavras. Também gostaria de agradecer aos colegas Nilton e Rafael Betito pelas discussões iniciais sobre bioinformática.

Finalmente, agradeço aos professores e pesquisadores da banca examinadora pela atenção e contribuição dedicadas a este trabalho.

"Cem vezes todos os dias lembro a mim mesmo que minha vida interior e exterior, depende dos trabalhos de outros homens, vivos ou mortos, e que devo esforçar-me a fim de devolver na mesma medida que recebi" – Albert Einstein.

"Todos os erros humanos são impaciência, uma interrupção prematura de um trabalho metódico" – Franz Kafka

"A ciência se compõe de erros que, por sua vez, são os passos até a verdade" – Julio Verne

"Que mar tan encrespado!" – Ned Land

RESUMO

VARGAS BENÍTEZ, César Manuel. Um Algoritmo Genético Paralelo para o Problema de Dobramento de Proteínas utilizando o modelo 3DHP com cadeia lateral. 156 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2010.

Este trabalho apresenta um algoritmo genético paralelo (AGP) para o problema de dobramento de proteínas, utilizando o modelo 3DHP-SC. Este modelo tem sido pouco abordado devido ao elevado grau de complexidade envolvido. Foi proposta uma função de *fitness* baseada na energia livre e na compacidade do dobramento. Operadores genéticos especiais foram desenvolvidos, além de estratégias para auxiliar o algoritmo no processo de busca de conformações de proteínas. Vários experimentos foram realizados para ajustar todos os parâmetros do sistema, incluindo os parâmetros básicos do AG (probabilidades de mutação e *crossover*, e o tamanho de torneio) e os parâmetros dos operadores especiais e das estratégias. O efeito da matriz de energias para o modelo no desempenho do algoritmo também foi estudado. Uma comparação com outra abordagem de computação evolucionária também foi realizada, a fim de verificar o desempenho do método proposto. Devido a não existir, até então, *benchmarks* para teste deste modelo, foi proposto um conjunto de 25 sequências baseado em outro modelo mais simplificado. Os resultados obtidos mostraram que o AGP alcançou um bom nível de eficiência e obteve dobramentos biologicamente coerentes, sugerindo a adequabilidade da metodologia proposta.

Palavras-chave: Dobramento de proteínas, Algoritmo Genético Paralelo, Bioinformática, 3DHP-SC

ABSTRACT

VARGAS BENÍTEZ, César Manuel. A Parallel Genetic Algorithm for Protein Folding Prediction Using the 3DHP Side Chain model. 156 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2010.

This work presents a parallel genetic algorithm (PGA) for the protein folding problem, using the 3DHP-SC model. This model has been sparsely studied in the literature due to its complexity. A new fitness function was proposed, based on the free-energy and compacity of the folding. Special genetic operators were developed, besides strategies to aid the algorithm in the search of protein conformations. Many experiments were done to adjust all the parameters of the system, including the basic parameters of the GA (mutation and crossover probability, and tournament size) and parameters of the special genetic operators and strategies. The effect of the energy matrix of the model in the performance of the algorithm was also studied. Moreover, a comparison with other evolutionary computation approach was done, to verify the performance of the proposed method. Since there is no benchmark available to date, a set of 25 sequences was used, based on a simpler model. Results show that the PGA achieved a good level of efficiency and obtained biologically coherent results, suggesting its adequacy for the problem.

Keywords: Protein Folding, Parallel Genetic Algorithm, Bioinformatics, 3DHP-SC

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

2D	bidimensional
3D	tridimensional
ABC	*Artificial Bee Colony*
ABC–HH	*Artificial Bee Colony* hierárquico
ABC–ME	*Artificial Bee Colony* mestre-escravo
ACO	*Ant Colony Optimization*
AG	Algoritmo Genético
AGP–HH	Algoritmo Genético Paralelo hierárquico
AGP–ME	Algoritmo Genético Paralelo mestre-escravo
AIS	*Artificial Immune Systems*
CE	Computação Evolucionária
CESDIS	*Center of Excellence in Space Data and Information Sciences*
cGA	*Cellular Genetic Algorithms*
CGE	*Charged Graph Embedding*
COW	*Cluster of Workstations*
cryo-ME	*Cryo-electron Microscopy*
DE	*Differential Evolution*
dGA	*Distributed Genetic Algorithms*
DHB	Dizimação *hot-boot*
EE	Estratégias Evolutivas
FA	*Firefly Algorithm*
HP	Hidrofóbico-Polar
HP-TSSC	*Hydrophobic-Polar Tangent Spheres Side Chain Model*
HSP	*Heat Shock Proteins*

HT	Hipótese Termodinâmica
LPE	*Lattice Polymer Embedding*
NASA	*National Aeronautics and Space Administration*
NIC	*Network Interface Controllers*
NMR	*Nuclear Magnetic Resonance*
NOW	*Network of Workstations*
MCL	Mutação de Cadeia Lateral
MIMD	*Multiple Instruction stream Multiple Data stream*
MISD	*Multiple Instruction stream Single Data stream*
MPI	*Message Passing Interface*
MPP	*Massive Parallel Processors*
MSC	Mutação Sem Colisão
MSM	Mutação Sempre Melhor
PDB	*Protein Data Bank*
PDP	Problema de Dobramento de Proteínas
PE	Programação Evolucionária
PGAs	*Parallel Genetic Algorithms*
PH	*Perturbed Homopolymer*
RE	Retículo Endoplasmático
SA	*Simulated Annealing*
SC	*Side Chain*
SIMD	*Single Instruction stream Multiple Data stream*
SISD	*Single Instruction stream Single Data stream*
SOM	*Self-Organizing Map*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
UniProtKB	*Universal Protein Knowledgebase*
VLSI	*Very-Large-Scale-Integration*

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

As proteínas são estruturas vitais para os seres vivos, pois são responsáveis por desempenhar funções estruturais (ou plásticas, por exemplo, colágeno – constituinte das cartilagens), de transporte (por exemplo, a hemoglobina carrega oxigênio até as células), de defesa (por exemplo, anticorpos), enzimáticas (por exemplo, as lipases transformam os lipídeos em ácidos graxos), entre outras.

Proteínas são polímeros compostos por uma cadeia de aminoácidos, também chamados de resíduos, que são ligados linearmente através de ligações peptídicas.

As proteínas, à medida que são formadas no ribossomo vão se dobrando sobre si, originando uma conformação tridimensional única, também conhecida como conformação nativa. Este processo é conhecido como dobramento de proteínas. A função biológica de uma proteína depende da sua estrutura tridimensional que, por sua vez, está estritamente ligada à sua estrutura primária, isto é, da sequência de aminoácidos que a compõe.

Sabe-se que proteínas mal-formadas (devido a um dobramento errôneo) podem originar diversas enfermidades, como o mal de Alzheimer, alguns tipos de câncer, fibrose Cística, a encefalopatia espongiforme bovina (doença da vaca louca), entre outras. Devido à importância do dobramento de proteínas para a medicina e a bioquímica, pesquisadores têm se concentrado no estudo deste processo e, consequentemente, gerado uma quantidade considerável de informações disponíveis para a comunidade científica. Portanto, adquirir conhecimento sobre a estrutura tridimensional de proteínas e, consequentemente, sobre a sua função é muito importante para o desenvolvimento de novas drogas com funcionalidade específica. Graças aos inúmeros projetos de sequenciamento genômico no mundo, uma grande quantidade de proteínas tem sido descoberta. No entanto, apenas uma pequena porção delas possui estrutura tridimensional conhecida. Neste contexto, o repositório de sequências de proteínas

UniProtKB/TrEMBL (LEINONEN et al., 2004) possui aproximadamente 10,9 milhões de registros, enquanto o Protein Data Bank – PDB (BERMAN et al., 2000) possui a estrutura de apenas 65378 proteínas (maio/2010).

Isto se deve à dificuldade envolvida no dobramento de proteínas tanto do ponto de vista bioquímico quanto computacional. A ciência da computação desempenha um papel importante nisto, desenvolvendo modelos computacionais e soluções para o problema de dobramento de proteínas (PDP).

Atualmente, a simulação de modelos computacionais que levam em consideração todos os átomos de uma proteína são inviáveis computacionalmente. Consequentemente, diversos modelos que abstraem a estrutura real da proteína foram propostos. O modelo mais simplificado para o estudo do dobramento de proteínas é conhecido como modelo Hidrofóbico-Polar (*Hydrophobic-Polar*–HP), nas versões bi (2D-HP) e tridimensional (3D-HP) (DILL et al., 1995). Contudo, a abordagem computacional para este modelo leva a um problema *NP*-difícil (BERGER; LEIGHTON, 1998). Este fato enfatiza a necessidade de se utilizar métodos heurísticos para lidar com o problema. Neste cenário, métodos de computação evolucionária têm se mostrado muito eficientes e, dentre estes, os algoritmos genéticos (AGs) têm se destacado (LOPES, 2008).

Neste trabalho é utilizado o modelo de representação computacional tridimensional Hidrofóbico-Polar com cadeia lateral (3DHP-SC). Este modelo é uma extensão do modelo HP que possui maior expressividade biológica e, consequentemente, aumenta o realismo da simulação. Entretanto, tem sido pouco abordado devido ao elevado grau de complexidade envolvido.

1.2 OBJETIVOS

Este trabalho tem por objetivo principal desenvolver um algoritmo genético paralelo hierárquico para o Problema de Dobramento de Proteínas (PDP), utilizando o modelo de representação tridimensional Hidrofóbico-Polar com cadeia lateral (3DHP-*Side Chain*). Os objetivos específicos do trabalho são:

- Elaborar um estudo sobre a fundamentação teórica do processo de dobramento de proteínas;
- Implementar um Algoritmo Genético para o problema de predição da estrutura terciária de proteínas utilizando o modelo 3DHP-SC, desenvolvendo operadores

genéticos, estratégias e função de *fitness* específicos;

- Realizar experimentos com o objetivo de avaliar a influência dos parâmetros do AG na predição de estruturas de proteínas;
- Realizar experimentos para determinar uma matriz de energia livre para o modelo 3DHP-SC;
- Estudar e desenvolver abordagens paralelas do Algoritmo Genético em *cluster* Beowulf;
- Realizar comparações com outra abordagem de computação evolucionária, chamada de Algoritmo de Colônia Artificial de Abelhas;
- Propor um conjunto de sequências de *benchmark* para o modelo 3DHP-SC.

1.3 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada em cinco capítulos divididos da seguinte forma. O Capítulo 2 faz uma revisão da literatura, explicando os conceitos básicos a respeito de proteínas, o Problema de Dobramento de Proteínas, computação evolucionária e computação paralela, bem como os trabalhos correlatos. O Capítulo 3 descreve detalhadamente o algoritmo desenvolvido. Os experimentos realizados juntamente com os resultados obtidos são expostos no Capítulo 4. Concluindo, o Capítulo 5 apresenta a discussão sobre os resultados, as conclusões e as propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 FUNDAMENTOS DE BIOQUÍMICA

2.1.1 Interações em meio aquoso

As células são compostas por água, íons inorgânicos e moléculas orgânicas. A molécula mais abundante nas células é a água, perfazendo, aproximadamente, 70% da massa total das células. Consequentemente, as interações entre a água e outros constituintes das células são de central importância para a bioquímica (COOPER, 2000).

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 uma molécula polar (BERG; TYMOCZKO; STRYER, 2002). Devido à sua natureza polar, moléculas de água podem formar pontes de hidrogênio com outras moléculas polares, como também com íons carregados. Como resultado destas interações, íons e moléculas polares são solúveis em água. Entretanto, moléculas apolares não são capazes de interagir com o meio aquoso.

2.1.2 Aminoácidos

Os aminoácidos são as unidades estruturais que compõem as proteínas. Cada aminoácido é caracterizado pela existência de um átomo de um carbono central (ou carbono α – Cα) ao qual estão ligados um átomo de hidrogênio, os grupos químicos amina (NH_2) e carboxílico (COOH), e uma cadeia lateral (também chamada de radical ou grupo R) que define a função do aminoácido. Dois aminoácidos formam uma ligação peptídica quando o grupo carboxílico de um deles reage com o grupo amina do outro. As propriedades químicas específicas da cadeia lateral do aminoácido determinam o papel de cada aminoácido na estrutura e função da proteína (COOPER, 2000).

A estrutura básica dos aminoácidos é apresentada na Figura 1. Nesta figura, pode-se observar que o Cα é o átomo de carbono que faz a ligação entre os dois grupos químicos da molécula e a cadeia lateral, representada pelo caractere R. A presença do Cα passa a nomear os aminoácidos que o possuem de α-aminoácidos (HUNTER, 1993).

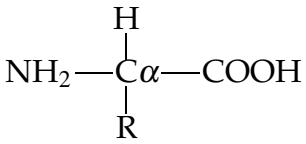


Figura 1: Estrutura básica de um α-aminoácido. O elemento R ligado ao Cα é diferente para cada aminoácido.

Fonte: Autoria própria

Na natureza existem inúmeros aminoácidos, porém apenas 20 deles são proteinogênicos. O primeiro a ser descoberto foi a asparagina, em 1806. O último, treonina, só foi identificado em 1938 (NELSON; COX, 2008). A lista dos 20 aminoácidos proteinogênicos com suas abreviações padronizadas e fórmulas químicas pode ser consultada no Anexo A.

Para entender as estruturas e funções das proteínas, é de fundamental importância ter conhecimento sobre as propriedades dos aminoácidos, determinadas pela sua cadeia lateral. Assim, os aminoácidos podem ser agrupados em quatro categorias: apolares, polares neutros, básicos ou ácidos (COOPER, 2000). Dez aminoácidos possuem cadeias laterais apolares (hidrofóbicas), isto é, que não são capazes de interagir com o meio aquoso e, portanto, tendem a se posicionar no interior das proteínas. Esta categoria inclui a Glicina, Alanina, Valina, Leucina, Isoleucina, Prolina, Cisteína, Metionina, Fenilalanina e Triptofano. A Glicina é o aminoácido mais simples, pois possui apenas um átomo de hidrogênio como cadeia lateral. A Alanina, Valina, Leucina e a Isoleucina possuem cadeias laterais hidrofóbicas formadas por até quatro átomos de carbono.

Cinco aminoácidos possuem cadeias laterais polares eletricamente neutras. Esta categoria inclui a Serina, Treonina e a Tirosina, que possuem um grupo hidroxila como cadeia lateral, como também a Aspargina e a Glutamina, que possuem um grupo amida polar como cadeia lateral. Devido à capacidade de formação de pontes de hidrogênio com água destas cadeias laterais, estes aminoácidos são polares (hidrofílicos), isto é, que são capazes de interagir com o meio aquoso (que é polar) e, tendem a se posicionar no exterior das proteínas.

Os aminoácidos Lisina, Arginina e Histidina possuem grupos básicos como cadeia lateral. Lisina e Arginina são aminoácidos altamente básicos com cadeias laterais carregadas positivamente. Consequentemente, são aminoácidos muito hidrofílicos e podem ser encontrados em contato com água na superfície de proteínas.

Finalmente, dois aminoácidos, Ácidos aspártico e glutâmico, possuem cadeias laterais com grupos carboxílicos. Estes aminoácidos são muito hidrofílicos e usualmente localizados na superfície de proteínas.

De acordo com esse comportamento em meio aquoso, pode-se concluir que a polaridade da cadeia lateral rege o processo de formação da estrutura tridimensional de proteínas (LODISH et al., 2000).

2.1.3 Proteínas

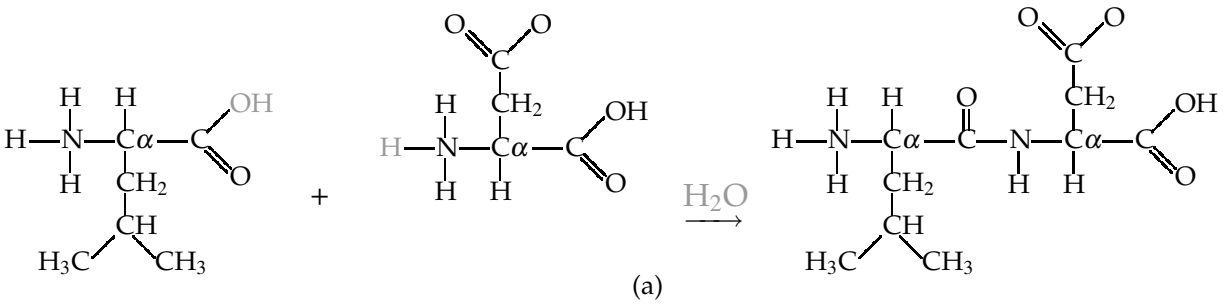
Proteínas são as estruturas básicas formadoras de todos os seres vivos (HUNTER, 1993). Do ponto de vista químico, as proteínas são as moléculas estruturalmente mais complexas e funcionalmente sofisticadas (ALBERTS et al., 2002).

Do ponto de vista da biologia molecular, uma proteína é um polímero formado por uma cadeia de aminoácidos que são ligados linearmente através de ligações peptídicas.

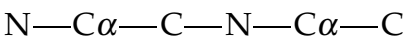
Na formação de uma ligação peptídica, dois aminoácidos são conectados entre si quando o grupo carboxila de um deles é condensado com o grupo amina do outro, da seguinte maneira. O grupo carboxila (COOH) de um dos aminoácidos e o grupo amina (NH_2) do outro são quimicamente modificados, perdendo um grupo hidroxila (OH) e um átomo de hidrogênio (H), respectivamente. Desta maneira, ocorre a ligação entre o carbono do grupo carboxila de um aminoácido com o nitrogênio do grupo amina do outro. Este processo de agregação de aminoácidos é conhecido como desidratação por liberar uma molécula de água (GRIFFITHS et al., 2000). Os grupos OH e H liberados pelos aminoácidos na formação da ligação peptídica se unem formando uma molécula de água (H_2O). A molécula formada pela ligação de dois aminoácidos é chamada de dipeptídeo. Quando vários aminoácidos são conectados, o produto resultante é chamado de polipeptídeo. Como uma proteína é composta por vários aminoácidos interconectados, ela pode ser considerada uma cadeia polipeptídica.

A Figura 2(a) apresenta um exemplo de ligação peptídica entre um aminoácido hidrofóbico (Leucina) e um hidrofílico (ácido aspártico). Pode-se observar que as unidades de aminoácidos que formam um peptídeo são chamados de resíduos de ami-

noácidos, pois estes aminoácidos não possuem mais a mesma estrutura original, já que perderam um átomo de hidrogênio do seu grupo amina e uma porção do grupo carboxila. O aminoácido que perde uma hidroxila é chamado de resíduo terminal amina ou terminal *N* e o aminoácido que perde uma porção do grupo carboxila é chamado de resíduo terminal carboxila ou terminal *C* (NELSON; COX, 2008).



(a)



(b)

Figura 2: Exemplo de ligação peptídica. Leucina + Aspartato = Leu-Asp + H_2O
Fonte: Autoria própria

Este processo ocorre ao longo da sequência de aminoácidos, formando uma sucessão de ligações peptídicas que gera uma "cadeia principal" ou *backbone* (ou esqueleto), ver Figura 2(b), a partir do qual as cadeias laterais são projetadas (ALBERTS et al., 2002; BRANDEN; TOOZE, 1999). Os átomos do *backbone* são o Cα, ao qual a cadeia lateral é conectada, um grupo NH limitado pelo Cα e um grupo carboxila (C==O) conectado ao Cα.

2.1.4 Estrutura hierárquica e função de proteínas

A organização estrutural de proteínas é comumente descrita em quatro níveis de complexidade (LODISH et al., 2000; GRIFFITHS et al., 2000; NELSON; COX, 2008; COOPER, 2000), em que as superiores englobam as propriedades das inferiores:

1. Estrutura primária:

   Este nível de organização estrutural visa representar apenas a sequência linear de aminoácidos que compõe a proteína ou polipeptídeo. Esse é o nível de organização mais simples, pois representa apenas as ligações peptídicas entre os aminoácidos. A sequência de aminoácidos é convencionalmente escrita do seu terminal *N* (à esquerda) para o seu terminal *C* (à direita). De acordo com o Anexo A, a estrutura primária pode ser representada com o nome dos aminoácidos por extenso, o código de três letras ou o código de uma letra. A Figura 3(a) apresenta uma sequência fictícia com a sua estrutura primária representada com o nome dos aminoácidos por extenso (Figura 3(b)), o código de três letras (Figura 3(c)) e código de uma letra (Figura 3(d)).

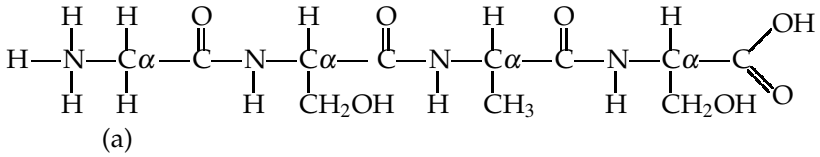


Figura 3: Exemplo de estrutura primária

Fonte: Autoria própria

2. Estrutura secundária:

   Este nível de organização estrutural consiste em arranjos espaciais de partes locais de uma cadeia polipeptídica. Existem basicamente três estruturas secundárias importantes: α-hélices (PAULING; COREY; BRANSON, 1951a), β-folhas (PAULING; COREY; BRANSON, 1951b) e *turns* (dobras) (LEWIS; MOMANY; SCHERAGA, 1973).

   A α-hélice é a forma mais comum de estrutura secundária (Figura 4(a)). Nesta estrutura, o *backbone* está enrolado em torno do eixo de uma hélice imaginária (espiral) e as cadeias laterais (grupos R) dos resíduos de aminoácido projetam-se para fora do *backbone* helicoidal. As α-hélices são formadas por 3,6 resíduos

por volta completa da espiral e estabilizadas por pontes de hidrogênio entre os grupos NH e CO do *backbone*.

A β-folha é formada por dois ou mais segmentos polipeptídicos da mesma molécula, ou de moléculas diferentes, dispostos lateralmente e estabilizados por pontes de hidrogênio entre os grupos NH e CO (Figura 4(b)). Polipeptídeos adjacentes em uma β-folha podem ter a mesma direção (β-folha paralela) ou direções opostas (β-folha antiparalela). Funcionalmente, as β-folhas antiparalelas estão presentes nos mais diversos tipos de proteínas, por exemplo, incluem enzimas, proteínas de transporte, anticorpos e células de superfície de proteínas (BRANDEN; TOOZE, 1999).

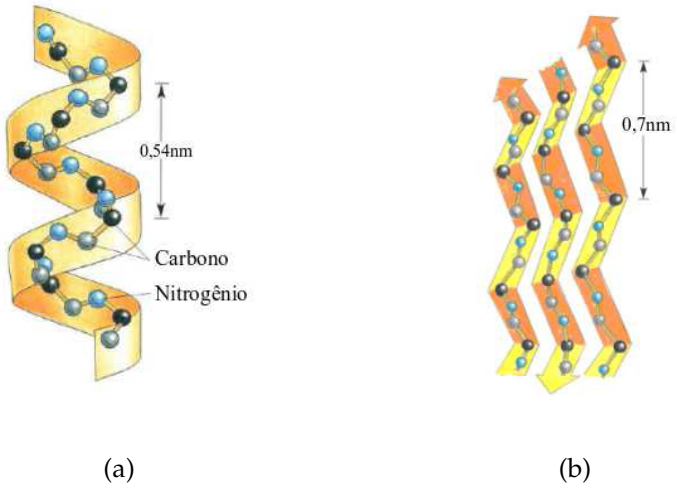


Figura 4: Estrutura de uma α-hélice (a) e de uma β-folha (b)
Fonte: Adaptado de (ALBERTS et al., 2002)

Turns são compostos de três ou quatro resíduos de aminoácidos e geralmente são localizados na superfície das proteínas formando dobras que redirecionam a cadeia polipeptídica para o interior da proteína. *Turns* permitem que proteínas grandes sejam dobradas em estruturas altamente compactas.

As estruturas secundárias podem ser associadas formando estruturas supersecundárias, chamadas de *motifs* (motivos) (BRANDEN; TOOZE, 1999; GRIFFITHS et al., 2000; NÖLTING, 2006). Os *motifs* são padrões frequentemente encontrados em estruturas tridimensionais. Alguns deles podem ser associados a funções específicas – por exemplo, o *motif* hélice-*turn*-hélice (HTH) está presente em proteínas regulatórias para DNA-*binding* (BRENNAN; MATTHEWS, 1989) e proteínas de ligação de cálcio (KRETSINGER, 1980).

3. Estrutura terciária:

   A estrutura terciária diz respeito à conformação de uma cadeia polipeptídica, isto é, o arranjo tridimensional dos resíduos de aminoácidos. Enquanto as estruturas secundárias são estabilizadas por pontes de hidrogênio, as estruturas terciárias são estabilizadas por interações entre cadeias laterais hidrofóbicas e pontes de hidrogênio entre cadeias laterais polares. A estrutura terciária representa o dobramento de um polipeptídeo como resultado de interações entre as cadeias laterais dos aminoácidos que se encontram em diferentes regiões da estrutura primária. A Figura 5(a) apresenta um exemplo de estrutura terciária, onde se pode observar a presença de duas estruturas secundárias: α-hélice e β-folha.

4. Estrutura quaternária:

   A estrutura quaternária é o nível de representação mais complexo e descreve o arranjo de duas ou mais subunidades polipeptídicas dobradas (estruturas terciárias) no espaço.

   A associação quaternária pode ser entre diferentes tipos de polipeptídeos (heterodímero) ou entre polipeptídeos idênticos (homodímero). Este nível de organização descreve o número e posições relativas de subunidades em proteínas multiméricas. A Hemoglobina é um exemplo de proteína multimérica, pois é composta de duas cópias de polipeptídeos diferentes que interagem entre si (Figura 5(b)).

   Diversas proteínas possuem estrutura compacta. Elas são chamadas de proteínas globulares – por exemplo, algumas enzimas e os anticorpos estão entre as proteínas globulares mais conhecidas. Proteínas com forma linear, chamadas de proteínas fibrosas, são componentes importantes em queratina dos cabelos ou miosina dos músculos.

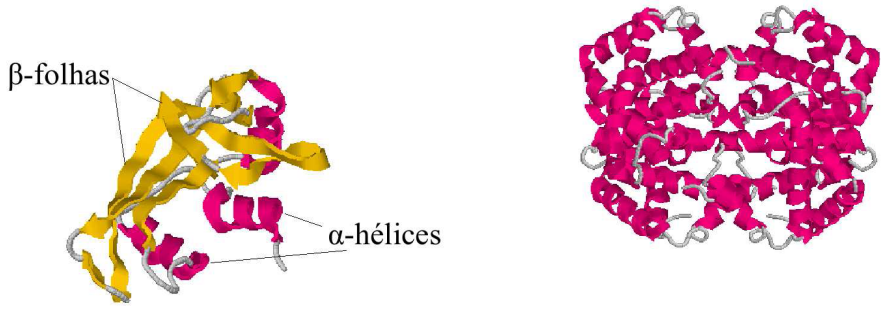
   As Figuras 5(a) e 5(b) foram obtidas pelo RASMOL[1] a partir de arquivos PDB[2].

2.1.5 Dobramento de proteínas

O dobramento de proteínas, ou *protein folding*, consiste no processo pelo qual cadeias polipeptídicas se convertem em estruturas compactas que exercem funções biológicas (PAIN, 2000). Estas funções incluem o controle e regulação de processos químicos

[1] O RASMOL é um aplicativo de visualização gráfica molecular utilizado por pesquisadores e cientistas para visualizar imagens de macromoléculas. Este aplicativo está disponível em: http://www.rasmol.org

[2] Os arquivos PDB estão disponíveis em: http://www.wwpdb.org



(a) (b)

Figura 5: Estrutura Terciária da Ribonuclease-A e estrutura quaternária da Hemoglobina (b). Fonte: Autoria própria

essenciais para para os organismos vivos (BRANDEN; TOOZE, 1999).

As proteínas, ao serem formadas no ribossomo, adquirem a forma tridimensional à medida que vão sendo sintetizadas. A estrutura tridimensional mais estável sob condições fisiológicas é chamada de conformação nativa e permite que uma proteína realize a sua função biológica (LODISH et al., 2000; PEDERSEN, 2000).

Os experimentos *in vitro* realizados por Christian Anfinsen e colaboradores (ANFINSEN, 1957; ANFINSEN et al., 1961; ANFINSEN, 1973) mostram que uma proteína pode ser desnaturada, ou seja, desdobrada através de modificações no meio onde ela se encontra. A maioria das proteínas podem ser desnaturadas com a mudança da temperatura, pH e teor de alguma substância desnaturante na solução (por exemplo, a uréia), afetando interações fracas (por exemplo, pontes de hidrogênio). Durante o processo de desnaturação, os polipeptídeos perdem a sua conformação nativa e, consequentemente, a sua função biológica.

Anfinsen demonstrou que as proteínas desnaturadas se dobram novamente em sua conformação nativa independentemente da conformação em que se encontrava (COOPER, 2000; CHANDRU; DATTASHARMA; KUMAR, 2003). Entretanto, o retorno da proteína à sua conformação nativa espontaneamente só é válido em proteínas de domínio-único (*single-domain proteins*) (PONTIN; RUSSELL, 2002). Domínios são regiões estáveis de uma proteína que possuem funções e que podem se dobrar de maneira autônoma (WETLAUFER, 1973). Em geral, os domínios apresentam restrições em seus tamanhos, variando de 36 a mais de 500 resíduos (SAVAGEAU, 1986; IS-

LAM; LUO; STERNBERG, 1995). Proteínas de domínio-único possuem sequências de aproximadamente 100 resíduos, podendo variar de 30 a 400 resíduos (HARTL, 1996). Proteínas maiores formam múltiplos domínios, onde cada um pode se dobrar independentemente. Em contraste com as proteínas de domínio-único, existe uma grande probabilidade de que o redobramento não seja possível nas proteínas de múltiplos-domínios, ou seja, a desnaturação é irreversível. Um exemplo clássico deste processo é a desnaturação da clara do ovo. A clara do ovo é composta de água e albumina. A albumina é uma proteína polar, portanto solúvel em água. Ao esquentar um ovo, provoca-se a desnaturação da albumina que, mesmo após retornar à temperatura ambiente, não consegue voltar à sua conformação nativa. Nesse caso, a conformação se torna insolúvel em água.

Através dos experimentos de Anfinsen, verificou-se que as proteínas possuem apenas uma única estrutura nativa e que as informações essenciais que codificam esta estrutura estão contidas na sequência de aminoácidos, pois estas proteínas podem alcançar a sua estrutura nativa *in vitro* na ausência de elementos auxiliares (CHANDRU; DATTASHARMA; KUMAR, 2003; ANFINSEN, 1973). Em outras palavras, a estrutura tridimensional nativa é baseada na estrutura primária da proteína.

Apesar da maioria das proteínas se dobrarem espontaneamente em sua conformação nativa, outras requerem da assistência de uma classe de proteínas chamada de chaperonas (BRANDEN; TOOZE, 1999; HARTL, 1996). As chaperonas estão localizadas em todos os tipos de células e compartimentos celulares e podem ser classificadas em duas famílias. A primeira família é composta de chaperonas moleculares, as quais basicamente auxiliam no dobramento de polipeptídeos incompletos ou desnaturados (desdobrados), impedindo que estes sofram a sua degradação no meio celular ou o dobramento errôneo (mal-dobramento) (GETHING; SAMBROOK, 1992). As chaperonas moleculares são divididas em várias subclasses, sendo a mais estudada a família das proteínas de choque térmico (*Heat Shock Proteins* – HSP) HSP70, cujo nível aumenta em células submetidas a condições de estresse como, por exemplo, temperaturas elevadas (MAYER; BUKAU, 2005).

A segunda familia inclui as chaperoninas, que são subunidades de proteína arranjadas em dois anéis empilhados formando um complexo cilíndrico que promovem o dobramento de proteínas no ambiente de sua cavidade central. Estas atuam basicamente como facilitadoras do processo de dobramento (ELLIS, 1996).

As chaperonas moleculares e as chaperoninas estão localizadas no citosol e em

organelas subcelulares – por exemplo, retículo endoplasmatico (RE), mitocôndrias e cloroplastos.

Chaperonas também auxiliam no processo de transporte de proteínas para organelas subcelulares – por exemplo, polipeptídeos desdobrados são transportados do citosol para a mitocôndria. Os polipeptídeos são transportados através da membrana mitocondrial em conformações desdobradas. As chaperonas citosólicas estabilizam a configuração desdobrada do polipeptídeo e as chaperonas mitocondriais facilitam o transporte e subsequente dobramento da cadeia polipeptídica dentro da organela.

A identificação de mutações responsáveis por uma grande variedade de doenças hereditárias tem sido muito acelerada nos últimos anos, devido à disponibilidade de ferramentas de mapeamento genético desenvolvidas nos projetos de sequenciamento genômico. Estudos abrangendo estas mutações tem revelado que alterações genéticas nas chaperonas e, consequentemente, a perda das suas funções bem definidas, dobramento errôneo e agregação de proteínas podem ser a causa de várias enfermidades (MACARIO; MACARIO, 2002) como, por exemplo, mal de Alzheimer (HUTTON et al., 2001; SELKOE, 2001), alguns tipos de câncer (BELL et al., 2002; DAWSON et al., 2003; ISHIMARU et al., 2003), fibrose cística (THOMAS; KO; PEDERSEN, 1992), arteriosclerose (URSINI et al., 2002), mal de Parkinson (MCNAUGHT et al., 2001), entre outros. Portanto, adquirir conhecimento sobre o processo de dobramento de proteínas e, consequentemente, da estrutura tridimensional e função de proteínas é de fundamental importância para a medicina/bioquímica, no que concerne ao desenvolvimento de novas drogas com funcionalidade específica (PEDERSEN, 2000). É importante ressaltar que apesar de uma grande quantidade de proteínas ter sido descoberta pelos projetos de sequenciamento genômico conduzidos no mundo, apenas uma pequena porção delas possui estrutura tridimensional conhecida. Neste contexto, o repositório de sequências de proteína UniProtKB /TrEMBL(LEINONEN et al., 2004) possui aproximadamente 10,9 milhões de registros de sequências de proteínas e o *Protein Data Bank* – PDB(BERMAN et al., 2000) possui a estrutura de apenas 65378 proteínas (maio/2010).

Em termos práticos, as informações detalhadas sobre a estrutura tridimensional de proteínas são obtidas através de técnicas experimentais que usam cristalografia de raios-X (DRENTH, 1999; SUNDE; BLAKE, 1997), ressonância nuclear magnética (NMR – *Nuclear Magnetic Resonance*) (WÜTHRICH, 1986; JARONIEC et al., 2004) e microscopia crio-eletrônica (cryo-ME)(FRANK, 2006; JIMENEZ et al., 2002).

A discrepância entre os repositórios UniProtKB/TrEMBL e PDB se deve ao custo

elevado e dificuldade envolvida no dobramento de proteínas, do ponto de vista bioquímico e biológico. A Ciência da Computação desempenha um papel importante nisto, propondo e desenvolvendo modelos e soluções computacionais para o estudo do Problema de Dobramento de Proteínas (PDP) (LOPES, 2008). Consequentemente, diversos modelos que abstraem a estrutura real da proteína foram propostos. Tais modelos, embora irreais, utilizam algumas propriedades bioquímicas dos aminoácidos, e apresentam algumas características interessantes e úteis para se observar o comportamento de proteínas sintéticas.

2.2 PREDIÇÃO DA ESTRUTURA TERCIÁRIA DE PROTEÍNAS

Apesar do considerável esforço teórico-experimental despendido ao estudo do dobramento de proteínas, ainda não há uma descrição detalhada sobre os mecanismos que regem o dobramento de uma proteína. Atualmente, há basicamente dois problemas relacionados ao processo de dobramento de proteínas. O Problema da Predição da Estrutura terciária de Proteínas (ou *Protein Structure Prediction* – PSP) pode ser definido como a determinação da estrutura tridimensional final de uma proteína utilizando apenas informações da sua estrutura primária, ou seja da sequência de aminoácidos que a compõe. Por outro lado, o Problema de Dobramento de Proteínas (PDP ou *Protein Folding Problem* – PFP) corresponde à descoberta dos comportamentos assumidos pela cadeia ao se dobrar para a sua conformação nativa, durante a síntese. Contudo, na literatura, estes termos são frequentemente utilizados sem nenhuma distinção, significando apenas o primeiro problema (LOPES, 2008).

Tendo surgido no contexto da biologia molecular, este problema é hoje claramente interdisciplinar, necessitando de ferramentas de várias áreas do conhecimento. Este problema representa um dos desafios abertos mais importantes da biologia e bioinformática (NICOSIA; STRACQUADANIO, 2008). Modelos computacionais foram desenvolvidos com o objetivo de abstrair características reais das proteínas, em um dado nível de detalhamento que permitam a representação fidedigna da estrutura tridimensional, sem a perda de viabilidade computacional.

Acredita-se que a conformação nativa de uma proteína é a sua estrutura mais estável, estando em um estado de energia livre mínima, o que gerou a chamada Hipótese da Termodinâmica (PEDERSEN, 2000) (ver Seção 2.2.1). Os modelos são, comumente, concebidos baseando-se nas leis da termodinâmica onde o problema da predição da estrutura de proteínas é modelado como um problema de minimização da energia li-

vre a respeito das possíveis conformações que uma proteína pode assumir. Assume-se que a minimização da energia livre é o fator mais importante para a formação da estrutura de uma proteína. Assim, a conformação nativa de uma proteína é definida como aquela com energia livre mínima (LOPES, 2008).

De acordo com (PEDERSEN, 2000), um modelo computacional que obedece a este princípio deve cumprir as seguintes características:

- Um modelo da proteína, definido por um conjunto de entidades representando átomos e ligações entre eles;
- Um conjunto de regras definindo as possíveis conformações da proteína;
- Uma função de avaliação da energia livre de cada conformação possível da proteína modelada que seja computacionalmente viável

Modelos de representação de polipeptídeos são apresentados na Seção 2.3. O modelo 3DHP-SC é apresentado na Seção 2.3.4.

2.2.1 Teoria da hipótese termodinâmica

Acredita-se que a conformação nativa de proteínas obedece à teoria da Hipótese Termodinâmica (HT), apresentada por Anfinsen [3] (ANFINSEN, 1973). Esta teoria é baseada na observação de que o processo de dobramento de proteínas ocorre de uma forma espontânea, onde os polipeptídeos tendem a assumir a estrutura tridimensional mais estável, em um estado global de mínima energia livre, que se acredita ser a força mais importante durante o processo de dobramento (PEDERSEN, 2000). Esta estrutura é chamada de conformação nativa, previamente citada na Seção 2.1.5 - página 34, e é estável termodinamicamente em relação ao meio onde se encontra, isto é, o polipeptídeo continua na mesma forma dobrada durante um tempo indefinido, na ausência de perturbações externas.

A partir desta teoria, sabendo-se que a estrutura tridimensional da proteína é baseada na sua estrutura primária (citado na Seção 2.1.5 - página 35), modelos discretos foram desenvolvidos permeando o objetivo de determinar a conformação nativa partindo-se da estrutura primária. Isto é realizado do ponto de vista termodinâmico, simulando o espaço conformacional da proteína utilizando uma função de energia livre, também chamada de energia livre de Gibbs (BOERIO-GOATES, 2000). A energia

[3] Prêmio Nobel de Química em 1972

livre de Gibbs é um dos conceitos mais importantes da termodinâmica, define a espontaneidade de uma reação química, de acordo com a primeira e segunda leis da termodinâmica, aplicável para reações isotérmicas (NELSON; COX, 2008). A Equação 1 apresenta a função de energia livre de Gibbs (ΔG) e mostra que é definida pelos fatores de entalpia (H), entropia (S) e temperatura (T). A entalpia é uma grandeza física que representa a capacidade calorífica ou energia armazenada e reflete o número e tipos de ligações entre os átomos da molécula. Por outro lado, a entropia expressa a aleatoriedade e desordem dos componentes de um sistema.

$$\Delta G = \Delta H - T\Delta S \tag{1}$$

No contexto de estrutura de proteínas, a magnitude da variação de energia (ΔG) discrimina proteínas dobradas de mal-dobradas (NELSON; COX, 2008). Onde as proteínas mal-dobradas são caracterizadas por um elevado nível de entropia. Esta entropia e interações por pontes de hidrogênio entre vários grupos da cadeia polipeptídica com o solvente (água) tendem a manter o estado desdobrado. As interações químicas envolvidas no processo de dobramento que estabilizam a conformação nativa incluem pontes de hidrogênio, interações de Van Der Waals e interações hidrofóbicas (KAUZMANN, 1959; DILL, 1990). As interações hidrofóbicas têm um papel muito superior às outras forças na formação do dobramento de uma proteína. Estas interações contribuem para a formação de conformações compactas com um núcleo hidrofóbico no interior das proteínas (DILL, 1999; NELSON; COX, 2008). Por este motivo, este trabalho objetiva a obtenção de conformações que maximizem o número de interações entre resíduos hidrofóbicos.

2.2.2 O paradoxo de Levinthal

Cyrus Levinthal propôs uma objeção sobre a idéia de que a procura da conformação nativa possa ser realizada utilizando a busca aleatória, que no pior caso se torna uma busca exaustiva, passando por todas as conformações possíveis da proteína.

O principal argumento de Levinthal baseia-se em experiências conceptuais (*gedanken experiment*), nas quais a complexidade e grandeza do espaço de busca podem ser estimadas a partir do tempo despendido para encontrar a conformação nativa de uma proteína. Este tempo é obtido pelo número de conformações da cadeia polipeptídica multiplicada pelo tempo necessário para encontrar uma configuração. Por exemplo,

considerando-se uma cadeia polipeptídica de 100 aminoácidos, onde cada aminoácido possui 5 movimentos possíveis (número bastante inferior aos números reais) e assumindo que o tempo hipotético necessário para encontrar uma conformação for de 10ps (dez picossegundos), o número total de conformações possíveis é igual a 5^{100} (ou 10^{70}) e o tempo estimado para encontrar a conformação nativa da proteína seria de $10^{70} \cdot 10 \times 10^{-12}s= 10^{59}$ segundos $\simeq 10^{52}$ anos(KARPLUS, 1997), o que é absurdamente maior que a idade do Universo, estimada na ordem de $1,4 \times 10^{10}$ anos. Entretanto, sabe-se que as proteínas se dobram em um tempo na ordem de milissegundos a segundos (SCHMID, 1992). Isto deu origem ao paradoxo de Levinthal (LEVINTHAL, 1968). Pode-se notar que a complexidade deste problema cresce com o aumento do número de aminoácidos da cadeia polipeptídica. Assim, este paradoxo também foi descrito na linguagem da complexidade computacional, demostrando que o problema da busca aleatória, proposto por Levinthal, é um problema *NP*-completo (NGO; MARKS; KARPLUS, 1994), isto é, que não pode ser resolvido em tempo polinomial.

2.3 MODELOS DE REPRESENTAÇÃO DE POLIPEPTÍDEOS

Há basicamente dois tipos de representação de polipeptídeos, a analítica e a discreta. A representação analítica descreve detalhadamente todas as informações sobre os átomos que compõem a proteína. Por outro lado, a representação discreta descreve uma proteína em um nível de detalhamento bastante reduzido. Os modelos analítico e discreto serão detalhados a seguir.

2.3.1 Modelo analítico

Este modelo apresenta o nível de detalhamento da estrutura de uma proteína mais completo possível e, consequentemente, é o mais complexo. Este modelo foi apresentado por (RICHARDSON, 1981) e (NGO; MARKS; KARPLUS, 1994). A proteína é considerada como uma coleção de átomos interconectados através de ligações. Assim, a estrutura tridimensional da proteína pode ser especificada utilizando ângulos, comprimentos e a torsão de cada ligação entre cada átomo ou grupos de átomos (por exemplo, aminoácidos) na sua estrutura. Consequentemente, esta é uma descrição bastante complexa que envolve informações, em nível atômico, sobre cada átomo da proteína.

Neste modelo, a função de energia livre é especificada através de termos que in-

dicam as contribuições de átomos ligados e não-ligados. Para os átomos ligados, os termos dependem dos comprimentos, ângulos e torções das ligações. Por outro lado, princípios físicos são utilizados para átomos não-ligados (por exemplo, forças de Coulomb e de Van der Waals) ou informações estatísticas inferidas de estruturas conhecidas (por exemplo, potenciais de força média).

Computacionalmente, o problema de dobramento de proteínas utilizando o modelo analitico é *NP*-difícil (NGO; MARKS; KARPLUS, 1994). Uma alternativa para reduzir a complexidade deste modelo é limitar o comprimento, ângulo e torsões das ligações para um conjunto de valores possíveis. Também alguns átomos podem ser omitidos do modelo, ou agrupados em unidades maiores e serem tratados como átomos individuais (PEDERSEN, 2000). Usualmente, estas limitações no modelo são obtidas a partir de estruturas de proteínas reais conhecidas (PEDERSEN; MOULT, 1997). Entretanto, estas limitações afetam o nível de detalhamento, diminuindo a equivalência visual com relação às conformações nativas reais.

2.3.2 Modelos discretos

Atualmente, a simulação de modelos computacionais que levem em conta todos os átomos de uma proteína é praticamente inviável computacionalmente, apesar da disponibilidade de recursos computacionais de alto desempenho. Consequentemente, pesquisadores tem desenvolvido vários modelos discretos (também chamados de simplificados) para abstrair as estruturas de proteínas, com o objetivo de encontrar soluções ótimas ou quase-ótimas para o PDP (CHANDRU; DATTASHARMA; KUMAR, 2003). Conforme citado na Seção 2.3.1, uma abordagem para obtenção de modelos simplificados é limitar a faixa de comprimentos, ângulos e torsões no modelo e utilizar um conjunto de valores predefinidos. Neste contexto, a classe de modelos mais simples para o PDP é chamada de *lattice models* ou modelos de treliça. Nesta classe, a proteína é modelada considerando-se apenas uma sequência de elementos simples, representando os resíduos da cadeia polipeptídica, posicionados em uma grade. Os ângulos de ligação possuem valores discretos definidos pela estrutura da grade, usualmente em um plano (2D) ou no espaço (3D).

Apesar da simplicidade dos modelos discretos, eles utilizam algumas propriedades bioquímicas e a sua simulação pode apresentar características interessantes e úteis para se observar o comportamento de proteínas (BENÍTEZ; LOPES, 2009; LOPES, 2008). Também permitem que o espaço conformacional possa ser explorado exten-

sivamente e podem servir como geradores de hipóteses que não podem ser obtidas de outra maneira, mas que podem ser reproduzíveis experimentalmente ou através de simulações refinadas (DILL, 1999). Por outro lado, o modelo analítico envolve muitos parâmetros e aproximações, fazendo com que a sua validade seja tão duvidosa quanto para os modelos discretos (CHANDRU; DATTASHARMA; KUMAR, 2003). Desta maneira, há uma importante motivação para o desenvolvimento de métodos computacionais para o estudo do PDP.

O modelo computacional mais simples para o PDP é o modelo Hidrofóbico-Polar (HP), nas versões bi (2D-HP) e tridimensional (3D-HP), apresentados a seguir.

2.3.3 Modelo Hidrofóbico-polar (HP)

O modelo Hidrofóbico-Polar (HP) proposto por Dill (DILL et al., 1995) divide os 20 aminoácidos proteinogênicos em duas classes, de acordo com a afinidade com o meio aquoso: Hidrofílicos (ou polares – P) e Hidrofóbicos (H) baseando-se em resultados experimentais. Neste modelo, a proteína é uma sequência de caracteres $S = s_1, \ldots, s_n$ com $s_i \in \{\mathrm{H}, \mathrm{P}\}$. Este é um modelo de treliça, onde as conformações devem estar embutidas em uma grade quadrada (bidimensional – 2DHP) ou cúbica (tridimensional – 3DHP), como apresentado nas Figuras 6(a) e 6(b). Ambos os modelos (2DHP e 3DHP) tem sido frequentemente explorados na literatura (SCAPIN; LOPES, 2007a; THACHUK; SHMYGELSKA; HOOS, 2007; LOPES, 2008; PATTON; III; GOODMAN, 1995; UNGER; MOULT, 1993b). Os pontos brancos e pretos representam os resíduos polares e hidrofóbicos, respectivamente. As ligações estão representadas pelas linhas pontilhadas.

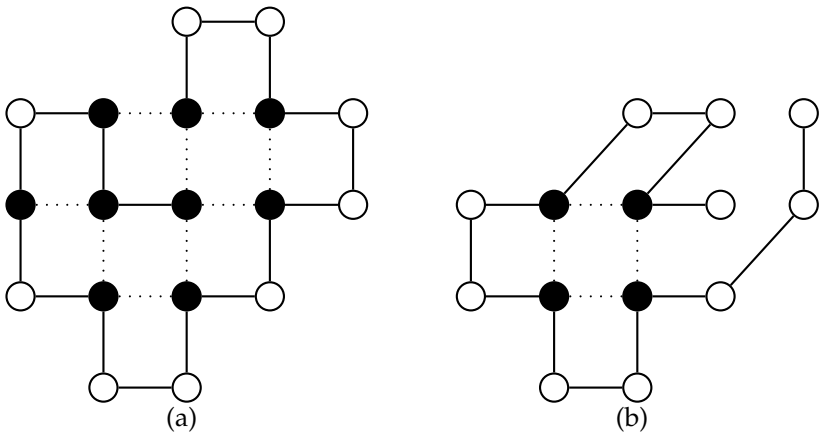


Figura 6: Exemplos de modelos 2DHP (a) e 3DHP (b).

Fonte: Autoria própria

Quando uma proteína é dobrada na sua conformação nativa, os resíduos hidrofóbicos tendem a se agrupar no interior da proteína, protegidos por resíduos polares posicionados no exterior. Portanto, um núcleo hidrofóbico é usualmente formado em uma proteína dobrada.

O objetivo deste modelo é a aplicação da Hipótese Termodinâmica, onde a energia livre deve ser minimizada através da maximização do número de interações hidrofóbicas. Em outras palavras, a energia livre e o número de interações hidrofóbicas são inversamente proporcionais. Cada conformação pode ser avaliada através de uma função de energia livre, sugerida por (LI et al., 1996), representada pela Equação 2:

$$E = \sum_{i<j} e_{v_i v_j} \delta(r_i - r_j) \tag{2}$$

Onde,
$\delta(r_i - r_j) = 1$, se os resíduos r_i e r_j formam uma ligação não-local.
$\delta(r_i - r_j) = 0$, caso contrário;
$e_{v_i v_j}$ é a energia que corresponde aos tipos de contatos entre os resíduos: e_{HH}, e_{HP} e e_{PP}, respectivamente, interações hidrofóbicas (hidrofóbico-hidrofóbico – H-H), hidrofóbica-polar (H-P) ou polares (ou hidrofílicas – P-P).

Este modelo considera que a interação entre os aminoácidos hidrofóbicos representa a contribuição mais significativa para a energia livre da proteína. Desta maneira, cada interação hidrofóbica possui um valor de energia ε, enquanto os outros tipos de interação possuem energia com valor δ ($e_{HH} = -\mid \varepsilon \mid$ e $e_{HP} = e_{PH} = e_{PP} = \delta$). Portanto, uma matriz de energia pode ser escrita da seguinte maneira:

$$E = \begin{array}{c} \\ \mathbf{H} \\ \mathbf{P} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{H} & \mathbf{P} \end{array} \\ \begin{pmatrix} -\varepsilon & \lambda \\ \lambda & \lambda \end{pmatrix} \end{array} \tag{3}$$

Onde,
$E(0,0) = e_{HH}$ corresponde a energia das interações hidrofóbicas (H-H);
$E(0,1) = e_{HP}$ corresponde a energia das interações H-P;
$E(1,0) = e_{PH}$ corresponde a energia das interações P-H;
$E(1,1) = e_{PP}$ corresponde a energia das interações polares (P-P);

A maneira mais simples de função de energia contabiliza o número de interações hidrofóbicas. O modelo envolve uma interação de atração (H-H, $\varepsilon = -1$) e três interações neutras (H-P, P-H, P-P, $\lambda = 0$), que tendem a produzir conformações compactas de energia livre mínima. Em outras palavras, a conformação nativa é aquela que maximiza o número de interações hidrofóbicas (H-H). Portanto, o procedimento algorítmico para o PDP que maximiza o número de interações hidrofóbicas encontrará, reciprocamente, a conformação com o estado de menor energia livre possível. Outra proposta usual para a matriz de energia é formada por três forças de atração (e_{HH}=-2,3 e e_{HP}=e_{PH}=-1) e uma força neutra (e_{PP}=0).

Há outras formas de codificação da matriz de energia livre que incluem interações de repulsão. Um modelo de proteínas que inclua *binding sites*, espaço vazio da grade rodeado pela cadeia polipeptídica dobrada, requer a introdução de forças de repulsão. Isto fornece uma maior contribuição no que diz respeito à relevância biológica do modelo, pois os *binding sites* estão presentes em um número significante de proteínas classificadas como funcionais (CHAN; DILL, 1996). Uma proposta de matriz de energia utilizando este conceito é chamada de modelo *shifted* HP e é formada por uma força de atração ($\varepsilon = -2$) e três de repulsão ($\lambda = 1$) (HIRST, 1999).

2.3.4 Modelo 3DHP com cadeia lateral (3DHP-SC)

Do ponto de vista biológico, os modelos 2DHP e 3DHP possuem pouca expressividade. Portanto, o próximo passo para simular características mais realistas das proteínas é incluir uma elemento representando a cadeia lateral (*Side Chain* – SC) dos aminoácidos (LI; KLIMOV; THIRUMALAI, 2002). Para isto, a proteína é representada por um *backbone* B, comum para todos os aminoácidos, e uma cadeia lateral, hidrofóbica (H) ou polar (P). Esta representação define o modelo 3DHP-SC, apresentado na Figura 7. Esta figura representa o dobramento de uma parte de uma proteína hipotética. Os resíduos hidrofóbicos e polares são representados, respectivamente, por esferas vermelhas e azuis. O *backbone* e as conexões entre aminoácidos são mostrados em cinza. Os tipos de interação também estão indicados na figura: cadeias laterais hidrofóbicas (HH), *backbone-backone* (BB), *backbone*-cadeia lateral hidrofóbica (BH), *backbone*-cadeia lateral polar (BP), cadeias laterais hidrofóbica-polar (HP), cadeias laterais polares (PP).

Ao contrário dos modelos 2DHP e 3DHP, o 3DHP-SC tem sido muito pouco explorado na literatura. Para este modelo, a energia livre de uma conformação leva em consideração a posição espacial da cadeia lateral, e pode ser descrita pela Equação 4

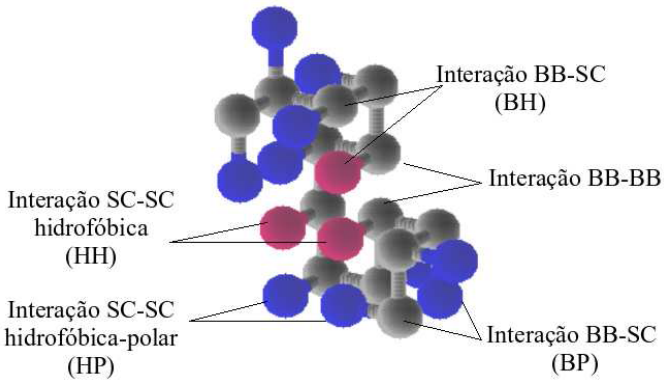


Figura 7: Exemplo do modelo 3DHP-SC

Fonte: Autoria própria

(LI; KLIMOV; THIRUMALAI, 2002):

$$H = \varepsilon_{bb} \sum_{i=1, j>i+1}^{N} \delta_{r_{ij}^{bb},a} + \varepsilon_{bs} \sum_{i=1, j\neq i}^{N} \delta_{r_{ij}^{bs},a} + \varepsilon_{ss} \sum_{i=1, j>i}^{N} \delta_{r_{ij}^{ss},a} \tag{4}$$

Onde ε_{bb}, ε_{bs} e ε_{ss} representam a ponderação na energia de cada tipo de interação possível: *backbone/backbone* (BB-BB), *backbone/side-chain* (BB-SC) e *side-chain/side-chain* (SC-SC). Em uma cadeia de N aminoácidos, r_{ij}^{bb}, r_{ij}^{bs} e r_{ij}^{ss} são as distâncias (no espaço tridimensional) entre o i-ésimo e o j-ésimo resíduos das interações BB-BB, BB-SC e SC-SC, respectivamente. O operador $\delta_{r_{ij},a}$ retorna 1 quando a distância entre o i-ésimo e o j-ésimo elementos é igual à constante a. Caso contrário, retorna 0.

2.3.5 Complexidade computacional do modelo HP

Apesar da simplicidade dos modelos de treliça, foi provado que o PDP para estes modelos é *NP*-completo, ou seja, não há algoritmo que resolva este problema em tempo polinomial. Isto foi demonstrado por (CRESCENZI et al., 1998) para o modelo 2DHP e por (BERGER; LEIGHTON, 1998) para o modelo 3DHP. Consequentemente, o modelo 3DHP-SC pode ser considerado um problema *NP*-Completo, pois é razoavelmente mais complexo de ser resolvido do que o 3DHP.

A Equação 5 representa o número de combinações de diferentes dobramentos que podem ser gerados para um polipeptídeo de n aminoácidos (utilizando a forma de representação apresentada na Seção 3.1 – página 69). Pode-se observar que o aumento linear do número de aminoácidos leva a um aumento exponencial de número de do-

bramentos possíveis. Portanto, o problema torna-se intratável, mesmo com poucos aminoácidos na cadeia, como exemplificado na Tabela 1. Esta tabela mostra o número de estados possíveis e uma estimativa do tempo de processamento, considerando uma situação hipotética de processar um estado a cada nanosegundo.

$$n_{dobramentos} = 25^{(n-1)} \tag{5}$$

Nesta equação, $n_{dobramentos}$ representa o número de dobramentos possíveis e n é o número de aminoácidos do polipeptídeo. É importante ressaltar que o expoente é $(n-1)$, pois o primeiro aminoácido da cadeia está fixo em seu lugar.

Tabela 1: Número de combinações e estimativa de tempo de processamento de acordo com o número de aminoácidos.

Nº de aminoácidos	Nº de dobramentos possíveis	Tempo estimado para explorar todo o espaço de busca
2	25	$25{,}00 \cdot 10^{-9}$s
5	$39{,}06 \cdot 10^{4}$	$39{,}06 \cdot 10^{-5}$s
10	$3{,}81 \cdot 10^{12}$	$3{,}81 \cdot 10^{-3}$s
20	$36{,}38 \cdot 10^{25}$	$11{,}53 \cdot 10^{9}$anos
30	$34{,}69 \cdot 10^{39}$	$11{,}00 \cdot 10^{23}$anos
40	$33{,}08 \cdot 10^{53}$	$104{,}91 \cdot 10^{36}$anos
50	$31{,}6 \cdot 10^{67}$	$10{,}01 \cdot 10^{51}$anos
⋮	⋮	⋮
300	$9{,}64 \cdot 10^{417}$	$30{,}57 \cdot 10^{400}$anos

Este fato enfatiza a necessidade de se utilizar métodos heurísticos para lidar com o problema. Neste cenário, métodos de computação evolucionária têm se mostrado muito eficientes e, dentre estes, os algoritmos genéticos (AGs) têm se destacado (LOPES, 2008; SONG et al., 2005; CUSTÓDIO; BARBOSA; DARDENNE, 2004).

2.3.6 Outros modelos

Além do modelo Hidrofóbico-Polar (HP), há outros modelos. Podem-se citar:

- Modelo PH (*Perturbed Homopolymer*): neste modelo apresentado por (SHAKHNOVICH; GUTIN, 1993), as reações entre aminoácidos hidrofóbicos não são le-

vadas em consideração, mas as interações entre aminoácidos do mesmo tipo são favorecidas, ou seja, H-H e P-P, desfavorecendo ligações H-P.

- Modelo LPE (*Lattice Polymer Embedding*): este modelo foi proposto por (UNGER; MOULT, 1993a). A proteína é modelada como uma sequência, $S = s_1, \ldots, s_n$ embutida em uma grade cúbica. Onde, cada aminoácido possui um coeficiente de afinidade, definido para cada par s_i, s_j ($c(s_i, s_j)$) e o objetivo da função de energia é minimizar o produto dos coeficientes pela distância entre os aminoácidos. (UNGER; MOULT, 1993a) mostraram que este problema é *NP*-completo.

- Modelo HP-TSSC (*Hydrophobic-Polar Tangent Spheres Side Chain Model*): este modelo proposto por (HART; ISTRAIL, 1997) é baseado no modelo HP, porém não utiliza uma grade para o posicionamento dos aminoácidos. Neste modelo a proteína é representada por um grafo tridimensional, onde a cadeia lateral e o *backbone* de cada aminoácido são esferas de mesmo raio.

- Modelo CGE (*Charged Graph Embedding*): este modelo foi descrito por (NGO; MARKS; KARPLUS, 1994). Neste modelo, uma carga (*charge*) é atribuida a cada resíduo ($C(s_i) \in \{-1, 0, 1\}$). Entretanto, as conformações permitidas não são realistas. (FRAENKEL, 1993) mostrou que este problema é *NP*-difícil.

- Modelo HPNX: modelo proposto por (BORNBERG; BAUER, 1997) que divide os 20 aminoácidos proteinogênicos em hidrofóbicos (H), positivos (P), negativos (N) e neutros (X). Este modelo também é baseado na mesma grade do modelo HP. Em geral, considera que interações entre aminoácidos hidrofóbicos (H-H) representam interações de atração e decrescem a energia do dobramento em -4.0, as interações entre positivos (P-P) e negativos (N-N) representam interações de repulsão e aumentam a energia livre em 1,0 e as interações entre N e P decrescem a energia em 1,0. O objetivo também consiste em minimizar a energia livre. Portanto, quanto mais interações hidrofóbicas melhor será o dobramento sem desprezar o valor das demais interações.

- Modelo HP-helicoidal (*Helical*-HP): este modelo proposto por (THOMAS; DILL, 1993) considera apenas uma grade bidimensional e inclui dois tipos de interação: interações não-locais através de energia de contatos hidrofóbicos e interações locais representadas por uma tendência à formação de α-hélices (chamada de propensão hélica).

- Modelo *off-lattice* AB: este modelo proposto por (STILLINGER; HEAD-GORDON; HIRSHFELD, 1993) divide os aminoácidos em dois tipos de acordo com a afinidade com o meio aquoso: hidrofóbicos (A) e Hidrofílicos (ou polares – B). Inicialmente, este modelo foi aplicado em duas dimensões (2D AB *off-lattice*) e posteriormente generalizado para três dimensões (3D AB *off-lattice*). A distância da ligação entre dois aminoácidos vizinhos na cadeia é unitária. Os aminoácidos não vizinhos interagem através de um potencial modificado de Lennard-Jones. Os ângulos de torsão entre ligações sucessivas também contribuem no cálculo da função de energia.

2.4 ALGORITMOS GENÉTICOS

A Computação Evolucionária (CE) é um ramo da Inteligência Computacional que utiliza técnicas que mimetizam o processo de evolução natural. As origens dos conceitos da CE remontam à década de 50, com os trabalhos publicados por (BOX, 1957; FRIEDBERG; DUNHAM; NORTH, 1958; BREMERMANN, 1962). Entretanto, as primeiras técnicas fundamentais surgiram nas décadas de 60 e 70, com os trabalhos de (HOLLAND, 1975) (Algoritmos Genéticos – AGs), (FOGEL, 1962)(Programação Evolucionária – PE) e (RECHENBERG, 1965) (Estratégias Evolutivas – EE). Desde então, o número de publicações e congressos nesta área tem crescido e novas implementações de algoritmos evolucionários tem surgido, oriundas das três técnicas citadas anteriormente: AGs, PE e EE.

Os Algoritmos Genéticos (AGs) foram apresentados por (HOLLAND, 1975) como resultado do seu estudo sobre fenômenos naturais que poderiam ser abstraídos para serem utilizados em sistemas computacionais. AGs são métodos probabilísticos de busca e otimização que simulam o processo de evolução de uma população de indivíduos (estruturas que representam as possíveis soluções para o problema em questão) de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas. Este processo é guiado por um mecanismo baseado no modelo Darwiniano de evolução e seleção natural princípio segundo o qual "os indivíduos mais bem adaptados ao ambiente, apresentam maior probabilidade de sobrevivência".

De modo geral, os AGs possuem as seguintes características (GOLDBERG, 1989):

- Operam em um conjunto de pontos (população) e não a partir de pontos isolados;
- Trabalham em um espaço de soluções codificadas e não no espaço de busca dire-

tamente;

- Necessitam somente de informação sobre o valor de uma função objetivo (informações de custo ou recompensa) para cada membro da população, e não requerem derivadas ou qualquer outro tipo de conhecimento.
- Usam transições probabilísticas e não regras determinísticas.

O pseudo-código do fluxo básico do AG é descrito pelo Algoritmo 1, onde o conjunto de soluções candidatas corresponde a uma população de indivíduos, $P(t)$. A cada iteração do algoritmo, denominada de "geração" (designada pelo índice t), uma nova população é criada a partir de indivíduos selecionados da geração anterior. O critério de parada de um AG pode ser definido, por exemplo, em termos do número máximo de gerações desejado ou após um determinado período de tempo, quando houver convergência da população ou quando não houver mais evolução durante uma determinada quantidade de gerações.

Algoritmo 1 Fluxo básico do AG com dois operadores genéticos (recombinação e mutação) e o procedimento de seleção

```
1: t ← 0
2: Gerar população inicial (P^0 = P(t))
3: Avaliar P(t) //Calcular o fitness de cada indivíduo
4: finalizar ← FALSO
5: Enquanto finalizar == FALSO Faça
6:    t ← t + 1
7:    Selecionar P(t) a partir de P(t − 1) com base no fitness de cada indivíduo
8:    Recombinar e Mutar P(t) //Aplicar os operadores genéticos
9:    Avaliar P(t) //Calcular o fitness de cada indivíduo
10:   Substituir a população antiga pela nova
11:   Se critério de parada for alcançado Faça
12:      finalizar ← VERDADEIRO
13:   Fim Se
14: Fim Enquanto
```

2.4.1 Representação cromossômica (codificação)

O primeiro passo para a aplicação de AGs a um problema real é a codificação das variáveis do problema. Usualmente, as variáveis são discretizadas em um determinado alfabeto e representadas por um conjunto de bits, chamado de "cromossomo". Um indivíduo pode ser formado por um (haplóide) ou mais (multiplóide) cromossomos. Em geral, o método de representação quanto ao alfabeto genético depende do problema. No caso mais simples, usa-se o alfabeto binário $\{0, 1\}$.

O cromossomo é composto de vários “genes”. Como cada gene pode assumir qualquer valor do alfabeto, cada elemento do alfabeto é equivalente a um “alelo”, ou seja, um valor possível para um dado gene. A posição de um gene em um cromossomo corresponde a um “locus gênico”. Um exemplo de cromossomo é apresentado na Figura 8.

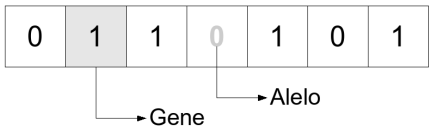


Figura 8: Exemplo de cromossomo

Fonte: Autoria própria

De acordo com (GOLDBERG, 1989), o sucesso da execução de um AG depende da codificação das variáveis do problema. Um problema mal codificado pode impedir a convergência do AG ou a obtenção de uma boa solução para o problema. Dependendo do problema, pode existir várias possibilidades de codificação de um indivíduo.

Comumente, assume-se que cada indivíduo é formado por um único cromossomo. Assim, é comum encontrar na literatura os termos indivíduo e cromossomo indistintamente. A maioria dos AGs na literatura usam uma população com número fixo de indivíduos, com cromossomos também de tamanho constante.

2.4.2 Geração da população inicial

Geralmente, a população inicial de indivíduos (P^0) é gerada aleatoriamente ou utilizando algum procedimento heurístico baseado em conhecimento do problema. Do ponto de vista biológico, não há evolução sem diversidade genética. Portanto, é importante que a população inicial seja dispersa na maior parte do espaço de busca.

2.4.3 Avaliação da população e função de *fitness*

A implementação de uma função de *fitness* (na falta de tradução melhor, esta função também é chamada de função de “adequabilidade” ou “adaptabilidade”) representa o ponto mais crítico na modelagem de AGs para uma aplicação real, pois mede a qualidade de cada indivíduo da população.

A função objetivo fornece uma medida de quão bem adaptado ao ambiente o indivíduo está. O valor de *fitness* de um indivíduo corresponde ao valor da função objetivo nos problemas de otimização sem restrições. Entretanto, para problemas de otimização com restrições, uma função de penalização pode estar associada à função de *fitness*.

2.4.4 Método de Seleção

A seleção é um processo que a priori seleciona indivíduos aos quais serão aplicados os operadores genéticos, normalmente privilegiando aos indivíduos da população mais bem adaptados ao ambiente (baseando-se no *fitness* dos indivíduos). Assim, o processo de seleção emula o princípio de seleção natural, pois os indíviduos mais bem adaptados ao ambiente têm maior probabilidade de sobreviver, se reproduzir e passar seu material genético para os descendentes. Este processo é utilizado para direcionar a evolução para melhores regiões do espaço de busca.

Dentre os vários métodos de seleção existentes, o mais utilizado é a seleção por torneio (*tournament selection*) (MILLER et al., 1995).

2.4.5 Operadores genéticos

Há dois operadores genéticos básicos: a recombinação (também chamada de *crossover*) e a mutação.

O operador de *crossover* permite a obtenção de indivíduos filhos a partir da combinação dos cromossomos dos pais. Este operador realiza busca local (*exploitation*). A forma mais simples é o operador de *crossover* de um ponto, que seleciona aleatoriamente uma posição do cromossomo (chamado de ponto de *crossover*) e a partir dela realiza a troca os genes entre os pais para gerar os filhos. Outra forma é operador de *crossover* de dois pontos, onde o parte contida entre os pontos de *crossover* é trocada entre os pais.

O operador de *crossover* é aplicado probabilisticamente com base em um parâmetro chamado de probabilidade de *crossover* (p_{cross}).

Por outro lado, o operador de mutação realiza uma busca global (*exploration*) através da mudança aleatória de alelos. Este operador tem como objetivo evitar a perda de diversidade genética durante a evolução do AG, reduzindo a possibilidade de convergência prematura.

A forma mais simples deste operador é a mutação aleatória de um bit do cromossomo, de acordo com o parâmetro de probabilidade de mutação (p_{mut}).

Os operadores genéticos também podem ser concebidos com base em conhecimentos sobre o problema.

2.4.6 Algoritmos Genéticos paralelos

Os Algoritmos Genéticos paralelos (AGPs ou *Parallel Genetic Algorithms*– PGAs) podem ser classificados em quatro modelos (CANTÚ-PAZ, 2000):

- Modelo Mestre-escravo: consiste na distribuição da função objetivo entre vários processadores escravos, sobre a coordenação de um processador central mestre. Esta abordagem é particularmente interessante para problemas onde a computação da função de *fitness* é muito custosa, como é o caso deste trabalho (ver seção 3.3). Este modelo é mostrado na Figura 9(a).

- Modelo distribuído (*Distributed Genetic Algorithms*– dGA) ou Multi-populacional: neste modelo, a população do AG é distribuída em várias subpopulações independentes. Geralmente, este modelo também é chamado de modelo insular, onde cada subpopulação é considerada como uma ilha. As ilhas evoluem independentemente e podem possuir parâmetros diferentes. Também é chamado de modelo de granularidade grossa ou AG *coarse-grained*, pois a taxa entre a computação e a comunicação é, usualmente, elevada. Por este motivo, este modelo é implementado em computadores MIMD (ver Seção 2.5.1) de memória distribuída (p.ex.: *clusters* Beowulf). A principal característica deste paradigma consiste na migração de indivíduos entre as subpopulações através de uma política de migração.

  A definição da política de migração é de suma importância, pois determina a velocidade de convergência e pode ajudar no controle da diversidade genética das subpopulações.

  Este modelo é mostrado na Figura 9(b), onde cada subpopulação (ou ilha) representa um AG simples. Notam-se, também, os canais de comunicação, os quais são mapeados para uma rede física e recebem atribuições específicas da estratégia de migração do algoritmo.

  Em (TAVARES; LOPES; ERIG, 2009) é apresentado um estudo sobre topologias de PGA insular.

- Modelo celular (*Cellular Genetic Algorithms* – cGA): este modelo trata com uma única população estruturada espacialmente. A estrutura da população é, usualmente, uma grade bi-dimensional, onde cada ponto da grade representa um indivíduo. Idealmente, cada indivíduo é associado a um processador. Por este motivo, este paradigma também é chamado de modelo de granularidade fina ou AG *fine-grained*. Assim, a função objetivo é computada simultaneamente para todos os indivíduos. A principal característica deste modelo é a estruturação da população em vizinhanças, onde os indivíduos podem interagir apenas com os seus vizinhos (ver Figura 8(c)).

- Modelos hierárquicos: os modelos hierárquicos (ou modelos híbridos) combinam características dos modelos anteriormente apresentados, tirando proveito dos benefícios de ambos paradigmas. Por exemplo, as Figuras 10(a) e 10(b) mostram duas arquiteturas hierárquicas compostas de dois níveis de paralelização. Nas Figuras 10(a) e 10(b), o nível superior é composto pelo modelo insular (dGA) e o inferior pelos modelos mestre-escravo e celular (cGA), respectivamente.

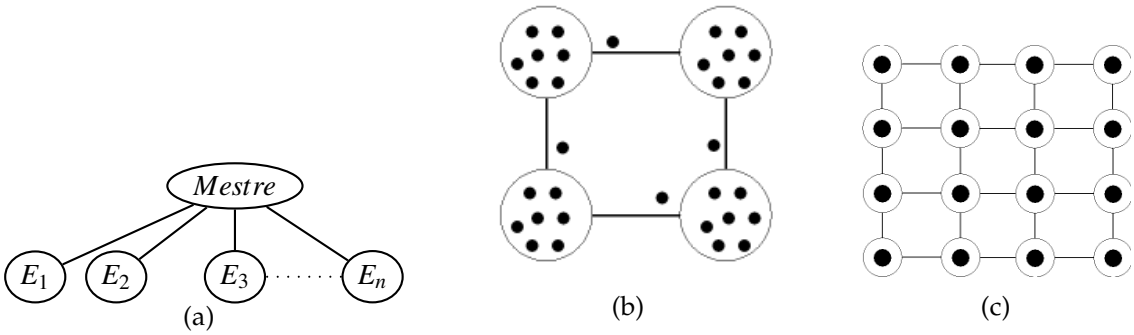


Figura 9: Modelos de AGP: mestre-escravo (a), Distribuído (b) e Celular (c)

Fonte: Autoria própria

2.5 PROCESSAMENTO PARALELO

Devido ao grande interesse por problemas complexos em diversas áreas de pesquisa (por exemplo, nas áreas de pesquisa aeroespacial, modelagem climática, bioinformática, entre outros), cientistas da computação tem se esforçado para melhorar o desempenho de computadores. Nas últimas décadas, o desempenho dos microprocesadores tem aumentado exponencialmente devido aos avanços tecnológicos em mi-

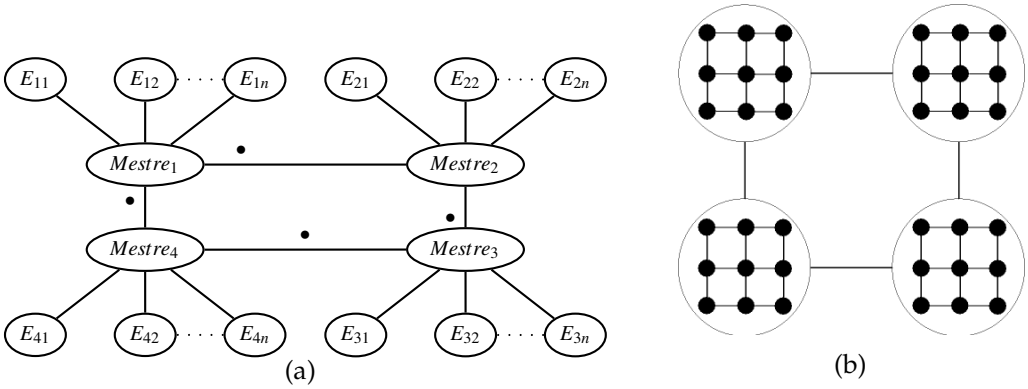


Figura 10: Modelos híbridos de AGP

Fonte: Autoria própria

croeletrônica, em particular no desenvolvimento de circuitos integrados VLSI (*Very-Large-Scale-Integration*) (PARHAMI, 2002). Isto permitiu a inclusão de melhorias a nível arquitetural cada vez mais complexas como *pipelines*, memórias cache, grandes *buffers* de instrução e unidades de processamento.

O aspecto financeiro tem orientado o desenvolvimento dos sistemas paralelos. Como o custo dos computadores pessoais tem diminuído, enquanto o custo de máquinas paralelas continua sendo bastante elevado (devido ao projeto e construção de *hardware* específico), as restrições econômicas levaram cientistas ao desenvolvimento de *clusters* locais (conjunto de computadores conectados utilizando alguma rede de interconexão). Por outro lado, as melhorias realizadas nas redes de comunicação também contribuíram no desenvolvimento de *clusters* formados por computadores distribuídos em diferentes locais geográficos (abordagem conhecida como *grid computing* ou computação em grade) (EL-REWINI; ABD-EL-BARR, 2005).

O processamento paralelo é uma modalidade de computação em que várias instruções são processadas simultaneamente para alcançar alto desempenho (DONGARRA et al., 2003). Os computadores paralelos são basicamente sistemas computacionais consistindo de múltiplas unidades de processamento conectadas por alguma rede de interconexão. Teoricamente, o processamento paralelo pode aumentar ilimitadamente o desempenho dos computadores. Entretanto, na prática, o desempenho destes computadores é limitado fortemente pelo *hardware* e pelo *software*.

2.5.1 Arquiteturas paralelas

Há diversas classificações possíveis para arquiteturas paralelas, dependendo dos critérios utilizados. Entretanto, a taxonomia proposta por Flynn (FLYNN, 1972) é comumente aceita como a referência neste domínio. Esta taxonomia organiza as arquiteturas em quatro classes, de acordo com os mecanismos de controle (ordem de execução) e de fluxo de dados (acesso aos operandos):

- SISD (*Single Instruction stream Single Data stream* – fluxo único de dados e instruções): esta classe representa os computadores monoprocessados que processam uma única instrução com um único dado a cada instante, correspondendo à tradicional arquitetura de Von Neumann.

- SIMD (*Single Instruction stream Multiple Data stream* – fluxo único de instruções e múltiplo de dados): esta classe representa processadores vetoriais com diversas unidades de processamento, permitindo a execução de uma instrução em vários dados simultaneamente. Este tipo de arquitetura também é conhecida como "*array processors*". Basicamente, o modelo SIMD consiste em duas partes: um computador *front-end* com arquitetura von Neumann, e um *array* de processadores. O *array* de processadores é um conjunto de elementos de processamento com a capacidade de processar uma instrução em diferentes dados. Cada processador do *array* possui memória local onde residem os dados distribuídos durante o processamento paralelo.

  Um programa pode ser desenvolvido e executado no computador *front-ent* utilizando uma linguagem de programação sequencial. A aplicação é executada pelo *front-end* de maneira sequencial, exceto comandos que são executados em paralelo no *array* de processadores.

- MISD (*Multiple Instruction stream Single Data stream*): nesta categoria, um conjunto linear de processadores executa diferentes instruções no mesmo dado simultaneamente. Na prática, a construção de máquinas MISD é inviável. Entretanto, máquinas *pipeline* são consideradas como exemplos de MISD (EL-REWINI; ABD-EL-BARR, 2005), apesar dos dados serem modificados após o processamento por cada fase do *pipeline*.

- MIMD (*Multiple Instruction stream Multiple Data stream* – fluxo múltiplo de instruções e dados): Esta categoria é caracterizada pela execução de múltiplos fluxos

de instruções sobre distintos fluxos de dados. A arquitetura MIMD é formada por vários processadores e módulos de memória interconectados através de uma rede de interconexão, cada um executando uma parte do programa simultaneamente para a solução do problema de forma cooperativa. Cada processador possui a sua própria unidade de controle e pode executar diversas instruções em diferentes dados. Basicamente, este tipo de arquitetura pode ser dividido em duas classes: memória compartilhada (*shared memory*) ou passagem de mensagem (*message-passing*). As classes de memória compartilhada e de passagem de mensagem também recebem o nome de multiprocessadores e multicomputadores, respectivamente.

O sistema de memória compartilhada (multiprocessadores) possui um processador de coordenação entre uma memória global compartilhada e todos os elementos processadores. O sistema de passagem de mensagem (multicomputadores, também chamado de memória distribuída) combina uma memória local e processador em cada nó da rede de interconexão. Neste sistema, não há memória global. Portanto, os dados devem ser movidos entre memórias locais através de comandos de envio/recebimento (*Send/Receive*) que devem ser escritos pelo programador utilizando o paradigma de *message-passing*. Os multicomputadores podem ser divididos em computadores massivamente paralelos MPP (*Massive Parallel Processors*) e COW (*Cluster of Workstations*). Os computadores MPP são formados por um conjunto de processadores fortemente acoplados através de uma rede de alta velocidade. Este tipo de arquitetura possui um custo bastante elevado, pois utiliza processadores específicos e redes de interconexão proprietárias. Um exemplo desde tipo de computador é a família Cray T3E. A arquitetura COW, também chamada de NOW (*Network of Workstations*) é composta por diversos computadores conectados através de redes de interconexão tradicionais. Um exemplo deste tipo de arquitetura é o cluster Beowulf, que é uma abordagem eficiente e de baixo custo (EL-REWINI; ABD-EL-BARR, 2005).

2.5.2 *Cluster* Beowulf

O conceito de *clusters* Beowulf [4] foi criado pelos cientistas Thomas Sterling e Donald Becker do *Center of Excellence in Space Data and Information Sciences* (CESDIS –

[4] O nome *Beowulf* deriva de um poema épico inglês datado do século XI, o mais antigo escrito existente em inglês. Neste conto, *Beowulf* é um herói que salvou o reino dinamarquês de dois monstros, exterminando-os.

NASA) com o principal objetivo de criar um computador paralelo de baixo custo, construído a partir de computadores pessoais disponíveis no mercado, para satisfazer requisitos computacionais específicos para aplicações científicas (STERLING, 2002).

Um *Cluster* Beowulf é formado por vários componentes de *hardware* e *software*, e composto de quatro componentes principais, dois componentes de *hardware* e dois de *software*. Os dois componentes de *hardware* são nós de computação e a rede que interconecta os nós para formar o sistema. Os dois componentes de *software* são coleções de ferramentas utilizadas para desenvolver as aplicações paralelas e o ambiente para a gestão dos recursos do *cluster* Beowulf (STERLING, 2002). A especificação do *cluster* determina os custos, capacidade, desempenho e a usabilidade do sistema.

Cada nó do *cluster* Beowulf é composto por um computador com um ou mais microprocessadores, unidades de armazenamento, memória e interfaces de entrada/saída. Alguns *clusters* são formados por nós *diskless*, isto é, sem disco rígido para reduzir custos e consumo de energia elétrica, como também melhorar a confiabilidade do sistema.

A rede de interconexão provê o meio de troca de dados entre os nós do *cluster*, coordenando as operações através de mecanismos de sincronização. Os componentes da rede são os controladores de interface de rede (*Network Interface Controllers* – NIC), canais de rede (ou *links*) e *switch*. Em um *cluster* Beowulf, os computadores são conectados entre si através do protocolo TCP/ IP sobre uma rede lógica *ethernet* convencional (EL-REWINI; ABD-EL-BARR, 2005).

A escolha das ferramentas de *software* para o desenvolvimento de aplicações depende do modelo de programação a ser utilizado. A comunidade de *clusters* Beowulf tem convergido para o modelo de passagem de mensagem (*message-passing*). Existem diversas bibliotecas de subrotinas de comunicação que auxiliam na implementação de aplicações neste modelo, sendo as principais: MPI (*Message Passing Interface*) (GROPP; LUSK; THAKUR, 1999) e PVM (*Parallel Virtual Machine*) (GEIST et al., 1994). A escolhida para ser utilizada neste trabalho é a MPI, mais especificamente a implementação MPICH2 do *Argonne Labs* [5]. Por outro lado, é importante destacar que a maioria dos *clusters* Beowulf utiliza o sistema operacional Linux, devido à sua estabilidade, robustez, desempenho e flexibilidade de configuração.

Além de proporcionar uma excepcional relação de custo/benefício em relação a outros computadores paralelos, os *clusters* Beowulf apresentam vários benefícios (STERLING, 2002): escalabilidade, flexibilidade de configuração/atualização e alta disponi-

[5]Disponível em: http://www.mcs.anl.gov/research/projects/mpich2/

bilidade.

2.5.3 A biblioteca MPI

A função da biblioteca MPI é prover uma biblioteca de rotinas para auxiliar no desenvolvimento de aplicações portáveis e eficientes utilizando o modelo de passagem de mensagem como, por exemplo, subrotinas de comunicação ponto-a-ponto, operações coletivas, gestão de processos, agrupamento de processos, criação de contextos de comunicação, topologia de processos, entre outras funções que automatizam tarefas usuais na computação paralela.

Uma aplicação MPI pode ser vista como um grupo de processos concorrentes que trabalham de forma conjunta, intercambiando mensagens para compartilhamento de dados e sincronização de operações em uma aplicação. No MPI, cada processo possui uma identificação única, denominada ***rank***, dentro de um determinado contexto. O *rank* de cada processo é atribuído como um número inteiro entre 0 e $n-1$ para uma aplicação com n processos. Durante a execução de uma aplicação paralela, o mesmo programa é executado em todos os processadores simultaneamente. Portanto, no código fonte do programa deve haver uma distinção sobre o que cada processo deve fazer. Em outras palavras, todo o paralelismo é explícito, ou seja, a identificação de pontos de paralelismo, a criação e organização dos processos fica a cargo do programador.

Em aplicações MPI não há o conceito de máquina virtual e o uso de *tags* (rótulos de mensagem) não é suficiente para distinguir mensagens (EL-REWINI; ABD-EL-BARR, 2005). O conceito de *communicator* (comunicador) foi introduzido na biblioteca MPI para garantir segurança na comunicação e evitar problemas no envio e recebimento de mensagens entre os processos da aplicação. O comunicador é um objeto local que representa o contexto (domínio) de uma comunicação. Em outras palavras, o comunicator é utilizado para representar o grupo de processos que podem estabelecer uma comunicação. Os comunicadores podem ser classificados em intracomunicadores para operações de comunicação dentro de um grupo de tarefas e intercomunicadores para operações entre diferentes grupos de processos.

Atualmente, a biblioteca MPI possui rotinas para Fortran, C e C++. O Anexo B apresenta as 6 funções básicas e algumas funções úteis da biblioteca MPI.

2.5.4 Considerações para projeto de algoritmos paralelos

A metodologia para projeto de algoritmo paralelos possui quatro fases (ROOSTA, 1999), como mostrado na Figura 11: particionamento, comunicação, agrupamento, mapeamento.

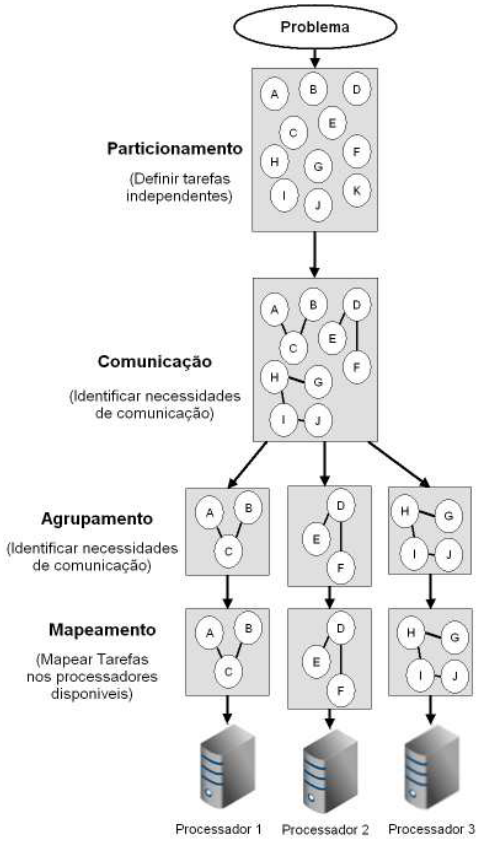


Figura 11: Diagrama de metodologia

Fonte: Autoria própria

Nas duas primeiras fases, aspectos independentes do *hardware* são levados em consideração, tais como concorrência de operações e decomposição de tarefas. Por outro lado, aspectos dependentes do *hardware* são considerados nas duas últimas fases, tais como desempenho e custo de comunicação.

- Particionamento: as oportunidades de paralelização são identificadas nesta fase, dividindo o processamento em várias partes pequenas. Há dois métodos para particionamento:
  - Particionamento do domínio: foco nos dados associados com o problema e determina a computação apropriada para estes dados;

  - Particionamento funcional: foco na decomposição do processamento em tarefas disjuntas.

  Estes métodos podem ser aplicados a diferentes componentes de um problema. Consequentemente, deve-se evitar a replicação de tarefas ou de dados. Nesta fase, também deve-se determinar o paradigma de paralelização.

- Comunicação: esta fase tem como objetivo estabelecer uma estrutura de comunicação e um meio de coordenar a execução de tarefas. Envolve a transferência de dados entre tarefas. Para isto, duas estruturas são envolvidas:

  - Estrutura de canal de comunicação: diz respeito à ligação direta ou indireta de tarefas que requerem dados de tarefas que produzem dados.
  - Estrutura de passagem de mensagens: especifica as mensagens que devem ser enviadas e recebidas pelo canal de comunicação.

  Esta fase é fundamental devido ao custo envolvido nas operações de comunicação. Em geral, procura-se otimizar o desempenho através da distribuição de operações de comunicação em várias tarefas, organizando-as de modo a permitir a execução concorrente. No entanto, pensar em termos de estruturas de canal pode ajudar a avaliar os algoritmos do ponto de vista de custo de comunicação.

- Agrupamento: esta fase avalia a granularidade das tarefas e o custo de comunicação entre elas. Nesta ponto as duas primeiras fases são revisadas para realizar as decisões apropriadas no que diz respeito às questões envolvidas. Deve-se levar em consideração o *hardware* disponível e, se necessário, voltar às duas fases anteriores. As tarefas com maior dependência de dados ou maior correlação entre si devem ser agrupadas.

- Mapeamento: esta fase especifica onde cada tarefa será fisicamente executada. Em outras palavras, cada tarefa é associada a um processador de forma a maximizar a utilização dos processadores e reduzir o custo de comunicação entre eles.

2.5.5 Métricas de desempenho em sistemas paralelos

As métricas de desempenho estabelecem parâmetros de comparação de desempenho dos algoritmos.

Provavelmente, a métrica de desempenho mais utilizada em computação paralela seja o *speedup* (ou fator de aceleração) (ALBA, 2005). Esta medida avalia o quanto mais rápido um algoritmo paralelo conseguiu ser executado em relação à sua versão sequencial. Em outras palavras, esta métrica quantifica o ganho de tempo de processamento de um sistema paralelo. O *speedup* (s_m) é definido como a divisão entre o tempo de execução sequencial em um processador (T_1) pelo tempo de execução do algoritmo paralelo, executado em m processadores (T_m), como mostrado na Equação 6.

$$s_m = \frac{T_1}{T_m} \tag{6}$$

A partir da Equação 6, três tipos de comportamento de *speedup* podem ser identificados: *speedup* sublinear ($s_m < m$), *speedup* linear ($s_m = m$) e *speedup* superlinear ($s_m > m$).

(ALBA, 2005) propôs uma taxonomia interessante para as medidas do *speedup* em sistemas paralelos:

- *Strong speedup* (aceleração forte): compara o tempo de processamento paralelo contra o algoritmo sequencial mais eficiente conhecido (algoritmo sequencial *best-so-far*);
- *Weak speedup* (aceleração fraco): compara o tempo de processamento do algoritmo paralelo desenvolvido por um pesquisador contra seu próprio algoritmo sequencial (supondo que ambos forneçam a mesma acurácia). Há duas variantes para este tipo de medida:
  - *Versus panmixia*: quando se compara o tempo de processamento do algoritmo paralelo com a versão sequencial simples;
  - *Orthodox*: quando se compara o tempo de processamento de algoritmo paralelo executando em um processador, com o mesmo algoritmo executando em m processadores.

Outra métrica de desempenho largamente utilizada em sistemas paralelos é a eficiência. Esta medida avalia a fração de tempo que um processador é, de fato, utilizado no processamento (ROOSTA, 1999). A Equação 7 apresenta como a eficiência de um sistema paralelo é calculada: é definida como o *speedup* dividido pelo número de processadores.

$$e_m = \frac{s_m}{m} \tag{7}$$

Idealmente, a eficiência deveria ser unitária ($e_m = 1$), mas isto não é possível porque os processadores não utilizam 100% do tempo para processamento, mas também para comunicação, alocação de memória e outras tarefas do sistema operacional (ROOSTA, 1999).

2.6 ALGORITMO DE COLÔNIA ARTIFICIAL DE ABELHAS

Os insetos sociais, como formigas e abelhas passam a maior parte da sua vida em busca de alimento. As colônias de abelhas têm um sistema descentralizado de coleta de alimentos e podem ajustar o padrão de busca para aumentar a quantide de néctar coletado (SEELEY, 1995).

Sabe-se que as abelhas são capazes de estimar a distância entre a colméia e as fontes de alimento através da medição da quantidade de energia consumida durante o vôo, além da direção e a qualidade da fonte de alimento. Esta informação é compartilhada com as demais abelhas da colméia através da trofalaxe (contato direto) e da realização de uma dança. O intercâmbio de informação entre abelhas é a ocorrência mais importante na formação de conhecimento coletivo. A pista de dança é o local da colméia onde as abelhas farejadoras realizam a dança para recrutar mais abelhas farejadoras. As abelhas que decidem farejar sem nenhuma orientação são chamadas de escoteiras. As abelhas que comparecem à dança podem escolher a fonte de alimento com base na sua qualidade. A qualidade de uma fonte de alimento é proporcional à quantidade de néctar encontrado nela, e esta informação é transmitida através da intensidade da dança e de contatos entre antenas. Quanto melhor a fonte de alimento, mais intensos são a dança e os contatos (REINHARD; SRINIVASAN, 2009). Portanto, cada abelha farejadora pode se comportar de três maneiras depois de descarregar o alimento: realizando a dança para recrutar mais farejadoras para a mesma fonte de alimento, abandonando a fonte de alimento devido ao esgotamento dos recursos disponíveis ou retornando diretamente à busca de alimento.

A idéia básica dos algoritmos baseados no comportamento das abelhas é que as abelhas farejadoras possuem uma solução potencial para um problema de otimização nas suas memórias (por exemplo, uma configuração para as variáveis de decisão do problema). Cada solução potencial corresponde à localização e qualidade da fonte

de alimento (por exemplo, valor da função objetivo). Baseado neste comportamento, (KARABOGA, 2005) propôs o algoritmo ABC (*Artificial Bee Colony*).

O algoritmo ABC (*Artificial Bee Colony Algorithm*) opera com um enxame de n soluções x (fontes de alimento) de dimensão d que são modificadas pelas abelhas artificiais. O objetivo das abelhas é descobrir locais de fontes de alimento v (posições no espaço de busca) com elevada quantidade de néctar (bom *fitness*).

No algoritmo ABC há três tipos de abelhas: abelhas escoteiras que voam aleatoriamente no espaço de busca sem orientação, abelhas empregadas que exploram a vizinhança de fontes de alimento selecionando uma solução perturbada aleatoriamente, e abelhas observadoras que são posicionadas nas fontes de alimento utilizando um processo de seleção probabilístico. A probabilidade P_i com que uma fonte de alimento é preferida pelas abelhas observadoras aumenta com o aumento da quantidade de néctar da fonte.

Se a quantidade de néctar de uma nova fonte encontrada por uma abelha empregada é maior do que a anterior armazenada na sua memória, ela atualiza a nova posição e esquece da anterior. Se uma solução não é melhorada por um número predeterminado de tentativas controlado através do parâmetro $limit$, a fonte de alimento é abandonada pela abelha empregada e esta se torna uma abelha escoteira.

Cada ciclo da busca consiste no movimento de abelhas empregadas e observadoras para fontes de alimento através do cálculo da quantidade de néctar, e determinação de abelhas escoteiras direcionadas para possíveis fontes de alimento. O pseudo-código do algoritmo ABC é apresentado no Algoritmo 2.

O algoritmo ABC tenta equilibrar a exploração através da combinação de métodos de busca local, realizados pelas abelhas empregadas e observadoras, com métodos de busca global, gerido por abelhas escoteiras.

2.7 TRABALHOS CORRELATOS

Há diversas abordagens para a solução do PDP, cada uma abordando o problema com o uso de um modelo de proteína e um método computacional para a obtenção de soluções ótimas ou quase-ótimas.

O método que parece ser o mais realista, do ponto de vista biológico, é chamado de Dinâmica Molecular (também conhecido como *ab initio*) (HARDIN; POGORELOV;

Algoritmo 2 Algoritmo ABC.

1: Parâmetros: n, $limit$
2: Função objetivo $f(x)$, $x = [x_1, x_2, ..., x_d]^T$
3: Inicializar fontes de alimento aleatoriamente x_i $i = 1, 2, ..., n$
4: ***Avaliar indivíduos*** //calcular a função de *fitness* $f(x_i)$ de cada indivíduo
5: **Enquanto** critério de parada não for alcançado **Faça**
6: Fase de abelhas empregadas:
7: Produzir novas soluções com valores aleatórios de k, j e ϕ
8: $v_{ij} = x_{ij} + \phi_{ij} \cdot (x_{ij} - x_{kj})$ $k \in \{1, 2, ..., n\}, j \in \{1, 2, ..., d\}, \phi \in [0,\ 1]$
9: ***Avaliar soluções***
10: Aplicar o processo de seleção gulosa para as abelhas empregadas entre v_i e x_i
11: Fase de abelhas observadoras:
12: Calcular a probabilidade para as soluções x_i
13: $P_i = \frac{f_i}{\sum_{j=i}^{n} f_j}$
14: Produzir novas soluções a partir da solução x_i selecionada usando P_i
15: ***Avaliar soluções***
16: Aplicar seleção gulosa para as abelhas observadoras
17: Fase de abelhas escoteiras:
18: Encontrar solução abandonada: Se o limite for excedido, substituir a solução por uma solução gerada aleatoriamente
19: Memorizar a melhor solução obtida
20: **Fim Enquanto**

LUTHEY-SCHULTEN, 2002). A idéia básica deste método é simular os movimentos de cada átomo de uma proteína em função do tempo, de acordo com regras da mecânica clássica. A energia da conformação leva em consideração todas as forças, acelerações e velocidades dos átomos. Assim, o número de operações matemáticas a serem realizadas é muito elevado, tornando o modelo inviável computacionalmente. Este tipo de simulação pode ser útil apenas para estudos do comportamento do dobramento durante um período de tempo pequeno, muitas vezes menor que o tempo necessário para dobrar uma proteína real. Entretanto, ainda assim propriedades da dinâmica do dobramento podem ser observadas, embora não se possa confirmar sua habilidade de convergir para a conformação nativa. Uma revisão completa sobre este modelo pode ser encontrada em (LEE; DUAN; KOLLMAN, 2001).

Algoritmos de aproximação podem ser utilizados para encontrar soluções próximas à solução ótima (também chamadas de soluções quase-ótimas) para problemas específicos com alguma garantia, isto é, dentro de um determinado limite de erro permitido (CHANDRU; DATTASHARMA; KUMAR, 2003). De acordo com (NGO; MARKS; KARPLUS, 1994), este tipo de algoritmo pode ser útil para o PDP, pois a exatidão não é um requisito absoluto. Por exemplo, utilizando um limite de erro suficientemente pe-

queno, o algoritmo pode encontrar estruturas que, embora não dobradas corretamente, estão próximas da estrutura nativa. Caso contrário, ainda poderia ser útil como parte de um esquema maior. Possivelmente, o primeiro algoritmo de aproximação para o PDP foi desenvolvido por (HART; ISTRAIL, 1996) usando os modelos 2D e 3DHP (ver Seção 2.3.3). Posteriormente, outros algoritmos foram desenvolvidos utilizando outros modelos de energia/grade (HART; ISTRAIL, 1997; HEUN, 2003; NEWMAN, 2002).

Também há diversas implementações de arquiteturas de redes neurais para o PDP. Por exemplo, (YANIKOGLU; ERMAN, 2002) utiliza um SOM (*Self-Organizing Map*) com o modelo 2D-HP. Entretanto, bons resultados foram obtidos apenas para sequências pequenas de até 36 aminoácidos.

Conforme foi mencionado na Seção 2.3.5, resolver o PDP, mesmo utilizando o modelo de treliça mais simples é *NP*-completo. Este fato tem motivado o desenvolvimento de várias metaheurísticas para lidar com o problema. Neste cenário, métodos de computação evolucionária e, especialmente, os algoritmos genéticos (AGs) têm se mostrado não somente adequados, mas também muito eficientes (CUSTÓDIO; BARBOSA; DARDENNE, 2004; LOPES, 2008; SONG et al., 2005).

Dentre as várias abordagens de computação evolucionária utilizadas para o PDP, certamente a mais utilizada é o AG, devido à sua eficiência em encontrar boas soluções em um espaço de busca complexo e fortemente restrito. Por exemplo, (SCAPIN; LOPES, 2007a) apresenta a aplicação de um AG ao PDP utilizando o modelo 2D-HP.

Métodos de busca local de Monte Carlo, busca tabu e *hill-climbing* foram empregados como operadores genéticos de AGs por (COX et al., 2004), (JIANG et al., 2003) e (TANTAR et al., 2007), respectivamente.

O algoritmo de otimização por colônia de formigas (*Ant Colony Optimisation*– ACO) é uma técnica evolucionária inspirada no comportamento de formigas na busca por alimento. A primeira aplicação de ACO para o PDP foi realizada por (SHMYGELSKA; HOOS, 2005) e é baseada em três fases: construção, busca local e atualização da matriz de feromônios. Na primeira fase, as formigas constroem um dobramento na grade inicializando em uma posição aleatória. Depois, um procedimento de busca local gulosa é realizado. Finalmente, a matriz de feromônios é atualizada utilizando dois mecanismos: taxa de evaporação uniforme e reforço de *motifs* estruturais. Um mecanismo de normalização da matriz de feromônios também foi utilizado para evitar a estagnação da busca. O ACO foi aplicado para várias instâncias de *bechmarks* utilizando os modelos 2D (PERRETO, 2005) e 3DHP (FIDANOVA, 2006; SONG et al., 2005; CHU;

TILL; ZOMAYA, 2005) e comparado com outros métodos heurísticos. (CHU; TILL; ZOMAYA, 2005) desenvolveram abordagens de *single* e *multiple-colony* ACO, com processamento centralizado e distribuído. Eles demostraram que a versão distribuída apresenta melhor desempenho comparando com as versões *single*.

O algoritmo de Evolução diferencial (*Differential Evolution* – DE) utiliza vetores de diferenças para gerar perturbações em uma população de vetores. Atualmente, os únicos trabalhos publicados utilizando Evolução Diferencial para o PDP são (BITELLO; LOPES, 2007) e (KALEGARI; LOPES, 2010) usando os modelos 2DHP e 2D AB *off-lattice*, respectivamente. Possivelmente, isto é devido ao fato do DE ser um algoritmo evolucionário relativamente recente e por ter sido criado para problemas contínuos de otimização.

Sistemas imunológicos artificiais (*Artificial Immune Systems*– AIS) também foram aplicados ao PDP, utilizando os modelos 2D e 3DHP por (CUTELLO; NARZISI; NICOSIA, 2005). (ALMEIDA; GONÇALVES; DELGADO, 2007) apresenta uma proposta de AIS hibridizado com busca tabu e um sistema de inferência *fuzzy*, onde um operador *fuzzy* decide quais anticorpos serão removidos da população após o procedimento de seleção, e a busca tabu é utilizada para definir um mecanismo de afinidade de maturação de anticorpos.

(LI, 2007) apresenta a implementação de um algoritmo de recozimento simulado (*Simulated Annealing*– SA), utilizando o modelo 2D-HP.

(DAUGHERITY, 1993) apresenta a aplicação de um sistema híbrido neuro-*fuzzy* aplicado ao PDP combinando a capacidade adaptativa das redes neurais com o poder de inferência dos sistemas *fuzzy*.

Métodos estatísticos aplicados ao PDP utilizando o modelo *ab initio* foram apresentados por (OSGUTHORPE, 2000).

(OSTROVSKY et al., 2001) descreve a implementação de um autômato celular para a simulação de polímeros, incluindo implicações no processo de dobramento de proteínas.

Algoritmos meméticos foram aplicados ao PDP utilizando os modelos HP de treliças triangular e quadrada bidimensional, e o modelo funcional de proteínas de treliça com forma de diamante por (KRASNOGOR et al., 2002). Os resultados obtidos demonstraram que o algoritmo é robusto para todos os modelos utilizados.

(THACHUK; SHMYGELSKA; HOOS, 2007) apresenta um implementação do mé-

todo de busca de Monte Carlo para o PDP utilizando os modelos 2D e 3D-HP.

(STILLINGER; HEAD-GORDON, 1995), (IRBACK; PETERSON; POTTHAST, 1997) e (TORCINI; LIVI; POLITI, 2001) apresentam a aplicação de redes neurais, método de busca de Monte Carlo e métodos biologicamente motivados ao PDP utilizando o modelo 2D AB *off-lattice*, respectivamente. Uma versão extendida do modelo 2D AB para a versão 3D foi apresentada por (HSU; MEHRA; GRASSBERGER, 2003). Recentemente, (ZHANG; CHENG, 2008) apresentaram uma implementação melhorada do método de busca tabu para o modelo 3D AB *off-lattice*. De acordo com os autores, este algoritmo apresenta um bom desempenho e pode ser efetivamente utilizado no PDP utilizando o modelo 3D AB *off-lattice*.

3 METODOLOGIA

Este capítulo apresenta a descrição detalhada da implementação do Algoritmo Genético Paralelo aplicado para o Problema de Dobramento de Proteínas com o modelo 3DHP-SC.

Três versões de AG foram desenvolvidas: as versões sequencial, mestre-escravo síncrono (AGP–ME) e hierárquica (AGP–HH).

O método proposto neste trabalho foi comparado com outra técnica de computação evolucionária. Para isto, versões paralelas do Algoritmo de colônia de abelhas artificial (ou *Artificial Bee Colony* – ABC) foram desenvolvidas.

3.1 CODIFICAÇÃO DOS INDIVÍDUOS

Há dois problemas principais na modelagem de AGs para um dado problema de otimização: como as variáveis do problema serão codificadas e como a qualidade das soluções será avaliada. O primeiro representa o problema da representação cromossômica (codificação) e o segundo, o problema da avaliação (função de *fitness*).

A representação cromossômica tem uma grande influência na dinâmica e eficiência dos AGs (LOPES, 2008). A codificação pode influenciar fortemente não somente o tamanho do espaço de busca, mas também na complexidade do problema, devido à presença de epistasia entre os genes do cromossomo. Para o problema de dobramento, há basicamente três formas de representação (KRASNOGOR et al., 1999; LOPES, 2008):

- Coordenadas cartesianas: este método de codificação descreve um dobramento como um vetor de elementos que representa a posição no espaço dos aminoácidos da sequência. Geralmente, sua utilização é inadequada para algoritmos baseados em população (como os algoritmos genéticos), pois estruturas idênticas ou semelhantes podem ter coordenadas totalmente diferentes;

- Coordenadas internas: uma dada conformação é representada como um conjunto de movimentos dos aminoácidos em relação ao seu predecessor na cadeia. Esta é a representação mais utilizada em abordagens com algoritmos evolucionários para o PDP, podendo ser classificada em dois tipos:
  - Coordenadas absolutas: este tipo de coordenadas é baseado na orientação do eixo da grade onde o dobramento está embutido (bi ou tridimensional). Para a grade tridimensional, este sistema de coordenadas é definido através do seguinte conjunto: $\{N, S, L, O, F, T\}$, correspondendo aos movimentos norte, sul, leste, oeste, para frente e para trás.
  - Coordenadas relativas: este tipo define a posição de cada aminoácido da cadeia em relação ao movimento do seu predecessor. O conjunto de movimentos possíveis é: $\{F, E, D, C, B\}$, correspondendo a para frente (continuando no mesmo sentido do último movimento realizado), à esquerda, à direita, para cima e para baixo.
- Matriz de distâncias: descreve a estrutura de um dobramento através de uma matriz quadrada que representa a distância entre aminoácidos. Este tipo de representação é raramente utilizado na literatura (PICCOLBONI; MAURI, 1998).

Um estudo sobre os tipos de sistema de coordenadas internadas foi realizado por (KRASNOGOR et al., 1999), utilizando vários tipos de grade. Segundo os resultados apresentados neste estudo, a codificação em coordenadas relativas internas pode levar o algoritmo genético a resultados melhores. Com base nisto, foram utilizadas coordenadas internas relativas neste trabalho. Neste sistema de coordenadas, uma dada conformação da proteína é representada como sendo um conjunto de movimentos sobre uma treliça cúbica. Assim, a posição de cada aminoácido na cadeia é relativa ao seu predecessor. Conforme mencionado na Seção 2.3.4, no modelo 3DHP-SC os aminoácidos de uma proteína são representados por um *backbone* (B) e uma cadeia lateral, hidrofóbica (H) ou polar (P). No espaço tridimensional há cinco movimentos relativos possíveis para o *backbone* (**E**squerda, **F**rente, **D**ireita, **B**aixo, **C**ima) e outros cinco para a cadeia lateral, relativos ao *backbone* (**e**squerda, **f**rente, **d**ireita, **b**aixo, **c**ima).

Para ilustrar os movimentos na treliça cúbica, as Figuras 12(a) e 12(b) mostram os possíveis movimentos para o *backbone* e cadeia lateral, respectivamente.

Portanto, a combinação dos possíveis movimentos do *backbone* e da cadeia lateral leva a 25 possibilidades, representadas pelo conjunto: {Ee, Ef, Ed, Eb, Ec, Fe, Ff, Fd,

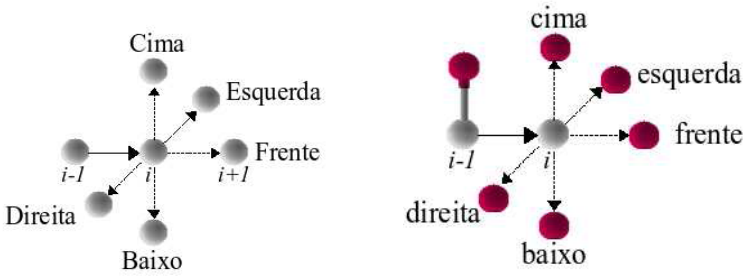


Figura 12: Exemplo de movimentos relativos para *backbone* (a) e cadeia lateral (b)
Fonte: Autoria própria

Fb, Dc, De, Df, Dd, Db, Dc, Be, Bf, Bd, Bb, Bc, Ce, Cf, Cd, Cb, Cc}. Cada elemento deste conjunto é representado como um único símbolo, conforme a Tabela 2, sendo este o alfabeto utilizado para codificar o cromossomo do AG. Os símbolos, por sua vez, são representados na forma binária com 5 *bits* (número de *bits* necessário para representar o alfabeto de 25 movimentos, com valores de 0 a 24). Os valores inválidos ($valor \geq 25 \wedge valor < 2^{25}$) são substituídos pelo maior valor possível ($valor = 24$). Considerando o dobramento de uma proteína com n aminoácidos, um cromossomo com $n-1$ genes representará o conjunto de movimentos do *backbone* e da cadeia lateral na treliça.

Tabela 2: Esquema de codificação das coordenadas relativas internas para o PDP.

Movimentos		*Backbone*				
		E	**F**	**D**	**B**	**C**
Cadeia Lateral	**e**	0	5	A	F	K
	f	1	6	B	G	L
	d	2	7	C	H	M
	b	3	8	D	I	N
	c	4	9	E	J	O

O fenótipo, ou seja, a representação do dobramento propriamente dito, pode ser decodificado a partir do genótipo, como apresentado na Figura 13. Em outras palavras, a posição dos aminoácidos na treliça cúbica é obtida a partir do cromossomo do indivíduo. A Figura 13 mostra um exemplo de mapeamento genótico→fenótipo para o dobramento de um polipeptídeo fictício. Por questão de clareza, apenas os quatro primeiros movimentos estão indicados no cromossomo e no dobramento. O *backbone* e a cadeia lateral do aminoácido inicial estão indicados também estão indicados na figura. Conforme citado na Seção 2.3.4, as cadeias laterais hidrofóbicas e polares são represen-

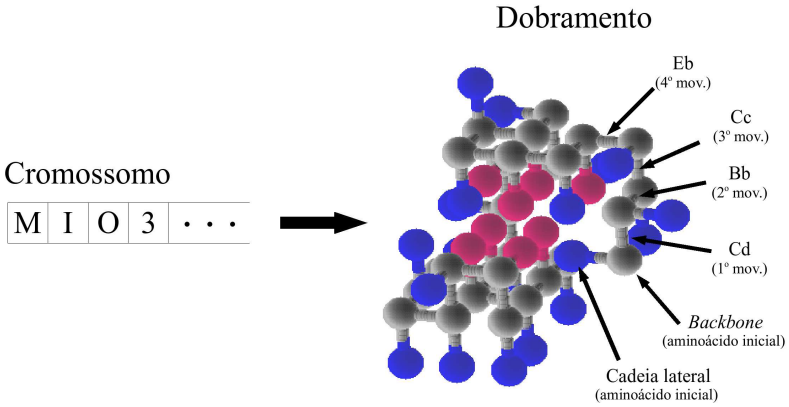


Figura 13: Mapeamento genótipo - fenótipo

Fonte: Autoria própria

tadas, respectivamente, por esferas vermelhas e azuis. O *backbone* e as conexões entre aminoácidos são mostrados em cinza.

Para representar a posição dos aminoácidos na treliça cúbica, as coordenadas cartesianas de cada elemento (*backbone* e cadeia lateral) são definidas por um vetor (x_i, y_i, z_i). Este vetor é obtido a partir do movimento relativo do aminoácido atual e da posição do aminoácido predecessor. Portanto, um procedimento sequencial progressivo é necessário, iniciando desde o primeiro *backbone*, situado na origem do sistema de coordenadas (posição (0, 0, 0)) e com cadeia lateral situada na posição (0, -1, 0).

O pseudo-código do algoritmo de mapeamento genótipo - fenótipo é detalhado de maneira simplificada no Algoritmo 3. O indivíduo é lido e decodificado em um *string* utilizando o alfabeto apresentado na Tabela 2. No próximo passo é construído o espaço tridimensional (matriz 3D) onde os elementos serão posicionados. O *backbone* e a cadeia lateral do primeiro aminoácido são posicionados nas coordenadas (0, 0, 0) e (0, -1, 0), respectivamente. Após a leitura do cromossomo e posicionamento dos elementos do primeiro aminoácido, a construção do fenótipo é realizada, onde os elementos de cada aminoácido são posicionados no espaço tridimensional. Para cada movimento a ser realizado, quatro passos são efetuados. Primeiro, o sentido do movimento é determinado a partir do movimento a ser realizado e do sentido do movimento realizado pelo aminoácido predecessor. Na sequência, as coordenadas do *backbone* do aminoácido são determinadas a partir do sentido do movimento e das coordenadas do aminoácido predecessor. O próximo passo consite na determinação das coordenadas da cadeia lateral do aminoácido a partir movimento e as coordenadas do *backbone*. Fi-

nalmente, os elementos do aminoácido são posicionados no espaço tridimensional a partir das suas coordenadas.

Algoritmo 3 Algoritmo de mapeamento genótipo - fenótipo

```
1: Ler_cromossomo()
2: Construir_espaço3D()
3: //Determinar as coordenadas de cada elemento (backbone e cadeia lateral):
4: Posicionar_1oAminoacido()
5: //Construir fenótipo:
6: Enquanto Movimento a ser realizado Faça
7:    Determinar_sentido()
8:    Determinar_coordenada_backbone()
9:    Determinar_coordenada_SC()
10: Fim Enquanto
11: Posicionar_elementos()
```

3.2 POPULAÇÃO INICIAL

O uso de coordenadas relativas internas para o PDP leva a um problema na inicialização do AG quando a população inicial é gerada. Como a geração é realizada de maneira aleatória, o número de colisões entre elementos *backbone* e cadeias laterais tende a ser grande (BENÍTEZ; LOPES, 2009). Consequentemente, na geração da população inicial não se pode garantir indivíduos válidos (sem colisões). Isto conduz o AG a uma perda de tempo de processamento e geração de conformações inválidas antes que bons resultados possam ser obtidos. Para contornar esta situação, este trabalho propõe um método especializado para a geração da população inicial. A população inicial é dividida em duas partes geradas aleatoriamente, sendo que uma delas é composta de indivíduos livres de colisão (20% da população). A taxa de indivíduos livres de colisão também pode ser configurada pelo usuário através de um parâmetro do arquivo de entrada ($Perc_{popsemColisoes}$). A geração dos indivíduos sem colisão é realizada utilizando uma estratégia de *backtracking*.

O dobramento é representado como um grafo orientado estruturado como uma árvore. Conceitualmente, cada nó da árvore representa uma solução candidata parcial c, desde o primeiro aminoácido da cadeia até o último sendo considerado. Portanto, um caminho até um nó folha representa um dobramento completo. Cada aresta do grafo representa o movimento de cada elemento (*backbone* e cadeia lateral) relativo ao seu predecessor.

A Figura 14 mostra um exemplo de um fragmento do espaço conformacional (ou

espaço de busca) para um polipeptídeo de 4 aminoácidos fictício com estrutura primária HPPH. Neste caso, há 3 ligações peptídicas para interconectar os aminoácidos do polipeptídeo. Assim, a árvore é de profundidade 4.

O espaço de busca completo não poderia ser mostrado na figura, pois sua representação ficaria ilegível (totalizando $25^3 = 15625$ dobramentos possíveis). Assim, diversos nós de profundidade 2, 3 e 4 não foram incluídos na figura para maior clareza.

A sequência de movimentos que levam a um dado dobramento pode ser seguida da raiz "início" até uma folha da árvore, como mostrado pelos caminhos em vermelho e azul na árvore apresentada na Figura 14. Ao final de cada caminho, encontra-se o nó folha que representa o dobramento completo.

O dobramento completo gerado pelos caminhos em vermelho e azul é Db-Df-Bf e Ce-Be-Be, respectivamente. A representação gráfica no espaço tridimensional destes dobramentos é apresentada nas Figuras 15(a) e 15(b). Nestas figuras, o *backbone* e a cadeia lateral do aminoácido inicial estão indicados. Na Figura 15(b), um aspecto importante pode ser observado. Neste caso, o dobramento Ce-Be-Be representa o melhor dobramento para o polipeptídeo HPPH, apresentando uma interação hidrofóbica entre o 2^o e 3^o aminoácidos da cadeia.

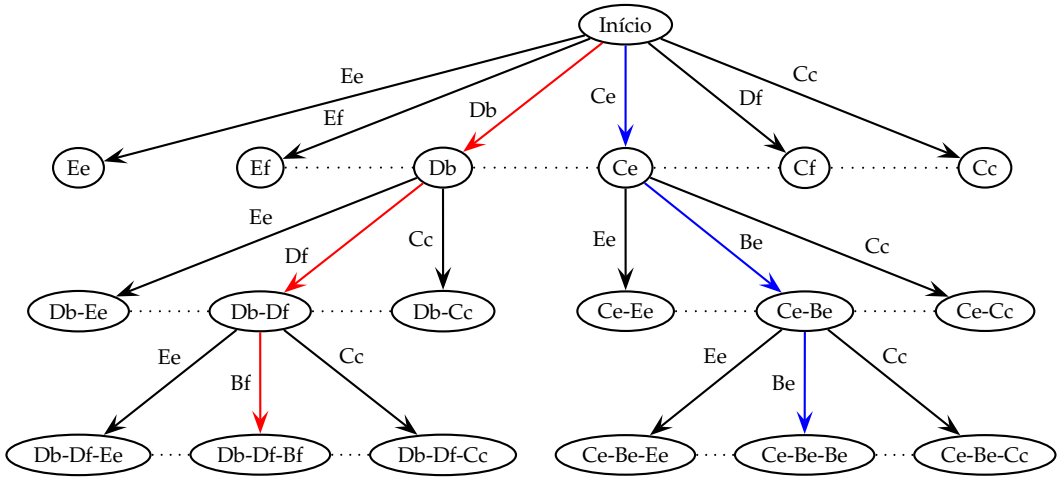
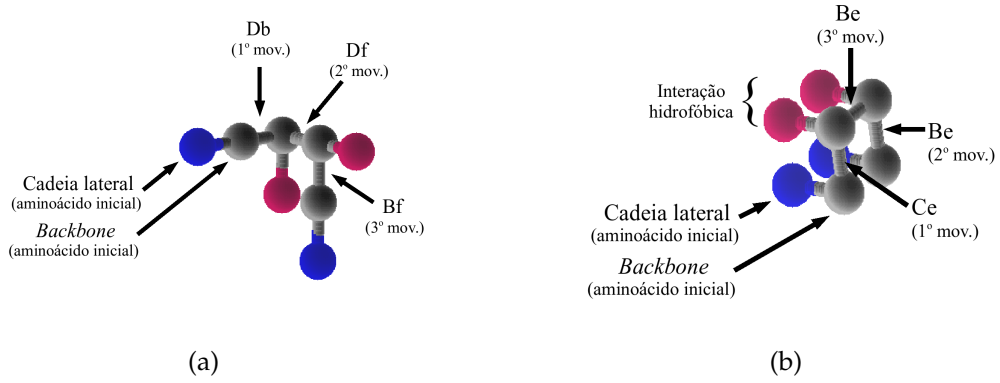


Figura 14: Fragmento do espaço conformacional de um polipeptídeo de 4 aminoácidos

Fonte: Autoria própria

O *backbone* do primeiro aminoácido é situado na origem, com a sua cadeia lateral na posição (0, -1, 0). O movimento do próximo aminoácido é selecionado aleatoriamente. Se o movimento leva a uma colisão com o *backbone* ou com a cadeia lateral de outro aminoácido previamente posicionado na treliça, o *backtracking* é realizado percorrendo



(a) (b)

Figura 15: Representação tridimensional dos dobramentos hipotéticos Db-Df-Bf (a) e Ce-Be-Be (b)

Fonte: Autoria própria

a árvore recursivamente, desde a raiz, seguindo uma busca em profundidade. Em cada nó c, o algoritmo verifica se c é um nó promissor, ou seja, se é uma solução parcial que não leva a uma colisão e que ainda pode levar a uma solução completa. Dobramentos parciais promissores são mantidos, pois levam a soluções válidas. Se c não é promissor, ou seja, com uma ou mais colisões, o algoritmo retorna para o nó anterior do grafo e seleciona aleatoriamente outro nó ainda não explorado. O procedimento é repetido até obter uma solução completa válida. O algoritmo de *backtracking* é detalhado no Algoritmo 4.

Algoritmo 4 Algoritmo de *Backtracking*

```
1:  i ← 0
2:  c ← selecionar_primeiro_nó()
3:  Enquanto solução s não completa Faça
4:     Se promissor(c) Faça
5:        s ← atualizar_solução(c, i)
6:        c ← selecionar_próximo_nó()
7:        i ← i + 1
8:     Senão
9:        podar_subárvore(c, i)
10:       Se verificar_vizinhança(i) Faça
11:          c ← reselecionar_nó(c, i)
12:       Senão
13:          i ← i − 1 //backtracking
14:       Fim Se
15:    Fim Se
16: Fim Enquanto
```

O método proposto para a geração da população inicial consome um tempo de

processamento significativo. Contudo, garante a qualidade dos indivíduos da primeira geração, permitindo a evolução do AG para boas soluções.

É importante ressaltar que o gerador de números aleatórios utilizado no AG desenvolvido é o Mersenne Twister (MATSUMOTO; NISHIMURA, 1998), que é conhecido como um dos melhores para este propósito.

3.3 FUNÇÃO OBJETIVO

A cada geração os indivíduos são avaliados de acordo com a sua capacidade em apresentar uma solução ao PDP. O cromossomo contendo um *string* de movimentos relativos é decodificado em um vetor de coordenadas Cartesianas. Estas, por sua vez, são utilizadas para cálculo de uma função objetivo que fornece o valor de *fitness* do indivíduo.

A função objetivo proposta neste trabalho é composta por termos que levam em consideração não somente a energia livre da conformação, já citada na Equação 4, mas também penaliza o número de colisões. Esta função, apresentada na Equação 8, também incorpora termos que medem a compacidade dos aminoácidos hidrofóbicos e polares. Esta função foi originalmente proposta por (LOPES; SCAPIN, 2005) para o modelo 2DHP e adaptada neste trabalho para o modelo 3DHP-SC.

$$fitness = Energia \cdot RadiusG_H \cdot RadiusG_P \tag{8}$$

Nesta equação, o termo *Energia* leva em consideração o número de contatos hidrofóbicos, interações hidrofílicas, e interações com o *backbone*. Também, o número de colisões (considerado como penalidades) e o peso destas penalidades. $RadiusG_H$ e $RadiusG_P$ representam o raio de giração das cadeias laterais hidrofóbicas e hidrofílicas, respectivamente. Estes termos são descritos detalhadamente a seguir.

3.3.1 Termo *Energia*

O termo *Energia* leva em consideração as interações citadas na Seção 2.3.4. Porém, há também algumas restrições que devem ser satisfeitas para uma conformação ser válida: a conformação não deve apresentar colisões (posição da grade ocupada por mais de um elemento) e os aminoácidos adjacentes na cadeia devem ser adjacentes na grade. Caso uma dada conformação tenha colisões ela é fisicamente inválida. Contudo,

o seu cromossomo correspondente pode carregar algum material genético promissor. Neste caso, há três abordagens para lidar com este problema:

- Remover conformações inválidas durante a evolução do algoritmo. Esta é a alternativa mais simples, porém pode desperdiçar indivíduos promissores e tornar a evolução muito lenta.
- Consertar a conformação inválida. Esta alternativa apresenta um custo computacional elevado, pois o número de alterações a serem realizadas é muito elevado.
- Admitir a conformação como solução válida para o problema, mas com a aplicação de uma penalização. Utilizando esta alternativa, o material genético presente em soluções inválidas pode ser recombinado durante o ciclo evolutivo de modo a formar soluções válidas. Para o PDP há duas maneiras de aplicar penalizações a conformações inválidas: considerando o número de pares de elementos (*backbone* ou cadeia lateral) que ocupam a mesma posição na grade ou considerando o número de posições que possuem mais de um elemento.

A terceira alternativa é utilizada neste trabalho. Neste caso, a conformação é penalizada no seu valor de *fitness* através do termo *Energia*. Para isto, um termo de penalização é decrementado do termo de energia livre H. Esta penalização é composta pelo número de posições na treliça ocupadas por mais de um elemento (NC – número de colisões), multiplicado por um peso de penalização (PP), conforme apresentado na Equação 9.

$$Energia = H - (NC \cdot PP) \tag{9}$$

É importante ressaltar que a energia livre de um dobramento (H) para o modelo 3DHP-SC, proposta por (LI; KLIMOV; THIRUMALAI, 2002) e já apresentada na Equação 4 (ver Seção 2.3.4 – página 44), considera apenas três tipos de interação (não fazendo distinção entre os tipos de cadeia lateral) . Neste trabalho, é proposta uma maneira mais realista de calcular a energia livre de um dobramento, levando em consideração todos os tipos de interação, como mostrado na Equação 10.

$$H = \varepsilon_{HH} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{HH},a} + \varepsilon_{BB} \cdot \sum_{i=1,j>i+1}^{n} \delta_{r_{ij}^{BB},a} + \varepsilon_{BH} \cdot \sum_{i=1,j\neq i}^{n} \delta_{r_{ij}^{BH},a} + \varepsilon_{BP} \cdot \sum_{i=1,j\neq i}^{n} \delta_{r_{ij}^{BP},a} + \varepsilon_{HP} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{HP},a} + \varepsilon_{PP} \cdot \sum_{i=1,j>i}^{n} \delta_{r_{ij}^{PP},a} \quad (10)$$

Nesta equação, ε_{HH}, ε_{BB} , ε_{BH} , ε_{BP} , ε_{HP} , ε_{PP} são as ponderações para as energias de cada tipo de interação, respectivamente: cadeias laterais hidrofóbicas (HH), *backbone-backone* (BB), *backbone*-cadeia lateral hidrofóbica (BH), *backbone*-cadeia lateral polar (BP), cadeias laterais hidrofóbica-polar (HP), cadeias laterais polares (PP).

Em uma cadeia de n aminoácidos, r_{ij}^{bb}, r_{ij}^{bs} e r_{ij}^{ss} são as distâncias (no espaço tridimensional) entre o i-ésimo e o j-ésimo elementos das interações BB-BB, BB-SC e SC-SC, respectivamente. O operador $\delta_{r_{ij}^{**},a}$ retorna 1 quando a distância entre o i-ésimo e o j-ésimo elementos é igual à constante a. Caso contrário, o operador retorna 0. Para efeitos de simplificação, neste trabalho foi utilizada distância unitária entre os elementos ($a = 1$).

De acordo com (LI; KLIMOV; THIRUMALAI, 2002), o peso para interações hidrofóbicas (ε_{HH}) é negativo. Consequentemente, a energia livre da proteína durante o processo tende a diminuir e a conformação converge para o seu estado nativo, de acordo com a hipótese da termodinâmica de Anfinsen (ANFINSEN, 1973). Neste trabalho, foi considerado o simétrico de H para tratar o problema como maximização.

A matriz de energia livre pode ser escrita a partir da Equação 10:

$$\varepsilon_{ab} = \begin{array}{c} \\ \textbf{SCH} \\ \textbf{SCP} \\ \textbf{BB} \end{array} \begin{array}{c} \begin{array}{ccc} \textbf{SCH} & \textbf{SCP} & \textbf{BB} \end{array} \\ \begin{pmatrix} \varepsilon_{HH} & \varepsilon_{HP} & \varepsilon_{HB} \\ \varepsilon_{PH} & \varepsilon_{PP} & \varepsilon_{PB} \\ \varepsilon_{BH} & \varepsilon_{BP} & \varepsilon_{BB} \end{pmatrix} \end{array} \quad (11)$$

3.3.2 Termo $RadiusG_H$

Uma questão importante a ser levada em consideração ao tentar predizer a estrutura tridimensional de uma proteína (em modelos de treliça) está relacionada com a sua (hiper)superfície de energia (ou *energy landscape*), ou seja, como a energia livre está distribuída ao se considerar todas as possíveis conformações.

O modelo 2DHP original utiliza apenas o número de interações entre cadeias laterais hidrofóbicas para avaliar indivíduos (DILL et al., 1995; LOPES, 2008). Esta abordagem gera grandes regiões planas na superfície de energia (KRASNOGOR et al., 1999)

e muitos mínimos locais, tornando ineficiente métodos de busca local.

Para aumentar a eficiência da busca na superfície de energia foi proposto por (SCAPIN; LOPES, 2007b) o uso do conceito físico de raio de giração, como parte da função objetivo. O raio de giração indica quão compacto se encontra um conjunto de pontos (neste caso, aminoácidos em uma treliça). Conjuntos mais compactos possuem menor raio de giração.

A Equação 12 mostra como o raio de giração entre cadeias laterais hidrofóbicas (Rg_H) é calculado.

$$RG_H = \sqrt{\frac{\sum_{i=1}^{n_H}(x_i - \overline{X})^2 + (y_i - \overline{Y})^2 + (z_i - \overline{Z})^2}{n_H}} \tag{12}$$

Onde,
x_i, y_i e z_i são as coordenadas da cadeia lateral do i-ésimo resíduo hidrofóbico da proteína;
$\overline{X}$, $\overline{Y}$ e $\overline{Z}$ são as médias de todos os x_i, y_i e z_i, respectivamente;
n_H é o número de resíduos hidrofóbicos da proteína.

Para obter um núcleo hidrofóbico compacto, típico em proteínas globulares, o raio de giração de um conjunto de cadeias laterais hidrofóbicas (RG_H) deve ser minimizado, aumentando o número de contatos entre elas. Neste trabalho, o problema é tratado como uma maximização da função de *fitness*. Portanto, para obter tal efeito, é necessário realizar a inversão deste valor de forma que conformações mais compactas possuam maiores valores. Isto é realizado conforme a Equação 13.

$$RadiusG_H = maxRG_H - RG_H \tag{13}$$

Onde, $maxRG_H$ é o valor de raio de giração calculado a partir da proteína totalmente esticada, assumindo que este seja o máximo valor que pode ser alcançado (pior caso).

3.3.3 Termo $RadiusG_P$

O termo $RadiusG_P$ segue o conceito do $RadiusG_H$. Neste caso, as cadeias laterais polares são consideradas para o cálculo do raio de giração. Este termo tem como objetivo fazer com que as cadeias laterais hidrofílicas se afastem do interior da estrutura dobrada, tendendo a se posicionar no exterior da proteína. Este termo é calculado

conforme mostrado na Equação 14.

$$RG_P = \sqrt{\frac{\sum_{i=1}^{n_P}(x_i - \overline{X})^2 + (y_i - \overline{Y})^2 + (z_i - \overline{Z})^2}{n_P}} \tag{14}$$

Onde,
x_i, y_i e z_i são as coordenadas da cadeia lateral do i-ésimo resíduo polar da proteína;
$\overline{X}$, $\overline{Y}$ e $\overline{Z}$ são as médias de todos os x_i, y_i e z_i, respectivamente;
n_P é o número de resíduos polares da proteína.

Conforme citado na Seção 2.3.3, os resíduos polares tendem a se posicionar no exterior da proteína, protegendo aos resíduos hidrofóbicos do contato com o meio aquoso. Portanto, RG_P deve ser maior que RG_H para que uma determinada conformação seja promissora.

Se o RG_P for menor que RG_H para uma determinada conformação, os resíduos polares estão mais agrupados do que os hidrofóbicos. Portanto, ela sofrerá uma penalização, diminuindo o seu valor de *fitness*.

Entretanto, não se sabe ao certo qual é a influência do raio de giração dos resíduos polares no processo de dobramento de proteínas. Desta maneira, quando o valor de RG_P for maior que RG_H, este termo não influencia a função objetivo, pois $RadiusG_P$ recebe o valor unitário. A Equação 15 apresenta o cálculo realizado para a obtenção do termo $RadiusG_P$.

$$RadiusG_P = \begin{cases} 1 & \text{, se } (RG_P - RG_H \geq 0) \\ \frac{1}{1-(RG_P - RG_H)} & \text{, caso contrário} \end{cases} \tag{15}$$

3.3.4 Exemplo de cálculo da função de *fitness*

Após a descrição dos termos que compõem a função de *fitness* proposta, um exemplo do seu cálculo é mostrado nesta seção. Para isto será considerada a conformação de um poliptídeo hipotético de 27 aminoácidos (HHHPPHHHHPPPHPHPPHHPPHPPPHH), apresentada na Figura 16. É importante lembrar que o *backbone* do primeiro aminoácido é situado na origem do sistema de coordenadas (posição (0, 0, 0)) e com a sua cadeia lateral situada na posição (0, -1, 0).

Para a realização do cálculo do termo *Energia* da Equação 9, é necessário calcu-

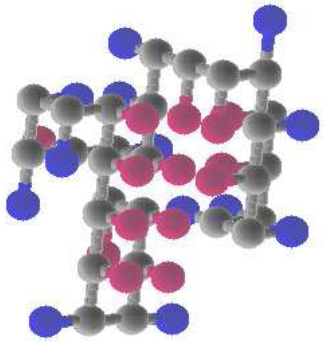


Figura 16: Exemplo de conformação

Fonte: Autoria própria

lar o valor da energia livre da conformação (H) (Equação 10). Para isto, os pesos da energia livre e o número de interações de cada tipo (HH,BB, BH,BP, HP,PP) devem ser conhecidos. A Equação 16 apresenta a matriz de pesos da energia livre utilizada neste exemplo, cujos valores foram obtidos no experimento apresentado na Seção 4.2.2. Esta matriz inclui interações de atração (entre cadeias laterais hidrofóbicas – HH, entre cadeias laterais polares – PP, *backbone-backbone*–BB e *backbone*- cadeia lateral polar–BP) e repulsão (entre cadeias laterais hidrofóbicas e o *backbone* – HB ou BH, e cadeias laterais hidrofóbica-polar – HP ou PH). A Tabela 3 apresenta o número de interações de cada tipo, obtidos a partir da análise da Figura 16 e da Tabela 4.

$$\varepsilon_{ab} = \begin{array}{c} \\ \textbf{SCH} \\ \textbf{SCP} \\ \textbf{BB} \end{array} \begin{array}{c} \begin{array}{ccc} \textbf{SCH} & \textbf{SCP} & \textbf{BB} \end{array} \\ \begin{pmatrix} 10 & -3 & -3 \\ -3 & 1 & 1 \\ -3 & 1 & 1 \end{pmatrix} \end{array} \tag{16}$$

Tabela 3: Número de interações entre elementos da conformação apresentada na Figura 16.

Tipo	Número
HH	16
PP	3
PH	0
BH	4
BB	7

Conhecendo a matriz de pesos e o número de interações, é possível calcular a energia livre da conformação (H) através da Equação 10, obtendo-se $H = 16 \cdot 10 + 7 \cdot 1 + 4 \cdot$

$(-3)+6\cdot 1+0\cdot(-3)=164$.

O termo *Energia* da função de $fitness$ pode ser, então, calculado a partir de H e supondo que o o peso de penalização utilizado seja $PP=10$, obtem-se $Energia=164-(0\cdot 10)=164$.

Para realizar o cálculo dos termos $RadiusG_H$ e $RadiusG_P$, é necessário conhecer as coordenadas da cadeia lateral de cada aminoácido da sequência do polipeptídeo. A Tabela 4 apresenta as coordenadas de cada elemento (B, H e P para *backbone*, cadeia lateral hidrofóbica e cadeia lateral polar, respectivamente).

Tabela 4: Coordenadas cartesianas de cada elemento da conformação apresentada na Figura 16.

Aminoácido	Elemento	Coordenadas	Aminoácido	Elemento	Coordenadas
1	B	(0,0,0)	15	B	(0,1,1)
	H	(0,-1,0)		H	(0,1,2)
2	B	(0,0,1)	16	B	(0,2,1)
	H	(0,0,2)		P	(0,2,0)
3	B	(0,-1,1)	17	B	(0,3,1)
	H	(0,-1,2)		P	(0,3,0)
4	B	(0,-2,1)	18	B	(0,3,2)
	P	(-1,-2,1)		H	(0,2,2)
5	B	(1,-2,1)	19	B	(0,3,3)
	H	(2,-2,1)		H	(0,2,3)
6	B	(1,-1,1)	20	B	(-1,3,3)
	H	(1,-1,2)		P	(-1,4,3)
7	B	(1,0,1)	21	B	(-1,2,3)
	H	(1,0,2)		P	(-2,2,3)
8	B	(1,1,1)	22	B	(-1,1,3)
	H	(1,1,2)		H	(0,1,3)
9	B	(1,2,1)	23	B	(-1,0,3)
	H	(1,2,2)		P	(-2,0,3)
10	B	(2,2,1)	24	B	(-1,0,2)
	P	(2,1,1)		P	(-1,0,1)
11	B	(2,2,0)	25	B	(-2,0,2)
	P	(1,2,0)		P	(-2,0,1)
12	B	(2,1,0)	26	B	(-2,1,2)
	P	(2,0,0)		H	(-1,1,2)
13	B	(1,1,0)	27	B	(-2,2,2)
	H	(1,1,-1)		H	(-1,2,2)
14	B	(0,1,0)			
	P	(-1,1,0)			

Sabendo-se que $N_H = 14$, $\overline{X} = 0,21$, $\overline{Y} = 0,71$ e $\overline{Z} = 1,79$, a aplicação da Equação 12 resulta em RG_H=1,6382. O valor $maxRG_H$ é calculado considerando a proteína totalmente esticada. Para esta proteína, $maxRG_H$=8,4491. Assim, o valor do termo $RadiusG_H$ é: $RadiusG_H = 8,4491 - 1,6382 = 6,8109$.

Para calcular o termo $RadiusG_P$, é necessário calcular o RG_P, considerando apenas as cadeias laterais polares. Sabendo-se que $N_P = 13$, $\overline{X} = -0,23$, $\overline{Y} = 0,85$ e $\overline{Z} = 1,08$, a aplicação da Equação 14 resulta em $RG_P = 2,5256$. Ao submeter os valores RG_H e RG_P à Equação 15, verifica-se que a primeira condição é satisfeita. Portanto, $RadiusG_P = 1$.

Finalmente, o $fitness$ pode ser calculado a partir dos valores obtidos de H, $RadiusG_H$ e $RadiusG_P$, sendo: $fitness = 164 \cdot 6,8109 \cdot 1 = 1116,99 \approx 1117$.

3.4 MÉTODO DE SELEÇÃO E OPERADORES GENÉTICOS BÁSICOS

Os operadores genéticos modificam indivíduos da população com o objetivo de criar novos indivíduos. Os indivíduos são selecionados através de um método de seleção, de acordo com algum critério previamente estabelecido. É importante ressaltar que o método de seleção não é um operador genético, pois ocorre após a aplicação da função de avaliação e antes da aplicação dos operadores genéticos.

3.4.1 Método de Seleção

A implementação proposta utiliza o método de seleção por torneio estocástico. A seleção por torneio estocástico não é baseada na competição entre todos os indivíduos da população. Este método escolhe aleatoriamente um número de indivíduos da população (tamanho do torneio, representado por um número t ou por uma porcentagem da população). O indivíduo com o maior nível de aptidão no grupo é selecionado.

3.4.2 *Crossover*

Este é o primeiro operador genético aplicado aos indivíduos previamente selecionados pelo método de seleção.

A implementação proposta utiliza o operador de *crossover* de dois pontos. Este operador escolhe 2 pontos de cruzamento de maneira aleatória, dividindo os indivíduos em 3 partes. Os genes localizados entre as duas posições selecionadas são trocados entre os pais de modo a gerar dois novos filhos, conforme mostrado na Figura 17.

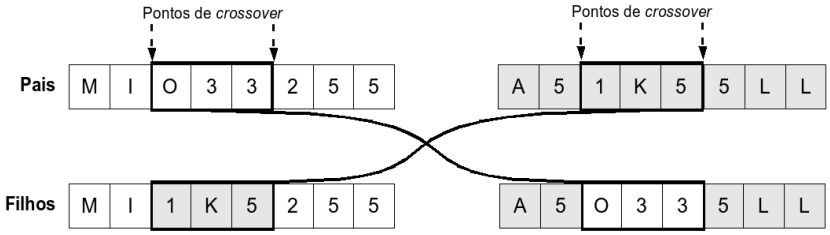


Figura 17: Exemplo de aplicação do operador de *crossover* de dois pontos
Fonte: Autoria própria

Espera-se que a fusão entre os indivíduos permita que alguma parte útil da estrutura de uma conformação possa ser útil para outras.

3.4.3 Mutação

Após a aplicação (ou não) do operador de *crossover*, cada um dos indivíduos recém-gerados (filhos) é submetido ao operador de mutação. O operador de mutação possui um papel secundário, que consiste em restaurar e manter a diversidade genética da população.

A implementação proposta utiliza um operador de mutação multi-ponto que consiste na alteração de um gene ou vários genes do cromossomo simultaneamente. Um indivíduo é submetido a este operador de acordo com o parâmetro de probabilidade de mutação (*pmut*). Caso o indivíduo deva sofrer mutação, o valor atual de um número de genes adjacentes é alterado para qualquer outro valor pertencente ao alfabeto utilizado (ver Seção 3.1 – página 69).

Este operador trata o cromossomo como um anel circular. A posição do primeiro gene a ser mutado é selecionada aleatoriamente e as próximas são envolvidas em sentido horário.

O número de genes adjacentes a serem mutados é determinado em relação a uma distribuição de Poisson com valor médio esperado igual a 3 genes.

A Figuras 18(a) e 18(b) apresentam dois exemplos da aplicação deste operador.

É importante ressaltar que este operador não força a mutação dos genes. Assim, existe a possibilidade do novo valor de um gene ser idêntico ao valor que existia antes do processo de mutação.

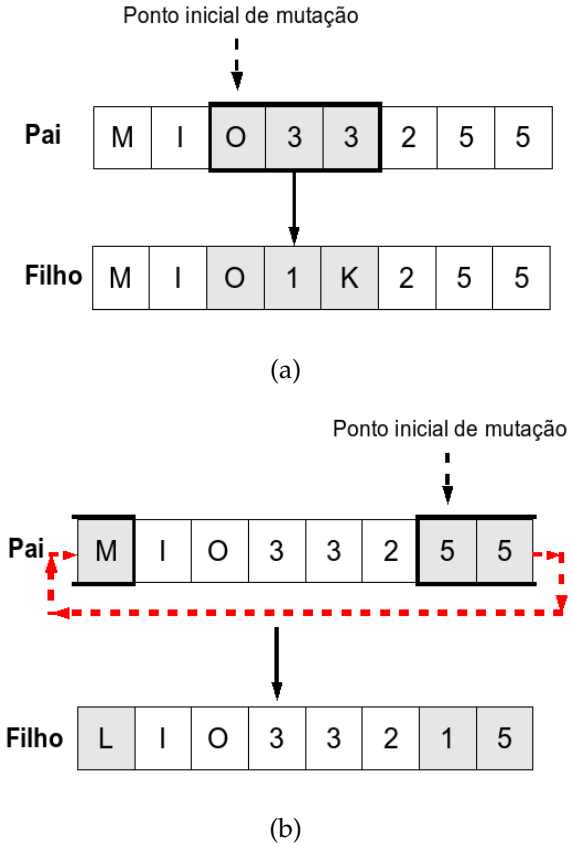


Figura 18: Exemplos de aplicação do operador de mutação multi-ponto
Fonte: Autoria própria

3.5 OPERADORES GENÉTICOS ESPECIAIS

Três operadores genéticos especializados foram desenvolvidos com o objetivo de auxiliar o processo de evolução para a obtenção de melhores resultados. Estes operadores são descritos nas subseções a seguir.

3.5.1 Operador de Mutação Sempre Melhor (MSM)

O operador de Mutação Sempre Melhor (MSM) funciona basicamente da mesma forma que a mutação simples (ver Seção 3.4.3). A única e principal diferença, é que o indivíduo é avaliado novamente a cada mutação realizada. Caso o *fitness* do indivíduo mutado seja maior do que o do original, a mudança é mantida. Caso contrário, o indivíduo original é mantido. Este operador é executado durante um número de tentativas configurado previamente. Os parâmetros do operador especializado MSM são: probabilidade de mutação sempre melhor ($pmut_{MSM}$) e número de tentativas ($tentativas_{MSM}$).

Na maioria das vezes, este tipo de operador especializado consegue melhorar o *fitness* do indivíduo. No pior caso, o *fitness* continuará como estava antes de ser submetido ao operador.

O pseudo-código simplificado do operador de Mutação Sempre Melhor é descrito pelo Algoritmo 5, mostrado a seguir.

Algoritmo 5 Operador de Mutação Sempre Melhor (MSM)

```
 1: fitness ← Avalia_indivíduo(individuo)
 2: Backup_indivíduo(individuo)
 3: FLAG_melhor ← FALSO
 4: Enquanto tentativas < tentativas_MSM Faça
 5:    mutação_indivíduo(individuo)
 6:    individuo_mutado.fitness ← Avalia_indivíduo(individuo)
 7:    Se individuo_mutado.fitness > fitness Faça
 8:       FLAG_melhor ← VERDADEIRO
 9:       individuo_melhor ← individuo
10:       fitness ← individuo_mutado.fitness
11:    Fim Se
12: Fim Enquanto
13: Se FLAG_melhor = VERDADEIRO Faça
14:    individuo ← individuo_melhor
15: Senão
16:    Restaurar_indivíduo()
17: Fim Se
```

No primeiro passo realizado pelo operador de Mutação Sempre Melhor, o indivíduo é submetido à função *fitness* "*Avalia_indivíduo(individuo)*", de acordo com a Seção 3.3, de modo a poder compará-lo com o indivíduo mutado. Depois, uma cópia do indivíduo original é realizada através da função "*Backup_indivíduo*(*individuo*)", de modo a poder restaurá-lo, caso o operador MSM não consiga melhorar a qualidade do mesmo. O próximo estado do operador MSM é um laço de repetição que realiza a mutação do indivíduo. Após a mutação, o indivíduo mutado é submetido à função *fitness*. Quando a condição de término for alcançada ($tentativas = tentativas\_MSM$), o algoritmo verifica se a qualidade do indivíduo melhorou com a aplicação do operador MSM. Caso o indivíduo não tenha melhorado, o indivíduo original é restaurado através da execução da função "*Restaurar_indivíduo*()".

3.5.2 Operador de Mutação Sem Colisão (MSC)

O operador de mutação especializado denominado Mutação Sem Colisão (MSC) funciona basicamente como o operador de mutação especializado MSM (Seção 3.5.1).

Entretanto, este operador avalia o indivíduo mutado, verificando se o mesmo apresenta colisões. Caso o indivíduo mutado não possua colisão, a mudança é mantida. Caso contrário, o indivíduo original é restaurado. Os parâmetros do operador especializado MSC são: probabilidade de mutação sem colisão ($pmut_{MSC}$) e número de tentativas ($tentativas_{MSC}$).

O pseudo-código simplificado do operador de Mutação Sem Colisão é descrito pelo Algoritmo 6, mostrado a seguir.

Algoritmo 6 Operador de Mutação Sem Colisão (MSC)

```
 1: FLAG_colisao = FALSO
 2: FLAG_backup = FALSO
 3: Mapeamento_genotipo_fenotipo(individuo)
 4: Se Verifica_colisoes() Faça
 5:    FLAG_colisao = VERDADEIRO
 6:    Backup_individuo(individuo)
 7:    Enquanto tentativas < tentativas_MSC & FLAG_colisao = VERDADEIRO Faça
 8:       mutacao_individuo(individuo)
 9:       Mapeamento_genotipo_fenotipo(individuo)
10:       Se !Verifica_colisoes() Faça
11:          FLAG_colisao = FALSO
12:       Fim Se
13:    Fim Enquanto
14:    Se FLAG_colisao = VERDADEIRO Faça
15:       Restaurar_individuo()
16:    Fim Se
17: Fim Se
```

O operador de Mutação Sem Colisão realiza o mapeamento genótipo→fenótipo, através da função *Mapeamento_genotipo_fenotipo*(*individuo*). Em outras palavras, a representação tridimensional do dobramento é obtida através da decodificação do cromossomo. Depois, o algoritmo verifica se a estrutura tridimensional apresenta colisões entre os resíduos.

Caso a estrutura possua colisões, uma cópia do indivíduo original é realizada, de modo a poder restaurá-lo, caso o operador MSC não consiga gerar um indivíduo sem colisão.

O próximo estado do operador MSC é um laço de repetição que realiza a mutação do indivíduo. Após a mutação, é realizado o mapeamento genótipo→fenótipo verificando se a representação fenotípica do indivíduo apresenta colisões.

Quando a condição de término for alcançada ($tentativas < tentativas\_MSC$

& FLAG_colisao = VERDADEIRO), o algoritmo verifica se o operador MSC conseguiu gerar um indivíduo válido (sem colisões). Caso isto não aconteça, o indivíduo original é restaurado através da execucação da função *Restaurar_indivíduo*().

3.5.3 Operador de Mutação de Cadeia Lateral (MCL)

O operador de Mutação de Cadeia Lateral (MCL) realiza a mutação de um gene do indivíduo selecionado aleatoriamente de acordo com uma probabilidade de mutação ($pmut_{MCL}$). O objetivo deste operador é maximizar o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado.

O movimento relativo da cadeia lateral do aminoácido, representado pelo gene selecionado, é mudado para outro movimento pertencente a um conjunto de movimentos possíveis. Então, o número de contatos hidrofóbicos para a cadeia lateral é determinado. Caso o número de contatos hidrofóbicos obtido seja maior do que o original, a mudança é mantida. Caso contrário, o indivíduo original é restaurado. Este operador é executado para todos os movimentos pertences ao conjunto de movimentos da cadeia lateral possíveis.

O pseudo-código simplificado do operador de Mutação de Cadeia Lateral é descrito pelo Algoritmo 7, mostrado a seguir.

Inicialmente, um aminoácido da cadeia polipeptídica é selecionado aleatoriamente para ser submetido ao processo de mutação, através da função *Seleciona_aminoacido*(). Depois, o operador verifica se a cadeia lateral do aminoácido é hidrofóbica através da função *Verifica_Cadeia_Lateral*(*sequencia*, *aminoacido*, 'H'). Caso a cadeia lateral seja hidrofóbica, uma cópia do indivíduo original é realizada, de modo a poder restaurá-lo, caso o operador MCL não consiga aumentar o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado. Após a verificação da cadeia lateral do aminoácido selecionado, o mapeamento genótipo→fenótipo é realizado através da função *Mapeamento_genotipo_fenotipo*(*individuo*). Em outras palavras, a representação tridimensional do dobramento é obtida através da decodificação do cromossomo. Depois, a função *Calcula_HH*() determina o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado antes de ser submetido à mutação. Em outras palavras, o número de interações hidrofóbicas do indivíduo original é determinado. Na sequência, a função *Verifica_movimentos_possiveis*(*individuo*, *aminoacido*) verifica os movimentos possíveis da cadeia lateral do aminoácido selecionado e monta um conjunto de movimentos possíveis que não geram colisão.

Algoritmo 7 Operador de Mutação de Cadeia Lateral (MCL)

```
1: aminoacido ← Seleciona_aminoacido()
2: Se Verifica_Cadeia_Lateral(sequencia, aminoacido, 'H') Faça
3:    FLAG_MCL = FALSO
4:    Backup_individuo(individuo)
5:    Mapeamento_genotipo_fenotipo(individuo)
6:    HH ← Calcular_HH()
7:    HH_max=HH
8:    Conjunto_mov_possiveis ← Verifica_movimentos_possiveis(individuo, aminoacido)
9:    Para cada movimento do Conjunto_mov_possiveis Faça
10:      Mutação_Cadeia_Lateral(individuo, aminoacido, movimento)
11:      Mapeamento_genotipo_fenotipo(individuo)
12:      HH ← Calcula_HH()
13:      Se HH > HH_max Faça
14:         FLAG_MCL = VERDADEIRO
15:         HH_max=HH
16:         individuo_melhor ← individuo
17:      Fim Se
18:   Fim Para
19:   Se FLAG_MCL = VERDADEIRO Faça
20:      individuo ← individuo_melhor
21:   Senão
22:      Restaurar_individuo()
23:   Fim Se
24: Fim Se
```

Após a determinação do conjunto de movimentos possíveis, cada um deles é testado com o objetivo de selecionar o movimento que maximize o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado. Para isto, o mapeamento genótipo → fenótipo e o cálculo do número de interações hidrofóbicas são realizados para todos os movimentos possíveis.

Caso o operador não consiga aumentar o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado, o indivíduo original é restaurado através da execucação da função *Restaurar_indivíduo*().

3.6 ESTRATÉGIAS

3.6.1 Dizimação *hot-boot* (DHB)

Quando um AG fica preso em torno de um máximo local no espaço de busca, espera-se um decréscimo na diversidade genética da população, principalmente como consequência do operador de *crossover*. Algumas vezes, este efeito pode ser contraba-

lanceado pela ação do operador de mutação. Em outras palavras, a mutação faria com que outra região do espaço de busca pudesse ser explorada, mas mesmo assim seria um evento do acaso. No entanto, frequentemente, isto não é suficiente e estratégias adicionais são necessárias para evitar a estagnação da busca.

Quando a população do AG está concentrada em torno de um máximo local, a única maneira de evitar esforço computacional inútil é escapar da região atual e redirecionar o AG para a exploração de outras regiões do espaço de busca. Para isto, é utilizada uma estratégia chamada dizimação e *hot-boot* (DHB), mostrada no Algoritmo 8 e descrita a seguir.

Durante a evolução do AG, uma cópia do melhor indivíduo de cada geração é mantida. Uma evidência indireta de que a população tenha possivelmente convergido e, consequentemente, a evolução do AG tenha estagnado em um máximo local é quando o melhor indivíduo continuar sendo o mesmo (não melhorar) após várias gerações. A estratégia utilizada verifica se o indivíduo *best-ever* não melhorou de uma geração para a próxima. Caso o melhor indivíduo da geração corrente não for melhor que o da geração anterior, um contador é incrementado em 1. Caso contrário, este contador é zerado. Assim, quando o contador atingir um valor predefinido de gerações ($gen2decimate$), a estratégia DHB é ativada.

Neste instante do processo de evolução, antes de gerar a população da próxima geração, 50% da população (selecionada aleatoriamente) é dizimada e substituída por indivíduos gerados de acordo com o mesmo procedimento utilizado para gerar a população inicial (Seção 3.2).

A aplicação da estratégia DHB melhora significativamente a diversidade genética da população e permite que o processo de evolução possa continuar por mais gerações. Finalmente, isto melhora a possibilidade de encontrar melhores soluções. Entretanto, deve ser levado em consideração que os novos indivíduos recém-criados pelo procedimento DHB provavelmente terão valores muito baixos de *fitness*. Apesar da diversidade genética tender a melhorar, a pressão seletiva aumenta devido à grande diferença entre os valores de *fitness* dos indivíduos. Sabe-se que a forte pressão seletiva leva à convergência prematura devido à perda da diversidade genética. Este é o efeito oposto ao desejado. Portanto, é necessário evitar a forte pressão seletiva durante algumas gerações, logo após a dizimação. Isto é conseguido através da diminuição do número de indivíduos que participam no processo de seleção (parâmetro $tourneysize$) para 2 durante um número fixo de gerações (parâmetro $gen2weakTourney$), em seguida

retornando ao seu valor original.

Algoritmo 8 Estratégia DHB

```
1:  Para cada geração Faça
2:    Se primeira geração Faça
3:      contador ← 0
4:      contador2 ← 0
5:    Fim Se
6:    Se FLAG_Dizimação = FALSO & indivíduo best-ever sem mudança Faça
7:      contador ← contador + 1
8:      Se contador = gen2decimate Faça
9:        habilitar DHB
10:       dizimar e substituir 50% da população
11:       contador ← 0
12:       tourneysize ← 2
13:       FLAG_Dizimação ← VERDADEIRO
14:     Fim Se
15:   Fim Se
16:   Se FLAG_Dizimação = VERDADEIRO Faça
17:     contador2 ← contador2 + 1
18:   Fim Se
19:   Se contador2 = gen2weakTourney Faça
20:     contador2 ← 0
21:     restaurar o valor de tourneysize
22:     FLAG_Dizimação ← FALSO
23:   Fim Se
24: Fim Para
```

3.7 FLUXO DE EXECUÇÃO DO AG

Após ter apresentado as características do AG implementado, uma visão geral do funcionamento do mesmo é apresentada nesta seção, de modo a facilitar a compreensão do processo. Para isto, foi utilizada a notação UML.

A Figura 19 apresenta um diagrama de estados mostrando a sequência básica de execução do AG. A população inicial é gerada conforme descrito na Seção 3.2. Depois, a população é submetida à função de *fitness*, de acordo com a Seção 3.3, com o objetivo de avaliar os indivíduos.

O próximo estado consiste na verificação do condição de parada do algoritmo que é quando o número máximo de gerações predefinido tenha sido alcançado. Caso esta condição não tenha sido satisfeita ($Geração \leq TotalGerações$), o algoritmo gerará uma nova população, baseando-se na população atual. A população nova será avaliada a

cada geração. Após a avaliação da população, o contador de gerações *Geração* será incrementado.

Caso a condição de parada seja alcançada (*Geração* $>$ *TotalGerações*), os arquivos de resultados serão exportados (dados de configuração do AG, do melhor indivíduo e para geração de gráficos de evolução e da figura do melhor dobramento).

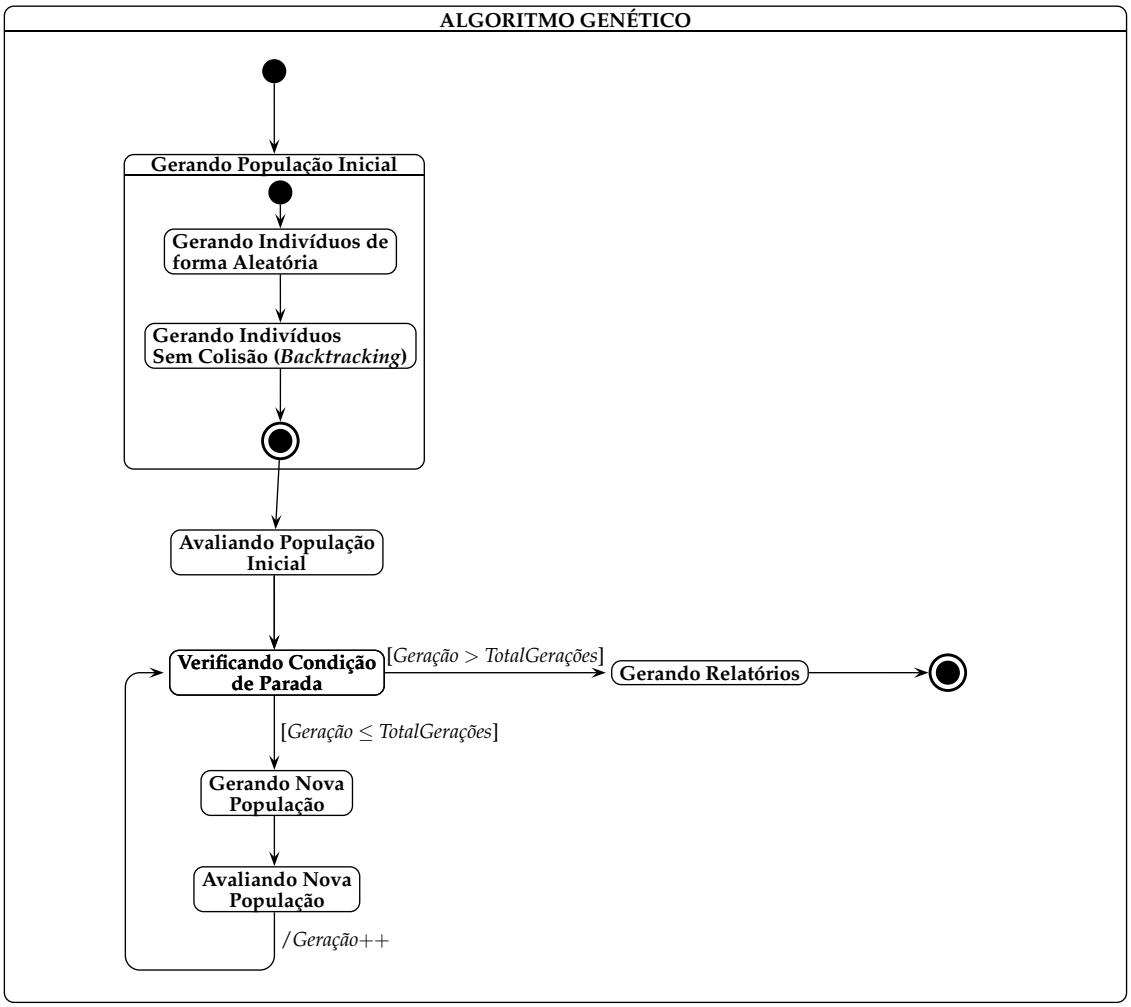


Figura 19: Diagrama de estados do AG

Fonte: Autoria própria

A Figura 20 apresenta um diagrama de estados do processo de geração de cada nova população (estado "Gerando Nova População" do diagrama apresentado na Figura 19), de modo a esclarecer as interações decorrentes da aplicação dos operadores e das estratégias implementadas.

Inicialmente, os invidíduos da população atual são selecionados para serem sub-

metidos aos operadores genéticos básicos de *crossover* e mutação, descritos nas Seções 3.4.2 e 3.4.3, respectivamente.

No próximo passo da evolução, os indivíduos selecionados são submetidos aos operadores especiais habilitados através do arquivo de parâmetros de entrada: MSM , MSC e MCL, descritos nas Seções 3.5.1, 3.5.2 e 3.5.3, respectivamente.

Na sequência é verificado se a estratégia de dizimação será aplicada a esta população. Caso positivo, o AG passa para o estado "Avaliando População", onde a população é submetida à função de *fitness* com o objetivo de avaliar os indivíduos e, consequentemente, atualizar os dados do indivíduo *best-ever*. Na sequência, o algoritmo verifica se o indivíduo *best-ever* foi atualizado. Caso tenha sido depois da aplicação dos operadores genéticos, o parâmetro de tamanho de torneio (*TamTorneio*) é restaurado. Caso contrário, o algoritmo verifica se será necessário aplicar a estratégia de dizimação na população. Caso a dizimação seja aplicada, o algoritmo prossegue para o estado "Ativando DHB", onde a população é submetida à estratégia de dizimação e *hot-boot*. Depois, o algoritmo passa para o estado "Alterando Parâmetro *TamTorneio*", onde o tamanho de torneio é modificado.

Após os indivíduos terem sido submetidos aos operadores genéticos e à estratégia de dizimação, o algoritmo verifica o número de indivíduos que já foram selecionados para fazerem parte da população da próxima geração. Caso este número for o tamanho da população predefinida pelo usuário, o estado "Gerando Nova População" é encerrado e o algoritmo prossegue, de acordo com a Figura 19. Caso contrário, novos indivíduos são selecionados e submetidos aos operadores genéticos até completar a população.

3.8 PARALELIZAÇÃO EM *CLUSTER* BEOWULF

Esta seção apresenta a descrição detalhada da implementação das versões paralelas do algoritmo genético proposto em *Cluster* Beowulf. As versões paralelas foram desenvolvidas seguindo a metodologia proposta por (ROOSTA, 1999), conforme mencionado na Seção 2.5.4. Primeiramente, a versão sequencial do algoritmo genético foi desenvolvida, segundo a descrição apresentada na Seção 2.4 e neste capítulo.

Os algoritmos foram desenvolvidos em linguagem de programação ANSI-C, sobre o sistema operacional Linux, utilizando a biblioteca de passagem de mensagens MPICH2.

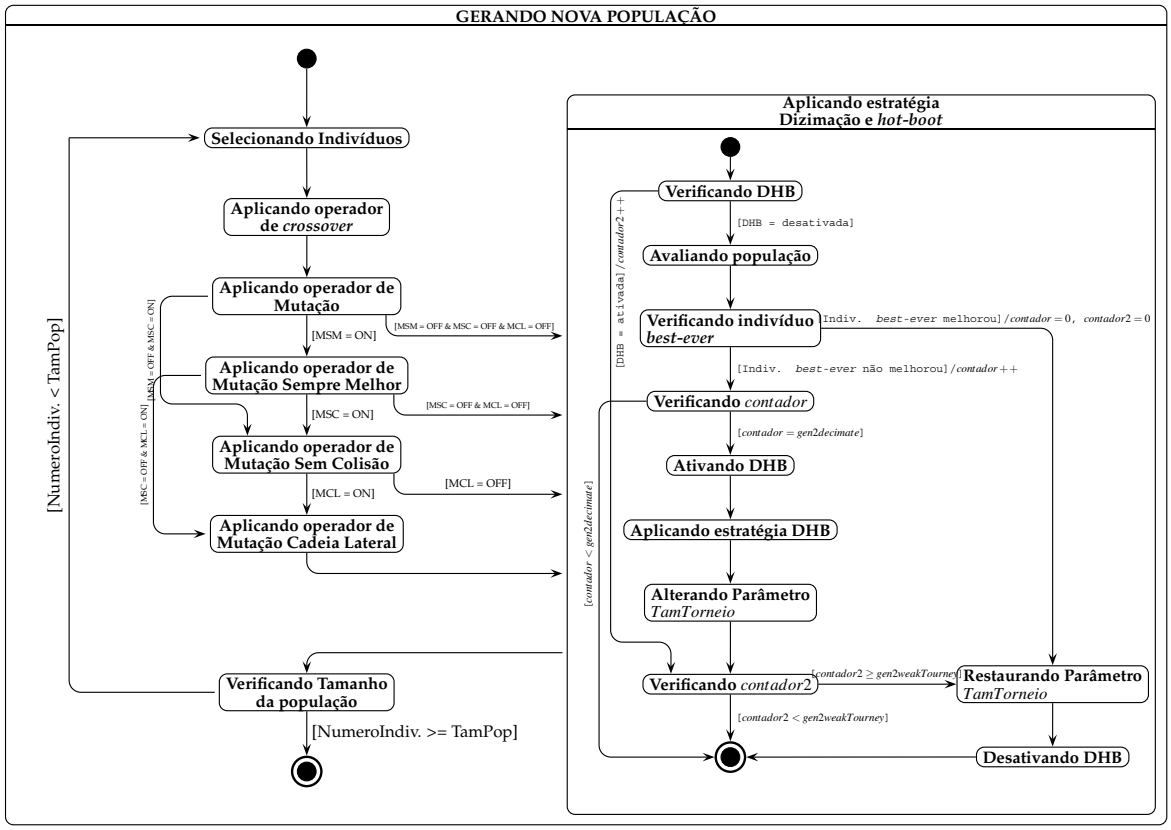


Figura 20: Diagrama de estados do estado "Gerando Nova População"
Fonte: Autoria própria

Conforme mencionado na Seção 2.5.2, um *Cluster* Beowulf é composto de vários nós de computação. Neste trabalho, cada nó de computação possui quatro núcleos de processamento, onde cada núcleo é denominado de processo. Os processos são integrados virtualmente utilizando o mecanismo de passagem de mensagens .

Cada processo é identificado com um número sequencial denominado de *rank* e ao conjunto de processos conectados dá-se o nome de comunicador (*communicator*). Utilizando o padrão MPI, uma cópia do programa em execução é enviada automaticamente para cada um dos processos. Em outras palavras, os processos são indepentendes e executam o mesmo algoritmo. O MPI apresenta um elevado desempenho, flexibilidade e se encarrega do tráfego das mensagens através da rede de interconexão entre os núcleos de processamento. Entretanto, o MPI não oferece um mecanismo que realize a divisão de tarefas automaticamente. Assim, o desenvolvedor é responsável por especificar a função ou tarefa para cada processo, assim como o protocolo de comunicação entre os processos. Isto deixa flexível a implementação da arquitetura do algoritmo.

3.8.1 Implementações paralelas do Algoritmo Genético

AGs possuem a capacidade de encontrar boas soluções em um tempo de processamento razoável. Porém, este tempo aumenta consideravelmente para problemas complexos, como é o caso do PDP. Para tais situações, AGs paralelos têm sido utilizados com mais eficácia do que as versões sequenciais. Neste trabalho, foram desenvolvidas duas abordagens de AG paralelo: AGP mestre-escravo e AGP hierárquico.

A primeira abordagem é do tipo mestre-escravo síncrono (AGP–ME). Este modelo é um sistema global de uma única população, onde o processo mestre divide a carga de processamento entre vários processos escravos, cada um executado em um processador diferente. A Figura 21 apresenta o modelo AGP–ME composto de um mestre e n escravos.

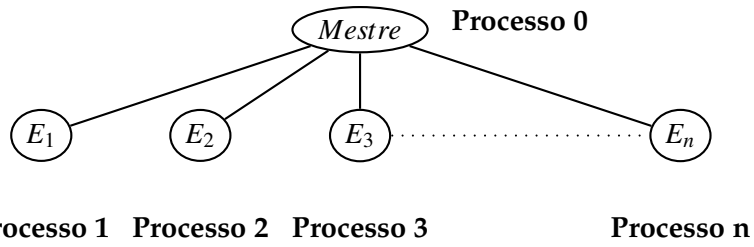


Figura 21: Modelo AGP mestre-escravo
Fonte: Autoria própria

Conforme mencionado na Seção 2.4.6, esta abordagem é particularmente interessante para problemas onde a computação da função de *fitness* é muito custosa, como é o caso deste trabalho (ver Seção 3.3). Também é importante ressaltar que este modelo é eficiente desde que a sobrecarga de comunicação seja desprezível em relação ao tempo de processamento da função objetivo.

Com o objetivo de apresentar como a comunicação é estabelecida entre os processos mestre e escravos, as Figuras 22 e 23 apresentam o diagrama de estados dos processos mestre e escravos, respectivamente.

O processo mestre é responsável por inicializar a população, executar o procedimento de seleção e os operadores genéticos (*crossover* e mutação) e por distribuir os indivíduos aos escravos após a geração das populações inicial e nova. Os escravos são responsáveis somente por processar a função de *fitness* de cada indivíduo recebido.

A distribuição de indivíduos pelo processo mestre inicia-se pela divisão de carga

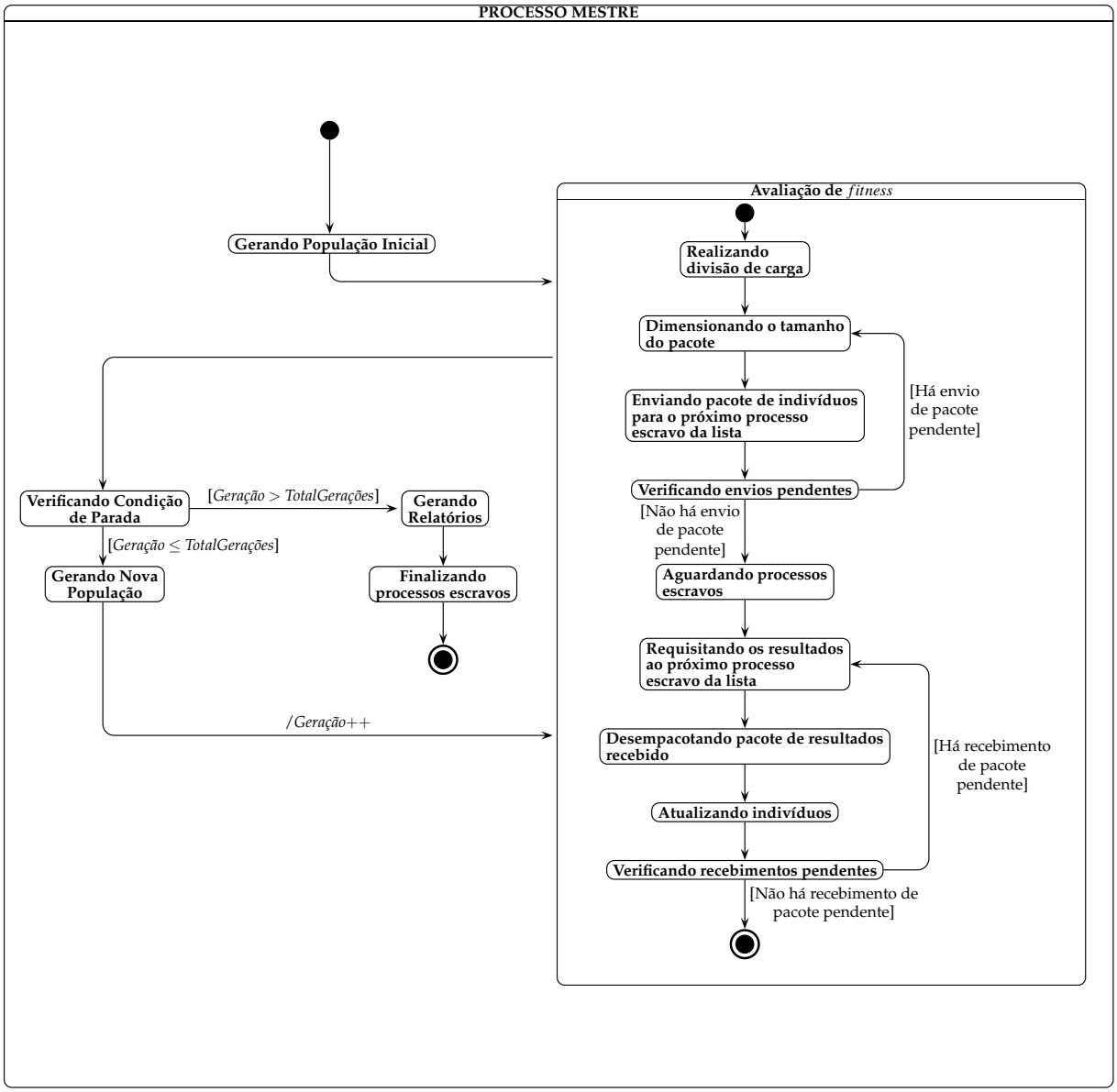


Figura 22: Diagrama de estados do processo mestre

Fonte: Autoria própria

de acordo com o número de indivíduos a serem avaliados e o número de processadores escravos disponíveis. Todos os indivíduos são avaliados após a geração da população inicial, e apenas os indivíduos que foram submetidos aos operadores genéticos são avaliados após a geração da população nova de cada geração. O próximo passo consiste no dimensionamento do tamanho do pacote de indivíduos a ser enviado para cada processador escravo. O pacote contém o cromossomo e identificação (*id*) de todos os indivíduos correspondentes a cada processo escravo. A Figura 24 ilustra como é formado o pacote de indivíduos.

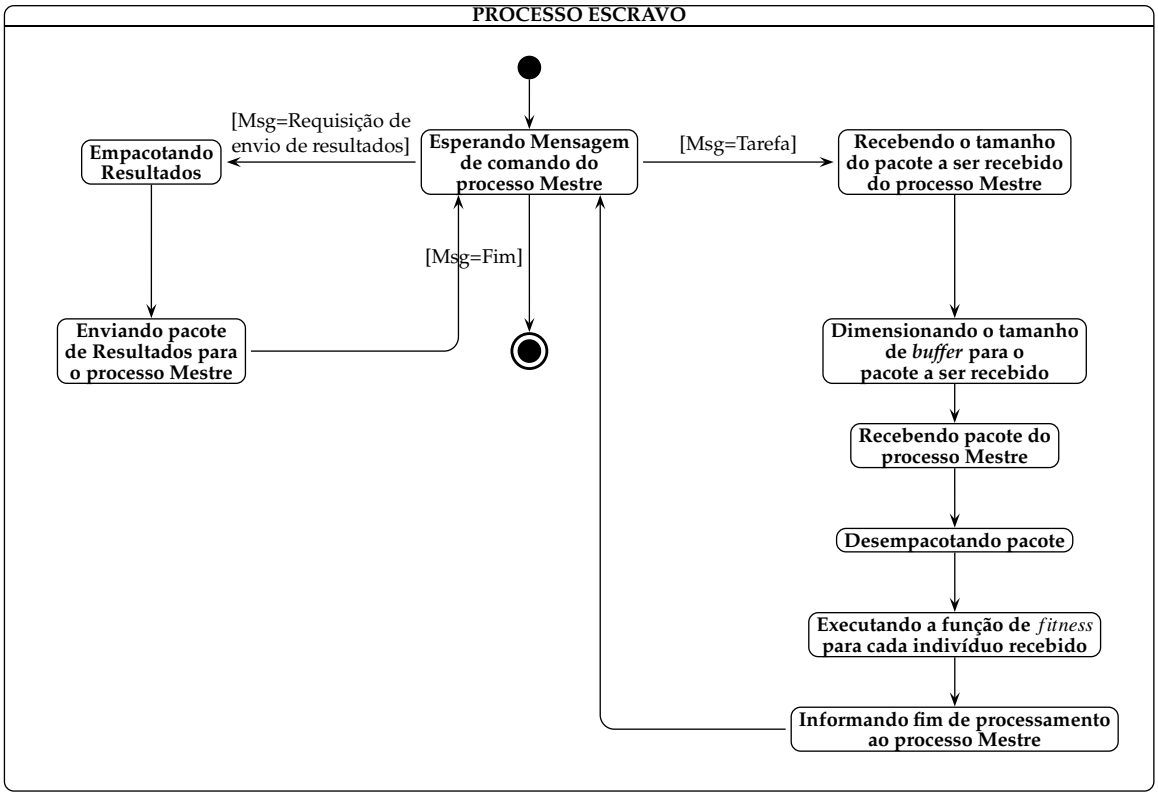


Figura 23: Diagrama de estados dos processos escravos

Fonte: Autoria própria

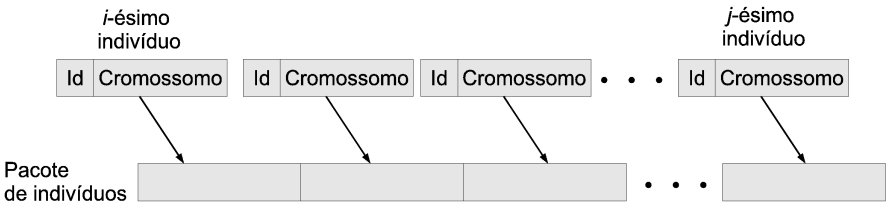


Figura 24: Pacote de indivíduos

Fonte: Autoria própria

O próximo passo consiste no envio dos pacotes de indivíduos para os processos escravos. Um comando de início de transferência e o tamanho do pacote são enviados antes de enviar o pacote de indivíduos. Para realizar o envio dos pacotes é utilizada uma função de envio bloqueante da biblioteca MPICH2 (ver Anexo B). A mensagem enviada pelo mestre é formada por duas partes:

- Envelope: representa o endereço do processo escravo destino formado pela identificação do processo (*rank*), rótulo da mensagem (*Tag*) e o comunicador (*communicator*) do grupo;
- Dado: representa o pacote de indivíduos correspondente.

Caso todos os pacotes tenham sido enviados para os seus respectivos processos escravos, o processo mestre fica aguardando a resposta de finalização de processamento de cada processo escravo. A cada resposta de finalização recebida, uma lista de escravos prontos é atualizada. Enquanto esta lista não estiver completa, o processo mestre fica aguardando a resposta dos escravos que ainda não finalizaram o processamento das funções de *fitness*. Caso contrário, o mestre faz a requisição dos pacotes de resultados para cada processo escravo. Após o recebimento de cada pacote, o mestre realiza o desempacotamento do pacote recebido e atualização do $fitness$ dos indivíduos, a partir do *id* e $fitness$ de cada indivíduo. Caso a condição de parada seja satisfeita, o processo mestre envia um comando de finalização para todos os processos escravos e finaliza o processamento do programa.

A Figura 23 mostra o diagrama de estados dos processos escravos. Eles esperam a mensagem de comando do processo mestre. O comando pode ser de execução de tarefa (recebimento de pacote de indivíduos e avaliação dos indivíduos recebidos), requisição de envio de resultados ($fitness$ dos indivíduos) ou de finalização de processamento. Caso o comando seja de execução de tarefa, o processo escravo aguarda o recebimento do tamanho do pacote de indivíduos a ser recebido do processo mestre. Após o recebimento do tamanho do pacote, o escravo dimensiona o tamanho do $buffer$ para o pacote. O próximo passo consiste no recebimento do pacote enviado pelo processo mestre. Após o recebimento do pacote, o processo escravo o desempacota e executa a função de $fitness$ para cada indivíduo recebido. Na sequência, o escravo informa ao processo mestre que finalizou o processamento da função de $fitness$ dos indivíduos recebidos e espera pelo recebimento da mensagem de requisição de envio de resultados. Ao receber a mensagem de requisição de envio de resultados, o escravo realiza o empacotamento e envio dos resultados obtidos.

O pacote de resultados é formado pela identificação (*id*) e $fitness$ de cada indivíduo recebido, como mostrado na Figura 25.

Como se pode observar, o modelo AG paralelo do tipo mestre-escravo possui exatamente as mesmas funcionalidades da versão sequencial. Na versão sequencial, tudo

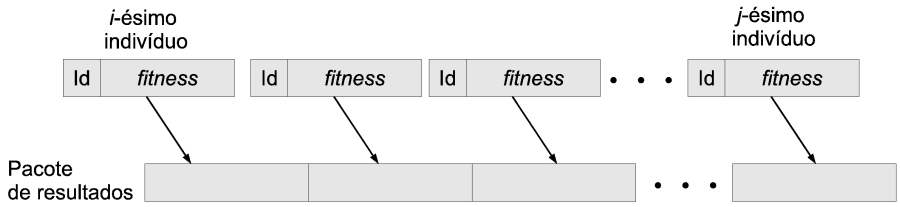


Figura 25: Pacote de resultados

Fonte: Autoria própria

é processado em um único processador. Entretanto, na versão paralela, a carga de processamento é dividida entre vários processadores, sobre a coordenação do processador mestre.

A segunda abordagem desenvolvida é um modelo hierárquico que combina o modelo insular (ou multi-populacional) no nível superior e o mestre-escravo no nível inferior (AGP–HH). No nível inferior a carga de processamento é dividida entre processadores escravos, sob a coordenação do mestre como no modelo descrito anteriormente. No nível superior, cada subpopulação (mestre e escravos correspondentes) pode ser vista como uma ilha trabalhando independentemente das demais. Neste nível, ocorrem migrações de indivíduos entre ilhas, controladas através de uma política de migração composta de quatro parâmetros: *Migration Gap* (número de gerações entre duas migrações sucessivas), *Migration Rate* (número de indivíduos migrantes que participarão em cada migração), critério de Seleção/Substituição e a topologia da conectividade entre ilhas.

A topologia utilizada neste trabalho é composta de quatro ilhas no nível superior e n escravos por ilha no nível inferior. As ilhas são conectadas em anel circular unidirecional, como mostrado na Figura 26.

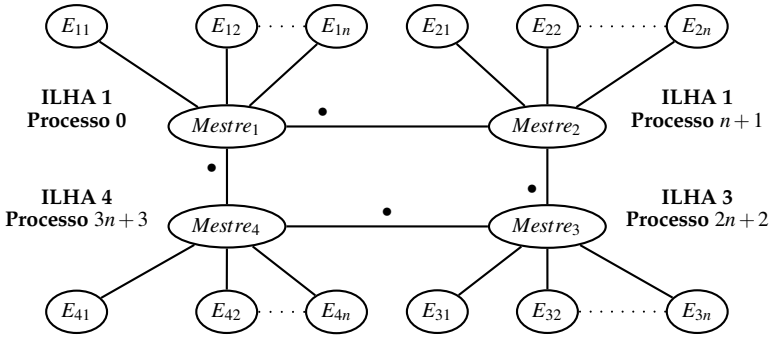


Figura 26: Modelo PGA hierárquico

Fonte: Autoria própria

A Figura 27 apresenta o diagrama de estados do processo mestre de cada ilha.

Conforme mencionado, os processos mestres das ilhas são responsáveis pela distribuição dos indivíduos aos escravos, mas também controlam o do processo migratório de indivíduos entre as ilhas. Para a especificação da topologia de conexão entre as ilhas foi utilizado o conceito de topologia virtual, que consiste no mapeamento dos processos mestres, utilizando coordenadas cartesianas. Cada processo mestre é identificado através de coordenadas cartesianas e conectado com os seus vizinhos em um *grid* virtual. É importante ressaltar que a construção da topologia fica a cargo do desenvolvedor, sendo que outras topologias podem ser implementadas utilizando este conceito como, por exemplo, hipercubos, malhas, toroidal, estrela, entre outras. Um estudo sobre as topologias paralelas de AGPs foi apresentado por (TAVARES; LOPES; ERIG, 2009).

Após a avaliação dos indivíduos da nova população, é verificada a condição para habilitar o processo migratório. Caso a condição seja satisfeita ($Migration\ Gap$ % Geração $\neq 0$), o algoritmo verifica o *rank* do processo mestre. Caso o processo seja o mestre da Ilha 1, é feita a sincronização com os processos mestres das demais ilhas, esperando que estes avisem que estão prontos para participar do processo migratório. Os próximos estados consistem na seleção, empacotamento e envio dos indivíduos emigrantes ao mestre da próxima ilha do anel. Após realizar o envio do pacote de indivíduos emigrantes, o mestre espera pelo pacote de indivíduos imigrantes enviado pelo mestre da ilha anterior do anel. Na sequência, o mestre seleciona e substitui indivíduos pelos indivíduos imigrantes.

Caso o processo seja o mestre de uma das demais ilhas (Ilhas 2, 3 e 4), o próximo estado consiste em informar ao mestre da Ilha 1 que está pronto para participar do

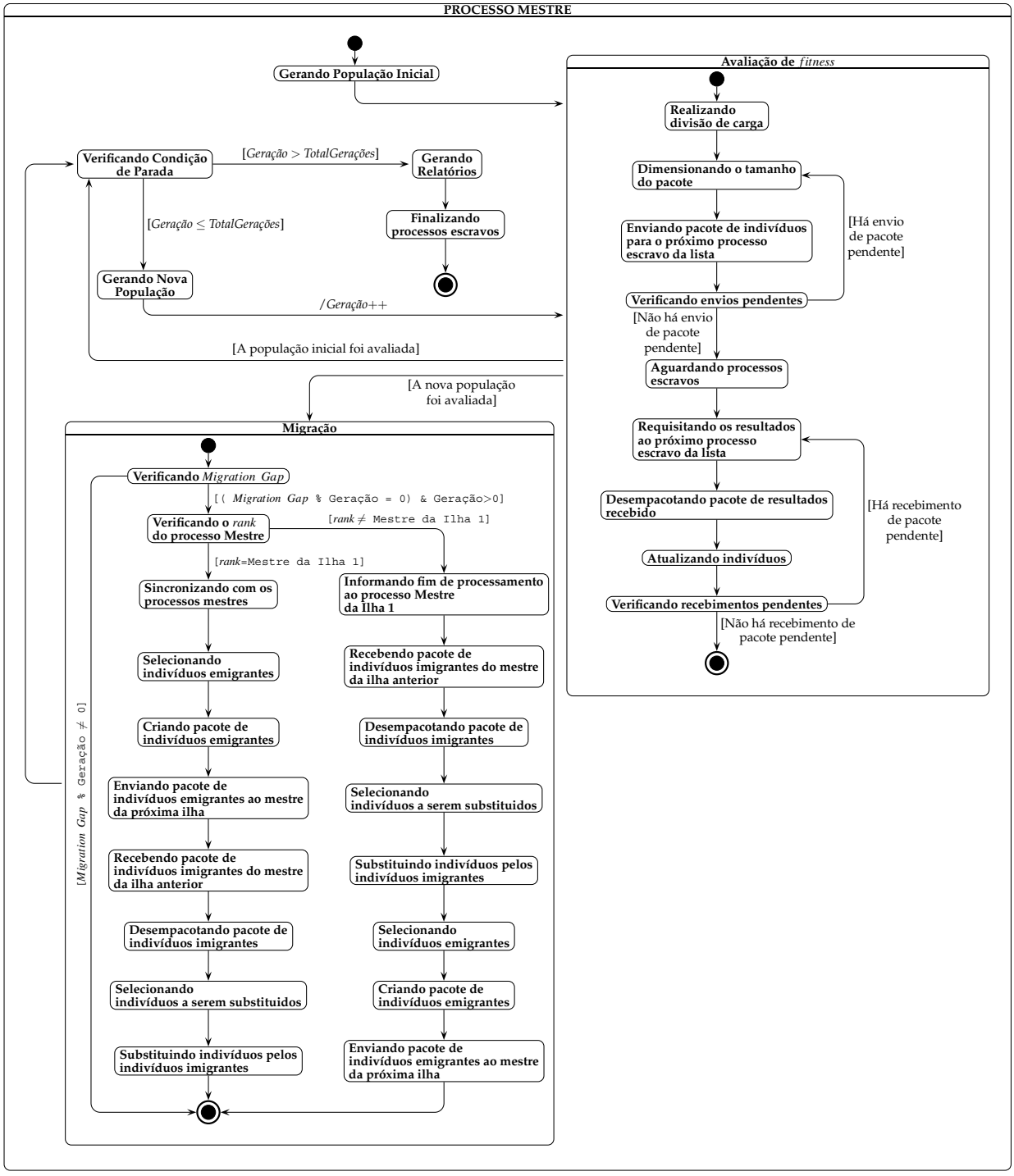


Figura 27: Diagrama de estados dos processos mestres do modelo Hierárquico
Fonte: Autoria própria

processo migratório. O próximo estado consiste no recebimento e desempacotamento do pacote de indivíduos imigrantes enviado pelo mestre da ilha anterior do anel. Na sequência, o mestre seleciona e substitui indivíduos pelos indivíduos imigrantes. Depois, o mestre seleciona, empacota e envia o pacote de indivíduos emigrantes para a

próxima ilha do anel. O pacote de indivíduos migrantes é formado pelo cromossomo e *fitness* de cada indivíduo, como mostrado na Figura 28.

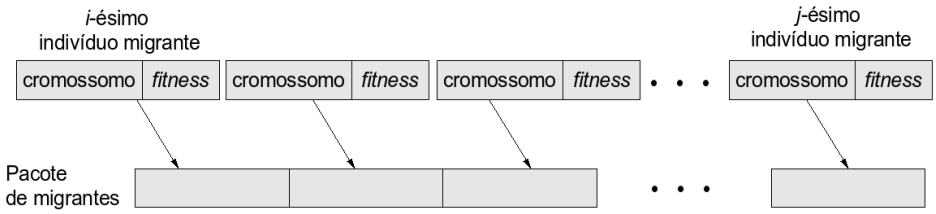


Figura 28: Pacote de indivíduos migrantes

Fonte: Autoria própria

3.9 COMPARAÇÃO COM OUTRA ABORDAGEM EVOLUCIONÁRIA

O método proposto neste trabalho foi comparado com outra técnica de computação evolucionária. Para isto, foi desenvolvido o algoritmo ABC (ver Seção 2.6–página 62). Duas versões paralelas do ABC foram implementadas: ABC mestre-escravo e ABC hierárquico, de modo a serem equivalentes às versões implementadas dos AGs.

A primeira abordagem é do tipo mestre-escravo (ABC–ME), apresentando a mesma estrutura que o AGP–ME (ver Seção 3.8.1–página 95). Este modelo é um sistema de uma única colônia de abelhas, onde o processo mestre divide a carga de processamento entre vários procesos escravos. O processo mestre é responsável por inicializar as fontes de alimento aleatoriamente, gerar novas soluções nas fases do algoritmo (abelhas empregadas, observadores e escoteiras – ver o Algoritmo 2 apresentado na Seção 2.6), aplicar o procedimento de seleção gulosa e distribuir as abelhas (soluções) aos processos escravos. Os processos escravos são responsáveis somente por processar a função de *fitness* das abelhas recebidas. Conforme mencionado, a computação da função de *fitness* é muito custosa, justificando a necessidade a abordagem paralela do algoritmo ABC. No Algoritmo 2 as instruções em negrito e itálico representam os locais de paralelização. Como se pode observar, o modelo ABC–ME apresenta as mesmas funcionalidades da versão sequencial do ABC.

A segunda abordagem é um modelo hierárquico composto de dois níveis (ABC–HH), apresentando a mesma estrutura que o AGP–HH (ver Seção 3.8.1). O nível superior pode ser considerado como um modelo multi-colônia, onde ocorrem migrações de abelhas entre as colônias. No nível inferior a carga de processamento é dividida entre

os escravos, como no modelo descrito anteriormente.

Os experimentos realizados são apresentados na Seção 4.8 do próximo capítulo.

4 EXPERIMENTOS E RESULTADOS

4.1 BALANCEAMENTO DE CARGA DE PROCESSAMENTO

Em um ambiente de processamento paralelo baseado em *cluster*, o tempo necessário para transmitir mensagens entre processos através da rede é significativo, quando comparado com a velocidade dos processadores. Portanto, é necessário estabelecer um balanceamento adequado entre a carga de processamento de cada processador e a quantidade de comunicação entre os processos.

Como a estratégia básica de paralelização usada para dividir a carga de processamento do AG é o modelo mestre-escravo, o número de indivíduos é dividido pelo número de processadores escravos. Consequentemente, o número de indivíduos determina, indiretamente, a carga de processamento para um dado número de processadores ativos.

Utilizando uma sequência de *benchmark*, um experimento foi realizado para calibrar a carga dos processadores. Para um número fixo de gerações, o tamanho da população (*popsize*) foi testada entre 200 e 2000 indivíduos, e o número de processadores entre 1 e 120. A Figura 29 apresenta o gráfico resultante, onde os eixos x, y e z representam o número de indivíduos, número de processadores e o tempo de processamento, respectivamente. É importante ressaltar que o tempo de processamento (T_p) inclui não somente o tempo de processamento, mas também o tempo de comunicação entre o processo mestre e os escravos.

Esta figura mostra claramente uma região onde a relação entre o tamanho da população e o número de processadores é melhor. Nesta região, é alcançado o melhor balanceamento entre o processamento e a comunicação, isto é, o melhor aproveitamento dos recursos computacionais.

Como não se conhece o algoritmo genético sequencial mais eficiente para o PDP utilizando o modelo 3DHP-SC, não se pode utilizar a medida "*Strong speedup*" (ace-

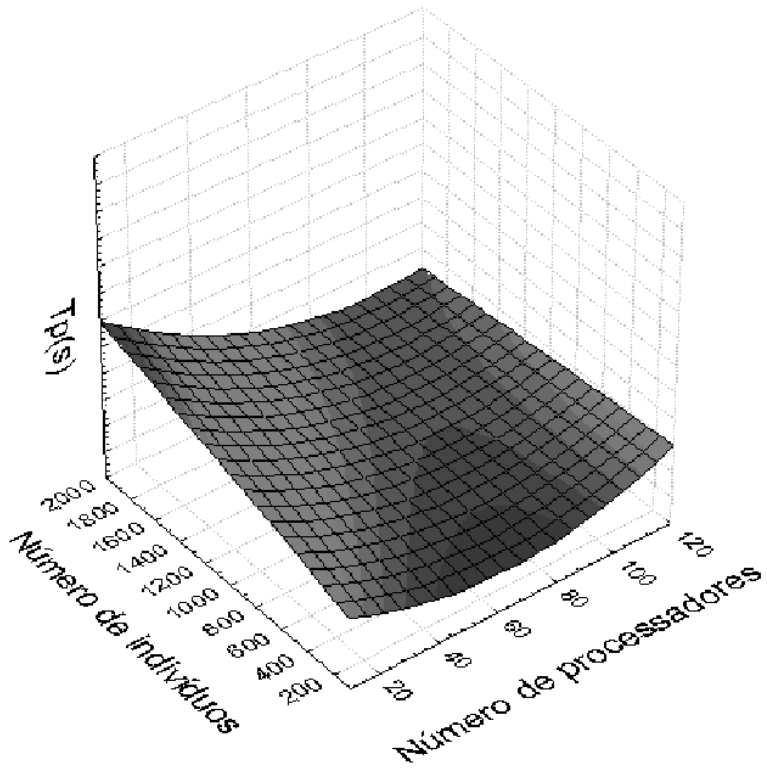


Figura 29: Gráfico de superfície mostrando o T_p para diferentes combinações de *popsize* e número de processadores

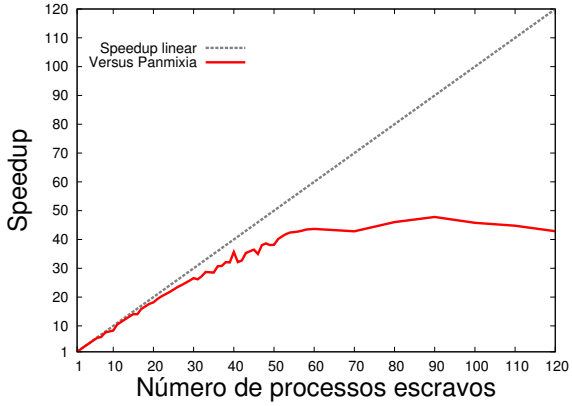
Fonte: Autoria própria

leração forte) para medir o *speedup* da versão paralela implementada. Como a versão paralela do algoritmo possui as mesmas características da versão sequencial implementada, utilizou-se a abordagem "*Versus panmixia*" para avaliar o desempenho da implementação paralela, a qual compara o tempo de processamento do algoritmo paralelo com a versão sequencial (ver Seção 2.5.5 – página 60).

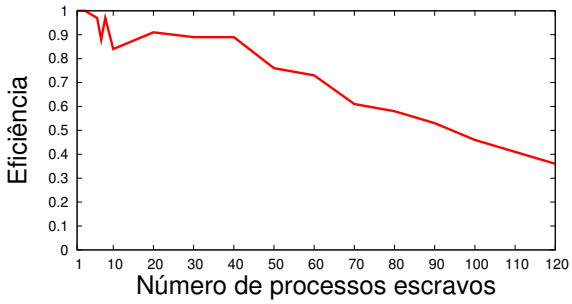
As Figuras 30(a) e 30(b) mostram, respectivamente, as curvas de *speedup* e eficiência da implementação paralela mestre-escravo com população de 500 indivíduos, respectivamente. Pode-se observar que o *speedup* é sempre sublinear e tende a se afastar da reta de *speedup* linear com o aumento do número de processadores.

Idealmente, o valor da eficiência deve ser próximo à unidade (mas não superior). Entretanto, na prática, isto não é sempre possível, pois os processadores não são utilizados 100% do tempo para processamento, mas também para comunicação, alocação de memória e outras tarefas fundamentais do sistema operacional. Na Figura 30(b), pode-se observar que há uma perda moderada da eficiência com o aumento do número de processadores. Isto se deve à sobrecarga de comunicação causada no processador

mestre. Também, há algumas situações em que o balanceamento de carga perfeito entre os processadores não é possível. Isto se deve ao fato de que número de indivíduos a serem processados é dividido pelo número de processadores, resultando em um valor inexato. Isto causa a perda de eficiência, em um dado instante de tempo, pois deveria haver alguns processadores trabalhando e outros esperando.



(a)



(b)

Figura 30: Curvas de *speedup* e eficiência

Fonte: Autoria própria

A partir destes experimentos, foi escolhida a configuração de um processo mestre e cem escravos. Portanto, cada escravo calcula a função de *fitness* de 5 indivíduos (para *popsize* = 500 indivíduos). Estes valores foram fixados para os demais experimentos com este modelo de paralelismo.

4.2 DEFINIÇÃO DE PARÂMETROS DO AG

O algoritmo genético proposto no Capítulo 3 apresenta um conjunto de parâmetros e estratégias que influenciam na qualidade das soluções obtidas. Antes de utilizar AGs para um determinado problema, é interessante realizar um estudo para encontrar um conjunto de parâmetros que tenda a levar a bons resultados. Vários experimentos foram realizados para ajustar todos os parâmetros do sistema, incluindo os parâmetros básicos do AG e os parâmetros dos operadores especializados e da estratégia de dizimação implementados. Para a realização destes experimentos foi escolhida uma sequência de *benchmark* formada por 27 resíduos e, devido à natureza estocástica do AG, todos os experimentos foram executados 100 vezes em um *cluster* Beowulf homogêneo composto por 31 computadores com mesma configuração de *hardware* (Intel *core* 2-quad de 3 GHz) rodando o sistema operacional Linux .

A avaliação dos experimentos leva em consideração não apenas a qualidade das soluções obtidas, mas também o esforço computacional despendido.

Para determinar a melhor combinação de parâmetros foi utilizado o conceito de "Otimalidade de Pareto" (DEB, 2001) em cada experimento. Um gráfico é contruído de modo a representar o comportamento de dois parâmetros que devem ser minimizados. Cada ponto do gráfico representa uma possível combinação dos parâmetros. Neste trabalho, o eixo x é [*1-normalizada(média HnC)*], uma função da média do número de contatos hidrofóbicos (*média HnC*) dividida pelo valor máximo encontrado.

O eixo y representa o tempo de processamento. Cada ponto do gráfico (x_i, y_i) é classificado como dominado quando há pelo menos outro ponto (x_j, y_j), tal que $(x_j < x_i) \bigwedge (y_j < y_i)$. Caso contrário, o ponto é não-dominado. Nesta análise, somente os pontos não dominados são realmente interessantes, pois representam o melhor balanço possível entre os dois critérios. O gráfico de Pareto também permite encontrar o ponto de trabalho mais adequado para situações particulares.

4.2.1 Parâmetros básicos

Não há um procedimento específico para realizar o ajuste dos parâmetros do AG para um determinado problema (LOBO; LIMA; MICHALEWICZ, 2007). Sabe-se que o auto-ajuste de parâmetros tende a ser mais eficiente que ajustes manuais. Contudo, isto não pertence ao escopo deste trabalho. Um exemplo de auto-ajuste de parâmetros pode ser encontrado em (MARUO; LOPES; DELGADO, 2005). Outra estratégia fre-

quentemente utilizada na literatura consiste em estabelecer uma faixa de valores para os parâmetros do AG e realizar experimentos com todas as combinações possíveis. Este procedimento é conhecido como experimento fatorial (BOX; HUNTER; HUNTER, 2005). A prática tem mostrado que um experimento fatorial é satisfatório na maioria dos casos de ajuste de parâmetros de AGs.

O experimento fatorial foi realizado para o ajuste dos parâmetros dos operadores genéticos básicos (*pmut* e *pcross*) e do tamanho de torneio do procedimento de seleção (*TamTorneio*) do AG. O número de gerações máximo (*maxgen*) e o tamanho da população (*popsize*) foram ajustados para 3000 e 500, respectivamente. A seguir são listados os parâmetros testados com seus respectivos valores: tamanho de torneio (*TamTorneio*), assumindo os valores 2%, 3% e 5%; probabilidade do operador de *crossover* (*pcross*), assumindo os valores 70%, 80% e 90%; probabilidade do operador de mutação (*pmut*), assumindo os valores 2%, 5% e 8%.

A combinação dos valores possíveis para estes parâmetros leva a um total de 27 experimentos diferentes. Os resultados foram computados em função da média de contatos hidrofóbicos, média de gerações (geração em que o melhor indivíduo foi encontrado), número máximo de contatos hidrofóbicos e o tempo total de processamento. A Tabela 5 apresenta os resultados obtidos.

A Figura 31 apresenta o gráfico de Pareto para os resultados obtidos. Nesta figura, pode-se observar que a diferença entre os pontos não-dominados (experimentos 15 e 9) é menor que 5%, em relação ao tempo de processamento. Portanto, o primeiro foi selecionado devido à melhor qualidade das soluções. Os parâmetros básicos do AG correspondentes ao experimento 15 são: Tamanho de torneio (*TamTorneio*): 3%; Probabilidade de *crossover* (*pcross*): 80% e Probabilidade de mutação (*pmut*): 8%. Estes valores foram fixados para os demais experimentos.

4.2.2 Matriz de Pesos de Energias

Um experimento fatorial também foi realizado para determinar o melhor conjunto de pesos para a matriz de energia livre apresentada na Equação 11. Os seguintes valores foram testados: ε_{HH}, assumindo os valores 10 e 15; $\varepsilon_{BB} = \varepsilon_{BP} = \varepsilon_{PP}$ assumindo os valores -5 e -3; $\varepsilon_{HP} = \varepsilon_{HB}$ assumindo os valores -1, 0 e 1. O efeito conjunto da penalização (*PP*) aplicada à função objetivo como uma consequência das colisões também foi avaliado. Este parâmetro foi testado para os valores: 5, 7 e 10. Todas as possíveis combinações dos valores mencionados acima leva a um total de 36 experimentos

Tabela 5: Resultados dos experimentos para estabelecer valores para os parâmetros básicos do AG

Exp.	*tamtorneio*	*pcross*	*pmut*	*HnC* média	*HnC* máx	média T_p(s)
1	2%	70%	2%	$12{,}78 \pm 1{,}71$	16	615,04
2	2%	70%	5%	$12{,}47 \pm 1{,}78$	16	605,56
3	2%	70%	8%	$14{,}00 \pm 1{,}89$	18	582,15
4	2%	80%	2%	$12{,}47 \pm 1{,}06$	16	625,91
5	2%	80%	5%	$13{,}79 \pm 1{,}88$	19	505,25
6	2%	80%	8%	$14{,}15 \pm 1{,}03$	19	493,55
7	2%	90%	2%	$12{,}26 \pm 1{,}82$	16	437,23
8	2%	90%	5%	$12{,}84 \pm 1{,}42$	16	440,98
9	2%	90%	8%	$14{,}16 \pm 1{,}19$	19	411,38
10	3%	70%	2%	$12{,}06 \pm 1{,}43$	15	597,35
11	3%	70%	5%	$13{,}63 \pm 1{,}41$	20	614,03
12	3%	70%	8%	$13{,}11 \pm 1{,}82$	16	589,27
13	3%	80%	2%	$12{,}42 \pm 1{,}43$	14	565,12
14	3%	80%	5%	$12{,}55 \pm 1{,}70$	16	486,45
15	3%	80%	8%	$14{,}42 \pm 1{,}59$	19	426,89
16	3%	90%	2%	$13{,}11 \pm 1{,}05$	17	426,49
17	3%	90%	5%	$14{,}16 \pm 1{,}83$	18	457,81
18	3%	90%	8%	$14{,}35 \pm 1{,}06$	18	429,52
19	5%	70%	2%	$11{,}83 \pm 1{,}77$	16	635,90
20	5%	70%	5%	$13{,}37 \pm 1{,}06$	18	653,17
21	5%	70%	8%	$13{,}78 \pm 1{,}70$	17	639,81
22	5%	80%	2%	$12{,}29 \pm 1{,}69$	15	539,10
23	5%	80%	5%	$13{,}00 \pm 1{,}73$	16	503,87
24	5%	80%	8%	$13{,}47 \pm 1{,}95$	16	526,89
25	5%	90%	2%	$12{,}22 \pm 1{,}83$	15	436,30
26	5%	90%	5%	$14{,}36 \pm 1{,}37$	18	436,01
27	5%	90%	8%	$13{,}94 \pm 1{,}41$	20	428,98

diferentes.

Os resultados foram computados em função da média de contatos hidrofóbicos, média de gerações (geração em que o melhor indivíduo foi encontrado), número máximo de contatos hidrofóbicos e o tempo de processamento. A Tabela 6 apresenta os resultados obtidos.

A Figura 32 apresenta o gráfico de Pareto para os resultados obtidos. Assim como na análise anterior, nesta figura pode-se observar que a diferença entre os pontos não-dominados (experimentos 7, 17, 18) é menor que 5%, em relação ao tempo de processamento. Portanto, o terceiro foi selecionado devido à melhor qualidade das soluções.

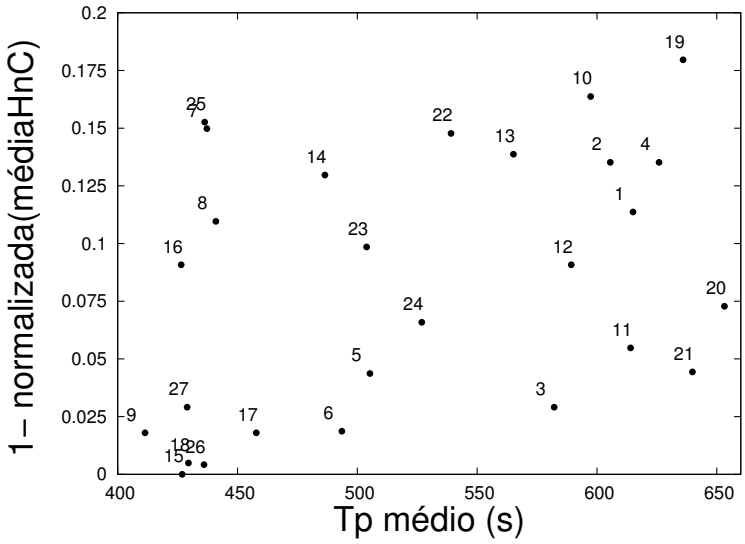


Figura 31: Gráfico de Pareto para selecionar o melhor conjunto de parâmetros básicos para o AG.

Fonte: Autoria própria

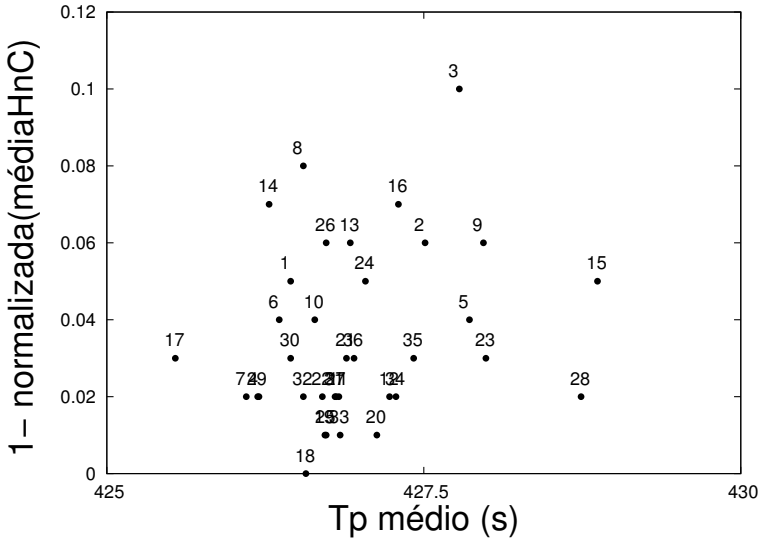


Figura 32: Gráfico de Pareto para selecionar a melhor matriz de pesos.

Fonte: Autoria própria

A matriz de pesos correspondente ao experimento 18 é mostrada na Equação 17 com penalização *PP*=10.

Tabela 6: Resultados dos testes para matriz de contatos e penalização

Exp.	*HnC*		Gerações		média T_p(s)
	média	máx	média	máx	
1	13,93±1,49	16	2042,14±1074,63	2990	426,45
2	13,72±1,71	18	1650,67±993,21	2982	427,51
3	13,20±1,64	15	1945,70±872,72	2994	427,78
4	14,31±1,89	19	1665,56±921,39	2996	426,20
5	14,05±2,07	19	1760,37±701,64	2617	427,86
6	14,05±1,68	17	2107,05±757,02	2993	426,36
7	14,33±1,46	18	1658,89±687,31	2889	426,10
8	13,50±2,12	17	1426,56±949,94	2994	426,55
9	13,68±1,83	16	1670,68±887,11	2983	427,97
10	14,06±1,48	16	1845,24±884,37	2991	426,64
11	14,33±1,85	17	2003,70±679,09	2988	426,83
12	14,39±1,90	18	2000,00±824,12	2955	427,23
13	13,80±1,74	17	1028,00±656,03	2354	426,92
14	13,59±1,94	17	1513,94±925,64	2896	426,28
15	13,94±2,30	18	1835,94±836,31	2998	428,87
16	13,53±1,98	17	1447,00±913,08	2852	427,30
17	14,25±2,10	19	1916,40±842,88	2993	425,54
18	14,63±1,80	20	1821,08±804,53	2985	426,57
19	14,53±1,77	20	1869,72±974,18	2978	426,72
20	14,48±1,72	18	1550,24±872,76	2997	427,13
21	14,25±2,77	21	1543,20±893,27	2941	426,89
22	14,37±1,38	19	1794,50±893,31	2991	426,70
23	14,25±1,04	17	1498,52±877,80	2999	427,99
24	13,89±1,70	16	1622,32±785,14	2944	427,04
25	14,44±1,71	19	1393,88±733,14	2840	426,73
26	13,74±1,24	16	1498,74±1033,12	2962	426,73
27	14,35±1,23	17	1514,95±886,83	2892	426,81
28	14,36±1,80	20	1443,96±870,32	2835	428,74
29	14,30±1,93	20	1604,70±829,50	2963	426,19
30	14,24±1,79	17	1361,41±655,53	2601	426,45
31	14,35±1,18	17	1698,80±762,84	2845	426,80
32	14,40±1,98	19	1842,20±673,68	2887	426,55
33	14,45±1,43	18	1894,50±695,31	2991	426,84
34	14,30±1,81	17	1746,40±1032,73	2978	427,28
35	14,25±2,49	20	1612,10±895,37	2969	427,42
36	14,15±1,84	18	1565,65±801,60	2744	426,95

$$\varepsilon_{ab} = \begin{array}{c} \\ \textbf{SCH} \\ \textbf{SCP} \\ \textbf{BB} \end{array} \begin{array}{c} \begin{array}{ccc} \textbf{SCH} & \textbf{SCP} & \textbf{BB} \end{array} \\ \begin{pmatrix} 10 & -3 & -3 \\ -3 & 1 & 1 \\ -3 & 1 & 1 \end{pmatrix} \end{array} \tag{17}$$

Na matriz obtida, pode-se observar que ela é formada por pesos de atração para

cadeias laterais hidrofóbicas ($\varepsilon_{HH} = 10$), hidrofílicas ($\varepsilon_{PP} = 1$), *backbone*-cadeia lateral polar ($\varepsilon_{BP} = \varepsilon_{PB} = 1$) e *backbone-backone* ($\varepsilon_{BB} = 1$), e pesos de repulsão para interações entre cadeias laterais hidrofóbica-polar ($\varepsilon_{HP} = \varepsilon_{PH}$=-3) e *backbone*-cadeia lateral hidrofóbica ($\varepsilon_{HB} = \varepsilon_{BH}$ = -3). Esta matriz de pesos foi utilizada nos demais experimentos.

4.3 PARÂMETROS DOS OPERADORES ESPECIAIS

Após escolher um conjunto de parâmetros básicos do AG, experimentos fatoriais foram realizados com o objetivo de selecionar os melhores parâmetros para os operadores especiais e estratégia de dizimação *hot-boot* (DHB), assim como avaliar a sua influência na qualidade das soluções. As seções a seguir apresentam os resultados obtidos.

4.3.1 Parâmetros do operador de mutação sempre melhor (MSM)

Conforme apresentado na Seção 3.5.1, o operador de Mutação Sempre Melhor (MSM) funciona da mesma forma que a mutação simples. A única diferença consiste em que o indivíduo é avaliado novamente a cada mutação realizada. Caso o *fitness* do indivíduo mutado seja maior do que original, a mudança é mantida. Caso contrário, o indivíduo original é restaurado. Este operador é executado durante um número de tentativas configurado previamente. A probabilidade de mutação sempre melhor ($pmut_{MSM}$) utilizada possui o mesmo valor que a probabilidade do operador de mutação simples, ou seja, $pmut_{MSM} = 8\%$. Experimentos preliminares mostraram que este tipo de mutação aumenta consideravelmente o tempo de processamento. Portanto, o número de tentativas foi configurado para 2 tentativas, ou seja, $tentativas_{MSM} = 2$.

A Tabela 7 apresenta os resultados obtidos com o operador de mutação sempre melhor.

Tabela 7: Resultados dos experimentos para o operador MSM

MSM	*HnC*		Geração		média T_p(s)
	média	máx	média	máx	
Não	14,63±1,80	20	1821,08±804,53	2985	426,57
Sim	14,68±1,76	19	1760,02±876,08	2977	1666,98

De acordo com a Tabela 7, o algoritmo obtém, em média, resultados ligeiramente

melhores com o operador MSM. O algoritmo básico (sem este operador) obteve um resultado máximo, mas gastou mais gerações para obtê-lo. Provavelmente, isto indica que o algoritmo básico mantém a diversidade genética durante mais tempo, podendo encontrar, eventualmente, um indivíduo melhor. Conforme esperado, o tempo de processamento deste operador é muito superior comparado ao algoritmo básico. Portanto, decidiu-se não utilizar este operador, pois o ganho de desempenho foi muito pequeno e o custo computacional muito alto.

Experimentos podem ser realizados utilizando um conjunto de valores de $pmut_{MSM}$ e $tentativas_{MSM}$ com o objetivo de realizar um estudo aprofundado sobre o comportamento deste operador. No entanto, isto não foi realizado devido ao elevado tempo de processamento despendido por este operador.

4.3.2 Parâmetros do operador de mutação sem colisão (MSC)

Conforme apresentado na Seção 3.5.2, o operador de mutação sem colisão (MSC) avalia o indivíduo mutado, verificando se o mesmo apresenta colisões. Caso o indivíduo mutado não possua colisão, a mudança é mantida. Caso contrário, o indivíduo original é restaurado.

Para testar a influência deste operador, foram realizados experimentos com a probabilidade de mutação sem colisão ($pmut_{MSC}$), assumindo os valores 2%, 5% e 8% e número de tentativas $tentativas_{MSC}$=2.

A Tabela 8 apresenta os resultados obtidos com o operador de mutação sem colisão.

Tabela 8: Resultados dos experimentos para o operador MSC

pmut_MSC	*HnC*		Geração		média T_p(s)
	média	máx	média	máx	
0%	14,63±1,80	20	1821,08±804,53	2985	426,57
2%	14,67±2,39	18	1732,50±795,47	2849	517,08
5%	14,77±1,82	19	1698,46±762,97	2938	631,98
8%	14,88±2,04	20	1450,30±840,34	2931	732,43

Na Tabela 8, pode-se observar que para qualquer valor de $pmut_{MSC}$, a qualidade das soluções obtidas é, em média, sempre melhor do que quando este operador não é aplicado e melhora com o aumento da $pmut_{MSC}$, pois o algoritmo tende a inserir me-

nos indivíduos inválidos após a geração da população nova. Entretanto, o tempo de processamento aumenta consideravelmente com o aumento da $pmut_{MSC}$, pois o mapeamento genótico→fenótipo da conformação é realizado em cada tentativa deste operador. De acordo com esta tabela, o melhor valor de probabilidade de mutação sem colisão é $pmut_{MSC}$ = 8%.

4.3.3 Parâmetros do operador de mutação de cadeia lateral (MCL)

Conforme apresentado na Seção 3.5.3, o operador de mutação de cadeia lateral (MCL) realiza a mutação de um gene do indivíduo selecionado aleatoriamente de acordo com uma probabilidade de mutação de cadeia lateral ($pmut_{MCL}$) com o objetivo de maximizar o número de interações hidrofóbicas da cadeia lateral do aminoácido selecionado. Para testar a influência deste operador, foram realizados experimentos com a probabilidade de mutação assumindo os valores 2%, 5% e 8%.

A Tabela 9 apresenta os resultados obtidos com o operador de mutação de cadeia lateral.

Tabela 9: Resultados dos experimentos para o operador MCL

pmut_MCL	*HnC*		Geração		média T_p(s)
	média	máx	média	máx	
0%	14,63±1,80	20	1821,08± 804,53	2985	426,57
2%	14,44±1,50	17	1976,33±940,33	2992	551,69
5%	14,84±1,54	19	1662,90±921,23	2934	682,68
8%	14,70±1,69	18	1820,44 ± 821,85	2885	752,79

Na Tabela 9, pode-se observar que a qualidade das soluções obtidas é, em média, melhor quando é aplicada a probabilidade *pmut_MCL* de 5% e 8% e que a quantidade máxima de contatos hidrofóbicos encontrada entre os melhores indivíduos é menor ao utilizar este operador, o que não era esperado. Provavelmente, isto deve-se ao fato do algoritmo perder diversidade genética ao utilizar este operador. Um exemplo claro disto pode ser observado no experimento com $pmut_{MCL}$ de 5%, onde o melhor indivíduo é obtido, em média, com um número médio menor de gerações. Também pode-se observar que o tempo de processamento aumenta consideravelmente com o aumento da $pmut_{MCL}$, pois o mapeamento genótico→fenótipo da conformação também é realizado em cada tentativa deste operador. De acordo com esta tabela, o melhor valor de

probabilidade de mutação de cadeia lateral é $pmut_{MCL}$ = 5%.

4.4 PARÂMETROS DA ESTRATÉGIA DE DIZIMAÇÃO E *HOT-BOOT* (DHB)

A estratégia DHB possui dois parâmetros que afetam no comportamento do AG. O primeiro (*gen2decimate*) indentifica o número de gerações sem melhoria da melhor solução para habilitar a estratégia DHB. O outro parâmetro (*gen2weakTourney*) representa o número de gerações após realizar a dizimação em que o valor do tamanho de torneio do procedimento (*tourneysize*) de seleção é diminuido para 2. Estes dois parâmetros foram testados com os seguintes valores: [300; 600] e [30; 60; 300; 600], respectivamente. A combinação destes valores leva a um total de 8 experimentos. Os resultados obtidos são apresentados na Tabela 10.

Tabela 10: Resultados dos experimentos para os conjuntos de parâmetros da estratégia DHB. O símbolo "$*$" representa o resultado para o AGP–ME básico apresentado na Tabela 6.

			HnC		Geração		média T_p(s)
Exp.	*gen2decimate*	*gen2weak Tourney*	média	máx	média	máx	
$*$	–	–	14,63 $\pm$ 1,80	20	1821,08 $\pm$ 804,53	2985	426,57
1	300	30	15,25 $\pm$ 1,57	19	1555,16 $\pm$ 858,56	2850	510,35
2	300	60	15,70 $\pm$ 1,53	18	2161,80 $\pm$ 688,61	2956	473,33
3	300	300	16,10 $\pm$ 1,97	21	2013,29 $\pm$ 776,86	2998	467,97
4	300	600	15,35 $\pm$ 1,58	20	1948,75 $\pm$ 842,49	2914	474,02
5	600	30	15,39 $\pm$ 2,00	19	1843,11 $\pm$ 863,36	2885	530,19
6	600	60	14,79 $\pm$ 1,55	18	1618,42 $\pm$ 1002,93	2991	491,73
7	600	300	14,52 $\pm$ 1,12	16	1992,47 $\pm$ 835,09	2974	449,89
8	600	600	14,40 $\pm$ 1,27	16	2005,50 $\pm$ 814,90	2974	556,15

Na Tabela 10 pode-se observar que a estratégia de dizimação leva, em média, a resultados melhores do que o experimento básico ($*$), exceto nos experimentos 7 e 8. Com relação ao valor máximo de contatos hidrofóbicos encontrado, o experimento 3 encontrou o maior número de contatos (21 contatos).

Na Figura 33, o gráfico de Pareto mostra que os pontos não-dominados correspondem aos experimentos 3 e 7. O valor dos parâmetros do experimento 3 foram escolhidos, pois leva a soluções de melhor qualidade. Estes valores são *gen2decimate*=300 e *gen2weakTourney*=300, e foram fixados para serem utilizados nos demais experimentos.

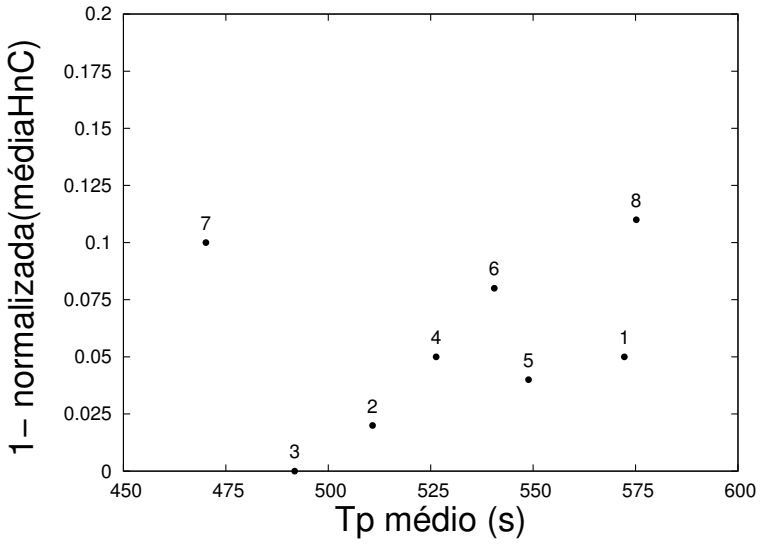
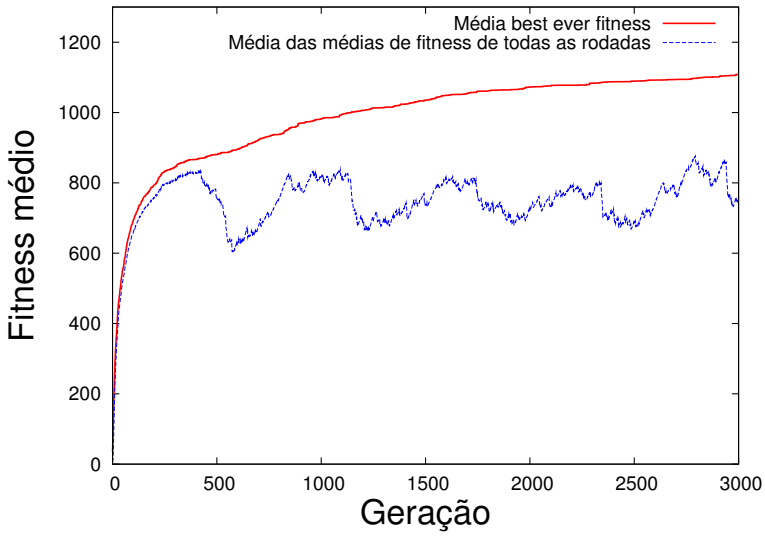


Figura 33: Gráfico de Pareto para selecionar o melhor conjunto de parâmetros da estratégia DHB

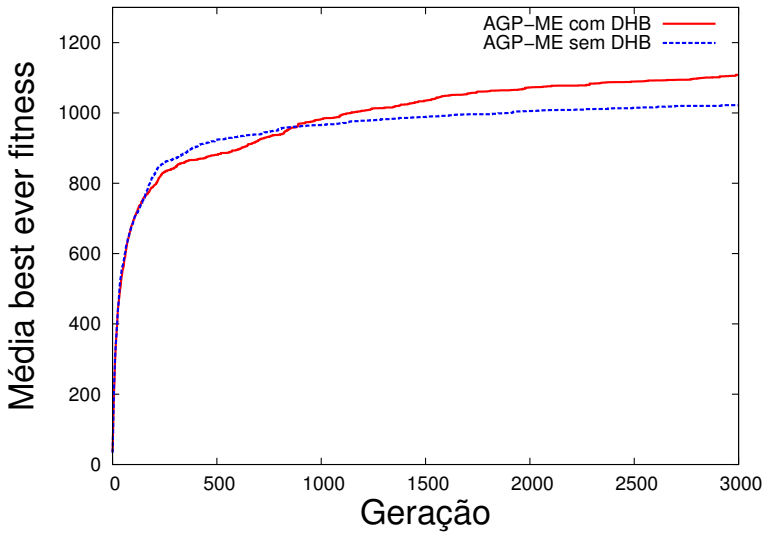
Fonte: Autoria própria

A Figura 34(a) apresenta o gráfico da média do *fitness* do melhor indivíduo obtido e da média dos *fitness* médios (sobre a população, para cada geração) para as 100 rodadas deste experimento. Nesta figura, pode-se observar como a estratégia de dizimação é eficiente para preservar a diversidade genética. Após sua aplicação, a distância entre a média e o melhor é mantida (ao todo, levemente oscilante). A manutenção de alta diversidade genética permite ao AG explorar o espaço de busca com maior eficiência, evitando a convergência prematura.

Para efeito de comparação, a Figura 34(b) apresenta o gráfico da média do *fitness* do melhor indivíduo obtido pelo AG com e sem a estratégia DHB. Nesta figura, pode-se observar que o uso da estratégia DHB leva a melhores indivíduos quando comparado com o AG sem DHB.



(a)



(b)

Figura 34: Curvas de *fitness*

Fonte: Autoria própria

4.5 PARÂMETROS DA POLÍTICA DE MIGRAÇÃO DO AGP HIERÁRQUICO

Experimentos foram realizados para ajustar os parâmetros da política de migração do AGP Hierárquico: *Migration Gap* (número de gerações entre duas migrações sucessivas), *Migration Rate* (número de indivíduos migrantes que participarão de cada migração), e o critério de Seleção/Substituição. Os três parâmetros foram testados com os seguintes valores: *Migration Gap*, assumindo os valores 30, 60, 90, 120, 150, 300, 600 e 900 gerações; *Migration Rate*, assumindo os valores 2, 3 e 5 indivíduos migrantes.

Critério de seleção/substituição: o melhor indivíduo e indivíduos selecionados aleatoriamente são selecionados para migrar para a próxima ilha da topologia e substituir indivíduos selecionados aleatoriamente. A combinação destes valores leva a um total de 24 experimentos. Os resultados obtidos são apresentados na Tabela 11.

É importante ressaltar que as ilhas são conectadas em anel circular unidirecional, onde cada ilha possui uma população de 500 indivíduos (valor determinado nos experimentos apresentados na Seção 4.2.1). Cada ilha é formada por 1 processo mestre e 30 escravos, totalizando 124 processos (este valor representa o número total de núcleos de processamento, considerando 1 núcleo por processo). Esta topologia foi escolhida devido a dois fatos: disponibilidade de *hardware* (alocando um núcleo de processamento por processo) e a expectativa de observar o efeito de coevolução o mais breve possível. Da literatura de algoritmos evolucionários paralelos, sabe-se que um número pequeno de ilhas em uma topologia em anel acelera o processo de coevolução, exigindo um número reduzido de gerações (CANTÚ-PAZ, 1998).

Na Tabela 11, pode-se observar que o modelo AGP–HH é melhor que o AGP–ME para qualquer uma das configurações testadas.

A Figura 35 apresenta o gráfico de Pareto para os resultados obtidos. Nesta figura, pode-se observar que a diferença entre os pontos não-dominados (experimentos 12, 19 e 20) é menor que 10% , em relação ao tempo de processamento. Portanto, o valor dos parâmetros do experimento 12 foram escolhidos, pois leva a soluções de melhor qualidade. Estes valores são $Migration\ Gap$=120 gerações e $Migration\ Rate$=5 indivíduos (o melhor mais quatro selecionados aleatoriamente), e foram fixados para serem utilizados nos demais experimentos.

A Figura 36 apresenta as curvas da média do $fitness$ do melhor indivíduo obtido pelo AGP–HH utilizando todos os valores do parâmetro $Migration\ Gap$ da política de migração. Esta figura mostra claramente que a velocidade de convergência do algoritmo depende do número de gerações entre duas migrações, estabelecido pelo parâmetro $Migration\ Gap$. Pode-se dizer que a velocidade de convergência do algoritmo aumenta para valores baixos e diminui para valores elevados deste parâmetro. Por exemplo, observa-se que para valores elevados ($Migration\ Gap$ = 600 e 900) a evolução do algoritmo ocorre lentamente, podendo levar a melhores soluções aumentando o número máximo de gerações através do parâmetro *maxgen*. Conforme esperado, a média do $fitness$ do melhor indivíduo obtido pelo AGP–HH com os parâmetros da política de migração escolhidos ($Migration\ Gap$=120 e $Migration\ Rate$=5) obteve o maior valor.

Tabela 11: Resultados dos experimentos sobre os parâmetros de migração do AGP hierárquico. O símbolo "$*$" representa o resultado para o AGP–ME básico apresentado na Tabela 6.

Exp.	*Migration gap*	*Migration rate*	*HnC* Gerações média	*HnC* Gerações máx	Gerações média	Gerações máx	média T_p(s)
$*$	–	–	14,63 ± 1,80	20	1821,08± 804,53	2985	426,57
1	30	2	15,13 ± 1,36	18	1387,60 ± 877,75	2910	554,63
2	30	3	15,45 ± 1,93	20	1392,30 ± 690,42	2610	599,57
3	30	5	15,07 ± 2,08	20	1813,67 ± 1044,98	2940	594,51
4	60	2	15,78 ± 2,14	21	1602,57 ± 879,16	2996	552,64
5	60	3	15,09 ± 1,37	18	1325,24 ± 861,81	2940	593,51
6	60	5	15,50 ± 1,15	18	1324,44 ± 746,61	3000	589,94
7	90	2	16,07 ± 2,08	21	1259,59 ± 813,99	2970	575,95
8	90	3	15,94 ± 2,79	22	1573,11 ± 802,49	2880	590,97
9	90	5	16,05 ± 1,95	21	1610,10 ± 748,05	2821	591,93
10	120	2	15,40 ± 1,67	19	1763,23 ± 916,99	2956	570,75
11	120	3	15,04 ± 1,60	19	1531,22 ± 839,99	2998	589,67
12	120	5	16,79 ± 2,22	21	1717,14 ± 738,44	2903	584,07
13	150	2	16,24 ± 1,94	20	1687,40 ± 843,23	2978	581,97
14	150	3	16,30 ± 1,63	19	1685,30 ± 660,61	2912	586,01
15	150	5	16,05 ± 2,09	22	1440,11 ± 654,50	2706	593,77
16	300	2	15,82 ± 1,72	19	1853,46 ± 745,21	2986	596,38
17	300	3	16,37 ± 1,70	20	1682,47 ± 833,68	2781	591,01
18	300	5	15,72 ± 1,63	21	2057,56 ± 622,57	2998	615,85
19	600	2	15,46 ± 1,75	19	1868,39 ± 688,71	2852	530,60
20	600	3	16,07 ± 2,01	21	1618,07 ± 722,25	2402	538,31
21	600	5	16,06 ± 1,95	21	1832,82 ± 666,43	2993	617,04
22	900	2	15,69 ± 1,60	19	2216,00 ± 671,48	2971	566,07
23	900	3	16,11 ± 2,15	21	2204,30 ± 737,93	2996	591,98
24	900	5	15,58 ± 1,74	19	2128,53 ± 745,70	2703	619,79

O efeito de coevolução do modelo AGP–HH é ilustrado na Figura 37. Esta figura mostra o valor do *fitness* do melhor indivíduo da população de cada ilha, entre as gerações 0 e 600. O procedimento de migração ocorre a cada 120 gerações como pode ser observado claramente nas gerações 120, 240, 360 e 480. Quando um indivíduo de boa qualidade chega a uma ilha não tende a melhorar apenas a melhor solução local, mas também, através do operador de *crossover*, induzir uma melhoria na qualidade da população.

Para avaliar a influência do tamanho da população de cada ilha na qualidade das soluções, um experimento adicional foi realizado mudando o tamanho da população das ilhas para 125 indivíduos (AGP–HH$_b$), totalizando 500 indivíduos (4 ilhas $\times$ 125 indivíduos/ilha = 500 indivíduos). O motivo deste experimento é que a versão AGP–

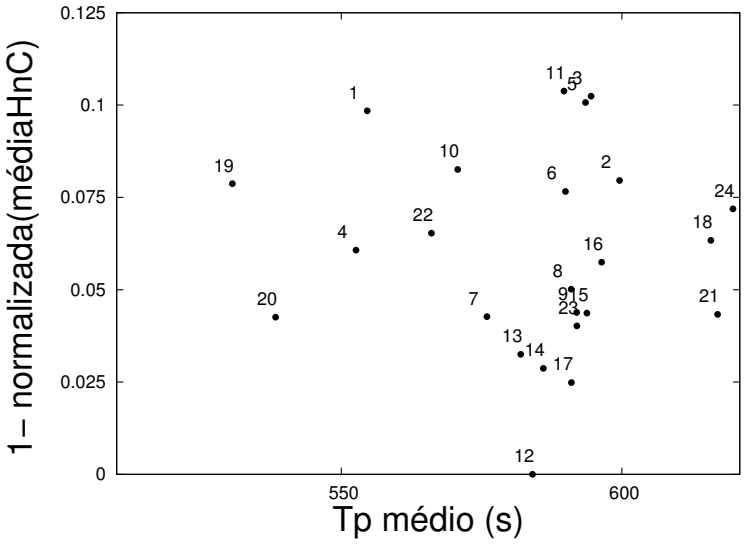


Figura 35: Gráfico de Pareto para selecionar os melhores parâmetros para a política de Migração

Fonte: Autoria própria

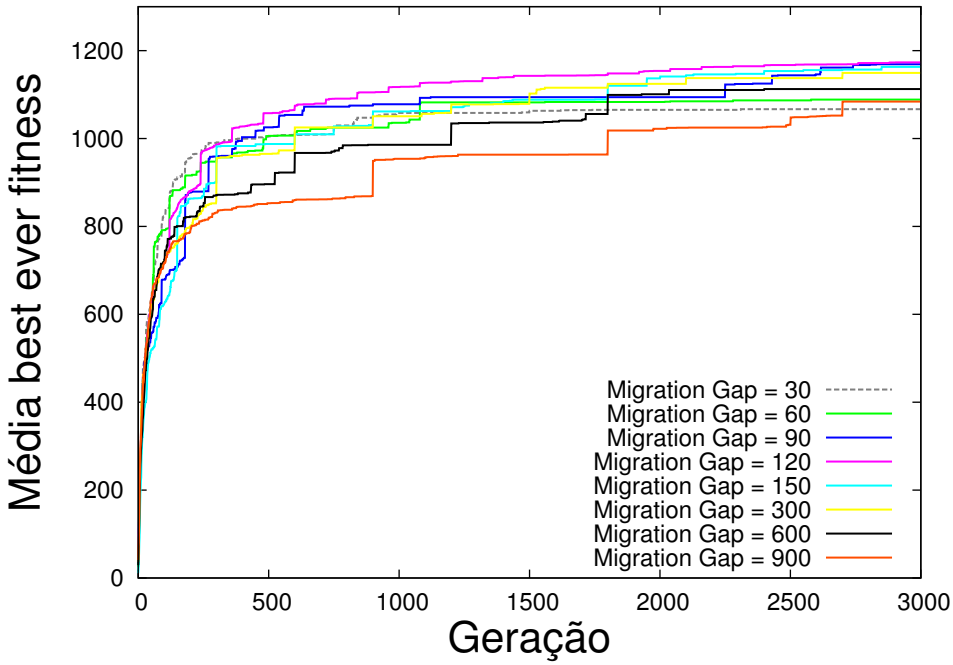


Figura 36: Curvas de *fitness*

Fonte: Autoria própria

ME utiliza uma população de 500 indivíduos e, se esta população fosse dividida em 4 ilhas, cada ilha teria 125 indivíduos. Apenas para efeito de comparação, o modelo AGP–ME também foi testado com uma população de 2000 indivíduos (AGP–ME$_b$), pois o modelo AGP–HH possui 500 indivíduos por ilha (totalizando 2000 indivíduos).

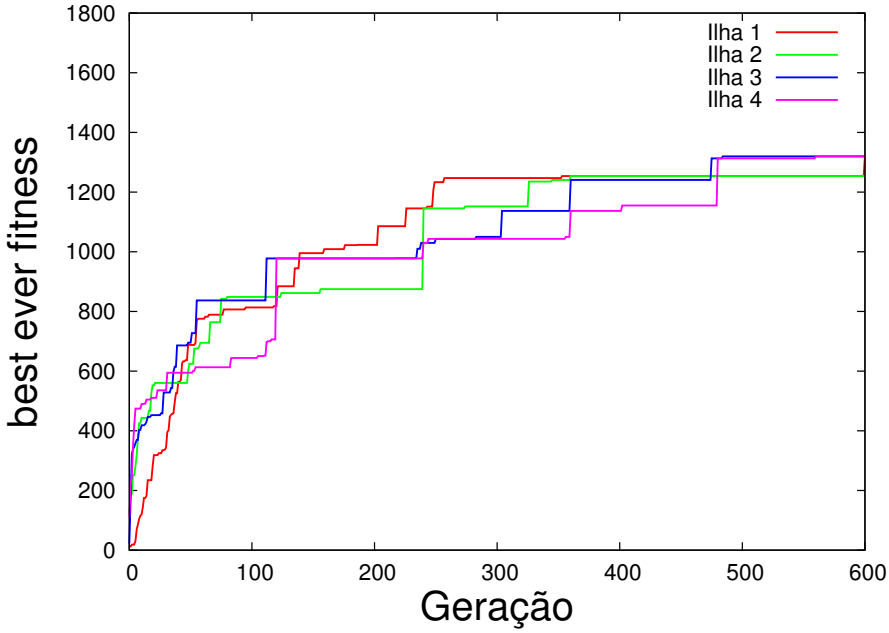


Figura 37: Exemplo do efeito da coevolução

Fonte: Autoria própria

A Tabela 12 apresenta os resultados obtidos. Comparando as abordagens hierárquicas (AGP–HH e AGP–HH$_b$) se pode observar que quanto maior o tamanho da população de cada ilha, melhores são os resultados obtidos. Por outro lado, comparando as abordagens hierárquicas (AGP–HH e AGP–HH$_b$) com as versões mestre-escravo (AGP–ME e AGP–ME$_b$), pode-se observar que as abordagens hierárquicas obtêm resultados melhores quando comparadas com as versões mestre-escravo com o mesmo tamanho de população total ($popsize_{total}$). Isto se deve ao efeito de coevolução entre as ilhas, pois os imigrantes injetam diversidade na população permitindo uma busca mais efetiva. Também pode-se observar que a implementação AGP–HH$_b$ (modelo hierárquico com 125 indivíduos por ilha, totalizando 500 indivíduos) apresenta resultados de melhor qualidade e menor tempo computacional quando comparado com o AGP–ME com o mesmo número de indivíduos. Isto sugere que a abordagem hierárquica pode levar a um bom balanço entre a qualidade da solução e o tempo de processamento.

A abordagem hierárquica AGP–HH é a melhor configuração, levando em consideração a qualidade das soluções obtidas como principal critério de escolha.

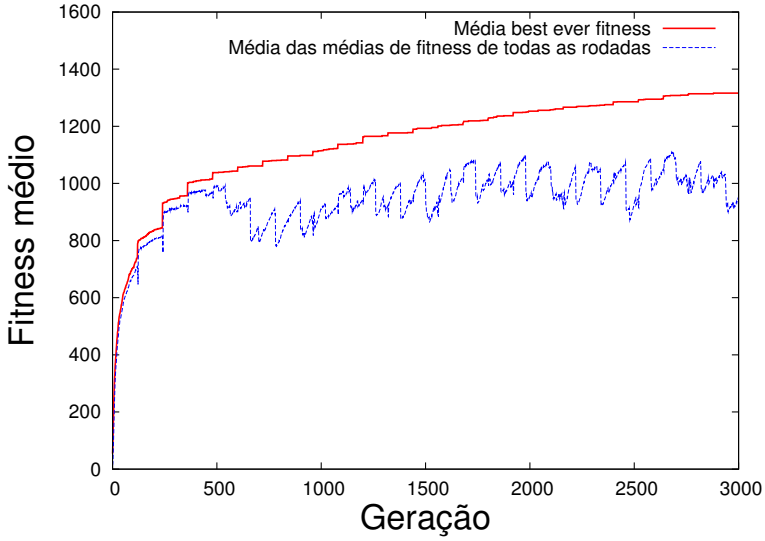
Tabela 12: Resultados dos experimentos para diferentes tamanhos de população $popsize_{total}$

Algoritmo	$popsize_{total}$	HnC		Geração		média T_p(s)
		média	máx	média	máx	
AGP–HH	2000	16,79± 2,22	21	1717,14±738,44	2880	584,07
AGP–ME$_b$	2000	14,86 ± 1,97	20	1613,63 ± 858,61	2999	536,94
AGP–HH$_b$	500	14,82 ±2,26	20	1867,77±729,19	2999	291,98
AGP–ME	500	14,63±1,80	20	1821,08±804,53	2985	426,57

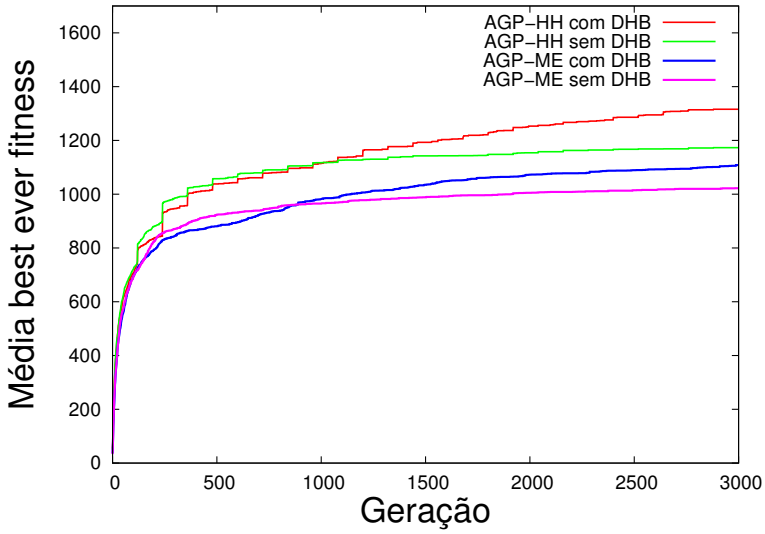
A Figura 38(a) apresenta o gráfico da média do $fitness$ do melhor indivíduo obtido e da média do $fitness$ médio (sobre a população para cada geração) para as 100 rodadas do AGP–HH com a estratégia DHB. Conforme esperado, pode-se observar que a estratégia de dizimação é eficiente para preservar a diversidade genética. Conforme dito, a manutenção da diversidade genética aumenta a eficiência do AG na exploração do espaço de busca, evitando a convergência prematura.

Para efeito de comparação, a Figura 38(b) apresenta o gráfico da média do $fitness$ do melhor indivíduo obtido pelas versões AGP–ME e AGP–HH com e sem a estratégia DHB. Pode-se observar que a estratégia DHB melhora a qualidade das soluções quando comparado com as versões sem DHB.

A Tabela 13 apresenta os resultados obtidos pelo AGP–HH com a estratégia DHB. Para efeito de comparação, o resultado obtido pelo AGP–ME com estratégia DHB também é apresentado nesta tabela. De acordo com esta tabela e a Figura 38(b), o AGP–HH com a estratégia DHB obtém, em média, resultados melhores que as demais abordagens. Com relação ao valor máximo de contatos hidrofóbicos encontrado, a abordagem AGP–HH com a estratégia DHB também foi superior (22 contatos). Também pode-se observar que o AGP–HH com DHB obtém o melhor indivíduo, em média, com um número médio maior de gerações que as demais abordagens. Isto se deve ao fato de que, quando a estratégia de dizimação é utilizada, a diversidade genética é mantida permitindo ao AGP–HH explorar o espaço de busca de maneira mais eficiente.



(a)



(b)

Figura 38: Curvas de *fitness*

Fonte: Autoria própria

Tabela 13: Resultados dos experimentos para o AGP–HH com DHB. O símbolo "$*$" representa o resultado para o AGP–ME com DHB apresentado na Tabela 10.

DHB	*HnC*		Geração		média T_p(s)
	média	máx	média	máx	
$*$	16,10 $\pm$ 1,97	21	2013,29 $\pm$ 776,86	2998	467,97
Não	16,79 $\pm$ 2,22	21	1717,14 $\pm$ 738,44	2880	584,07
Sim	17,79 $\pm$ 2,20	22	2405,63 $\pm$ 498,65	2988	566,93

4.6 CONSIDERAÇÕES FINAIS

A partir dos resultados apresentados nas seções anteriores, determinou-se um conjunto de parâmetros mais adequado para ser utilizado no problema de dobramento de proteínas utilizando o modelo 3DHP-SC. Optou-se por utilizar o modelo AGP–HH e a estratégia de dizimação, bem como não utilizar os operadores especiais (MSM, MSC e MCL), pois estes, apesar de tenderem a melhorar a qualidade das soluções, a melhoria não é significativa, quando comparados com a estratégia de dizimação. Por outro lado, levam a um aumento significativo no tempo de processamento.

Os parâmetros estabelecidos para o algoritmo AGP–HH são mostrados na Tabela 14.

4.7 APLICAÇÃO DO ALGORITMO AGP–HH PARA SEQUÊNCIAS DE *BENCHMARK*

Um total de 25 sequências sintéticas de aminoácidos foram utilizadas nos experimentos, como mostrado na Tabela 15. Estas sequências possuem tamanho de 27, 31, 36 ou 48 aminoácidos. Os *benchmarks* utilizados foram divididos em três grupos: "Dill.*", proposto inicialmente por (YUE; DILL, 1993), "Unger273d.*" (UNGER; MOULT, 1993a) e "S48.*" (YUE; FIEBIG; AL., 1994). Para estas proteínas, não se conhece o valor máximo de contatos hidrofóbicos não-locais para o modelo 3DHP-SC. Apenas para efeito de comparação, o valor máximo de contatos hidrofóbicos conhecido para os grupos de sequências "Dill.*", "Unger273d.*" and "S48.*" usando o modelo 3DHP é apresentado na coluna (E) da tabela, seguindo os melhores resultados obtidos por (YUE; DILL, 1993), (PATTON; III; GOODMAN, 1995) e (THACHUK; SHMYGELSKA; HOOS, 2007), respectivamente. Supõe-se que a solução ótima para estas sequências, usando o modelo 3DHP-SC, não terá um número inferior de contatos hidrofóbicos (do modelo 3DHP) mas, provavelmente, maior.

4.7.1 Resultados numéricos

Devido à natureza estocástica do AG, o AGP–HH foi executado 100 vezes com diferentes sementes aleatórias para cada uma das 25 sequências de *benchmark*. Os resultados são apresentados na Tabela 16. Nesta tabela, a primeira coluna identifica a sequência, a segunda, a terceira e a quarta identificam, respectivamente, a geração em que o melhor indivíduo foi encontrado, a média e o número máximo de gerações que

Tabela 14: Conjunto de parâmetros.

	Parâmetro	Significado	Valor
Parâmetros básicos	*maxgen*	Número máximo de gerações	3000 gerações
	popsize	Tamanho da população de cada ilha	500 indivíduos
	Porc_popsemColisoes	porcentagem de indivíduos gerados sem colisão	20% de *popsize*
	pcross	Probabilidade de *Crossover*	80%
	pmut	Probabilidade de mutação	8%
	TamTorneio	Tamanho de torneio	3% de *popsize*
Matriz de energias	ε_{HH}	Ponderação para energia de interações entre cadeias laterais hidrofóbicas (HH)	10
	$\varepsilon_{HP} = \varepsilon_{PH}$	Ponderação para energia de interações entre cadeias laterais hidrofóbica-polar (HP)	-3
	$\varepsilon_{HB} = \varepsilon_{BH}$	Ponderação para energia de interações entre *backbone*-cadeia lateral hidrofóbica (BH)	-3
	ε_{PP}	Ponderação para energia de interações entre cadeias laterais polares (PP)	1
	$\varepsilon_{PB} = \varepsilon_{BP}$	Ponderação para energia de interações entre *backbone*-cadeia lateral polar (PB)	1
	ε_{BB}	Ponderação para energia de interações entre *backbone-backone* (BB)	1
Operadores especiais	$pmut_{MSM}$	Probabilidade de mutação do operador MSM	0%
	$pmut_{MSC}$	Probabilidade de mutação do operador MSC	0%
	$pmut_{MCL}$	Probabilidade de mutação do operador MCL	0%
Estratégia DHB	*gen2decimate*	Número de gerações sem melhoria da melhor solução para habilitar a estratégia DHB	300 gerações
	gen2weakTourney	Número de gerações após realizar a dizimação em que o valor do tamanho de torneio do procedimento (*tourneysize*) de seleção é diminuido para 2	300 gerações
Migração	*Migration Gap*	Número de gerações entre duas migrações sucessivas	120 gerações
	Migration Rate	Número de indivíduos migrantes que participarão em cada migração	5 indivíduos

foram necessárias para encontrar o melhor indivíduo. A próxima coluna representa a média do tempo de processamento e as duas últimas representam, respectivamente, a média e o número máximo de contatos não-locais entre cadeias laterais hidrofóbicas.

Nesta tabela, pode-se observar que o AGP–HH necessita, em média, menos gerações que o máximo configurado ($maxgen = 3000$) para encontrar a melhor solução. Este fato sugere que um número menor de gerações pode ser utilizado em experimentos futuros com estes *benchmarks*.

Tabela 15: Sequências de *Benchmark* para o modelo 3DHP-SC: n indica o número de aminoácidos da sequência, E o número máximo de contatos hidrofóbicos não locais para o modelo 3DHP.

Referência	n	*Sequência*	E_{3DHP}
Dill.1	27	$HP^4H^4P(PH)^3H(HP)^2PH^2P^2H$	16
Dill.2	27	$HP^3H^4(PH)^2HP^3HPH(HP)^2P^2HP$	15
Dill.3	27	$HPH^2(PPHH)^2H(HPPP)^2H^3P^2H$	16
Dill.4	31	$(HHP)^3H(HHHHHPP)^2H^7$	28
Dill.5	36	$PH(PPH)^{11}P$	14
Unger273d.1	27	$(PH)^3H^2P^2(HP)^2P^{10}H^2P$	9
Unger273d.2	27	$PH^2P^{10}H^2P^2H^2P^2HP^2HPH$	10
Unger273d.3	27	$H^4P^5HP^5H^3P^8H$	8
Unger273d.4	27	$H^3P^2H^4P^3(HP)^2PH^2P^2HP^3H^2$	15
Unger273d.5	27	$H^4P^4HPH^2P^3H^2P^{10}$	8
Unger273d.6	27	$HP^6HPH^3P^2H^2P^3HP^4HPH$	11
Unger273d.7	27	$HP^2HPH^2P^3HP^5HPH^2(PH)^3H$	13
Unger273d.8	27	$HP^{11}(HP)^2P^7HPH^2$	4
Unger273d.9	27	$P^7H^3P^3HPH^2P^3HP^2HP^3$	7
Unger273d.10	27	$P^5H(HP)^5(PHH)^2PHP^3$	11
S48.1	48	$HPH^2P^2H^4PH^3P^2H^2P^2HPH^3(PH)^2HP^2H^2P^3HP^8H^2$	32
S48.2	48	$H^4(PHH)^2H^3(PPH)^2HP^2HP^6(HPP)^2PHP^2H^2P^2H^3PH$	34
S48.3	48	$(PH)^2HPH^6P^2(HP)^2(PH)^2(HP)^3(PPH)^2HP^2H^2P^2(HP)2PHP$	34
S48.4	48	$(PH)^2HP^2HPH^3P^2H^2PH^2P^3H^5P^2HPH^2(PH)^2P^4HP^2(HP)^2$	33
S48.5	48	$P^2HP^3HPH^4P^2H^4(PHH)^2HP(PH)^3P^2HP^5(PHH)^2PH$	32
S48.6	48	$H^3P^3H(HP)^2(HHP)^3HP^7(HP)^2PHP^3HP^2H^6PH$	32
S48.7	48	$PHP^4HPH^3(PH)^2H^3(PHH)^2P^3(HP)^2P^2H^3(PPHH)^2P^3H$	32
S48.8	48	$(PHH)^2HPH^4P^2H^3P^6HPH^2P^2H(HP)^2P^2H^2(PH)^3HP^3$	31
S48.9	48	$(PH)^2P^4(HP)^3(PH)^2H^5P^2H^3PHP(PH)^2HP(PH)^2H^2P^4H$	34
S48.10	48	$PH^2P^6H^2P^3H^3PHP(PH)^2(HPP)^3H^2P^2H^7P^2H^2$	33

Se o AG consegue alcançar o número máximo de contatos hidrofóbicos não-locais (apresentado na última coluna da Tabela 16) em todas as rodadas, sua eficiência seria de 100%. Considerando todas as sequências de *benchmark*, o AGP–HH alcançou uma eficiência média que excede 80%. Este valor pode ser considerado como bom, levando em consideração as diferenças entre as instâncias, a natureza estocástica do AG e o número de parâmetros a serem ajustados. Portanto, pode-se inferir que o algoritmo proposto é consistente.

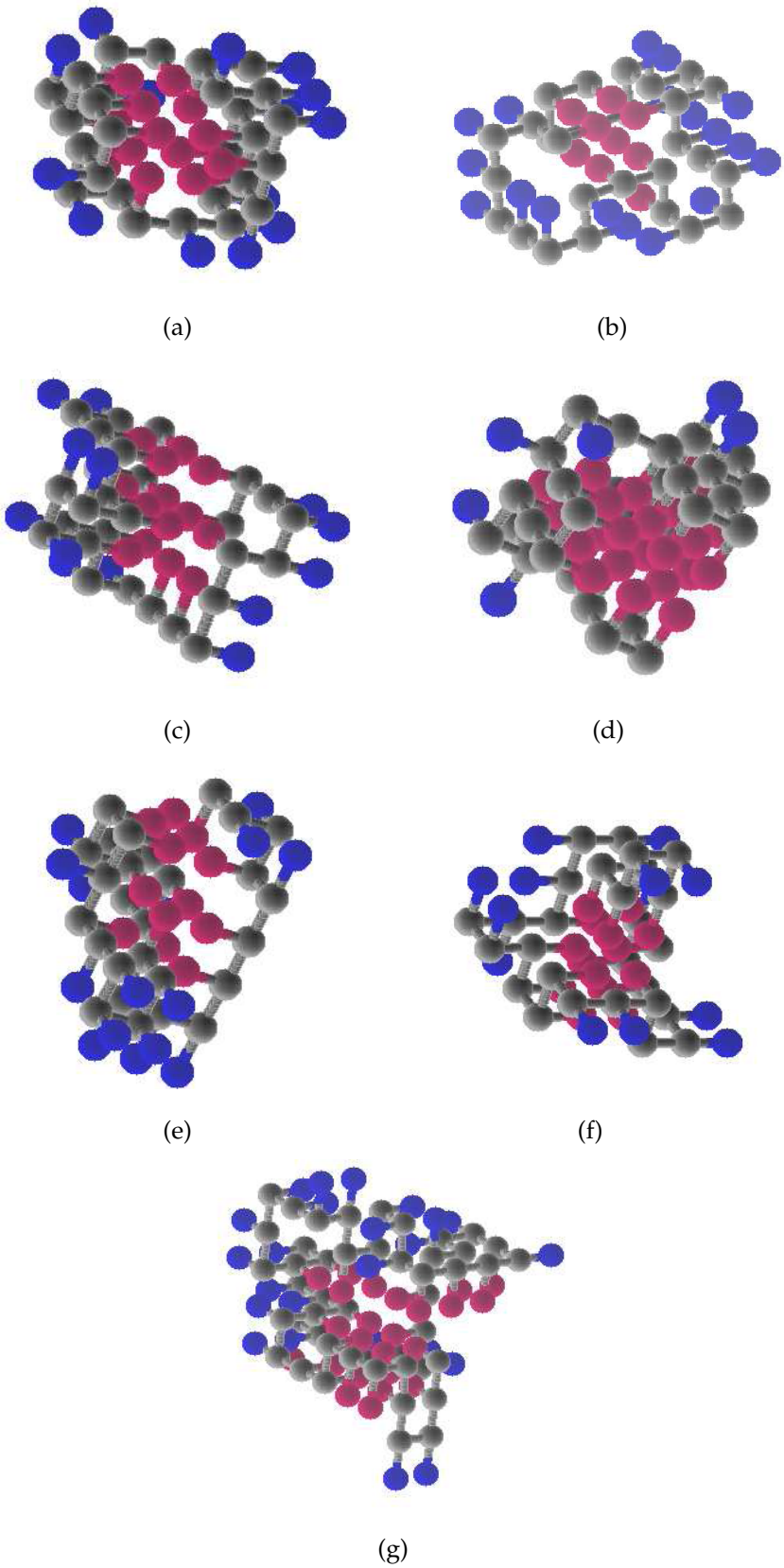
Também pode-se observar que o tempo de processamento total é dependente do tamanho da sequência, com possível crescimento exponencial com o número de aminoácidos. Este fato sugere que o algoritmo perde desempenho para proteínas de tamanho elevado.

Tabela 16: Resultados das sequências de *benchmark* para o modelo 3DHP-SC.

Referência	Geração		média T_p(s)	HnC		E_{3DHP}
	melhor	média		média	máx	
Dill.1	2160	2151,80 ± 687,28	573.50	17,42 ± 1,86	21	16
Dill.2	2904	2084,25 ± 770,10	566,59	14,78 ± 1.56	19	15
Dill.3	2880	2378,29 ± 583,31	566,68	18,00 ± 1,79	23	16
Dill.4	2760	2154,31 ± 603,6	749,10	32.81 ± 2.40	41	28
Dill.5	2880	2104,10 ± 741,37	1141,11	11,29 ± 1,27	14	14
Unger273d.1	2400	2153,13 ± 702,58	572,80	10,06 ± 1,18	12	9
Unger273d.2	2640	2025,67 ± 725,73	569,10	11,89 ± 0,93	13	10
Unger273d.3	2880	1996,71 ± 740,37	569,48	10,71 ± 0,94	13	8
Unger273d.4	1778	2405,63 ± 498,65	566,93	17,79 ± 2,20	22	15
Unger273d.5	2779	2340,37 ± 438,18	560,04	10,83 ± 1,04	13	8
Unger273d.6	1680	2126,62 ± 612,23	568,20	11,28 ± 1,29	14	11
Unger273d.7	1680	2019,89 ± 763,72	566,53	12,57 ±1,50	16	13
Unger273d.8	1560	1523,00 ± 817,37	569,05	4,68 ± 0,85	6	4
Unger273d.9	2400	1818,24 ± 846,19	568,18	8,06 ± 1,03	10	7
Unger273d.10	2400	2068,67 ± 871,71	568,99	11,08 ± 1,23	14	11
S48.1	2882	2467,10 ± 593,50	2640,50	26,24 ± 3,13	32	32
S48.2	2520	2523,32 ± 373,44	2640,97	26,68 ± 3,07	32	34
S48.3	2880	2367,69 ± 619,59	2644,03	24,63 ± 3,32	30	34
S48.4	2760	2541,63 ± 469,90	2647,95	25.75 ± 3,47	35	33
S48.5	2964	2340,37 ± 723,44	2647,43	26,37 ± 2,87	31	32
S48.6	2780	2275,26 ± 658,95	2644,14	26.29 ± 3.04	33	32
S48.7	2144	2319,56 ± 687,14	2645,08	24,63 ± 3,56	32	32
S48.8	2743	2335,81 ± 596,25	2647,55	25,94 ± 2,89	31	31
S48.9	2084	2378,71 ± 516,47	2650,81	26,12 ± 2,99	32	34
S48.10	1800	2135,00 ± 480,95	2655,54	27,75 ± 2,89	33	33

4.7.2 Resultados gráficos

As conformações que representam os melhores indivíduos obtidos são apresentadas nas Figuras 39(a), 39(b), 39(c), 39(d), 39(e), 39(f) e 39(g). Nestas figuras, pode ser observada a formação de um núcleo hidrofóbico no interior do dobramento, parcialmente protegido por cadeias laterais polares. Este tipo de conformação, típica de proteínas globulares, foi esperada como consequência da função de *fitness*. Este fato sugere que a função de *fitness* proposta é adequada para o PDP usando o modelo 3DHP-SC, pois é capaz de levar a conformações que mimetizam propriedades bioquímicas de proteínas reais durante o processo de dobramento.



(a) (b) (c) (d) (e) (f) (g)

Figura 39: Melhor dobramento 3D para as sequências Dill.2 (a), Unger273d.3 (b), Unger273d.4 (c), Dill.4 (d), Unger273d.7 (e), Dill.3 (f) e S48.10 (g). As esferas vermelhas e azuis representam cadeias laterais hidrofóbicas e polares, respectivamente. O *backbone* e as conexões entre os elementos são mostrados em cinza.

Fonte: Autoria própria

4.8 COMPARAÇÃO COM OUTRA ABORDAGEM EVOLUCIONÁRIA

O AGP–HH implementado foi comparado com uma implementação do algoritmo ABC (*Artificial Bee Colony Algorithm*) para dobramento de proteínas. Inicialmente, o algoritmo ABC foi submetido a experimentos com o objetivo de estabelecer um conjunto valores para os seus parâmetros de controle. Depois, quatro sequências foram utilizadas para realizar a comparação entre o AGP–HH e o ABC.

De maneira similar aos estudos realizados para o algoritmo genético, alguns parâmetros do ABC podem ser ajustados. Neste caso, também não há um procedimento específico para realizar o ajuste dos parâmetros do ABC para um dado problema. Foram testados os seguintes parâmetros do algoritmo ABC, utilizando a mesma sequência de *benchmark* utilizada no ajuste dos parâmetros do AG: Tamanho da Colônia (*Colonysize*), e a porcentagem de abelhas empregadas (n_e) e observadoras (n_o). Três conjuntos de valores para *Colonysize* e *MCN*, respectivamente {250, 500, 1000} e {6000, 3000, 1500} para o mesmo número de avaliações (para abelhas empregadas e observadoras) foram utilizados. Para cada combinação, três pares de valores para n_e e n_o foram testados: {75%, 25%}, {50%, 50%}, e {25%, 75%}, resultando em 9 conjuntos de parâmetros. Para cada conjunto, 100 rodadas independentes com sementes aleatórias diferentes foram realizadas. Os resultados são apresentados na Tabela 17. O melhor conjunto de parâmetros obtido corresponde ao experimento 2 e é formado por: *Colonysize*=250, *MCN*=6000, n_e=50% e $n_e = n_o$=50%. Estes parâmetros foram fixados para serem utilizados nos demais experimentos do ABC.

Tabela 17: Resultados dos experimentos para os conjuntos de parâmetros do ABC

Exp.	*Colonysize*	*MCN*	n_o	*HnC*		Geração		média T_p(s)
				Média	Máx	Média	Máx	
1	250	6000	25%	12,94±1,65	17	4198,16±1497,46	6000	1259,03
2	250	6000	50%	13,53±1,74	18	4155,67±1530,56	5960	1406,21
3	250	6000	75%	12,40±1,50	15	3612,61±1867,82	5993	1431,37
4	500	3000	25%	12,32±1,57	16	2300,97±707,86	2981	737,71
5	500	3000	50%	12,22±1,22	14	2298,05±551,71	2986	944,94
6	500	3000	75%	12,76±1,85	19	2190,05±580,20	2990	1018,79
7	1000	1500	25%	12,86±1,31	15	1305,95±124,22	1484	666,40
8	1000	1500	50%	12,74±1,31	15	1206,16±231,66	1489	672,13
9	1000	1500	75%	12,13±1,33	15	1322,84±153,93	1493	671,57

Assim como no AGP–HH, o modelo ABC–HH é formado no nível superior por uma topologia de 4 ilhas conectadas em anel unidirecional. O parâmetro *Migration Gap* foi configurado para 10% do número máximo de ciclos (*MCN*), ou seja, a migração ocorre a cada 600 gerações (lembrando que foi selecionado *MCN*=6000 ciclos). O parâmetro *Migration Rate* para 2 abelhas, de tal forma que uma cópia da melhor abelha e uma selecionada aleatoriamente emigram para a próxima ilha da topologia e substituem abelhas selecionadas aleatoriamente.

Duas versões de ABC–HH foram implementadas para avaliar a influência do tamanho da colônia (*Colonysize*) na qualidade das soluções. Uma utiliza 250 abelhas por ilha (ABC–HH_1) e a outra utiliza 63 abelhas por ilha (ABC–HH_2), totalizando 1000 e 252 abelhas, respectivamente. O motivo destas duas versões é que a versão ABC–ME utiliza uma colônia de 250 abelhas e, divindo esta colônia temos 62,5 abelhas por ilha (como somente números inteiros fazem sentido, a versão ABC–HH_2 utiliza 63 abelhas por ilha).

O modelo ABC–ME utiliza 50 processos escravos e, portanto, cada escravo calcula a função de *fitness* de 5 abelhas (como no AGP–ME).

No nível inferior, o modelo ABC–HH utiliza uma configuração ABC–ME com um processo mestre e 25 escravos por ilha e, portanto, cada escravo calcula a função de *fitness* de 10 abelhas. Portanto, cada escravo calcula a função de $fitness$ para 10 abelhas da versão HH–ABC_1 e 2 ou 3 do HH–ABC_2. Devido à natureza estocástica do ABC, 100 rodadas independentes com sementes aleatórias diferentes foram executadas. Os resultados são apresentados na Tabela 18.

Tabela 18: Resultados dos experimentos para as modelos paralelos do ABC

Algoritmo	*HnC*		Geração		média T_p(s)
	média	máx	média	máx	
ABC–ME	13,53±1,74	18	4155,67±1530,56	5960	1406,21
ABC–HH_1	14,59±1,67	19	3829,68±1353,36	5961	1788,19
ABC–HH_2	14,20±1,03	16	4130,78±1393,10	5819	926,67

A Figura 40 apresenta as curvas da média do $fitness$ da melhor abelha encontrada pelas três abordagens, para cada ciclo. Na Tabela 18 e na Figura 40, pode-se observar que as abordagens ABC–HH (ABC–HH_1 e ABC–HH_2) levam a melhores soluções quando comparadas com o ABC–ME. Os resultados sugerem que o efeito da coevolu-

ção entre as ilhas é vantajoso também para o algoritmo ABC.

A diferença entre ABC–HH_1 e ABC–HH_2 é o tamanho da colônia de cada ilha. Os resultados indicam que quanto maior a colônia de cada ilha, melhores são os resultados obtidos. Portanto, o ABC–HH_1 será utilizado nos experimentos de comparação com o AGP–HH.

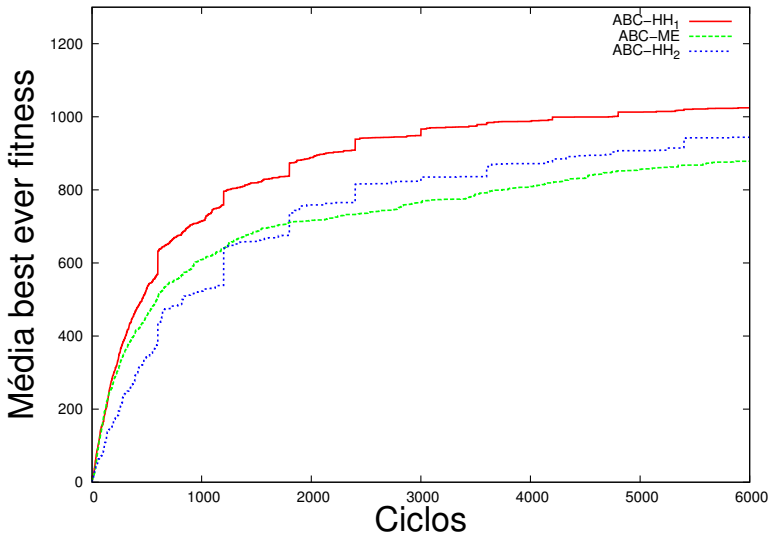


Figura 40: Curvas de *fitness*

Fonte: Autoria própria

Uma vez escolhido o conjunto de parâmetros do ABC, o ABC–HH foi comparado com o AGP–HH. Para isto foram utilizadas quatro sequências de *benchmark*, previamente apresentadas na Tabela 15: Unger273d.1, Unger273d.2, Unger273d.3 e Unger273d.4. Os resultados obtidos são apresentados na Tabela 19. Nesta tabela pode-se observar que o AGP–HH obteve, para todos os casos, resultados de melhor qualidade em um menor tempo computacional. Estes resultados desencorajam o uso do algoritmo ABC para testes posteriores.

Tabela 19: Resultados das sequências de *benchmark* para o modelo 3DHP-SC.

Algoritmo	Referência	*HnC*		média T_p(s)
		média	máx	
AGP–HH	Unger273d.1	$10{,}06 \pm 1{,}18$	12	572,80
	Unger273d.2	$11{,}89 \pm 0{,}93$	13	569,10
	Unger273d.3	$10{,}71 \pm 0{,}94$	13	569,48
	Unger273d.4	$17{,}79 \pm 2{,}20$	22	566,93
ABC–HH	Unger273d.1	$7{,}20 \pm 1{,}14$	9	1849,98
	Unger273d.2	$8{,}13 \pm 1{,}89$	11	1848,15
	Unger273d.3	$8{,}67 \pm 1{,}22$	11	1758,01
	Unger273d.4	$14{,}59 \pm 1{,}67$	19	1788,19

5 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

5.1 CONCLUSÕES

O PDP ainda é um problema em aberto do ponto de vista computacional, posto que sua solução recai em um problema *NP*-completo. Nem mesmo utilizando os modelos mais simplificados é possível encontrar a conformação nativa de proteínas reais (com várias dezenas ou centenas de aminoácidos). A alternativa tem sido a utilização de métodos heurísticos, notadamente aqueles oferecidos pela computação evolucionária.

O modelo 3DHP tem sido bastante explorado na literatura recente. Entretanto, o 3DHP-SC, embora tenha maior expressividade biológica do que o 3DHP, tem sido muito pouco abordado devido à maior complexidade envolvida. Como consequência, também não há *benchmarks* para este modelo e neste trabalho foram adaptados *benchmarks* para o 3DHP. Portanto, os resultados obtidos neste trabalho representam uma contribuição importante em relação a este assunto.

Primeiramente, experimentos foram realizados com o objetivo de estabelecer um conjunto de parâmetros capaz de obter bons resultados com tempo de processamento aceitável para o problema. Também foi estudado o efeito dos pesos de energia de cada tipo de interação no desempenho do algoritmo. Portanto, outra contribuição relevante é a matriz de pesos otimizada para o modelo 3DHP-SC, até então não disponível na literatura. Estes parâmetros talvez não possam ser considerados como ótimos para quaisquer instâncias, mas servem como referência inicial para outros pesquisadores e serão objeto de estudo em trabalhos futuros.

Este trabalho mostra que os AGs são capazes de obter bons resultados para o PDP utilizando o modelo 3DHP-SC. Embora não se possa afirmar que todos os resultados obtidos para os *benchmarks* são ótimos, eles são coerentes com o modelo, pois se pode observar que a função de *fitness* proposta consegue simular as forças de atração entre as cadeias laterais hidrofóbicas, bem como as forças de repulsão entre as cadeias laterais hidrofóbica-polar e *backbone*-cadeia lateral hidrofóbica. Nas figuras apresenta-

das na Seção 4.7.2 pode ser observada a formação de um núcleo hidrofóbico no interior das conformações, parcialmente protegido por cadeias laterais polares posicionadas na parte externa. Este fato sugere que este tipo de conformação, típica de proteínas globulares, foi obtido de acordo com os princípios utilizados na implementação da função de *fitness*. Portanto, pode-se concluir que a função de *fitness* proposta é adequada para o PDP usando o modelo 3DHP-SC, pois é capaz de levar a conformações que mimetizam propriedades bioquímicas de proteínas reais durante o processo de dobramento.

Conforme esperado, a estratégia de dizimação implementada foi eficiente no controle da diversidade genética da população, permitindo que o processo de evolução do AG possa continuar por mais gerações e melhorando a possibilidade de obtenção de melhores soluções. Talvez, com uma análise mais aprofundada dos parâmetros desta estratégia, o algoritmo possa obter resultados ainda melhores.

Os resultados obtidos demonstram que a combinação dos modelos de paralelização mestre-escravo e insular em dois níveis obtém melhores resultados com menor esforço computacional, devido ao efeito de coevolução entre as ilhas (como pode ser observado na Figura 37 – página 122) e divisão da carga de processamento entre processos escravos, desde que o algoritmo seja projetado adequadamente. No nível superior, o processo de migração entre as ilhas melhora a diversidade genética das populações e, consequentemente, a qualidade da busca. No nível inferior, o modelo mestre-escravo introduz uma melhoria significativa no desempenho do algoritmo.

O projeto de algoritmos genéticos hierárquicos (AGP–HH) é bastante complexo, pois, além do efeito dos parâmetros básicos de controle, o seu comportamento é afetado também pela migração de indivíduos entre as ilhas. Cada parâmetro da política de migração (*Migration Rate*, *Migration Gap*, critério de seleção/substituição de indivíduos e a topologia de conectividade entre as ilhas, apresentados na Seção 3.8.1 – página 99) afeta a qualidade e eficiência da busca do algoritmo de maneira não-linear, tornando mais difícil a configuração da política de migração.

Vários experimentos foram realizados para ajustar a política de migração. Verificou-se que, para qualquer configuração da política de migração, o AGP–HH é superior ao algoritmo simples, obtendo, em média, resultados melhores. Observou-se que a velocidade de convergência e a qualidade de busca do algoritmo dependem da política de migração. Valores maiores do parâmetro *Migration Rate* podem levar à obtenção de melhores soluções.

No nível inferior do algoritmo genético hierárquico, o tempo de processamento do

mestre-escravo é devido a dois componentes: o tempo utilizado na computação das funções de *fitness* e o tempo despendido na comunicação entre os processos. A comunicação entre os processos ocorre a cada geração quando o processo mestre envia o pacote de indivíduos para os escravos e quando os escravos retornam as avaliações de *fitness* ao mestre. De acordo com a Figura 30(a) (ver Seção 4.1 – página 107), pode-se observar que há uma dependência não-linear entre o *speedup* atingido pelo algoritmo e o número de processos escravos observa-se um compromisso entre a diminuição do tempo utilizado pelos processos escravos na computação do *fitness* dos indivíduos e o aumento do tempo despendido na comunicação entre os processos. O *speedup* é sempre sublinear e tende a se afastar da reta de *speedup* linear com o aumento do número de processadores, pois o tempo de comunicação tende a ser superior ao tempo de computação de cada processo escravo. Portanto, um número adequado de processos escravos deve ser determinado com o objetivo de minimizar o tempo de processamento do algoritmo. Por exemplo, quando a comunicação é custosa em relação à computação das avaliações de *fitness*, um número pequeno de processos escravos e várias ilhas podem ser a melhor escolha. Por outro lado, quando a computação é mais custosa, a configuração pode ser formada por mais escravos por ilha.

O algoritmo AGP–HH foi aplicado a 25 sequências de *benchmark* divididas em três grupos. Os dois primeiros grupos são formados por sequências mais curtas (27, 31 e 36 aminoácidos) e o terceiro grupo é formado por sequências com 48 aminoácidos. Os resultados obtidos são apresentados na Tabela 16 (ver Seção 4.7.1 – página 128).

Comparando os resultados obtidos para os dois primeiros grupos com o valor máximo conhecido usando o modelo 3DHP (ver Tabela 15 – página 127) observou-se que o algoritmo obteve um número superior de contatos hidrofóbicos usando o modelo 3DHP-SC, estando de acordo com o que foi preconizado na Seção 4.7. Contudo, para a sequência maior (36 aminoácidos), o algoritmo obteve o mesmo número de contatos hidrofóbicos, sugerindo que este resultado ainda pode ser melhorado.

Para a sequência de 48 aminoácidos, observa-se que os resultados obtidos apresentam um número inferior ou igual de contatos hidrofóbicos na maioria dos casos. Portanto, mais experimentos deverão ser feitos principalmente para as instâncias mais longas utilizando um número maior de gerações, pois os resultados certamente poderão ser melhorados. Esta hipótese é reforçada pela Figura 38(a), apresentada na página 124, na qual pode se observar que o *fitness* do melhor indivíduo tende a melhorar, em média, com o número de gerações, não apresentando estagnação durante a evolução

do algoritmo. Isto foi obtido devido ao controle da diversidade genética realizado pela estratégia de dizimação e ao efeito de coevolução entre as ilhas.

Também se pode observar que o tempo total de processamento depende do tamanho da sequência, com possível crescimento exponencial em relação ao número de aminoácidos.

O algoritmo proposto foi comparado com o algoritmo ABC. Para isto, duas versões paralelas do ABC foram implementadas: ABC mestre-escravo e ABC hierárquico. O AGP–HH obteve, para todos os casos, resultados melhores com um tempo de processamento menor. Os resultados obtidos pelo ABC desencorajam o seu uso. Entretanto, uma análise mais aprofundada sobre os seus parâmetros ainda pode ser realizada. A diferença no tempo de processamento se deve à natureza dos algoritmos, pois o AG avalia a população no final de cada geração e o ABC avalia a colônia após cada fase do algoritmo (abelhas empregadas, observadoras e escoteiras) como pode ser observado nos Algoritmos 1 (AG) e 2 (ABC). A aplicação de técnicas de paralelização para outras abordagens de computação evolucionária também pode ser estudada e comparada com este trabalho como, por exemplo, ACO (*Ant Colony Optimization*) (STÜTZLE, 1998) e PSO (*Particle Swarm Optimization*) (VENTER; SOBIESZCZANSKI-SOBIESKI, 2005).

De maneira geral, os resultados são bons e promissores para suportar a continuidade deste trabalho. Em resumo, acredita-se que este trabalho seja uma contribuição interessante para esta área de pesquisa devido a três motivos: exploração do modelo 3DHP-SC sugerindo uma matriz de pesos otimizada para a função de energia, fornecimento de resultados de referência para a comparação com outras abordagens, e a modelagem de um algoritmo genético paralelo hierárquico eficiente para o PDP.

5.2 TRABALHOS FUTUROS

Há uma elevada complexidade em determinar o conjunto de parâmetros de controle do AG que leve ao melhor balanço entre a obtenção de soluções de boa qualidade e a geração de novas soluções, pois o controle da diversidade genética da população é realizado através da combinação de efeitos destes parâmetros. Portanto, uma autoadaptação eficiente dos parâmetros poderia não somente liberar o usuário de possíveis ajustes, como também oferecer a possiblidade de utilizar valores mais adequados para cada etapa da busca.

Este trabalho apresenta diversas melhorias em relação ao AG simples e foi capaz de encontrar boas soluções para as instâncias de *benchmarks* do problema. Não obstante, observando a Figura 39(g) é possível notar que pequenas (mas relevantes) melhorias poderiam ser realizadas. Tais melhorias poderiam ser realizadas através de operadores genéticos inteligentes biologicamente inspirados e hibridização de estratégias de busca local com o AG.

Devido ao elevado esforço computacional necessário para lidar com o problema, o processamento paralelo foi essencial, permitindo a obtenção de resultados satisfatórios em tempos de processamento razoáveis, estando de acordo com o que foi preconizado por (LOPES, 2008). Trabalhos futuros considerarão, também, o uso de abordagens baseadas em *hardware* reconfigurável, tal como (ARMSTRONG; LOPES; LIMA, 2007), e *General-Purpose Graphics Processing Units* (GPGPU), tal como (POSPICHAL; JAROS; SCHWARZ, 2010).

Apesar do algoritmo paralelo hierárquico implementado ter apresentado resultados satisfatórios, outras políticas de migração entre ilhas podem ser estudadas, assim como topologias dinâmicas e outras estratégias de paralelização. Tendo a população distribuída em várias ilhas, vários estudos podem ser realizados. Por exemplo, os parâmetros de cada população podem ser ajustados durante a evolução do algoritmo baseando-se em alguma medida de diversidade ou desempenho. A política de migração também pode ser ajustada dinamicamente. O critério de seleção/substituição e o parâmetro *Migration Rate* podem ser ajustados de acordo com o *fitness* dos indivíduos da população da ilha que receberá os imigrantes. Por exemplo, o parâmetro *Migration Rate* pode ser aumentado com o objetivo de melhorar a diversidade genética da população. Topologias de conectividade entre as ilhas que possam se adaptar à busca também podem ser estudadas. Em adição, uma análise sobre a relação entre a política de migração e a pressão seletiva deve ser realizada, pois a adição/remoção de conexões entre ilhas e ajuste dos demais parâmetros da política de migração certamente devem impactar na pressão seletiva.

(KIRLEY; GREEN, 2000) apresentaram resultados interessantes quando removeram indivíduos aleatoriamente, simulando catástrofes geográficas. Esta idéia pode ser incorporada ao algoritmo implementado, utilizando algum mecanismo de seleção de indivíduos afetados pela catástrofe baseando-se na função de fitness. Em algumas situações esta estratégia poderia afetar ilhas selecionadas aleatoriamente ou baseando-se em alguma medida de diversidade, podendo afetar apenas uma região ou a popula-

ção inteira. Um mecanismo de colonização também poderia ser implementado com o objetivo de controlar o tamanho da população de cada ilha durante a evolução do algoritmo, pois foi demonstrado que o tamanho da população influencia significativamente na qualidade dos resultados obtidos. De acordo com os experimentos realizados, uma população maior leva a melhores resultados quando comparada com populações menores. Entretanto, o aumento no custo computacional sugere a implementação de um mecanismo de configuração dinâmica do número de processos escravos, de forma a garantir a minimização do tempo de processamento do algoritmo.

Um ecossistema artificial composto por várias ilhas também pode ser implementado, onde as ilhas podem ser formadas por comunidades de espécies diferentes, interagindo entre si e agindo sob a ação de um mecanismo que simule os fatores abióticos de um ecossistema natural (fatores externos físico-químicos do meio ambiente). Diversas abordagens baseadas em população podem ser utilizadas, por exemplo: ACO (DORIGO; STÜTZLE, 2004), PSO (KENNEDY; EBERHART, 1995), ABC (KARABOGA, 2005) e FA (*Firefly Algorithm*) (YANG, 2008).

Em trabalhos futuros serão feitos mais experimentos com objetivo de propor mais instâncias de *benchmarks*, bem como o estudo de outras funções de *fitness* mais complexas. Também podem ser considerados modelos de representação mais próximo das proteínas reais, isto é, com maior expressividade biológica.

REFERÊNCIAS

ALBA, H. **Parallel Metaheuristics: A New Class of Algorithms**. New York, NY, USA: Wiley-Interscience, 2005.

ALBERTS, B. et al. **Molecular Biology of The Cell**. New York, NY, USA: Garland Science, 2002.

ALMEIDA, C.; GONÇALVES, R.; DELGADO, M. Hybrid immune-based system for the protein folding problem. **Lecture Notes in Computer Science**, p. 13–24, 2007.

ANFINSEN, C. Reductive cleavage of disulfide bridges in ribonuclease. **Science**, v. 125, p. 691–692, 1957.

ANFINSEN, C. Principles that govern the folding of protein chains. **Science**, v. 181, n. 96, p. 223–230, 1973.

ANFINSEN, C. et al. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. **Proceedings of the National Academy of Sciences of the USA**, v. 47, n. 9, p. 1309–1314, 1961.

ARMSTRONG, N.; LOPES, H.; LIMA, C. Reconfigurable computing for accelerating protein folding simulations. **Lecture Notes in Computer Science**, v. 4419, p. 314–325, 2007.

BELL, S. et al. p53 contains large unstructured regions in its native state. **Journal of Molecular Biology**, v. 322, n. 5, p. 917–927, 2002.

BENÍTEZ, C.; LOPES, H. A parallel genetic algorithm for protein folding prediction using the 3DHP side-chain model. In: **Proc. IEEE Congr. on Evolutionary Computation**. Piscataway, NJ, USA: IEEE Computer Society, 2009. p. 1297–1304.

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

BERGER, B.; LEIGHTON, F. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. **Journal of Computational Biology**, v. 5, n. 1, p. 27–40, 1998.

BERMAN, H. et al. UniProt archive. **Nucleic Acids Research**, v. 28, n. 1, p. 235–242, 2000.

BITELLO, R.; LOPES, H. A differential evolution approach for protein folding. **Journal of Computer Science and Technology**, Springer, Beijing, China, v. 22, n. 6, p. 904–908, 2007.

BOERIO-GOATES, B. O. J. **Chemical Thermodynamics – Principles and Applications**. 5^{th}. ed. San Diego, CA, USA: Academic Press, 2000.

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

BOX, G. Evolutionary operation: A method for increasing industrial productivity. **Applied Statistics**, v. 6, n. 2, p. 81–101, 1957.

BOX, G.; HUNTER, W.; HUNTER, J. **Statistics for Experimenters: Design, Innovation, and Discovery**. 2^{nd}. ed. New York, NY, USA: J. Wiley & Sons, 2005.

BRANDEN, C.; TOOZE, J. **Introduction to Protein Structure**. New York, NY, USA: Garland Publishing, 1999.

BREMERMANN, H. Optimization through evolution and recombination. In: **Self-Organizing Systems**. Washington, DC, USA: Spartan Books, 1962. p. 93–106.

BRENNAN, R.; MATTHEWS, B. The helix-turn-helix DNA binding motif. **Journal Biochemical Chemistry**, v. 264, n. 4, p. 1903–1906, 1989.

CANTÚ-PAZ, E. A survey of parallel genetic algorithms. **Calculateurs Paralleles**, p. 611–615, 1998.

CANTÚ-PAZ, E. **Efficient and Accurate Parallel Genetic Algorithms**. New York, NY, USA: Springer, 2000.

CHAN, H.; DILL, K. Comparing folding codes for proteins and polymers. **Proteins: Structure, Function and Genetics**, v. 24, p. 335–344, 1996.

CHANDRU, V.; DATTASHARMA, A.; KUMAR, V. The algorithmics of folding proteins on lattices. **Discrete Applied Mathematics**, Elsevier Science Publishers B. V., Amsterdam, Holland, v. 127, n. 1, p. 145–161, 2003.

CHU, D.; TILL, M.; ZOMAYA, A. Parallel ant colony optimization for 3D protein structure prediction using the HP lattice model. In: **Proc.** 19^{th} **IEEE Int. Parallel and Distributed Processing Symp.** Washington, DC, USA: IEEE Computer Society, 2005. p. 193–199.

COOPER, G. **The Cell: A Molecular Approach**. Sunderland, UK: Sinauer Associates, 2000.

COX, G. et al. Development and optimisation of a novel genetic algorithm for studying model protein folding. **Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)**, v. 112, n. 3, p. 163–178, 2004.

CRESCENZI, P. et al. On the complexity of protein folding. **Journal of Computational Biolology**, v. 5, p. 423–446, 1998.

CUSTÓDIO, F.; BARBOSA, H.; DARDENNE, L. Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm. **Genetics and Molecular Biology**, v. 27, n. 4, p. 611–615, 2004.

CUTELLO, V.; NARZISI, G.; NICOSIA, G. A class of pareto archived evolution strategy algorithms using immune inspired operators for ab-initio protein structure prediction. In: **Applications on Evolutionary Computing**. Heidelberg, Germany: Springer-Verlag, 2005. v. 3449, p. 54–63.

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

DAWSON, R. et al. The N-terminal domain of p53 is natively unfolded. **Journal of Molecular Biology**, v. 332, p. 1131–1141, 2003.

DEB, K. **Multi-Objective Optimization using Evolutionary Algorithms**. Chichester, UK: John Wiley & Sons, 2001.

DILL, K. Dominant forces in protein folding. **Biochemistry**, v. 29, n. 31, p. 7133–7155, 1990.

DILL, K. Polymer principles and protein folding. **Protein Science**, Cambridge University Press, USA, v. 8, n. 6, p. 1166–1180, 1999.

DILL, K. et al. Principles of protein folding - a perspective from simple exact models. **Protein Science**, v. 4, n. 4, p. 561–602, 1995.

DONGARRA, J. et al. **Sourcebook of Parallel Computing**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. Cambridge, MA, USA: MIT Press, 2004.

DRENTH, J. **Principles of Protein X-Ray Crystallography**. New York, NY, USA: Springer-Verlag, 1999.

EL-REWINI, H.; ABD-EL-BARR, M. **Advanced Computer Architecture and Parallel Processing**. Hoboken, NJ, USA: Wiley, 2005.

ELLIS, R. Chaperonins: introductory perspective. In: ELLIS, R. (Ed.). **The Chaperonins**. San Diego, CA, USA: Academic Press, 1996. p. 2–25.

FIDANOVA, S. 3D HP protein folding problem using ant algorithm. In: **Proceedings of BioPS'06 International Conference**. Sofia, Bulgaria: CD-ROM, 2006. p. III.19–III.26.

FLYNN, M. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, v. 21, n. C, p. 948–960, 1972.

FOGEL, L. Autonomous automata. **Industrial Research**, v. 4, n. 2, p. 14–19, 1962.

FRAENKEL, A. Complexity of protein folding. **Bulletin of Mathematical Biology**, v. 55, n. 6, p. 1199–1210, 1993.

FRANK, J. **Three-Dimensional Electron Microscopy of Macromolecular Assemblies**. New York, NY, USA: Oxford University Press, 2006.

FRIEDBERG, R.; DUNHAM, B.; NORTH, J. A learning machine: Part I. **IBM Journal**, v. 2, n. 1, p. 2–13, 1958.

GEIST, A. et al. **PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Network Parallel Computing**. Cambridge, MA, USA: MIT Press, 1994.

GETHING, M.-J.; SAMBROOK, J. Protein folding in the cell. **Nature**, v. 355, p. 33–45, 1992.

GOLDBERG, D. **Genetic Algorithms in Search, Optimization and Machine Learning**. Boston, USA: Addison-Wesley, 1989.

GRIFFITHS, A. et al. **An Introduction to Genetic Analysis**. 7th. ed. New York, NY, USA: Freeman, 2000.

GROPP, W.; LUSK, E.; THAKUR, R. **Using MPI-2: Advanced Features of the Message-Passing Interface**. Cambridge, MA, USA: MIT Press, 1999.

HARDIN, C.; POGORELOV, T.; LUTHEY-SCHULTEN, Z. Ab initio protein structure prediction. **Current Opinion in Structural Biology**, v. 12, n. 2, p. 176–181, 2002.

HART, W.; ISTRAIL, S. Fast protein folding in the hydrophobic-hydrofilic model within three-eights of optimal. **Journal of Computational Biology**, v. 3, p. 53–96, 1996.

HART, W.; 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.

HEUN, V. Approximate protein folding in the HP side chain model on extended cubic lattices. **Discrete Applied Mathematics**, v. 127, p. 163–177, 2003.

HIRST, J. The evolutionary landscape of functional model proteins. **Protein Engineering**, v. 12, n. 9, p. 721–726, 1999.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Cambridge, MA, USA: MIT Press, 1975.

HSU, H.; MEHRA, V.; GRASSBERGER, P. Structure optimization in an off-lattice protein model. **Physical Review E**, v. 68, n. 3, p. 037703, 2003.

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

HUTTON, M. et al. Analysis of taupathies with transgenic mice. **Trends in Molecular Medicine**, v. 7, p. 467–470, 2001.

IRBACK, A.; PETERSON, C.; POTTHAST, F. Identification of amino acid sequences with good folding properties in an off-lattice model. **Physical Review E**, v. 55, n. 1, p. 860–867, 1997.

ISHIMARU, D. et al. Fibrillar aggregates of the tumor suppressor p53 core domain. **Biochemistry**, v. 42, n. 30, p. 9022–9027, 2003.

ISLAM, S.; LUO, J.; STERNBERG, M. Identification and analysis of domains in proteins. **Protein Engineering**, v. 8, n. 6, p. 513–525, 1995.

JARONIEC, C. et al. High resolution molecular structure of a peptide in an amyloid fibril determined by magic angle spinning NMR spectroscopy. **Proceedings of the National Academy of Sciences**, USA, v. 101, n. 3, p. 711–716, 2004.

JIANG, T. et al. Protein folding simulations of the hydrophobic-hydrophilic model by combining tabu search with genetic algorithms. **Journal of Chemical Physics**, v. 119, n. 8, p. 4592–4596, 2003.

JIMENEZ, J. L. et al. The protofilament structure of insulin amyloid fibrils. **Proceedings of the National Academy of Sciences**, USA, v. 50, n. 14, p. 9196–9201, 2002.

KALEGARI, D.; LOPES, H. A differential evolution approach for protein structure optimisation using a 2d off-lattice model. **International Journal of Bio-Inspired Computation**, v. 2, n. 3/4, p. 242–250, 2010.

KARABOGA, D. **An idea based on honey bee swarm for numerical optimization.** Kayseri, Turkey, 2005.

KARPLUS, M. The Levinthal paradox: yesterday and today. **Folding & design**, v. 2, n. 4, p. S69–S75, 1997.

KAUZMANN, W. Some factors in the interpretation of protein denaturation. **Advances in Protein Chemistry**, v. 97, p. 1–63, 1959.

KENNEDY, J.; EBERHART, R. Particle Swarm Optimization. In: **Proceedings of the IEEE International Conference on Neural Networks**. Piscataway, NJ, USA: IEEE Computer Society, 1995. v. 4, p. 1942–1948.

KIRLEY, M.; GREEN, D. Adaptation and spatial patterns: Optimization using the cellular genetic algorithm. In: **Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program**. San Francisco, CA, USA: Morgan Kaufmann, 2000. p. 12–16.

KRASNOGOR, N. et al. Multimeme algorithms for protein structure prediction. In: **Proceedings of the 7^{th} International Conference on Parallel Problem Solving from Nature**. Heidelberg, Germany: Springer, 2002. p. 769–778.

KRASNOGOR, N. et al. Protein structure prediction with evolutionary algorithms. In: **Proc. Genetic and Evolutionary Computation Conf.** San Francisco, CA, USA: Morgan Kaufmann, 1999. p. 1596–1601.

KRETSINGER, R. Structure and evolution of calcium-modulated proteins. **Critical Reviews in Biochemestry**, v. 8, p. 119–174, 1980.

LEE, M.; DUAN, Y.; KOLLMAN, P. State of the art in studying protein folding and protein structure prediction using molecular dynamics methods. **Journal of Molecular Graphics and Modelling**, v. 19, p. 146–149, 2001.

LEINONEN, R. et al. UniProt archive. **Bioinformatics**, v. 20, n. 17, p. 3236–3237, 2004.

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

LEWIS, P.; MOMANY, F.; SCHERAGA, H. Chain reversals in proteins. **Biochim Biophys Acta 303**, n. 2, p. 211–229, 1973.

LI, H. et al. Emergence of preferred structures in a simple model of protein folding. **Science**, v. 273, p. 666–669, 1996.

LI, M.; KLIMOV, D.; THIRUMALAI, D. Folding in lattice models with side chains. **Computer Physics Communications**, v. 147, n. 1, p. 625–628, 2002.

LI, X. Protein folding based on simulated annealing algorithm. In: **Proceedings of International Conference on Natural Computation**. Los Alamitos, CA, USA: IEEE Computer Society, 2007. v. 4, p. 256–259.

LOBO, F.; LIMA, C.; MICHALEWICZ, Z. **Parameter Setting in Evolutionary Algorithms**. New York, NY, USA: Springer, 2007.

LODISH, H. et al. **Molecular Cell Biology**. 4^{th}. ed. New York, NY, USA: Freeman, 2000.

LOPES, H. Evolutionary algorithms for the protein folding problem: A review and current trends. In: SMOLINSKI, T.; MILANOVA, M.; HASSANIEN, A.-E. (Ed.). **Computational Intelligence in Biomedicine and Bioinformatics**. Heidelberg, Germany: Springer-Verlag, 2008. v. I, p. 297–315.

LOPES, H.; SCAPIN, M. An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. **Lecture Notes in Computer Science**, v. 3871, p. 238–246, 2005.

MACARIO, A.; MACARIO, E. de. Sick chaperones and ageing: a perspective. **Ageing Research Reviews**, v. 1, n. 2, p. 295–311, 2002.

MARUO, M.; LOPES, H.; DELGADO, M. Self-adapting evolutionary parameters: encoding aspects for combinatorial optimization problems. **Lecture Notes in Computer Science**, v. 3448, p. 154–165, 2005.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. **ACM Transactions on Modeling and Computer Simulation**, v. 8, n. 1, p. 3–30, 1998.

MAYER, M.; BUKAU, B. Hsp70 chaperones: Cellular functions and molecular mechanism. **Cellular and Molecular Life Sciences**, v. 62, p. 670–684, 2005.

MCNAUGHT, K. et al. Failure of the ubiquitin-proeasome system in Parkinson's disease. **Nature Reviews**, v. 2, p. 589–594, 2001.

MILLER, B. L. et al. Genetic algorithms, tournament selection, and the effects of noise. **Complex Systems**, v. 9, p. 193–212, 1995.

NELSON, D.; COX, M. **Lehninger Principles of Biochemistry**. 5^{th}. ed. Boston, USA: W.H. Freeman, 2008.

NEWMAN, A. A new algorithm for protein folding in the hp model. In: **Proceedings of the 13th Annual Symposium on Discrete Algorithms (SODA)**. San Francisco, CA, USA: Society for Industrial and Applied Mathematics, 2002. p. 876–884.

NGO, J.; MARKS, J.; KARPLUS, M. Computational complexity, protein structure prediction, and the levinthal paradox. In: MERZ, K.; LEGRAND, S. (Ed.). **The Protein Folding Problem and Terciary Structure Prediction**. Boston, USA: Birkhäuser, 1994. p. 433–506.

NICOSIA, G.; STRACQUADANIO, G. Generalized pattern search algorithm for peptide structure pediction. **Biophysical Journal**, v. 95, p. 4988–4999, 2008.

NÖLTING, B. **Protein Folding Kinetics**. 2nd. ed. Berlin, Germany: Springer, 2006.

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

OSTROVSKY, B. et al. Cellular automata for polymer simulation with application to polymer melts and polymer collapse including implications for protein folding. **Parallel Computing**, v. 27, n. 5, p. 613–641, 2001.

PAIN, R. **Protein Folding**. Oxford, UK: Oxford University Press, 2000.

PARHAMI, B. **Introduction to Parallel Processing**. New York, NY, USA: Plenum, 2002.

PATTON, A.; III, W. P.; GOODMAN, E. A standard GA approach to native protein conformation prediction. **Proc. 6th Int. Conf. on Genetic Algorithms**, p. 574–581, 1995.

PAULING, L.; COREY, R.; BRANSON, H. 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**, USA, v. 37, p. 729–740, 1951.

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

PEDERSEN, C. **Algorithms in Computational Biology**. Tese (PhD Thesis) — Department of Computer Science, University of Aarhus, Denmark, 2000.

PEDERSEN, J.; MOULT, J. Protein folding simulations with genetic algorithms and a detailed molecular description. **Journal of Molecular Biology**, v. 269, p. 240–259, 1997.

PERRETO, M. **Aplicação do algoritmo de otimização por colônia de formigas aos problemas de reconstrução de árvores filogenéticas e dobramento de proteínas**. Dissertação (Mestrado) — UTFPR, 2005.

PICCOLBONI, A.; MAURI, G. **Application of Evolutionary Algorithms to Protein Folding Prediction**. 1998.

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

POSPICHAL, P.; JAROS, J.; SCHWARZ, J. Parallel Genetic Algorithm on the CUDA Architecture. In: **Lecture Notes in Computer Science**. Heidelberg, Germany: Springer-Verlag, 2010. v. 6024, p. 442–451.

RECHENBERG, I. **Cybernetic solution path of an experimental problem**. Royal Aircraft Establishment, translation No. 1122, Ministry of Aviation, Farnborough Hants, UK, 1965.

REINHARD, J.; SRINIVASAN, S. The role of scents in honey bee foraging and recruitment. In: **Food Exploitation by Social Insects: Ecological, Behavioral, and Theoretical Approaches**. 1st. ed. Boca Raton, FL, USA: CRC Press, 2009. cap. 9, p. 165–182.

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

ROOSTA, S. **Parallel Processing and Parallel Algorithms: Theory and Computation**. New York, NY, USA: Springer-Verlag, 1999.

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 of the USA**, v. 83, n. 5, p. 1198–1202, 1986.

SCAPIN, M.; LOPES, H. A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model. In: YANG, A.; SHAN, Y.; BUI, L. (Ed.). **Success in Evolutionary Computation**. Heidelberg, Germany: Springer, 2007. p. 205–224.

SCAPIN, M.; LOPES, H. A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model. In: **Success in Evolutionary Computation**. Heidelberg, Germany: Springer-Verlag, 2007. p. 205–224.

SCHMID, F. Kinetics of unfolding and refolding of single domain proteins. In: **Protein Folding**. New York, NY, USA: W.H. Freeman & Sons, 1992. p. 197–241.

SEELEY, T. **The Wisdom of the Hive**. Cambridge, MA, USA: Harvard University Press, 1995.

SELKOE, D. Clearing the brain's amyloid cobwebs. **Neuron**, v. 32, n. 2, p. 177–180, 2001.

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

SHMYGELSKA, A.; HOOS, H. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. **Bioinformatics 6**, p. 6–30, 2005.

SONG, J. et al. A novel genetic algorithm for HP model protein folding. In: **Proc.** 6^{th} **Int. Conf. on Parallel and Distributed Computing Applications and Technologies**. Los Alamitos, CA, USA: IEEE Computer Society, 2005. p. 935–937.

STERLING, T. **Beowulf Cluster Computing with Linux**. Cambridge, MA, USA: MIT Press, 2002.

STILLINGER, F.; HEAD-GORDON, T. Collective aspects of protein folding illustrated by a toy model. **Physical Review E**, v. 52, n. 3, p. 2872–2877, 1995.

STILLINGER, F.; HEAD-GORDON, T.; HIRSHFELD, C. Toy model for protein folding. **Physical Review E**, v. 48, n. 2, p. 1469–1477, 1993.

STÜTZLE, T. Parallelization strategies for ant colony optimization. In: **Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature**. London, UK: Springer-Verlag, 1998. p. 722–731.

SUNDE, M.; BLAKE, C. The structure of amyloid fibrils by electron microscopy and x-ray diffraction. **Advances in Protein Chemistry**, v. 50, 1997.

TANTAR, A. et al. A parallel hybrid genetic algorithm for protein structure prediction on the computational grid. **Future Generation Computer Systems**, v. 23, n. 3, p. 398–409, 2007.

TAVARES, L.; LOPES, H.; ERIG, C. A study of topology in insular parallel genetic algorithms. In: **Proc. World Congress on Nature and Biologically Inspired Computing (NABIC)**. Piscataway, NJ, USA: IEEE Computer Press, 2009. p. 632–635.

THACHUK, C.; SHMYGELSKA, A.; HOOS, H. A replica exchange Monte Carlo algorithm for protein folding in the HP model. **BMC Bioinformatics**, v. 8, p. 342+, 2007.

THOMAS, P.; DILL, K. Local and nonlocal interactions in globular proteins and mechanisms of alcohol denaturation. **Protein Science**, v. 2, p. 2050–2065, 1993.

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

TORCINI, A.; LIVI, R.; POLITI, A. A dynamical approach to protein folding. **Journal of Biological Physics**, v. 27, n. 2–3, p. 181–203, 2001.

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 3D protein folding simulations. In: **Proc. 5^{th} Ann. Int. Conf. on Genetic Algorithms**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993b. p. 581–588.

URSINI, F. et al. Atherosclerosis: another protein misfolding disease? **Trends in Molecular Medicine**, v. 8, p. 370–374, 2002.

VENTER, G.; SOBIESZCZANSKI-SOBIESKI, J. A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. In: HERSKOVITS, J.; MAZORCHE, S.; CANELAS, A. (Ed.). **6th World Congresses of Structural and Multidisciplinary Optimization**. Rio de Janeiro - Brazil: CD-ROM, 2005.

WETLAUFER, D. Nucleation, rapid folding and globular intrachain regions in proteins. **Proceedings of the National Academy of Sciences of the USA**, USA, v. 70, n. 3, p. 697–701, 1973.

WÜTHRICH, K. **NMR of Proteins and Nucleic Acids**. New York, NY, USA: John Wiley & Sons, 1986.

YANG, X.-S. **Nature-Inspired Metaheuristic Algorithms**. York, UK: Luniver Press, 2008. ISBN 1905986106, 9781905986101.

YANIKOGLU, B.; ERMAN, B. Minimum energy configurations of the 2-dimensional hp-model of proteins by self-organizing networks. **J. Comput. Biol.**, v. 9, n. 4, p. 613–620, 2002.

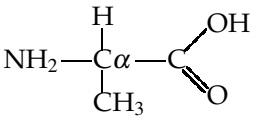
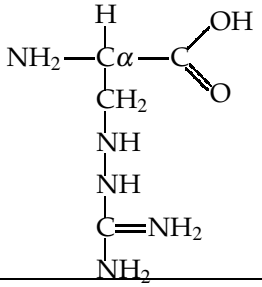
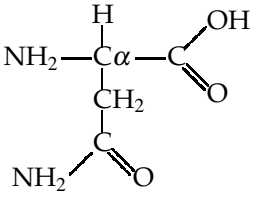
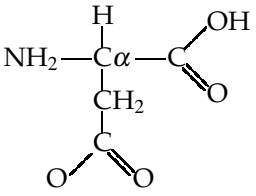
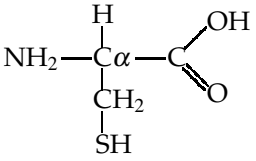
YUE, K.; DILL, K. Sequence-structure relationships in proteins and copolymers. **Physical Review E**, v. 48, n. 3, p. 2267–2278, 1993.

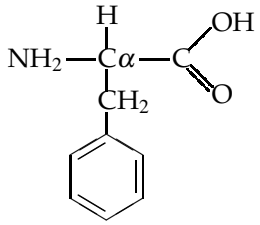
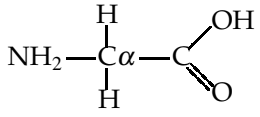
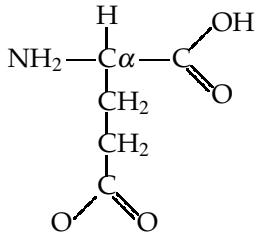
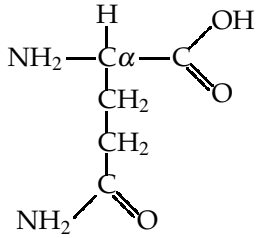
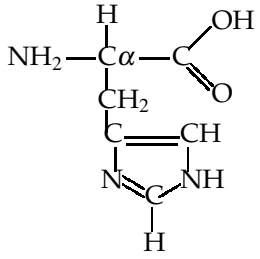
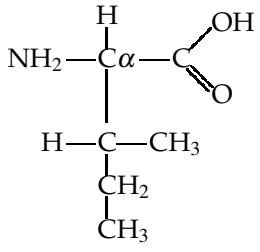
YUE, K.; FIEBIG, K.; AL., P. T. et. A test of lattice protein folding algorithms. **Proceedings of the National Academy of Sciences of USA**, V. 91, p. 581–588, 1994.

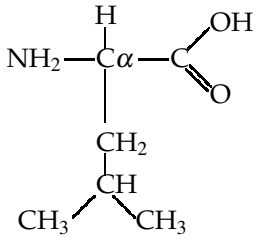
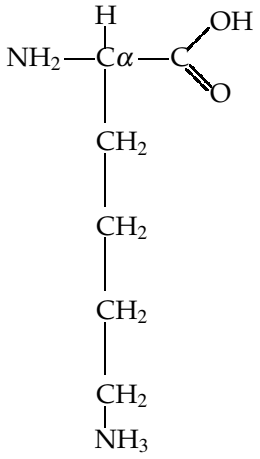
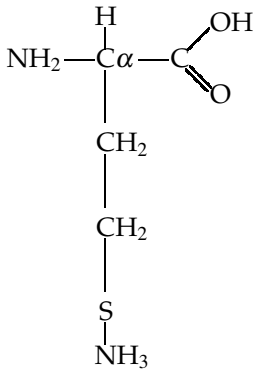
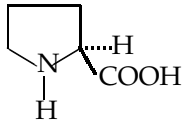
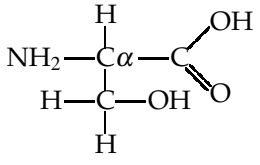
ZHANG, X.; CHENG, W. An Improved Tabu Search Algorithm for 3D Protein Folding Problem. In: **PRICAI 2008: Trends in Artificial Intelligence**. Heidelberg, Germany: Springer-Verlag, 2008. v. 5351, p. 1104–1109.

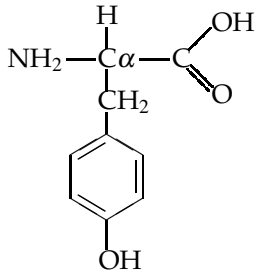
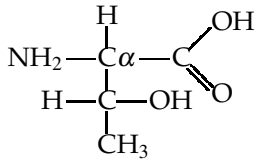
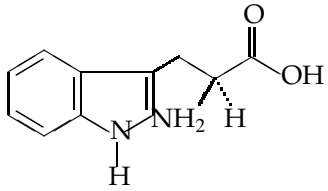
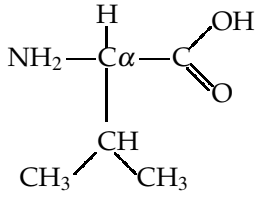
ANEXO A – LISTA DE AMINOÁCIDOS

Tabela 20: Lista de aminoácidos

Aminoácido	Código de 3 letras	Código de 1 letra	Sequência	Polaridade
Alanina	Ala	A		H
Arginina	Arg	R		H
Asparagina	Asn	N		P
Aspartato	Asp	D		P
Cisteína	Cis	C		P

Fenilalanina	Fen	F		H
Glicina	Gli	G		P
Glutamato	Glu	E		P
Glutamina	Gln	Q		P
Histidina	His	H		H
Isoleucina	Ile	I		H

Leucina	Leu	L		H
Lisina	Lis	K		H
Metionina	Met	M		H
Prolina	Pro	P		H
Serina	Ser	S		P

Tirosina	Tir	Y		P
Treonina	Ter	T		P
Triptofano	Trp	W		H
Valina	Val	V		H

ANEXO B – FUNÇÕES BÁSICAS DA BIBLIOTECA MPI

Tabela 21: Funções básicas da biblioteca MPI

Função	Efeito
MPI_Init	Inicializa o ambiente MPI. Esta função deve ser chamada por todos os processadores, pois estabelece o ambiente necessário para executar os processos. Também sincroniza todos os processos na inicialização de uma aplicação paralela.
MPI_Comm_size	Determina o número de processos de um grupo definido por um comunicador
MPI_Comm_rank	Descobre o *rank* de um processo. Todos os processos devem executar esta função para , posteriormente, saberem as ações que deverão executar.
MPI_Send	Esta função realiza o envio de mensagens de um processo remetente para um processo destinatário específico ou a todos os processos. Esta função é do tipo bloqueante, ou seja, interrompe a execução do programa até ter certeza que a mensagem foi recebida pelo destinatário.
MPI_Recv	Esta função recebe uma mensagem enviada por um remetente específico ou de qualquer processo do grupo. Esta função é do tipo bloqueante, ou seja, interrompete a execução do programa até ter certeza que a mensagem foi recebida.
MPI_Finalize	Finaliza a e execução do processo no ambiente MPI. Esta função deve ser a última rotina MPI chamada pelo programa. Deve ser chamada por todos os processos inicializados.
MPI_Pack	Empacotamento de dados para transmissão.
MPI_Unpack	Desempacotamento de um conjunto de dados.
MPI_Cart_Create	Cria uma topologia cartesiana de processos.