

VIDAL DANIEL DA FONTOURA

META-HEURÍSTICAS E HIPER-HEURÍSTICAS APLICADAS AO PROBLEMA DE
DOBRAMENTO DE PROTEÍNAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Aurora Trinidad Ramirez Pozo.

Co-orientador: Roberto Santana .

CURITIBA PR

2016

Resumo

Proteínas são estruturas, compostas por aminoácidos, que exercem um papel de suma importância na natureza. Estas estruturas são formadas a partir de um processo de dobramento, onde uma sequência de aminoácidos inicialmente desdobrada irá adotar uma conformação/estrutura espacial única/nativa. Entretanto, o processo de dobramento ainda não é completamente compreendido e é considerado um dos maiores desafios das áreas de biologia, química, medicina e bioinformática. Este desafio é conhecido como o problema de dobramento de proteínas (PDP) e trata da predição de estruturas de proteínas.

O PDP pode ser visto como um problema de minimização, pois afirmam que a estrutura nativa de uma proteína é aquela que minimiza sua energia global livre. Dessa maneira, diversas estratégias heurísticas utilizando modelos simplificados de representação de proteínas, tais como modelo HP, vem sendo desenvolvidas para buscar a estrutura nativa de proteínas. Por se tratar de um problema complexo com muitos mínimos locais e restrições, surge a demanda de estratégias que possuam mecanismos robustos para manter a diversidade assim como intensificar a busca quando necessário. É nesse contexto que hiper-heurísticas se apresentam como boas opções para explorar o espaço de busca de problemas complexos.

Nesta dissertação, são apresentadas duas abordagens para o PDP. A primeira visa a aplicação de algoritmos evolucionários multi objetivos para resolver o PDP. A segunda consiste no *design* automático de heurísticas de alto nível utilizando uma técnica de programação genética chamada evolução gramatical, a qual utiliza uma gramática BNF para gerar programas de computador.

As estratégias propostas foram aplicadas utilizando um conjunto de benchmark com diferentes sequências de aminoácidos. Os resultados foram comparados com outros trabalhos que utilizam o mesmo conjunto de *benchmark*. Alguns resultados obtidos se mostraram promissores dessa maneira motivando novos estudos que visem estratégias adaptativas para o PDP.

Palavras-chave: PDP, hiper heurísticas, evolução gramatical, otimização multi objetiva.

Abstract

Proteins are structures, composed by amino-acids, which plays a important role in nature. These structures are built by a process called protein folding, where a sequence of amino-acids initially unfolded will obtain your native structure. However, the protein folding process is not entire understood and it is considered one of the most challenging problem from biology, chemistry, medicine and bio-informatics. This problem is knows as the protein folding problem (PFP) and handles the prediction of protein structures.

The PFP is a minimization problem, because the proteins native structures are the one within minimum energy. Thus, many heuristics strategies make use of simplified models, such as the HP model, to find the proteins native structures within the HP model. Since the PDP is a very complex problem with lots of local optima, it raises the necessity of building strategies that have robust mechanisms to maintain the diversity. In this context, that adaptive strategies fits well as good options to explore the fitness landscape from complex problems.

In this dissertation two approaches are presented for the PFP. The first one aims the application of multi objective evolutionary algorithms to solve the PFP. The second consisted on the automated design of high level heuristics through a type of genetic programming called grammatical evolution, which uses a BNF grammar to generate computer programs.

Both strategies were applied using a benchmark set with different amino-acids sequences. The results were compared with other previous studies which used the same amino-acids sequences. In some cases the results obtained were promising that way motivating the development of new adaptive strategies to the PFP.

Keywords: PFP, hyper heuristics, evolução grammatical, otimização multo objetiva.

Sumário

1	Introdução	1
1.1	Organização do Texto	2
2	Problema de Dobramento de Proteínas	3
2.1	PDP - Problema de Dobramento de Proteínas	3
2.2	Dobramento de proteínas	3
2.3	Modelos de Representação de Proteínas	6
2.3.1	Modelos Discretos	6
2.3.2	Conclusão	9
3	Referencial Teórico	10
3.1	AEs - Algoritmos Evolucionários	10
3.1.1	NSGAI - Non-dominated sorting Genetic Algorithm II	11
3.1.2	IBEA (Indicator-Based Evolutionary Algorithm)	11
3.2	Hiper-Heurísticas	13
3.2.1	Hiper-heurísticas de Geração	16
3.3	Programação Genética (PG)	16
3.4	Evolução Gramatical (EG)	17
3.5	Programação Genética como Hiper-Heurística de Geração de Heurísticas . . .	20
3.6	Conclusão	22
4	Trabalhos Relacionados	24
4.1	Conclusão	25
5	Metodologia	27
5.1	AEMOs aplicados ao PDP	30
5.1.1	Funções Objetivo	31
5.2	EGHyPDP	32
5.3	Heurísticas de alto nível	32
5.3.1	Função de <i>Fitness</i>	35
5.3.2	Critério de Parada	35

5.4	Heurísticas de baixo nível	35
5.4.1	Conjunto de heurísticas de baixo nível	35
5.5	Processo geral do EGHyPDP	36
5.5.1	Mecanismo de Memória	36
5.6	Considerações Finais	37
6	Experimentos	38
6.1	Resultados dos AEMOs aplicados ao PDP	39
6.1.1	Comparação com outras abordagens mono-objetivas	40
6.1.2	Conclusão dos experimentos utilizando os AEMOs	41
6.2	Resultados obtidos com EGHyPDP	42
6.2.1	Resultados obtidos com EGHyPDP-1	43
6.2.2	Resultados obtidos com EGHyPDP-2	44
6.2.3	Resultados obtidos com EGHyPDP-3	45
6.2.4	Comparação com <i>framework</i> hiper heurístico GIHH	46
6.2.5	Discussão	46
6.2.6	Conclusão dos experimentos utilizando o EGHyPDP	47
7	Conclusão	49
	Referências Bibliográficas	51

Lista de Figuras

3.1	Framework Geral Hiper-Heurístico. Adaptado de [Sabar et al., 2015]	15
3.2	Classificação Hiper-heurísticas. Adaptado de [Sabar et al., 2015]	16
5.1	Cromossomo decodificado que representa uma possível conformação para a cadeia HHPPHPPHHH	28
5.2	Exemplo de aplicação do operador 2x. Fonte Autoria Própria	28
5.3	Exemplo de aplicação do operador MPX. Fonte Autoria Própria	28
5.4	Exemplo de aplicação do operador SMUT. Fonte Autoria Própria	29
5.5	Exemplo de aplicação do operador LM. Fonte Autoria Própria	29
5.6	Exemplo de aplicação do operador LPM. Fonte Autoria Própria	29
5.7	Exemplo de aplicação do operador OM. Fonte Autoria Própria	30
5.8	<i>Estrutura geral do EGHyPDP</i> . Fonte: Adaptado de [Sabar et al., 2015]	32

Lista de Tabelas

3.1	<i>Regras de produção</i> e o número de escolhas para cada uma.	18
6.1	Instâncias de <i>benchmark</i> utilizadas nos experimentos.	38
6.2	Tamanho da população, número máximo de avaliações para cada instância . . .	39
6.3	Resultado de média/desvio padrão dos AEMOs	40
6.4	Comparação dos melhores AEMOs com o estudos anteriores do PDP	41
6.5	Resultados da execução do melhor indivíduo encontrado pelo grupo de experi- mento EGHyPDP-1	44
6.6	Resultados da execução do melhor indivíduo encontrado no grupo de experi- mento EGHyPDP-2	45
6.7	Resultados da execução do melhor indivíduo encontrado no grupo de experi- mento EGHyPDP-3	45
6.8	Os melhores resultados conhecidos e resultados encontrados com o EGHyPDP e GIHH.	46

Lista de Acrônimos

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná

Lista de Símbolos

α	alfa, primeira letra do alfabeto grego
β	beta, segunda letra do alfabeto grego
γ	gama, terceira letra do alfabeto grego
ω	ômega, última letra do alfabeto grego
π	pi
τ	Tempo de resposta do sistema
θ	Ângulo de incidência do raio luminoso

Capítulo 1

Introdução

As proteínas são responsáveis por muitas funções importantes das células vivas. Estas estruturas garantem o correto funcionamento de um amplo número de entidade biológicas. As proteínas são o produto de um processo chamado de dobramento de proteínas, no qual uma cadeia de aminoácidos inicialmente desdobrada será transformada em sua estrutura final/nativa. A predição de estruturas de proteínas possui um campo amplo de aplicações biotecnológicas e médicas. Por exemplo: síntese de novas proteínas e dobramentos [Wang et al., 2012, Röthlisberger et al., 2008], síntese de novas drogas baseada nas estruturas [Qian et al., 2004, Krieger et al., 2009] e obtenção experimental de estruturas a partir de dados incompletos de ressonância magnética nuclear [Shen et al., 2009].

Determinar a estrutura nativa de proteínas uma tarefa desafiadora até mesmo para modernos super computadores. Isto ocorre por conta do imenso espaço de busca para testar todas as possíveis configurações que uma dada proteína pode adotar. Diferentes formas de representar as estruturas/conformações de proteínas existem e podem ser utilizadas para simular o processo de dobramento. Embora existam modelos extremamente detalhados, estas representações são computacionalmente muito custosas. Consequentemente, muitos autores [Custódio et al., 2004, Hsu et al., 2003, Lin e Su, 2011, Unger e Moult, 1993b, Sant’Anna, 2008, Custódio et al., 2014, Garza-Fabre et al., 2012] utilizam modelos simplificados para representar as estruturas de proteínas. Um modelo bastante conhecido para este propósito, criado por Lau and Dill [Lau e Dill, 1989], é o modelo Hidrofóbico-Polar (HP). Este modelo simplifica os amino ácidos em apenas dois tipos: hidrofóbico (H) e polar (P).

Para avaliar as estruturas representadas pelo modelo HP é preciso computar o valor de energia associado a uma dada conformação [Unger e Moult, 1993b]. Para isto, é necessário considerar as interações entre os aminoácidos. Uma interação ocorre quando o par de aminoácidos é adjacente no *grid*/cubo e não é adjacente na sequência. No modelo HP existem apenas três: (HP, HH e PP), mas somente interações hidrofóbicas (HH) influenciam no valor de energia referente a uma conformação [Unger e Moult, 1993b]. A questão que surge é como buscar, dentre as possíveis conformações, aquela cuja a energia seja mínima. Embora, o modelo HP seja um modelo simplificado este se apresenta como um problema NP-Completo e as instâncias apresentam

uma grande variabilidade entre si. Dessa maneira, tornando muito difícil para os algoritmos manter bons resultados em várias instâncias do modelo HP. Outra dificuldade encontrada por pesquisadores é a configuração dos parâmetros para os algoritmos

É nesse contexto que as hiper-heurísticas e estratégias adaptativas, tais como utilizar operadores genéticos de maneira dinâmica, geralmente são aplicadas e vem apresentando bons resultados [Burke et al., 2013]. Entretanto, não existem muitas abordagens que visam o projeto automático de novas heurísticas ou automatização da seleção de heurísticas existentes para o PDP. Assim como a maioria dos trabalhos trata o PDP de maneira mono objetiva. Consequentemente, existem poucos trabalhos que propõem estratégias multi objetivas.

Esta dissertação apresenta duas abordagens para o PDP. A primeira trata-se de uma estratégia multi objetiva para o PDP utilizando algoritmos evolucionários multi objetivos (AEMOs) do estado da arte. Duas adaptações foram propostas para os AEMOs com objetivo de melhorar os resultados obtidos. A segunda abordagem consistiu em uma estratégia de design automático de heurísticas de alto nível para um *framework* hiper heurístico.

1.1 Organização do Texto

O restante desta dissertação está organizado da seguinte maneira: no capítulo 2 é introduzido o referencial teórico necessário para uma boa compreensão sobre o PDP. Já no capítulo 3 são apresentados conceitos relacionados com os AEMOs, assim como os algoritmos que foram utilizados nesta dissertação. Ainda são discutidos os aspectos referentes as hiper-heurísticas, sua classificação e também conceitos de programação genética (PG) e evolução gramatical (EG). Em seguida, o capítulo 4 apresenta estudos relacionados com esta dissertação. As duas abordagens propostas são apresentadas no capítulo 5. Os experimentos realizados com as duas abordagens são apresentados no capítulo 6. Finalmente, o capítulo 7 apresenta a conclusão desta dissertação.

Capítulo 2

Problema de Dobramento de Proteínas

2.1 PDP - Problema de Dobramento de Proteínas

Proteínas são estruturas básicas, essenciais para vida e possuem incontáveis funções biológicas. Proteínas são sintetizadas pelos ribossomos seguindo um formato provido pelo mensageiro RNA (mRNA). Durante a síntese, as proteínas dobram (enovelam) em uma estrutura tridimensional única, conhecida como conformação nativa. Este processo é chamado de dobramento de proteínas (*protein folding*). A função biológica de uma proteína depende da sua estrutura tridimensional.

As proteínas são polímeros compostos por sequências de aminoácidos (também chamados de resíduos) conectados linearmente por ligações peptídicas. Cada aminoácido é composto por um átomo central de carbono ($C\alpha$) conectado a um átomo de hidrogênio, um grupo amina, um grupo carboxila e uma cadeia lateral (*side-chain*) a qual confere a cada aminoácido uma função distinta. Uma ligação peptídica é formada por dois aminoácidos quando o grupo carboxila de uma molécula reage com o grupo amina da outra. Este processo de agregação de aminoácidos é conhecido como desidratação pois libera uma molécula de água (H_2O) [Suzuki et al., 1986]. Proteínas podem ser chamadas de cadeias polipeptídicas. Todos os aminoácidos tem o mesmo *backbone* e se diferem dos outros apenas pela *side-chain*, a qual pode ser um simples átomo de hidrogênio ou até um grupo heterocíclico complexo. A *side-chain* define as propriedades físicas e químicas dos aminoácidos de uma proteína [Cox et al., 2013].

2.2 Dobramento de proteínas

É o processo em que cada cadeia polipeptídica é transformada em uma estrutura compacta que realiza alguma função biológica [Grantcharova et al., 2001]. Estas funções incluem controle e regulação de processos químicos essenciais para os organismos vivos [Branden et al., 1999]. A estrutura tridimensional mais estável é chamada de conforma-

ção nativa e é a qual permite que a proteína exerça corretamente sua função biológica [Lodish et al., 2000, Pedersen, 2000].

Experimentos conduzidos por Anfinsen et al. [Sela et al., 1957, Anfinsen, 1972, Anfinsen et al., 1961], mostraram que as proteínas possuem apenas uma conformação nativa e que as informações essenciais que codificam a estrutura estão contidas na sequência de aminoácidos. A conformação tridimensional nativa é dada pela estrutura primária (sequência de aminoácidos) de uma proteína.

Muitas proteínas podem desnaturar por modificações no ambiente em que estão inseridas, conforme demonstrado por [Sela et al., 1957, Anfinsen, 1972, Anfinsen et al., 1961]. Durante o processo de desnaturação as proteínas perdem sua forma nativa (desdobram) e, conseqüentemente, perdem sua função. O exemplo mais conhecido de desnaturação proteica é o da clara do ovo. A clara do ovo é composta por água e albumina. A albumina é uma proteína polar, portanto solúvel em água. Ao fritar ou cozinhar o ovo, eleva-se a temperatura, levando à desnaturação da albumina que, mesmo ao retornar à temperatura original, não consegue voltar à sua conformação nativa. Além de se desdobrarem é possível que ocorram erros de dobramento na formação das proteínas causando com que a proteína não exerça sua função biológica corretamente. Estudos tentam identificar causas para os erros de dobramento das proteínas pois, muitas enfermidades são causadas por conta disto, 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 et al., 1992], arteriosclerose [Ursini et al., 2002], mal de Parkinson [McNaught et al., 2001], entre outras. Portanto, entender como o processo de dobramento de proteínas ocorre é de fundamental importância.

Um dos objetivos comuns das ciências biológicas é caracterizar funcionalmente sequências de proteínas através da resolução de suas conformações nativas [Eswar et al., 2003]. Varias áreas da ciência, tais como Biologia, Medicina, Química Orgânica, realizam diferentes estudos das proteínas. Muitos destes estudos são voltados para o processo de dobramento das proteínas que pode sofrer alterações: tanto em como a conformação estará disposta no espaço, como ela estará agrupada e sobre sua má formação. Isto é muito relevante para estudos que visam à produção de medicamentos, suplementos alimentares, técnicas que manipulam o DNA, ou para formação de novos compostos proteicos sintéticos em laboratório [Devlin e Toma, 1998]. É importante mencionar que apesar do avanço na grande quantidade de proteínas que se tem conhecimento por conta de projetos de sequenciamento genômico, apenas uma pequena fração de estruturas tridimensionais é conhecida.

A cristalografia de raios-X e espectroscopia de RNM são os métodos experimentais mais poderosos para o estudo da estruturas de proteínas [Ilari e Savino, 2008] [Göbl e Tjandra, 2012]. Entretanto estes métodos são altamente custosos tanto em esforços computacionais, de tempo e financeiros, e estão disponíveis apenas para algumas instituições.

Embora o conceito de dobramento de proteínas tenha surgido da área de biologia molecular, este problema é um problema interdisciplinar, o qual requer apoio de muitas áreas

do conhecimento, e é considerado como um dos desafios atuais mais importantes da biologia e bioinformática [Nicosia e Stracquadanio, 2008].

Na biologia computacional existem dois problemas que tratam sobre o dobramento de proteínas. São eles: problema de predição estrutura de proteínas (ou PSP - *Protein Structure Prediction*), que trata de predizer a estrutura tridimensional (conformação) a partir de sua sequência (estrutura primária); e o problema de dobramento de proteínas (PDP ou PFP - *Protein Folding Problem*), o qual trata da determinação dos passos/eventos que conduzem o dobramento a partir da estrutura primária até a conformação nativa [Lopes, 2008]. Porém, na literatura, são encontrado ambos os termos sendo utilizados sem nenhuma distinção, normalmente se referindo apenas ao primeiro problema [Lopes, 2008].

A ciência da computação desempenha um papel importante nisto, propondo e desenvolvendo modelos e soluções computacionais para o estudo de ambos os problemas PSP e PDP [Lopes, 2008]. Muitas estratégias computacionais descrevem modelos de predição para estruturas proteicas com diferentes níveis de detalhamento e complexidade. Embora, existam modelos com um elevado grau de detalhamento. Simular a predição de estruturas das proteínas é computacionalmente muito custosa. É nesta lacuna que se abre espaço para estratégias que proponham modelos simplificados para representar estruturas de proteínas sem perda de viabilidade computacional [Benítez, 2010]. Consequentemente, é possível evitar a obrigatoriedade de métodos caros e assim aumentar a capacidade de centros de pesquisa, com recursos escassos, desenvolverem abordagens com modelos simplificados para representar as estruturas. Apesar de complexidade reduzida estes modelos apresentam uma maneira fidedigna de representar as estruturas de proteínas.

Segundo a lei da termodinâmica [Anfinsen, 1972], a estrutura de uma proteína se torna estável quando adquire o seu estado nativo, em qual a sua energia livre é mínima. Os modelos de predição de estruturas normalmente são baseados nesta lei. Dessa maneira, o problema é modelado com objetivo de minimizar energia livre das possíveis conformações que uma proteína pode assumir [Benítez, 2010]. Estes modelos assumem que o principal fator para formação das estruturas de proteínas segue a lei da termodinâmica. Portanto, a conformação nativa de uma proteína é dada por aquela que possui o menor valor de energia livre.

[Pedersen, 2000] sugere que um modelo computacional deve possuir algumas características:

- Um conjunto de entidades que representam os átomos e as ligações entre eles.
- Regras que definem as possíveis conformações.
- Uma função que seja computacionalmente factível para calcular a energia livre das possíveis conformações.

A próxima subseção irá discorrer sobre alguns modelos para representar as estruturas de proteínas.

2.3 Modelos de Representação de Proteínas

Em suma, existem duas classes de modelos de representação de estruturas de proteínas: analítico (também conhecido como *all atom*) e discreto (chamado também de *coarse-grained*). Os modelos analíticos possuem uma descrição detalhada da estrutura tridimensional incluindo informações de todos os átomos que constituem uma proteína. Já os modelos discretos descrevem as proteínas com um nível bastante reduzido de detalhes. Recentemente, os modelos discretos ganharam maior interesse, por conta de dois fatores [Benítez, 2010]:

- A simulação de modelos analíticos nem sempre é computacionalmente possível por conta do alto custo envolvido.
- Modelos discretos possibilitam simulações biologicamente relevantes com melhor aproveitamento computacional.

Esta proposta irá descrever apenas os modelos discretos pois visa a utilização do modelo hidrofóbico polar (HP) 2D.

2.3.1 Modelos Discretos

Os modelos computacionais mais simples são os conhecidos como modelos de grade (*lattice models*). Estes modelos consideram as estruturas de proteínas como um colar de esferas posicionado em uma grade. O grau de liberdade dos movimentos é restrito à estrutura da grade, que pode ser 2D (plano) ou 3D (espacial). Conformações válidas são aquelas que os aminoácidos adjacentes na sequência também são adjacentes na grade e cada aminoácido ocupe uma posição distinta na grade. Muitos modelos de grade têm sido propostos e aplicados ao PDP. Os modelos 2D-HP e 3D-HP são exemplos de modelos de grade.

Modelo Hidrofóbico-Polar HP

No modelo HP os aminoácidos são classificados em 2 tipos: Hidrofílicos (Polar) e Hidrofóbico. Consequentemente, uma proteína é representada por uma *string* de caracteres definida por um alfabeto binário $\{H, P\}$. Este modelo considera que as interações entre aminoácidos hidrofóbicos (H) representam a contribuição mais importante para a energia livre de uma proteína. Portanto existe uma relação inversamente proporcional: quanto maior for a quantidade de interações hidrofóbicas (H-H), menor será a energia livre de uma proteína. Uma interação hidrofóbica (também conhecida como contato hidrofóbico) é definida como um par de aminoácidos do tipo H-H que não sejam consecutivos na sequência mas sejam adjacentes na grade. Como dito anteriormente, uma conformação é dita válida quando nenhuma posição da grade é ocupada por mais que um aminoácido. Conformações inválidas possuem colisões entre os aminoácidos. Dada uma conformação válida para o modelo HP e n o número de interações hidrofóbicas, a energia da conformação pode ser facilmente calculada utilizando a equação 2.1:

$$E(c) = n.(-1) \quad (2.1)$$

Quando uma proteína é dobrada na sua conformação nativa, os aminoácidos hidrofóbicos tendem a se agrupar no interior da estrutura, protegidos por aminoácidos polares posicionados no exterior. Dessa maneira, um núcleo hidrofóbico é formado em proteínas dobradas [Benítez, 2010]. Embora simples, a estratégia computacional de buscar uma solução para o PDP utilizando modelo HP é considerada como um problema *NP*-completo [Atkins e Hart, 1999, Berger e Leighton, 1998, Crescenzi et al., 1998]. O espaço de busca do modelo HP possui algumas características mencionadas na literatura [Bastolla et al., 1997, Berger e Leighton, 1998, Crescenzi et al., 1998, Krasnogor et al., 1999, Vendruscolo et al., 2000] :

- Elevada degenerescência.
- Espaço de busca multimodal.
- Muitas regiões com conformações inválidas.

A figura ?? apresenta um exemplo para os modelos HP (2D e 3D). Os pontos pretos são aminoácidos do tipo H e os brancos são aminoácidos do tipo P. As linhas pontilhadas representam as interações hidrofóbicas.

Diversos trabalhos aplicam algoritmos evolucionários e bio-inspirados ao problema de dobramento de proteínas utilizando o modelo HP. Uma decisão comum a todos trabalhos que utilizam o modelo HP é a de como representar as variáveis de entrada. Na literatura é possível encontrar basicamente três representações [Krasnogor et al., 1999, Lopes, 2008]:

- Coordenadas cartesianas: Este método representa a posição de cada aminoácido utilizando suas coordenadas espaciais (x,y) no plano cartesiano 2D ou (x,y,z) no plano cartesiano 3D. Geralmente, sua utilização não é adequada para algoritmos baseados em população, pois estruturas idênticas ou semelhantes podem ter coordenadas totalmente diferentes [Benítez, 2010];
- Coordenadas internas: Nesta representação as conformações são representadas por conjuntos de movimentos que ditam como a estrutura final irá se parecer. Esta representação é a mais utilizada em abordagens com algoritmos evolucionários para o PDP [Benítez, 2010]. Existem duas possibilidades de se representar conformações utilizando coordenadas internas:
 - Coordenadas absolutas: Este tipo de coordenada é baseado na orientação do eixo da grade onde a conformação esta contida. No caso de uma grade bidimensional os possíveis movimentos são: {N, S, L, O} ou norte, sul, leste e oeste. Já em uma grade

3D os possíveis movimentos são: $\{N, S, L, O, F, T\}$ que correspondem aos mesmos movimentos no caso 2D porém com dois movimentos a mais: para frente e para trás.

- Coordenadas relativas: Este tipo de representação define a posição de cada aminoácido da cadeia em relação ao movimento do seu predecessor. O conjunto de movimentos possíveis para a grade 2D é definido por $\{F, E, D\}$, que correspondem aos movimentos: frente (continuar no mesmo sentido do aminoácido anterior), à esquerda e à direita. Em um cubo 3D, os possíveis movimentos são $\{F, E, D, C, B, \}$, possuindo dois movimentos a mais: para cima e para baixo.
- Matriz de distâncias: descreve a conformação de uma proteína através de uma matriz quadrada que representa a distância entre os aminoácidos. Este tipo de representação é raramente utilizado na literatura [Benítez, 2010].

Outros Modelos

Além do modelo HP, outros modelos simples em grade são utilizados para representar a estrutura de proteínas em outros estudos encontrados na literatura. Por exemplo:

- Modelo PH (*Perturbed Homopolymer*): Proposto por Shakhnovich et al. [Shakhnovich e Gutin, 1993], as reações entre aminoácidos hidrofóbicos não são levadas em consideração, mas as interações entre aminoácidos do mesmo tipo são favorecidas, ou seja, H-H e P-P, desfavorecendo ligações H-P [Benítez, 2010].
- Modelo LPE (*Lattice Polymer Embedding*): Modelo proposto por Unger e Moulton [Unger e Moulton, 1993a]. A modelagem é feita a partir de uma sequência de aminoácidos, $A = a_1, \dots, a_n$ atrelada a uma grade cúbica. Cada aminoácido possui um coeficiente de afinidade, definido para cada par $a_i, a_j (c(a_i, a_j))$. O objetivo da função de energia é minimizar o produto dos coeficientes pela distância entre os aminoácidos [Benítez, 2010].
- Modelo HP-TSSC (*Hydrophobic-Polar Tangent Spheres Side Chain Model*): este modelo proposto por Hart et al. [Hart e Istrail, 1997] é baseado no modelo HP, porém não utiliza uma grade. Neste modelo a proteína é modelada via um grafo tridimensional, onde a cadeia lateral e o *backbone* de cada aminoácido são esferas de mesmo raio [Benítez, 2010].
- Modelo CGE (*Charged Graph Embedding*): Modelo descrito por Ngo et al. [Ngo et al., 1994]. Neste modelo, uma carga (*charge*) é atribuída a cada resíduo. Entretanto, as conformações permitidas não são realistas [Benítez, 2010].
- Modelo HPNX: modelo proposto por Bornberg-Bauer [Bornberg-Bauer, 1997]. Divide os 20 aminoácidos em 3 classes: hidrofóbicos (H), positivos (P), negativos (N) e neutros (X). Este modelo, assim como o modelo HP, utiliza uma grade. Interações entre aminoácidos

hidrofóbicos (H-H) representam interações de atração e diminuem a energia da conformação 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. Da mesma maneira que o modelo HP, quanto mais interações hidrofóbicas melhor será o dobramento. Porém este modelo não desconsidera o valor das demais interações [Benítez, 2010].

- **Modelo HP-helicoidal (Helical-HP):** este modelo proposto por Thomas e Dill [Thomas e 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) [Benítez, 2010].
- **Modelo *off-lattice* AB:** este modelo proposto por Stillinger et al. [Stillinger et al., 1993] divide os aminoácidos em duas classes de acordo com sua polaridade: Hidrofóbicos (A) e Hidrofílicos (ou polares B). Inicialmente, este modelo foi aplicado em duas dimensões (2D AB *off-lattice*) e posteriormente aplicado para três dimensões (3D AB *off-lattice*). Os aminoácidos não consecutivos interagem através de um potencial modificado de Lennard-Jones. Os ângulos de torção entre ligações sucessivas também contribuem no cálculo da função de energia [Benítez, 2010].

2.3.2 Conclusão

Nesta capítulo foi apresentado o problema de dobramento de proteínas, bem como sua importância para biológica computacional, química orgânica e medicina. Também foi mencionado que existem diversos modelos para representar estruturas de proteínas. Cada modelo tem suas peculiaridades e considera interações diferentes. Não existe um modelo que represente de maneira real o dobramento de proteínas, pois se trata de um processo ainda não completamente compreendido pelos cientistas e pesquisadores. Os modelos propostos tem diferentes níveis de detalhe e complexidade. O modelo mais simples é o HP mas apesar da sua simplicidade se apresenta como um problema *NP*-completo. O modelo HP será utilizado nesta proposta por conta de sua simplicidade de implementação, assim como o baixo custo computacional para realizar simulações do cálculo de energia. Utilizando o modelo HP diferentes maneiras de representar as variáveis existem. Nesta dissertação será utilizada a representação relativa pois, é mencionado na literatura que esta tem uma maior capacidade de guiar algoritmos de busca a melhores resultados [Krasnogor et al., 1999].

Capítulo 3

Referencial Teórico

3.1 AEs - Algoritmos Evolucionários

Um algoritmo evolucionário (AE) é uma técnica de busca, altamente paralela, inspirada na teoria da seleção natural e reprodução genética de Charles Darwin. De acordo com a teoria de Darwin, a seleção natural irá favorecer os indivíduos que forem mais aptos, dessa maneira, estes indivíduos tem uma maior probabilidade de reprodução. Indivíduos com mais descendentes tem uma chance maior de perpetuarem seus códigos genéticos nas gerações futuras. O código genético é a identidade de cada indivíduo e é representado por cromossomos. Estes princípios são utilizados na implementação de algoritmos computacionais, que procuram por soluções melhores para um dado problema, evoluindo uma população de soluções codificadas em cromossomos artificiais – estruturas de dados utilizadas para representar soluções factíveis para um dado problema [Pacheco, 1999].

De maneira geral, problemas reais de otimização possuem múltiplos objetivos a serem minimizados/maximizados e estão presentes em muitas áreas do conhecimento. Para otimizar problemas multiobjetivos dois ou mais objetivos são considerados, os quais geralmente são conflitantes. Para estes problemas é impossível encontrar uma única solução ótima. Um conjunto de soluções é encontrado avaliando a dominância de Pareto [Baudrillart, 1872] entre as soluções. O objetivo principal é encontrar o conjunto de soluções que sejam não dominadas entre si. Uma solução domina outra, se e somente se, for melhor em pelos um dos objetivos, sem ser pior em qualquer outro. Este conjunto de soluções constitui a fronteira de Pareto. Encontrar a fronteira real de Pareto é um problema NP-Completo [Fonseca et al., 2005], dessa maneira, o objetivo é encontrar uma boa aproximação da fronteira real.

Algoritmos Evolucionários Multi-Objetivos (AEMOs) são extensões de AEs para problemas multi-objetivos, os quais aplicam conceitos da dominância de Pareto para criar diferentes estratégias para evoluir e manter a diversidade das soluções. Nesta dissertação foram explorados dois AEMOs: NSGAII [Deb et al., 2002b] and IBEA [Zitzler e Künzli, 2004].

3.1.1 NSGAII - Non-dominated sorting Genetic Algorithm II

O algoritmo 1 apresenta o pseudo código do NSGAII. O algoritmo recebe como parâmetro N o tamanho da população e T o número máximo de avaliações. O algoritmo inicia criando uma população com tamanho N chamada P_0 . A população P_0 é classificada de acordo com aptidão e a relação de não dominância. A população P_0 é submetida ao operador de seleção: torneio binário para selecionar duas soluções que serão utilizadas para gerar descendentes. Operadores de cruzamento e mutação são aplicados na soluções selecionadas gerando duas soluções distintas descendentes. Ao fim do processo de reprodução, as soluções descendentes são avaliadas e adicionadas a população chamada Q_0 .

Após esta etapa, P_0 e Q_0 são adicionadas em uma população auxiliar chamada R . Utilizando o conceito de não dominância, R é ordenada criando fronteiras, onde cada solução da primeira fronteira não é dominada por nenhuma outra solução, já soluções da segunda fronteira são dominadas apenas por soluções contidas na primeira fronteira, e assim por diante. Para cada fronteira, as soluções são avaliadas utilizando um mecanismo de *Crowding-Distance* (CD). Soluções com maiores valores de CD irão ser adicionadas na população da próxima geração chamada P_t , no qual t é a avaliação corrente.

Após criar e preencher P_t com as soluções não dominadas de todas as fronteiras, a população P_t é avaliada e então passa para um novo torneio binário e reprodução, dessa maneira, iniciando uma novo ciclo do algoritmo.

3.1.2 IBEA (Indicator-Based Evolutionary Algorithm)

No contexto de otimização multiobjetiva, otimizar consiste em tentar encontrar a fronteira com uma boa aproximação da fronteira real de Pareto. Entretanto, não existe uma definição geral para "uma boa aproximação". Consequentemente, indicadores de qualidade vem sendo utilizados para avaliar a qualidade da aproximação de fronteiras. O indicador *hypervolume* é um exemplo de indicador utilizado para avaliação e comparação das fronteiras.

No algoritmo IBEA, indicadores de qualidade são utilizados para avaliar o conjunto de soluções não dominadas [Figueiredo et al., 2013]. Para utilizar o IBEA, é necessário definir qual indicador será utilizado para associar cada solução a um valor scalar. Um dos indicadores bastante utilizados é o *hypervolume* por conta da sua capacidade de avaliar a convergência e a diversidade do espaço de busca ao mesmo [Ishibuchi et al., 2008].

$$F(x_i) = \sum_{x_j \in (P - x_i)} -e^{\frac{-I_{Hy}(x_j, x_i)}{k}} \quad (3.1)$$

A equação de *fitness* do IBEA é apresentada pela equação 3.1 e é utilizada para calcular a contribuição de uma dada solução para o valor do indicador referente a população, onde k é um fator escalar dependente do I_{Hy} , o qual representa o indicador de qualidade sendo utilizado.

Algoritmo 1 NSGAI

```

1:  $N \leftarrow \text{Population Size}$ 
2:  $T \leftarrow \text{Max evaluations}$ 
3:  $P_0 \leftarrow \text{CreatePopulation}(N)$ ;
4:  $\text{CalculateFitness}(P_0)$ ;
5:  $\text{FastNonDominatedSort}(P_0)$ ;
6:  $Q_0 \leftarrow 0$ 
7: while  $Q_0 < N$  do
8:    $\text{Parents} \leftarrow \text{BinaryTournament}(P_0)$ ;
9:    $\text{Offspring} \leftarrow \text{CrossoverMutation}(\text{Parents})$ ;
10:   $Q_0 \leftarrow \text{Offspring}$ 
11: end while
12:  $\text{CalculateFitness}(Q_0)$ ;
13:  $t \leftarrow 0$ 
14: while  $t < T$  do
15:   $R_t \leftarrow P_t \cup Q_t$ ;
16:   $\text{Fronts} \leftarrow \text{FastNonDominatedSort}(R_t)$ ;
17:   $P_{t+1} \leftarrow 0$ 
18:   $i \leftarrow 0$ 
19:  while  $P_{t+1} + \text{Front}_i < N$  do
20:     $\text{CrowdingDistanceAssignment}(\text{Front}_i)$ ;
21:     $P_{t+1} \leftarrow P_{t+1} \cup \text{Front}_i$ 
22:     $i \leftarrow i + 1$ 
23:  end while
24:   $\text{CrowdingDistanceSort}(\text{Front}_i)$ ;
25:   $P_{t+1} \leftarrow P_{t+1} \cup \text{Front}_i[1 : (N - P_{t+1})]$ 
26:   $\text{Parents} \leftarrow \text{BinaryTournament}(P_{t+1})$ ;
27:   $Q_{t+1} \leftarrow \text{CrossoverMutation}(\text{Parents})$ ;
28:   $t \leftarrow t + 1$ 
29: end while
30: return  $P \leftarrow \text{Set of non-dominated solutions.}$ 

```

O valor $F(x_i)$ corresponde à perda de qualidade da aproximação, da fronteira real de Pareto, se a solução x_i for removida da população [Figueiredo et al., 2013].

O Algoritmo 2 recebe como parametro o tamanho da população N , o número máximo de avaliações T e o fator escalar k . O processo se inicia criando uma população P de tamanho N . Os seguintes passos irão se repetir até que o critério de parada seja atingido: um torneio binário para selecionar indivíduos, reprodução (cruzamento e mutação) dos indivíduos selecionados para gerar descendentes, adicionar os descendentes na população auxiliar \bar{P} . Após a reprodução, a população \bar{P} é unida com P . Enquanto o tamanho de P exceder N , o pior indivíduo avaliado pela equação 3.1 é removido da população P e os indivíduos restantes são re-avaliados. Quando o algoritmo terminar irá retornar o conjunto de soluções não dominadas é retornado.

Algoritmo 2 IBEA

```

1:  $N \leftarrow$  Population Size
2:  $\bar{N} \leftarrow$  AuxiliaryPopulationSize
3:  $T \leftarrow$  Max Evaluations
4:  $k \leftarrow$  Scale factor of Fitness
5:  $P \leftarrow$  CreatePopulation( $N$ );
6:  $\bar{P} \leftarrow$  CreateEmptyAuxiliaryPopulation( $\bar{N}$ );
7:  $m \leftarrow 0$ 
8: CalculateFitness( $P$ );
9: while  $m \geq T$  or other stop criterion is not reached do
10:    $\bar{P} \leftarrow$  BinaryTournament( $P$ );
11:    $\bar{P} \leftarrow$  CrossoverMutation( $\bar{P}$ );
12:    $P \leftarrow P \cup \bar{P}$ 
13:    $m \leftarrow m + 1$ 
14:   while Size( $P$ ) >  $N$  do
15:      $x^* \leftarrow$  WorstIndividualByFitness();
16:     RemoveFromPopulation( $x^*$ ,  $P$ );
17:     CalculateFitness( $P$ );
18:   end while
19: end while
20: return  $P \leftarrow$  Set of non-dominated solutions

```

3.2 Hiper-Heurísticas

Apesar do sucesso de métodos heurísticos e outros métodos de busca na tarefa de resolver problemas de busca computacional difíceis, ainda existem dificuldades em generalizar estes métodos para diferentes problemas ou até mesmo para diferentes instâncias de um mesmo problema. Esta dificuldade provém principalmente da necessidade de selecionar os parâmetros e configurações mais adequados dos algoritmos para um problema ou para uma dada instância de um problema. Também vale mencionar a pouca orientação na tarefa de definir estes parâmetros. É neste contexto que surge uma questão: é possível automatizar o projeto e parametrização de métodos heurísticos para resolver problemas de busca computacional difíceis? [Burke et al., 2013]. A ideia principal é desenvolver algoritmos que sejam mais genéricos do que as implementações de metodologias atuais [Burke et al., 2013]. As principais abordagens já propostas para este desafio podem ser classificadas em duas categorias: configuração estática (*offline*) e controle dinâmico (*online*). Abaixo são apresentadas algumas abordagens já propostas na literatura:

- Configuração Estática (Offline):
 - Seleção de algoritmos;
 - Portfólio de algoritmos;
 - Configuração de algoritmos;
 - Ajuste de parâmetros;

- Hiper-Heurísticas.
- Controle Dinâmico (Online):
 - Seleção adaptativa de operadores (AOS);
 - Controle de parâmetros;
 - Algoritmos meméticos adaptativos;
 - Hiper-Heurísticas

Esta seção tratará apenas de hiper-heurísticas e suas particularidades. Uma hiper-heurística pode ser vista como uma metodologia de alto nível, a qual seleciona ou cria heurísticas para resolver um dado problema ou instância de um problema. [Burke et al., 2013]. O objetivo principal é tentar encontrar ou construir a heurística mais adequada para cada situação. As hiper-heurísticas diferem dos métodos padrão de busca, pois operam sobre o espaço de busca de heurísticas que por sua vez operam sobre o espaço de busca de um problema. Além disso, hiper-heurísticas são independentes do problema. Tradicionalmente frameworks hiper-heurísticos possuem dois níveis: [Sabar et al., 2015]

Heurísticas de baixo nível: Um conjunto de heurísticas de baixo nível específicas. Estas heurísticas costumam diferir entre domínios de problemas. São exemplos: operadores de cruzamento, mutação e buscas locais. Em alguns casos, meta-heurísticas, dependendo da modelagem do *framework* hiper-heurístico, também podem assumir o papel de heurísticas de baixo nível.

Heurísticas de alto nível: Geralmente consistem em dois componentes: mecanismo de seleção, que gerencia quais heurísticas de baixo nível devem ser aplicadas durante a busca; um critério de aceitação, que tem a responsabilidade de decidir se irá aceitar ou não uma solução gerada, a partir da aplicação de uma heurística de baixo nível. A responsabilidade do mecanismo de seleção é selecionar, de um *pool* de heurísticas de baixo nível, a heurística que for mais adequada naquele momento. Geralmente, a escolha da heurística de baixo nível é crucial para uma boa exploração do espaço de busca, evitando que a busca fique confinada em uma região específica [Sabar et al., 2015]. O objetivo do critério de aceitação é auxiliar o processo de busca a evitar mínimos locais assim como explorar diferentes regiões através do aceite ou rejeição de soluções geradas [Chakhlevitch e Cowling, 2008]. Espera-se que um bom critério de aceitação deva atingir um bom equilíbrio entre aceitar soluções melhores assim, como soluções diversificadas caso a busca estiver presa em um mínimo local [Sabar et al., 2015]. Ambos os componentes devem ser independentes de conhecimento sobre o problema.

A imagem 3.1 apresenta um diagrama exemplificando os níveis de um *framework* hiper-heurístico e suas características. Note que entre os níveis (alto e baixo) existe uma barreira de domínio, ou seja, apenas as heurísticas de baixo nível são dependentes de conhecimento do problema ou instância enquanto as heurísticas de alto nível não são dependentes do problema.

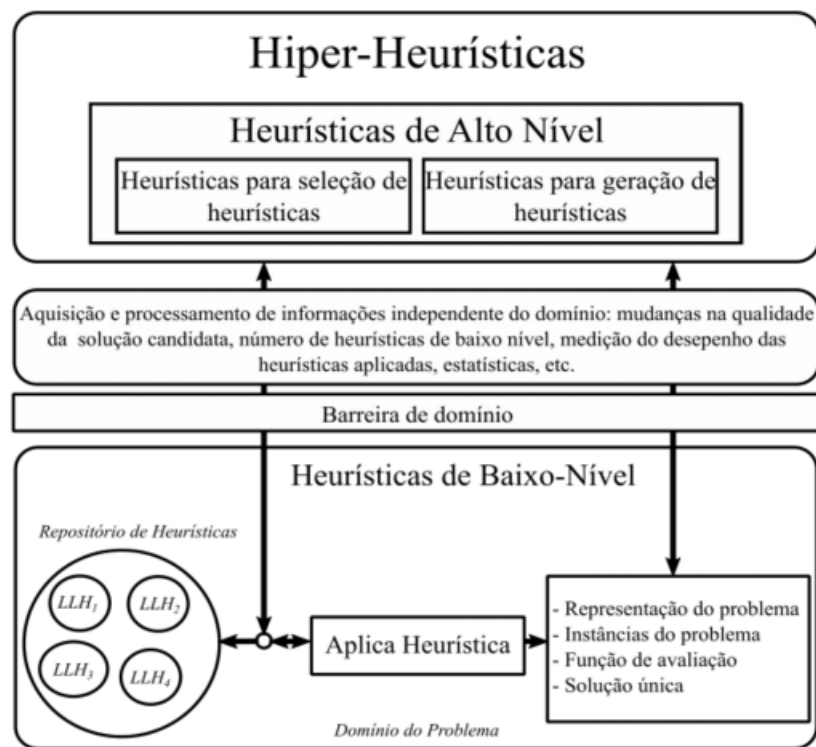


Figura 3.1: Framework Geral Hiper-Heurístico. Adaptado de [Sabar et al., 2015]

Como cada instância ou problema possui um espaço de busca com diferentes características, os componentes da heurística de alto nível têm um grande impacto no desempenho de um *framework* hiper-heurístico. Esta é uma das razões de existir um grande interesse de pesquisa em desenvolver novos mecanismos de seleção, assim como diferentes critérios de aceitação [Burke et al., 2013]. Um bom mecanismo de seleção deve selecionar a heurística mais adequada em um dado momento, para guiar a busca para regiões promissoras do espaço de busca. Ao utilizar hiper-heurísticas, espera-se encontrar o método correto ou a sequência de heurísticas que mais se adequam a um problema ou instância em vez de tentar resolver o problema diretamente. Entretanto, um importante objetivo é desenvolver métodos genéricos, que têm potencial em produzir soluções com uma qualidade aceitável, utilizando um conjunto de heurísticas de baixo nível com fácil implementação. As hiper-heurísticas podem ser classificadas de diversas maneiras. A figura 3.2 apresenta as possíveis classificações descritas na literatura.

A primeira classificação de hiper-heurísticas é baseada na sua fonte de conhecimento durante a busca: *Online* é quando a hiper-heurística toma decisões de maneira instantânea, baseando-se em métricas durante sua execução, não necessitando de treinamento prévio. *Offline* necessita de treinamento prévio; estes *frameworks* tomam suas decisões baseados no que foi aprendido apenas durante o treinamento, sem atualização deste conhecimento. Os *frameworks* classificados como *No-Learning* não possuem nenhuma forma de aprendizagem. Outra classificação considera a forma como as heurísticas de baixo nível operam sobre as soluções do problema. As heurísticas ditas perturbativas realizam pequenas perturbações nas soluções gerando novas soluções. Já heurísticas construtivas criam soluções do zero passo a passo e normalmente avaliam

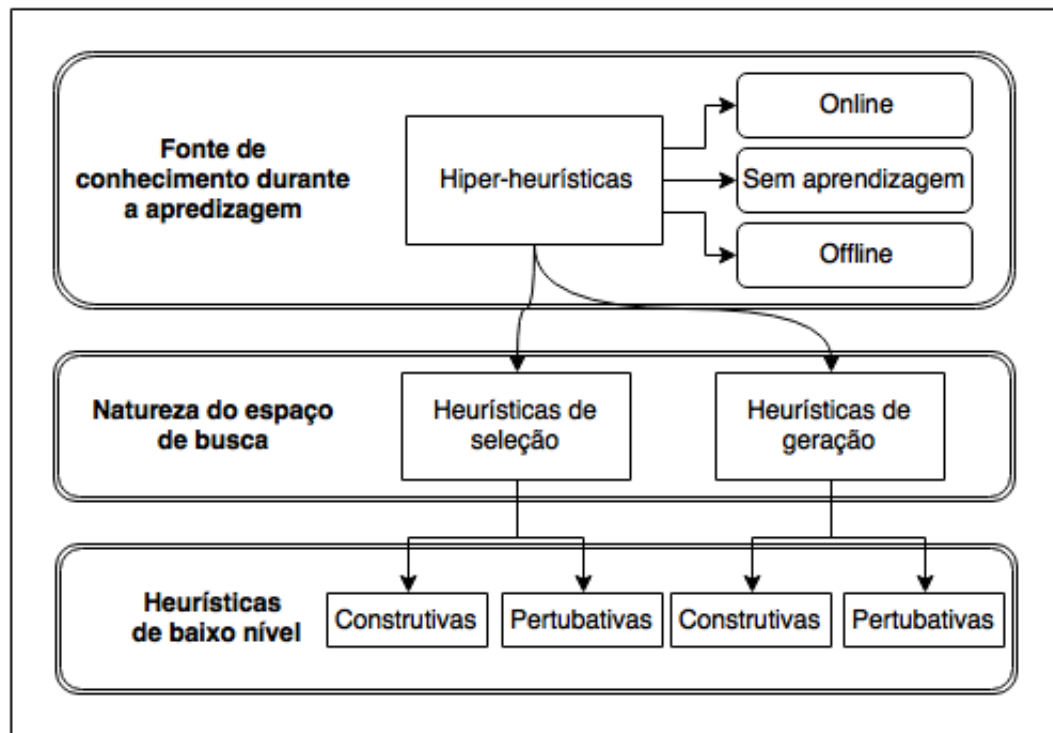


Figura 3.2: Classificação Hiper-heurísticas. Adaptado de [Sabar et al., 2015]

cada etapa da construção para obter *feedback* sobre o seu desempenho. Uma última classificação, mas não menos importante, divide as hiper-heurísticas de acordo com a natureza do seu espaço de busca. As hiper-heurísticas de seleção selecionam sequências de heurísticas a serem aplicadas para resolver um dado problema ou instância. Já as hiper-heurísticas de geração operam gerando novas heurísticas com objetivo de resolver um problema ou instância.

3.2.1 Hiper-heurísticas de Geração

Estas hiper-Heurísticas geram novas heurísticas combinando componentes de heurísticas existentes. Geralmente se utiliza programação genética (GP), ou alguma vertente de GP, como evolução gramatical [Ryan et al., 1998] ou programação gênica [Ferreira, 2006], como hiper-heurística para gerar heurísticas. A próxima sub-seção irá introduzir o conhecimento necessário para a compreensão da programação genética, assim como irá introduzir evolução gramatical, que se trata de um tipo de programação genética e que será utilizada nesta proposta.

3.3 Programação Genética (PG)

Programação Genética [Burke et al., 2009] é um ramo da síntese de programas que utiliza ideias oriundas da teoria da evolução natural para produzir programas. Os principais componentes da computação evolucionária são herança (cruzamento/reprodução), seleção e variação (mutação). A herança significa que os descendentes têm alguma semelhança com seus

pais, pois quase todo material genético vem dos pais. A seleção trata de escolher quais pais irão se reproduzir para gerar novos descendentes; pais com maior aptidão tendem a ter maior probabilidade de serem selecionados. Esta pressão de seleção define quais indivíduos estão mais aptos que outros. Variação realiza pequenas alterações em um descendente a fim de criar novo material genético neste indivíduo e que não estava presente em nenhum dos indivíduos que o geraram. Computação evolutiva pode ser pensada como a interação destes três componentes. Uma população aleatória de programas de computador é gerada, e os operadores geneticamente inspirados (cruzamento e mutação) são repetidamente aplicados com objetivo de produzir novos programas de computador. Estes programas são avaliados utilizando uma função de *fitness* (normalmente dependente do desempenho obtido pela aplicação do programa em um problema), que determina quais destes programas são mais suscetíveis a sobreviver para gerações futuras. Os programas com maior aptidão tem mais chances de serem selecionados para o cruzamento e perpetuarem parte de seus códigos genéticos durante o processo evolutivo. Programação genética é um método de geração de programas sintaticamente válidos e a função de *fitness* é utilizada para decidir quais programas são mais adequados para o problema. Na programação genética, os programas que compõem a população são tradicionalmente representados utilizando estruturas de árvore. Existem outras estruturas que podem ser evoluídas, por exemplo: sequências lineares de instruções ou gramáticas. Nesta proposta será utilizada uma representação gramatical linear que será explicada na seção 3.4.

3.4 Evolução Gramatical (EG)

Evolução gramatical é uma técnica relativamente nova de computação evolutiva, proposta por Ryan et al. [Ryan et al., 1998]. Trata-se de um tipo de programação genética. Assim como na programação genética, o principal objetivo é encontrar um programa executável ou trecho de um programa, que obtenha um bom valor de *fitness* para o problema em questão. Na maioria dos trabalhos publicados de programação genética, expressões que representam estruturas de árvore são manipuladas, enquanto na evolução gramatical os operadores genéticos são aplicados em vetores de inteiros que posteriormente são mapeados para um programa (ou trecho de programa) através de uma gramática específica. Um dos benefícios de EG é que este mapeamento generaliza a aplicação para diferentes linguagens de programação. Ryan et al. [Ryan et al., 1998] propõem uma técnica para gerar programas ou fragmentos de programas para qualquer linguagem de programação utilizando definições BNF. A técnica pode ser utilizada para evoluir programas por um processo evolutivo. A evolução gramatical adota um mecanismo de mapeamento entre o genótipo (indivíduos codificados em um vetor de inteiros) e o fenótipo (programas gerados para resolver algum problema). A notação *Backus Naur Form* (BNF) é a notação utilizada para expressar a gramática de uma linguagem na forma de regras de produção. Uma gramática BNF consiste em um conjunto de terminais, os quais são itens que podem aparecer na linguagem, por exemplo: +, -, *, / etc e não terminais, que podem ser expandidos

em um ou mais terminais e não terminais. Uma gramática pode ser expressada como uma tupla N, T, P, S , onde N é o conjunto de não terminais, T o conjunto de terminais, P um conjunto de regras de produção que mapeia os elementos N para T ; e, por último, S , um símbolo de início e que está contido em N .

$$\begin{aligned} N &= \langle expr \rangle, \langle op \rangle, \langle pre-op \rangle \\ T &= Sin, Cos, Tan, Log, +, -, /, *, X \\ S &= \langle expr \rangle \end{aligned}$$

E P pode ser representada como:

$$\langle expr \rangle ::= \langle expr \rangle \langle op \rangle \langle expr \rangle \quad (0)$$

$$| (\langle expr \rangle \langle op \rangle \langle expr \rangle) \quad (1)$$

$$| \langle pre-op \rangle (\langle expr \rangle) \quad (2)$$

$$| \langle var \rangle \quad (3)$$

$$\langle op \rangle ::= + \quad (0)$$

$$| - \quad (1)$$

$$| / \quad (2)$$

$$| * \quad (3)$$

$$\langle pre-op \rangle ::= Sin \quad (0)$$

$$| Cos \quad (1)$$

$$| Tan \quad (2)$$

$$\langle var \rangle ::= X \quad (0)$$

Gramática 3.1: Gramática exemplo para demonstrar como decodificar vetores de inteiros em programas de computador.

Tabela 3.1: Regras de produção e o número de escolhas para cada uma.

Regra de produção	Número de escolhas
$\langle expr \rangle$	4
$\langle op \rangle$	4
$\langle pre-op \rangle$	3
$\langle var \rangle$	1

Ryan et al. [Ryan et al., 1998] propôs o uso de um algoritmo genético (AG) para controlar quais escolhas devem ser feitas, permitindo dessa maneira que o AG controle quais regras de produção serão utilizadas. Um indivíduo (cromossomo) consiste em um vetor de tamanho variável de valores inteiros que representa o genótipo. Para fins de compreensão o processo de mapeamento de um cromossomo será demonstrado utilizando a Gramática 3.1

apresentada anteriormente nesta seção. O Algoritmo 3 apresenta o *template* geral dos programas gerados pela Gramática 3.1. A expressão $\langle expr \rangle$ apresentada na linha 2 do Algoritmo 3 é substituída por expressões matemáticas que estão codificadas pelos cromossomos (vetores de inteiros).

Algoritmo 3 *Template* geral dos algoritmos gerados

float symb(float x)

a = $\langle expr \rangle$;

return a;

Suponha o seguinte vetor de inteiros:

[220, 203, 17, 6, 108, 215, 104, 30]

Este vetor será utilizado para mapear o cromossomo (genótipo) em um trecho de programa (fenótipo) utilizando a gramática BNF. A tabela Table 3.1 resume o número de escolhas associada à cada regra de produção da Gramática 3.1. Existem 4 opções de regras de produção que podem ser selecionadas para a expressão $\langle expr \rangle$. Para decidir qual será selecionada, o primeiro valor no cromossomo deve ser utilizado. Sendo que o valor é 220. Devemos realizar o módulo deste valor pelo número de escolhas, neste caso 4. Portanto, $220 \text{ MOD } 4 = 0$, o que significa selecionar a primeira opção: $\langle expr \rangle \langle op \rangle \langle expr \rangle$.

Note que a primeira expressão é novamente $\langle expr \rangle$ e da mesma maneira devemos obter o próximo valor de inteiro e realizar o módulo. O próximo valor inteiro é 203; realizando o módulo de 4, resulta em 3, que portanto seleciona a quarta opção: $\langle var \rangle$. Substituindo na expressão anterior, obtemos: $\langle var \rangle \langle op \rangle \langle expr \rangle$

Nenhuma escolha é necessária para a expressão $\langle var \rangle$, pois existe apenas uma opção X . A expressão pode ser reescrita da seguinte maneira: $X \langle op \rangle \langle expr \rangle$

Neste momento é necessário decodificar a expressão não terminal $\langle op \rangle$. Obtendo o próximo valor inteiro do cromossomo, temos 17 e para o $\langle op \rangle$ temos 4 opções (+ | - | / | *). O resultado de $17 \text{ MOD } 4$ é igual a 1, que significa selecionar: -. Substituindo na expressão, temos: $X - \langle expr \rangle$

Novamente é necessário fazer uma nova escolha para resolver a expressão não terminal $\langle expr \rangle$. O próximo valor do cromossomo é 6 e novamente existem 4 opções. Realizando o módulo $6 \text{ MOD } 4$, obtém-se 2, que seleciona $\langle pre - op \rangle (\langle expr \rangle)$. Atualizando a expressão, obtemos: $X - \langle pre - op \rangle (\langle expr \rangle)$

Resolvendo a expressão $\langle pre - op \rangle$, obtemos $108 \text{ MOD } 4 = 0$ que por sua vez seleciona a primeira expressão terminal *Sin*. Atualizando a expressão, obtemos: $X - \text{Sin}(\langle expr \rangle)$

Expandindo $\langle expr \rangle$, obtemos $215 \text{ MOD } 4 = 3$, que seleciona a expressão não terminal $\langle var \rangle$. Já que para a expressão $\langle var \rangle$ existe apenas uma opção, nenhuma escolha é necessária e a expressão final decodificada (fenótipo) é: $X - \text{Sin}(X)$

Note que nem todos os genes do cromossomo foram necessários para obter o fenótipo. Nos casos em que isto ocorre, os genes que não forem utilizados são desconsiderados. Além disso, pode ocorrer que um cromossomo não tenha genes suficientes para mapear um programa. Neste caso a estratégia é reutilizar os genes do cromossomo a partir do primeiro gene.

Operadores genéticos tradicionais (cruzamento e mutação) também são utilizados na EG. Além dos operadores tradicionais outros dois operadores *Prune* e *Duplicate* são peculiares à EG e serão descritos em seguida:

- *Duplicate*: Este operador (dada uma probabilidade) realiza a cópia de alguns genes. Os genes duplicados são adicionados após a última posição do cromossomo. O número de genes a serem duplicados é selecionado de maneira aleatória. A motivação por trás deste operador é que ao duplicar genes ocorre um aumento da presença de genes que são potencialmente bons, pois pertencem a um indivíduo com boa aptidão selecionado pelo operador de seleção.
- *Prune* : Este operador leva em consideração que nem sempre todos os genes, de um cromossomo, são utilizados para decodificar um programa. Dessa maneira (dada uma probabilidade) realiza o truncamento de cromossomos. O objetivo é diminuir a probabilidade que o operador de cruzamento opere em regiões dos cromossomos que não sejam utilizadas realmente.

O Algoritmo 4 apresenta o pseudocódigo da evolução gramatical (EG). Note que o pseudocódigo é muito similar a um algoritmo genético simples. Nas linhas 3 e 4 ocorre a inicialização da população e o mapeamento para programas utilizando a gramática que foi provida como entrada. Em seguida, na linha 5 ocorre a execução dos programas e na linha 6 acontece a avaliação dos indivíduos da população, baseando-se na saída obtida pelos respectivos programas. Dentro do laço principal, apresentado na linha 7, podemos observar o processo de seleção dos indivíduos pais na linha 8 e na linha 9 o processo de cruzamento destes indivíduos. Nas linhas 10 e 11 ocorre a aplicação dos operadores *Prune* e *Duplicate* respectivamente e na linha 12 podemos observar a aplicação do operador de mutação. Em seguida, nas linhas 13,14 e 15 ocorre o mapeamento dos indivíduos descendentes para programas, execução dos programas e finalmente a atribuição de *fitness* para os descendentes. Por fim, na linha 16 do laço principal, ocorre a substituição dos descendentes na população.

3.5 Programação Genética como Hiper-Heurística de Geração de Heurísticas

Nesta seção serão apresentadas questões relativas ao uso de EG como mecanismo de geração de heurísticas [Burke et al., 2009] descrevem que muitos autores mencionam a melhor adequação de programação genética, em relação a outras técnicas de aprendizagem de máquina,

Algoritmo 4 Pseudocódigo da evolução gramatical

```

1: AG ← Arquivo da gramática;
2: população ← Inicialização a população;
3: programas ← Mapeia população para programas utilizando AG;
4: Executa os programas;
5: Atribui valor de fitness para as soluções of população de acordo com a saída obtida pelos
   respectivos programas decodificados;
6: while Condição de parada não atingida do
7:   pais ← Seleção de indivíduos para cruzamento;
8:   descendentes ← Cruzamento pais;
9:   Aplica o operador Prune nas soluções descendentes;
10:  Aplica o operador Duplicate nas soluções descendentes;
11:  Aplica o operador de mutação nas soluções descendentes;
12:  programas ← Mapeia descendentes para programas utilizando AG;
13:  Executa programas;
14:  Atribui valor fitness para as soluções descendentes de acordo com a saída obtida pelos
   respectivos programas decodificados;
15:  população ← Realiza substituição;
16: end while
17: return Melhor programa da população;

```

para gerar heurísticas de maneira automática. Burke et al [Burke et al., 2009] também apontam algumas vantagens desta técnica:

- PG utiliza cromossomos de tamanho variável. Geralmente, não se sabe um tamanho ótimo para representar heurísticas de um dado domínio de problema.
- PG produz estruturas de dados executáveis. E heurísticas são tipicamente expressadas como programas ou algoritmos.
- Facilidade em identificar boas características do domínio do problema, afim de definir o conjunto terminal que será utilizado pela PG.
- Heurísticas desenvolvidas por humanos podem facilmente ser expressadas na mesma linguagem utilizada para criar o espaço de busca da PG. O conjunto de funções, relevante para o problema pode ser determinado facilmente. E adicionalmente PG pode ser suplementada com uma gramática específica.

Todas estas vantagens descritas por Burke et al. [Burke et al., 2009] também são consideradas ao utilizar EG, visto que se trata de uma extensão de programação genética e possui as mesmas características (cromossomo de tamanho variável, produz estruturas executáveis, etc). Burke et al. [Burke et al., 2009] também mencionam desvantagens, por exemplo: a cada execução da programação genética é encontrada uma melhor heurística que, por se tratar de uma técnica estocástica, os resultados podem ser distintos em diferentes execuções. Portanto, se fazem necessárias múltiplas execuções a fim de se obter um melhor conhecimento da qualidade

das heurísticas que podem ser produzidas. Outra desvantagem é referente à configuração de parâmetros, que normalmente é encontrada via tentativa e erro.

Abordagem Básica

Burke et al. [Burke et al., 2009] descrevem uma abordagem básica para aplicar programação genética para gerar heurísticas:

1. Examinar as heurísticas existentes: Avaliar se as heurísticas já propostas para um dado problema podem ser descritas em um *framework* comum. Estas heurísticas podem ter sido criadas por humanos ou até mesmo concebidas via outras técnicas de aprendizagem. Este passo não é trivial, pois envolve o entendimento de um número diverso de heurísticas existentes, que podem operar de diferentes maneiras. Geralmente heurísticas desenvolvidas por humanos são produtos de anos de pesquisa, e portanto, uma boa compreensão das heurísticas existentes pode ser um trabalho difícil.
2. Um framework que utilizará as heurísticas: neste momento a preocupação é em como as heurísticas serão aplicadas para um dado problema. Em geral, os frameworks tendem a ser bem diferentes dependendo do domínio do problema.
3. Definição do conjunto terminal: neste passo a preocupação refere-se a variáveis que expressem o estado do problema. Estas variáveis irão compor os terminais da programação genética/evolução gramatical. Outros terminais também podem ser utilizados. Particularmente, constantes aleatórias podem ser úteis.
4. Definição do conjunto de funções: é necessário definir como as variáveis estarão relacionadas ou combinadas entre si. Estes relacionamentos irão compor o conjunto de funções da programação genética/evolução gramatical.
5. Identificar uma função de *fitness*: uma função de *fitness* precisa ser identificada para o problema. Geralmente, uma função simples de aptidão não irá avaliar bem os cromossomos. Introduzir alguns parâmetros pode ajudar a encontrar uma mais adequada.
6. Executar o framework: geralmente ao executar pela primeira vez um framework hiper-heurístico com programação genética, não serão produzidos bons resultados, devido à escolha dos parâmetros. Isto é observado especialmente em casos que o pesquisador é iniciante. Portanto, é essencial que as definições de parâmetros sejam cuidadosamente investigadas.

3.6 Conclusão

Neste capítulo foram apresentados os conceitos que permeiam a área de estudo sobre hiper-heurísticas, tendo sido discutidos os seus níveis (alto e baixo) e as classificações encontradas

na literatura. Foram discutidas algumas estratégias para hiper-heurísticas de seleção e geração. As hiper-heurísticas de geração foram mais detalhadas, pois esta proposta visa o projeto automático de heurísticas de alto nível. Foram apresentados os conceitos de PG e sua extensão EG, por se tratarem de estratégias comumente utilizadas para o projeto de hiper-heurísticas de geração de heurísticas. Também foram discutidas algumas vantagens e desvantagens referentes ao uso de PG para geração de heurísticas, além de demonstrar que a EG possui as mesmas características da PG, pois se trata de uma extensão que utiliza uma gramática para gerar os programas. O funcionamento geral da EG foi demonstrado utilizando uma gramática exemplo e um vetor de inteiros e, por fim, o pseudocódigo da evolução gramatical foi apresentado. A Capítulo 2.1 apresenta o Problema de Dobramento de Proteínas e o Capítulo 5 apresentará a proposta da aplicação de EG a este problema.

Capítulo 4

Trabalhos Relacionados

Este capítulo irá apresentar alguns trabalhos relacionados com a presente proposta. Serão apresentados trabalhos que buscam construir/adaptar estratégias heurísticas para encontrar melhores soluções ao PDP utilizando o modelo HP. Também serão apresentados alguns trabalhos que utilizam técnicas de programação genética para gerar heurísticas para diferentes problemas.

O estudo, desenvolvido por [Unger e Moulton, 1993b], foi pioneiro, ao aplicar um algoritmo genético ao PDP com o modelo HP, utilizando operadores de cruzamento e mutação aprimorados. Os resultados apresentados superaram um número significativo de estratégias tradicionais anteriores que utilizam métodos Monte Carlo para explorar as conformações.

Um algoritmo genético multi memético foi proposto por [Krasnogor et al., 2002]. Esta estratégia combina um algoritmo genético e buscas locais selecionando a busca local que mais se adequar com a instância (sequência) sendo otimizada. Mais tarde este trabalho foi aprimorado com uma estratégia fuzzy para as buscas locais, dessa maneira produzindo melhores resultados para o PDP.

Em [Hsu et al., 2003], os autores utilizam um algoritmo de crescimento de cadeia, chamado *pruned-enriched Rosenbluth method* (PERM). Esta estratégia se baseia em iterativamente construir uma conformação adicionando os aminoácidos um a um.

A otimização de colônia de formigas também foi aplicada para o PDP nos trabalhos [Shmygelska et al., 2002, Shmygelska e Hoos, 2003]. Estas abordagens utilizam formigas artificiais com objetivo de construir as conformações para o modelo HP. Uma busca local também foi introduzida com objetivo de melhorar e manter a qualidade das soluções.

No trabalho de [Santana et al., 2008] é proposto a aplicação de diferentes algoritmos de estimação de distribuição (EDA) para o PDP. Os EDAs são capazes de aprender a explorar as regularidades do espaço de busca utilizando modelos de dependência probabilísticos. Os autores compararam os resultados com as abordagens descritas anteriormente neste capítulo e constataram que a sua abordagem conseguiu atingir os valores ótimos para várias sequências de aminoácidos.

O estudo desenvolvido por [Lin e Su, 2011] utiliza um algoritmo genético híbrido combinando um operador de mutação baseado na otimização por exame de partículas. Os

resultados apresentados por Lin et al. se mostraram superiores aos apresentados por outros estudos, da época, que utilizam algoritmos evolutivos. Este trabalho também utiliza operadores de buscas locais que serão utilizados como heurísticas de baixo nível na presente proposta.

Custódio et al. [Custódio et al., 2014] desenvolveram um metodologia que consistiu modificar um algoritmo genético para selecionar os operadores de cruzamento e mutação de maneira dinâmica. Além disso, utilizaram um mecanismo baseado em *crowding* para manter a diversidade durante o processo de busca. Este trabalho apresentou bons resultados em relação a outros estudos que exploram algoritmos evolutivos. Os operadores genéticos utilizados deste trabalho também serão implementados como heurísticas de baixo nível nesta proposta.

Lourenço et al. [Lourenço et al., 2012] desenvolveram uma estratégia hiper-heurística utilizando evolução gramatical para geração e *tuning* automático de algoritmos evolutivos. Neste trabalho uma gramática foi desenvolvida e contém os principais componentes de algoritmos evolutivos. Os resultados apresentados por Lourenço et al. provaram a habilidade da abordagem para evoluir algoritmos evolutivos. Os resultados obtidos pelos algoritmos evolutivos gerados pela evolução gramatical são competitivos com outras abordagens padrão.

O trabalho desenvolvido [Sabar et al., 2015] propõe uma estratégia, utilizando Gene Expression Programming (GEP), de geração de heurísticas de alto nível para um *framework* hiper-heurístico aplicado a diversos problemas de *benchmark* contidos no framework HyFlex [Ochoa et al., 2012]. Este trabalho se difere dos apresentados anteriormente pois foi aplicado a domínios de problemas diferentes do PDP. Este trabalho motivou a presente proposta pois os resultados apresentados se mostraram promissores. A aplicação de uma vertente de programação genética para geração de heurísticas tem uma maior capacidade de explorar espaços de busca complexos (com muitos mínimos locais) e com muitas restrições.

A presente proposta visa aplicar evolução gramatical (EG), pois apesar de possuir as mesmas características da GEP ela torna mais amigável a manipulação dos indivíduos. Isto ocorre pelo fato de representá-los utilizando vetores de inteiros enquanto a GEP utiliza vetores de strings para representação. Dessa maneira, a EG facilita a manipulação dos indivíduos por operadores genéticos (cruzamento de mutação). A principal diferença entre esta proposta e os outros trabalhos relacionados [Santana et al., 2008, Shmygelska et al., 2002, Shmygelska e Hoos, 2003, Hsu et al., 2003, Krasnogor et al., 2002, Krasnogor et al., 2002, Unger e Moulton, 1993b] é que este irá trabalhar em um nível acima: gerando heurísticas de alto nível para um *framework* hiper-heurístico que será aplicado ao PDP enquanto os outros trabalham aplicando meta-heurísticas diretamente ao PDP.

4.1 Conclusão

Neste capítulo foram discutidos alguns estudos que utilizam algoritmos de busca para explorar o espaço de busca do PDP utilizando o modelo HP. Também foram discutidos trabalhos que aplicam PG como hiper-heurística de geração de heurísticas. Foram mencionadas diferentes

estratégias de busca para o PDP e algumas destas estratégias servem de base para alguns componentes que esta proposta possui. Os operadores genéticos utilizados [Custódio et al., 2014] e [Lin e Su, 2011] serviram de matéria prima para as heurísticas de baixo nível desta proposta. O trabalho desenvolvido por [Sabar et al., 2015] será utilizado como base na implementação da presente proposta, pois obteve bons resultados dessa maneira, demonstrando a habilidade do *framework* proposto generalizar bem entre diferentes domínios de problemas.

Capítulo 5

Metodologia

Neste capítulo serão apresentadas as duas estratégias propostas e desenvolvidas nesta dissertação para o PDP simplificado. A primeira trata de uma abordagem multi-objetiva utilizando dois AEMOs. Já a segunda visa aplicação da evolução gramatical para gerar heurísticas de alto nível para um *framework* hiper-heurístico intitulada EGHyPDP.

O problema PDP simplificado foi modelado utilizando a representação relativa, descrita na subseção 2.3.1, afim de codificar as possíveis estruturas de proteínas em vetores de inteiros. Segundo o estudo realizado por [Krasnogor et al., 1999] esta representação possui um maior potencial em conduzir os algoritmos a resultados melhores. Cada gene do cromossomo especifica a direção que o amino ácido atual deve ser posicionado. Cada amino ácido é posicionado na direção codificada pelo respectivo gene em relação ao amino ácido anterior. O genes podem assumir apenas 3 valores:

- 0 indica que o próximo amino ácido deve ser posicionado à direita do amino ácido anterior
- 1 indica que o próximo amino ácido deve ser posicionado à frente do amino ácido anterior
- 2 indica que o próximo aminoácido deve ser posicionado à esquerda do amino ácido anterior.

A Figura 5.1 apresenta um exemplo de um cromossomo hipotético e a conformação gerada no *grid* para o modelo HP-2D.

Para ambas as abordagens o mesmo conjunto de heurísticas de baixo nível (operadores de cruzamento/mutação e busca locais) foi selecionado a partir dos estudos anteriores [Custódio et al., 2014, Custódio et al., 2004, Garza-Fabre et al., 2012, Benítez, 2010]. O conjunto de heurísticas de baixo nível será descrito abaixo:

- *Single Point Crossover* (1X): Este operador seleciona, de maneira aleatória, 1 ponto de cruzamento dividindo os indivíduos em 2 partes. Os genes entre as posições selecionadas são trocados entre os pais de modo a gerar dois novos filhos [Benítez, 2010].

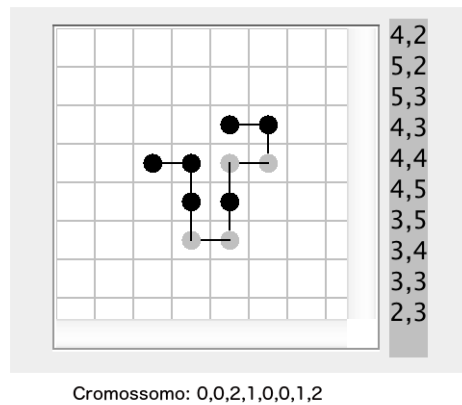


Figura 5.1: Cromossomo decodificado que representa uma possível conformação para a cadeia HHPPHPPHHH

- *Two Points Crossover (2X)*: Este operador seleciona, de maneira aleatória, 2 pontos de cruzamento dividindo os indivíduos em 3 partes. Os genes entre as posições selecionadas são trocados entre os pais de modo a gerar dois novos filhos [Benítez, 2010], conforme apresentado na figura 5.2.

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
4	3	3	0	2	2	1	3	2	0	3	4	0	1	3	4	4	2	3
				p1						p2								
0	0	4	3	2	2	1	3	2	0	3	4	1	2	2	4	0	3	3
4	3	3	0	1	1	1	3	3	3	2	2	0	1	3	4	4	2	3

Figura 5.2: Exemplo de aplicação do operador 2x.

Fonte Autoria Própria

- *Multi Points Crossover (MPX)*: Semelhante ao 2X porém com c pontos, baseado na função $c = \text{int}(n * 0.1)$, onde n é o tamanho da sequência. O operador MPX é utilizado para promover diversidade estrutural realizando uma mescla randômica entre os pais. Embora, não tão radical quanto o *Uniform Crossover* [Sabar et al., 2015]. Um exemplo de aplicação do operador MPX é apresentado na imagem 5.3

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
4	3	3	0	2	2	1	3	2	0	3	4	0	1	3	4	4	2	3
				p1				p2				p3		p4				
0	0	4	3	2	2	1	3	2	3	2	2	0	1	3	4	0	3	3
4	3	3	0	1	1	1	3	3	0	3	4	1	2	2	4	4	2	3

Figura 5.3: Exemplo de aplicação do operador MPX.

Fonte Autoria Própria

- *Segment Mutation (SMUT)*: Altera um número aleatório (5 a 7) de genes consecutivos para direções distintas. Esta heurística introduz grandes mudanças na conformação, e tem uma

grande probabilidade de criar colisões. Um mecanismo de reparação simples é aplicado no descendente gerado. A imagem 5.4 apresenta um exemplo da aplicação do SMUT.

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
				p1					p2									
0	0	4	3	3	4	2	2	1	4	2	2	1	2	2	4	0	3	3

Figura 5.4: Exemplo de aplicação do operador SMUT.
Fonte Autoria Própria

- *Exhaustive Search Mutation* (EMUT): Esta heurística seleciona um gene aleatório e testa todas as outras direções possíveis. Manterá a alteração que conseguir aumentar a qualidade da estrutura. O *tradeoff* deste operador é demandar 4 avaliações de *fitness*, há mais que as demais. Esta heurística tem grande potencial de melhorar o *fitness* de uma estrutura.
- *Local Move Operator* (LM): Esta heurística troca direções entre dois genes aleatórios consecutivos. Existem algumas condições para que esta heurística possa ser executada, por exemplo, as novas direções não podem criar movimentos redundantes. A figura 5.5 apresenta um exemplo da aplicação do operador LM.

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
									p1	p2								
0	0	4	3	1	1	1	3	3	2	3	2	1	2	2	4	0	3	3

Figura 5.5: Exemplo de aplicação do operador LM.
Fonte Autoria Própria

- *Loop Move Operator* (LPM): Da mesma maneira que a heurística LM, esta heurística troca direções entre dois genes que estão a 5 genes de distância na sequência. A figura 5.6 apresenta um exemplo da aplicação do operador LPM.

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
							p1			p2								
0	0	4	3	1	1	2	3	3	3	1	2	1	2	2	4	0	3	3

Figura 5.6: Exemplo de aplicação do operador LPM.
Fonte Autoria Própria

- *Opposite Mutation* (OM): Esta heurística troca as direções, para direção oposta, de uma sequência de genes entre dois genes (i, j) selecionados de maneira aleatória. A direção 0 (F) não possui oposta, portanto é mantida. Para exemplificar, suponha esta solução hipotética para uma sequência de 5 aminoácidos: $\{0, 1, 2, 1, 2\}$. Ela se tornaria $\{0, 2, 1, 2, 1\}$. A figura 5.7 apresenta um exemplo da aplicação do operador OM.

Este capítulo está dividido em duas seções para melhor apresentar ambas as estratégias propostas nesta dissertação. A seção 3.1 apresenta a estratégia multi objetiva onde foi utilizada

0	0	4	3	1	1	1	3	3	3	2	2	1	2	2	4	0	3	3
	i				j													
0	0	3	4	2	2	1	3	3	3	2	2	1	2	2	4	0	3	3

Figura 5.7: Exemplo de aplicação do operador OM.

Fonte Autoria Própria

dois algoritmos evolucionários, do estado da arte de otimização multi objetiva. Já a seção 5.2 irá apresentar o design automático de heurísticas de alto nível para um *framework* hiper heurístico para resolver o PDP.

5.1 AEMOs aplicados ao PDP

Esta abordagem utiliza uma modelagem multi-objetiva para PDP baseado no estudo desenvolvido por [Gabriel et al., 2012]. Onde o primeiro objetivo trata de maximizar a quantidade de contatos topológicos das estruturas de proteínas e o segundo de minimizar a máxima distância euclidiana entre os aminoácidos.

Duas abordagens multi-objetivas foram desenvolvidas neste capítulo, utilizando os AEMOs (NSGAI e IBEA) descritos no capítulo 3. A primeira abordagem consistiu em aplicar os algoritmo IBEA and NSGAI utilizando suas versões padrão. Nesta abordagem os operadores genéticos (cruzamento e mutação) são fixos com: *Single Point Crossover (1x)* and *Bit Flip Mutation (BM)*. Esta foi a combinação que obteve os melhores resultados em experimentos preliminares. No caso da segunda abordagem, o IBEA e NSGAI foram modificados com objetivo de aprimorar os resultados em relação às versões padrões. Duas modificações foram propostas e serão descritas abaixo:

- *Pool* de operadores: O uso de operadores fixos geralmente não conseguem guiar a busca para regiões promissoras. Com objetivo de aprimorar os AEMOs, um *pool* de operadores foi desenvolvido. Os operadores que compõem *pool* foram selecionados de estudos anteriores e foram apresentados no início deste capítulo. A cada operação de cruzamento e mutação os operadores são selecionados de maneira aleatória a partir do *pool*. Os operadores são sempre executados independente de probabilidades conforme a versão padrão dos algoritmos.
- Inicialização via *backtracking*: Tradicionalmente, a população inicial é gerada de maneira aleatória no caso dos algoritmos NSGAI e IBEA. Este tipo de inicialização tem grande potencial de gerar muitas soluções inválidas ao modelo HP-2D. Soluções que não sejam *self-avoiding walk* (SAW) são consideradas inválidas pois dois ou mais aminoácidos estariam ocupando a mesma posição no espaço. Se a população for integralmente gerada de maneira aleatória os algoritmos de otimização perdem um tempo considerável avaliando soluções inválidas. Para evitar este problema uma estratégia de *backtracking* pode ser

utilizada. A estratégia de inicialização com *backtracking* irá começar posicionando o primeiro aminoácido na posição 0,0. Para posicionar o próximo aminoácido, um movimento é selecionado de maneira aleatória. Caso o movimento cause uma colisão, este movimento será marcado como uma má escolha e um novo movimento é selecionado aleatoriamente (do conjunto que restou sem os movimentos marcados como más escolhas). Caso todos os movimentos estejam marcados como má escolha, a estratégia de *backtracking* irá retornar de maneira recursiva para o aminoácido anterior e marcar a escolha em questão como uma má escolha. A estratégia de *backtracking* termina quando gerar uma conformação que não possua colisões. Entretanto, a inicialização via *backtracking* é computacionalmente custosa. Dessa maneira, apenas 20% da população inicial foi inicializada utilizando esta estratégia.

Portanto 4 algoritmos foram implementados (IBEA, NSGAI, M_IBEA e M_NSGAI) foram propostos para avaliar a abordagem multi objetiva para o PDP simplificado.

5.1.1 Funções Objetivo

- **Valor de Energia:** Este é o objetivo principal e sua responsabilidade é avaliar o valor de energia associado com as possíveis conformações codificadas pelos cromossomos. O objetivo é minimizar o valor de energia, o qual, é calculado conforme descrito no capítulo 2. Este objetivo guia a busca na direção onde os valores energia associados com as estruturas de proteínas sejam mínimos. Dessa maneira, obtendo conformações mais próximas ao estado nativo das estruturas de proteínas.
- **Distância euclidiana entre os resíduos mais distantes:** Este é um objetivo secundário inspirado pelo estudo desenvolvido por [Gabriel et al., 2012]. A motivação por de trás deste objetivo é que estruturas mais compactas tendem a possuir mais contatos hidrofóbicos, oque resultaria em um valor menor de energia. A distância entre os resíduos é calculada utilizando a distância euclidiana.

Geralmente para avaliar e comparar a performance dos AEMOs, indicadores de qualidade são utilizados. Neste estudo o indicador *hypervolume* foi utilizado. Este indicador considera o volume do espaço de busca dominado pela fronteira conhecida de Pareto obtida por um algoritmo [Zitzler et al., 2003]. Um maior valor de *hypervolume* significa maior qualidade na cobertura do que um algoritmo com valor inferior.

Os 4 algoritmos foram implementados utilizando a arquitetura *open source* disponível no *framework* jMetal. A arquitetura do jMetal é de fácil extensão e possui uma ativa comunidade.

5.2 EGHyPDP

Esta abordagem é baseada no trabalho desenvolvido por [Sabar et al., 2015], o qual utilizou GEP (*gene expression programming*) com objetivo de gerar, de maneira *online*, os componentes de um *framework* hiper-heurístico para diversos domínios de problemas. Os testes de generalidade realizados, utilizando os 6 domínios providos pelo *framework* hiper-heurístico HyFlex, apresentaram bons resultados em relação às outras estratégias hiper-heurísticas do estado da arte. Nesta proposta pretende-se utilizar EG ao invés de GEP e aplicar ao PDP simplificado utilizando o modelo HP-2D. Da mesma maneira que a abordagem que utilizou AEMOs a representação de coordenadas relativas descrita na subseção 2.3.1, será utilizada. Como mencionado anteriormente, um *framework* hiper-heurístico possui dois níveis: alto (*high-level heuristics*) e baixo (*low-level heuristics*). Nesta proposta as heurísticas de alto nível são compostas por: um mecanismo de seleção e um critério de aceitação. Já as heurísticas de baixo nível consistem em um conjunto de heurísticas, selecionadas de estudos anteriores, um mecanismo de memória e uma função de *fitness*.

5.3 Heurísticas de alto nível

Esta abordagem foi desenvolvida para gerar de maneira *offline* os componentes de uma heurística de alto nível (mecanismo de seleção e critério de aceitação) para um *framework* hiper-heurístico. A figura 5.8 apresenta a estrutura geral do EGHyPDP.

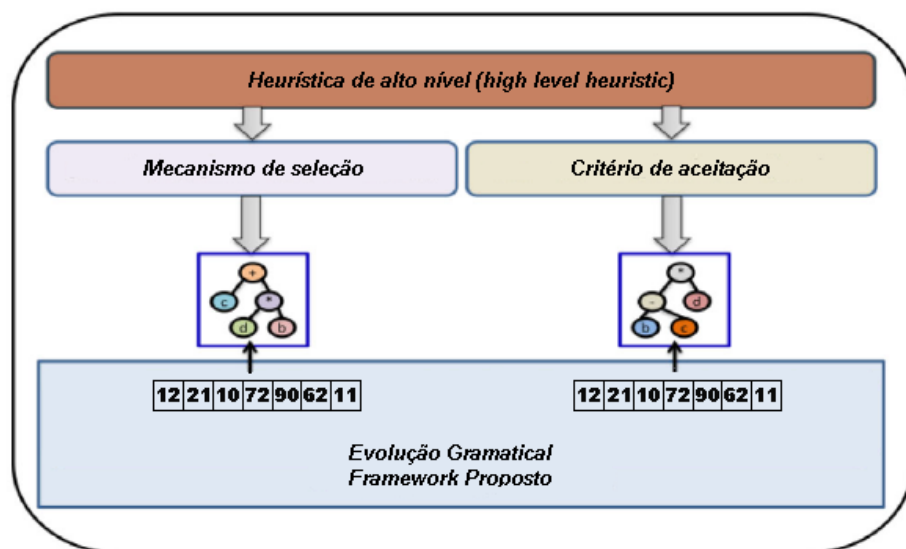


Figura 5.8: Estrutura geral do EGHyPDP.
Fonte: Adaptado de [Sabar et al., 2015]

Heurísticas de alto nível geralmente levam em consideração uma ou mais informações referentes ao histórico das aplicações das heurísticas de baixo nível para tomar suas decisões. Tradicionalmente, informações tais como desempenho (capacidade de melhorar soluções), tempo

(desde a última aplicação de uma dada heurística) e intervalo de confiança (no caso de estratégias que utilizam MAB) são utilizadas como base de conhecimento. [Sabar et al., 2015] propõem a utilização de vários critérios para avaliar as heurísticas de baixo nível. A razão disto se dá pelo fato de que cada critério irá favorecer a seleção de uma heurística de baixo nível a partir de um aspecto diferente. Por exemplo, algumas heurísticas de baixo nível podem ter bom desempenho apenas no início da busca, enquanto outras podem obter melhores resultados apenas ao final. Estes critérios propostos por [Sabar et al., 2015] contém estatísticas referente às aplicações das heurísticas de baixo nível e são genéricos o suficiente para serem aplicados ao PDP. Os critérios propostos por Sabar et al. [Sabar et al., 2015] são detalhados em seguida:

- *RC (Reward Credit)*: Representa a recompensa que uma determinada heurística de baixo nível deve receber baseado no seu desempenho durante o processo de busca. Quando a i -ésima heurística é aplicada, a melhoria para a solução é computada. O cálculo da melhoria é dado por: $M(i) = (|f1 - f2|/f1) * 100$ se $f2 < f1$, onde $f1$ é a qualidade da solução corrente e $f2$ é a qualidade da solução resultante após a aplicação da i -ésima heurística. A melhoria obtida é salva em uma janela deslizando (FIFO) de tamanho W . O crédito de qualquer heurística de baixo nível é então atribuído como o máximo valor na janela deslizando correspondente. A ideia por trás deste critério é: heurísticas de baixo nível que não são usadas com frequência mas que alteram a solução com grandes melhorias tendem a ter mais preferência do que aquelas que geram pequenas melhorias. Portanto as heurísticas que trazem frequentes, mas pequenas melhorias irão ter menos probabilidade de serem selecionadas.
- C_{best} : Número de vezes que a i -ésima heurística de baixo nível atualizou a melhor solução conhecida. Este critério favorece as heurísticas de baixo nível que obtiveram êxito em melhorar a melhor solução conhecida até o momento. Este critério é útil para sistematicamente melhorar o atual mínimo local.
- $C_{current}$: Número de vezes que a i -ésima heurística de baixo nível atualizou a solução atual. Este critério favorece as heurísticas de baixo nível que obtêm êxito em atualizar a solução corrente. Este critério serve para deixar a busca concentrada próxima à solução corrente.
- C_{accept} : Número de vezes que a solução gerada pela i -ésima heurística de baixo nível foi aceita pelo critério de aceitação. Irá favorecer heurísticas de baixo nível que podem ajudar a escapar de um mínimo local.
- C_{ava} : A média de melhorias anteriores da i -ésima heurística de baixo nível durante o progresso da busca. Este critério favorece heurísticas de baixo nível que realizaram grandes melhorias em média.
- C_r : O número de vezes que a i -ésima heurística de baixo nível foi classificada como primeira.

Da mesma maneira [Sabar et al., 2015] propõem o uso de dados referentes ao histórico de aplicações das heurísticas de baixo nível para compor critérios de aceitação que irão definir limites para aceitar soluções com qualidade inferior. Dessa forma, um conjunto de fatores também foi proposto e será detalhado em seguida:

- Delta: A diferença da qualidade entre a solução corrente e a solução descendente.
- PF: A qualidade da solução anterior.
- CF: A qualidade da solução atual.
- CI: Iteração corrente.
- TI: Número de iterações.

Utilizando estes dados estatísticos e um conjunto de funções matemáticas simples, tais como soma, subtração, multiplicação e divisão, uma gramática foi desenvolvida para suportar a geração das heurísticas de alto nível. A gramática desenvolvida para gerar mecanismos de seleção e critérios de aceitação é apresentada na Gramática 5.1.

Para inicializar os dados dos terminais: todas as heurísticas foram executadas uma vez e os dados para cada terminal foi calculado. Toda iteração seguinte irá atualizar os dados dos terminais e essas informações são utilizadas durante a busca.

$$\begin{aligned} \langle hh\text{-}selection \rangle &::= \langle selection\text{-}mechanism \rangle \langle acceptance\text{-}criterion \rangle \\ \langle selection\text{-}mechanism \rangle &::= \langle selection\text{-}terminal \rangle \\ &\quad | \langle selection\text{-}mechanism \rangle \langle math\text{-}function \rangle \langle selection\text{-}mechanism \rangle \\ &\quad | (\langle selection\text{-}mechanism \rangle \langle math\text{-}function \rangle \langle selection\text{-}mechanism \rangle) \\ \langle selection\text{-}terminal \rangle &::= RC \mid Cbest \mid Ccurrent \mid Caccept \mid Cava \mid Cr \\ \langle math\text{-}function \rangle &::= + \mid - \mid * \mid \% \\ \langle acceptance\text{-}criterion \rangle &::= \langle acceptance\text{-}terminal \rangle \\ &\quad | \langle acceptance\text{-}criterion \rangle \langle math\text{-}function \rangle \langle acceptance\text{-}criterion \rangle \\ &\quad | (\langle acceptance\text{-}criterion \rangle \langle math\text{-}function \rangle \langle acceptance\text{-}criterion \rangle) \\ \langle acceptance\text{-}terminal \rangle &::= PF \mid CF \mid CI \mid TI \end{aligned}$$

Gramática 5.1: Gramática definida para gerar heurísticas de alto nível

O conjunto de funções matemáticas para combinar de diferentes maneiras os dados históricos das aplicações das heurísticas de baixo nível é apresentado abaixo:

- +: Adiciona as duas entradas.
- -: Subtrai a segunda entrada da primeira.

- *: Multiplica as duas entradas.
- %: Divisão protegida, isto é, se o denominador for 0, o altera para 0,001.

Utilizando a Gramática 5.1 e vetores de inteiros é possível gerar heurísticas de alto nível. Os conjuntos terminais da gramática apresentam estatísticas sobre as heurísticas de baixo nível e estas são a matéria-prima para a construção dos componentes das heurísticas de alto nível de um *framework* hiper-heurístico.

O próximo passo consiste em evoluir uma população de vetores de inteiro utilizando o processo evolutivo descrito na subseção 3.4.

5.3.1 Função de *Fitness*

Com objetivo de avaliar os indivíduos gerados durante a busca, uma função de *fitness* foi desenvolvida. A função executa a heurística de alto nível, representada por um dado indivíduo, em 3 instâncias aleatórias de um total de 11 selecionadas dos estudos anteriores. Para cada instância a heurística de alto nível foi executada com um tempo máximo de 30 minutos e o retorno é a melhor solução para o PDP simplificado com o modelo HP-2D. O valor de *fitness* associado com a solução retornada é então normalizado entre 0 e 1. O *fitness* de um indivíduo, da EG, consiste na soma das saídas das execuções com cada uma das 3 instâncias. Dessa maneira, o melhor valor possível é 3 e o pior é 0. A razão de executar a heurística de alto nível (indivíduo) com 3 instâncias é que treinar a EG com menos instâncias não seria suficiente para obter bons resultados em várias instâncias na fase de validação.

5.3.2 Critério de Parada

Para terminar o processo da EG um número máximo de iterações que não obtêm melhora será utilizado como condição de parada. Note que este critério de parada é referente à parada do processo da EG e não das execuções dos indivíduos dentro do *framework* hiper-heurístico, que ocorrem durante o progresso da EG.

5.4 Heurísticas de baixo nível

Nas heurísticas de baixo nível o EGHYPDP possui 2 componentes principais: um conjunto de heurísticas de baixo nível e um mecanismo de memória.

5.4.1 Conjunto de heurísticas de baixo nível

O conjunto de heurísticas de baixo nível foi desenvolvido baseado estudos anteriores e foi apresentados no início deste capítulo.

5.5 Processo geral do EGHYPDP

As principais etapas da EGHYPDP proposta serão apresentadas nesta seção. Inicialmente uma população de indivíduos (heurísticas de alto nível: mecanismos de seleção e critérios de aceitação) é gerada conforme o procedimento que será descrito posteriormente nesta seção. O *fitness* da população é calculado inserindo os indivíduos em um *framework* hiper-heurístico e o executando com 3 instâncias por 30 minutos. E de maneira iterativa selecionar indivíduos pais e aplicar os operadores de cruzamento, *prune*, mutação, e *duplicate* para gerar descendentes. Posteriormente, estes indivíduos são submetidos ao processo de avaliação descrito na subseção 5.3.1.

O processo da EGHYPDP irá parar apenas quando o critério de parada discutido na subseção 5.3.2 for atingido e será retornado o indivíduo (heurística de alto nível) que possuir o maior valor de *fitness*. Também será retornada a solução ao PDP que tiver maior qualidade no mecanismo de memória.

5.5.1 Mecanismo de Memória

A maioria dos *frameworks* hiper-heurísticos propostos na literatura operam sobre uma única solução [Chakhlevitch e Cowling, 2008, Burke et al., 2013]. Blum et al. [Blum et al., 2011] menciona que utilizar uma única solução pode restringir a capacidade de explorar complexos espaços de busca e com alta variância de características. Dessa maneira, [Sabar et al., 2015] propôs uma abordagem que utiliza um mecanismo de memória, assim como [Talbi e Bachelet, 2006], o qual contém um conjunto de soluções com alta qualidade e diversificadas, atualizado durante o progresso da busca. Nesta proposta o mecanismo de memória tem a responsabilidade de armazenar soluções para o problema PDP utilizando a representação de coordenadas relativas para o modelo HP-2D.

Inicialização do Mecanismo de Memória

Tradicionalmente algoritmos evolutivos inicializam suas populações iniciais de maneira aleatória, por conta disto muitas soluções inválidas ao modelo HP, são geradas na inicialização. Isto geralmente ocasiona perda de tempo de processamento, por conta da grande quantidade de conformações inválidas antes que bons resultados sejam obtidos. Diante disto, [Benítez, 2010] propôs uma estratégia especializada de inicialização. A população de seu algoritmo genético, é dividida em duas partes. Uma gerada aleatoriamente, com indivíduos que potencialmente possuem colisões. E uma segunda parte onde todos os indivíduos são livres de colisões. Uma configuração é utilizada para definir a proporção entre as duas partes da população inicial. Para garantir que os indivíduos não possuam colisões, uma estratégia de *backtracking* deve ser utilizada. Nesta abordagem, a mesma estratégia de inicialização via *backtracking*, descrita na seção 5.1 foi implementada.

Atualização do Mecanismo de Memória

Para cada indivíduo (heurística de alto nível) será selecionada de maneira aleatória uma solução do mecanismo de memória e a busca irá iniciar em torno desta solução, quando o framework hiper-heurístico atingir o seu número máximo de iterações a solução final tem que ser avaliada para verificar sua qualidade e diversidade. A qualidade de uma solução para o PDP utilizando o modelo HP é inversamente proporcional à quantidade de interações entre aminoácidos hidrofóbicos. Portanto a qualidade de uma solução é dada pela quantidade de iterações H-H multiplicada por -1, conforme descrito na subseção 2.3.1. As soluções geradas que tiverem a qualidade maior que todas as soluções contidas no mecanismo de memória substituirão a solução que tiver menor similaridade segundo a distância de [Hamming, 1950]. Se a qualidade de uma solução gerada não for maior que todas as soluções, mas melhor em relação a um sub-conjunto do mecanismo de memória, esta substituirá a solução que tiver menor qualidade e menor similaridade do sub-conjunto. E por fim se a qualidade da solução gerada for pior que todas contidas no mecanismo de memória, esta é descartada. A similaridade é considerada a fim de manter a diversidade entre as soluções.

5.6 Considerações Finais

Neste capítulo foram discutidos os principais aspectos relativos à duas estratégias propostas nesta dissertação. Inicialmente, foi discutido sobre a aplicação de AEMOs e em seguida o EGHYPDP foi apresentado. A representação relativa foi utilizada para codificar as soluções para ambas as estratégias. O conjunto de heurísticas de baixo nível, utilizado em ambas abordagens, também foi apresentado. No caso dos AEMOs 4 versões de algoritmos foram apresentadas. As funções objetivas também foram discutidas. Duas adaptações foram propostas que consistiram em adicionar um *pool* de operadores para tornar os AEMOs mais adaptativos e a inicialização via *backtracking*. Para aplicar o EGHYPDP com objetivo de gerar heurísticas de alto nível de um framework hiper-heurístico para o PDP simplificado, uma gramática foi desenvolvida. Também foi apresentado um conjunto de terminais referente ao histórico das aplicações das heurísticas de baixo nível. Funções matemáticas simples compõem a gramática e combinadas com os dados estatísticos diferentes heurísticas de alto nível foram geradas. Posteriormente, a função de *fitness* avalia estas heurísticas executando o *framework* composto por tais três vezes em três instâncias distintas selecionadas de maneira aleatória.

O próximo capítulo apresenta os experimentos realizados afim de avaliar o desempenho das abordagens aqui propostas.

Capítulo 6

Experimentos

Neste Capítulo serão apresentados os experimentos para validar as duas estratégias apresentadas no Capítulo 5. O conjunto de *benchmark* para avaliar o desempenho dos das abordagens ao PDP simplificado com modelo HP-2D

A partir da revisão bibliográfica foi possível selecionar 11 instâncias dos trabalhos relacionados [Unger e Moulton, 1993b, Krasnogor et al., 2002, Shmygelska et al., 2002, Shmygelska e Hoos, 2003, Hsu et al., 2003] para compor um conjunto de *benchmark*. Este conjunto foi utilizado para avaliar ambas estratégias propostas, a Tabela 6.1 apresenta o tamanho, o melhor valor de energia conhecido e a fórmula da sequência de aminoácidos para o modelo HP. Tanto os AEMOs e o EGHYPDP foram submetidos a esse conjunto de instâncias o qual possui diferentes níveis de complexidade e características. Todos os experimentos realizados nesta dissertação foram executados 30 vezes por conta do comportamento estocástico inerente às heurísticas utilizadas. Por conta das múltiplas execuções em casos que forem necessários serão apresentados os valores de média, desvio padrão, mínimo e máximo referente as execuções.

Este capítulo esta dividido em duas seções para melhor apresentar os resultados obtidos por cada uma das estratégias propostas. A primeira seção descreve os experimentos realizados para avaliar a aplicação de AEMOs para o PDP utilizando o modelo HP-2D. A segunda seção apresenta os experimentos realizados para avaliar a habilidade do EGHYPDP em gerar heurísticas de alto nível para um *framework* hiper heurístico, utilizando um procedimento de EG.

Tabela 6.1: Instâncias de *benchmark* utilizadas nos experimentos.

Instância	Tamanho	Melhor Valor de Energia	Fórmula HP
1	20	-9	HPHPHPHPHPHPHPHPHPHP
2	24	-9	HHPPHPHPHPHPHPHPHPHP
3	25	-8	PPHPHPHP ⁴ HH ⁴ HH ⁴ HH
4	36	-14	P ³ HHPPHP ⁵ H ⁷ PPHP ⁴ HHPPHP
5	48	-23	PPHPHPHPHP ⁵ H ¹⁰ P ⁶ HHPPHPHPHP ⁵
6	50	-21	HHHPHPHPHP ⁴ HP ³ HP ³ HP ⁴ HP ³ HP ³ HP ⁴ HP ⁴ H
7	60	-36	PPH ³ PH ⁸ P ³ H ¹⁰ PH ³ H ¹² P ⁴ H ⁶ PHHPHP
8	64	-42	H ¹² PHHPHPHP ² PPHPHP ² PPHPHP ² PPHPHP ¹²
9	85	-53	H ⁴ P ⁴ H ¹² P ⁶ H ¹² P ³ H ¹² P ³ H ¹² P ³ H ² H ² P ² H ² P ² HPH
10	100	-48	P ⁶ HPH ² P ⁵ H ³ PH ⁵ PH ² P ⁴ H ² P ² H ² PH ⁵ PH ¹⁰ PH ² PH ⁷ P ¹ PH ⁷ P ² HP ³ P ⁶ HPH
11	100	-50	P ³ H ² P ² H ⁴ P ² H ³ PH ² PH ² PH ⁴ P ⁸ H ⁶ P ² H ⁶ P ⁹ HPH ² PH ¹ PH ² H ³ PH ² PH ² HPH ³ P ⁶ H ³

6.1 Resultados dos AEMOs aplicados ao PDP

Nesta seção serão apresentados o conjunto de experimentos realizados utilizando a abordagem multi-objetiva com os algoritmos NSGAII e IBEA. Inicialmente, serão apresentadas as configurações utilizadas para os algoritmos. Em seguida, uma comparação entre os resultados de cada algoritmo é apresentada. Por fim, uma comparação com os resultados dos trabalhos relacionados, que tratam o PDP de maneira mono-objetiva, é apresentada.

As configurações utilizadas nos AEMOs foi definida baseada no tamanho das instâncias. Para instâncias menores o tamanho da população e o número máximo de avaliações são menores. Já para as sequências maiores os valores utilizados são superiores. A Tabela 6.2 apresenta para cada instância: o tamanho da instância, o tamanho da população e número máximo de avaliações.

No caso das versões padrão dos algoritmos IBEA e NSGAII as probabilidades de cruzamento e mutação foram configuradas com 0.9 e 0.01 respectivamente. A segunda abordagem não necessita de probabilidades, pois os operadores sempre serão aplicados para gerar novos indivíduos. No caso dos algoritmos IBEA e M_IBEA a população auxiliar foi mantida em 200 para todas as instâncias.

Tabela 6.2: Tamanho da população, número máximo de avaliações para cada instância

Instância	Tamanho	Tamanho População	N Max Avaliações
1	20	100	25000
2	24	100	25000
3	25	500	250000
4	36	500	250000
5	48	1000	2500000
6	50	1000	2500000
7	60	2500	2500000
8	64	2500	2500000
9	85	2500	2500000
10	100	3500	3500000
11	100	3500	3500000

Conforme mencionado no Capítulo 5 o indicador *hypervolume* foi utilizado para comparar o desempenho dos AEMOs. Para cada algoritmo a média e o desvio padrão do *hypervolume* referente a 30 execuções é apresentada na Tabela 6.3. Os maiores valores estão em negrito e indicam uma maior aproximação da fronteira de Pareto [Barr, 1998].

Observando a Tabela 6.3 é possível notar que, exceto para a instância 1, em todas instâncias o algoritmo M_IBEA (versão modificado com *backtracking* e o *pool* de operadores) obteve a maior média de *hypervolume*. No caso da instância 1, o IBEA sem modificações obteve a média mais alta. Também vale mencionar, que quando comparando apenas o NSGAII e o M_NSII, a versão modificada obteve as melhores médias. De maneira geral, os AEMOs com *backtracking* e o *pool* de operadores (M_IBEA and M_M_NSII) apresentaram uma melhoria

Tabela 6.3: Resultado de média/desvio padrão dos AEMOs

Instância	<i>Hypervolume</i> Média (Desvio padrão)			
	NSGAI	M_NSGAI	IBEA	M_IBEA
1	0.742827 (0.106315)	0.720864 (0.131351)	0.789712 (0.067660)	0.786571 (0.099424)
2	0.680572 (0.083445)	0.712275 (0.137226)	0.719960 (0.080727)	0.737086 (0.095299)
3	0.671171 (0.129417)	0.709898 (0.124201)	0.716438 (0.148112)	0.738017 (0.155638)
4	0.702280 (0.0689832)	0.740153 (0.075271)	0.751755 (0.092427)	0.785728 (0.055607)
5	0.707654 (0.082611)	0.758128 (0.062315)	0.733464 (0.128757)	0.807637 (0.039620)
6	0.667771 (0.132218)	0.774017 (0.063231)	0.728699 (0.080679)	0.821177 (0.048124)
7	0.784483 (0.063257)	0.792843 (0.033062)	0.801778 (0.067111)	0.810351 (0.054576)
8	0.677464 (0.041287)	0.705798 (0.053048)	0.7450656 (0.036454)	0.811439 (0.050087)
9	0.687454 (0.041287)	0.710798 (0.043546)	0.7150656 (0.026561)	0.771439 (0.044087)
10	0.650798 (0.062157)	0.690891 (0.033661)	0.7250126 (0.031211)	0.761439 (0.013186)
11	0.630496 (0.052751)	0.670812 (0.031235)	0.713126 (0.053422)	0.752445 (0.042326)

considerável em relação aos AEMOs tradicionais. Na Tabela 6.3, as células do M_IBEA que estão marcados com cinza apresentaram diferença estatística de acordo com o teste de Kruskal-Wallis [McKight e Najab, 2010] entre os outros algoritmos (NSGAI, M_NSGAI e IBEA).

6.1.1 Comparação com outras abordagens mono-objetivas

Esta subseção apresenta a comparação dos resultados obtidos pelas melhores variantes de cada AEMO com outras abordagens mono-objetivas propostas em trabalhos anteriores. Os trabalhos comparados são: EDA [Santana et al., 2008], GA [Unger e Moul, 1993b], MMA [Krasnogor et al., 2002], ACO [Shmygelska et al., 2002], NewACO [Shmygelska e Hoos, 2003] e PERM [Hsu et al., 2003]. Todos os trabalhos mencionados tratam-se de abordagens mono-objetivas, considerando apenas o valor de energia referente as estruturas de proteínas. A Tabela 6.4 apresenta os melhores resultados, em termos da energia, pelas versões modificadas dos AEMOs, assim como os melhores resultados obtidos pelos estudos anteriores.

Tabela 6.4: Comparação dos melhores AEMOs com o estudos anteriores do PDP

Instância	M_IBEA	M_NSGAII	EDA	GA	MMA	ACO	NewACO	PERM
1	-9	-9	-9	-9	-9	-9	-9	-9
2	-9	-9	-9	-9	-9	-9	-9	-9
3	-8	-8	-8	-8	-8	-8	-8	-8
4	-14	-13	-14	-14	-14	-14	-14	-14
5	-23	-22	-23	-22	-22	-23	-23	-23
6	-21	-21	-21	-21		-21	-21	-21
7	-35	-34	-35	-34		-34	-36	-36
8	-42	-39	-42	-37		-32	-42	-38
9	-49	-44	-52	-37		-32	-52	-53
10	-43	-39	-47				-47	-48
11	-41	-37	-48				-47	-50

No caso das instâncias 1, 2 e 3 a versões modificadas dos AEMOs (M_NSGAII e M_IBEA) obtiveram os mesmos valores mínimos de energia que os estudos anteriores considerados nesta comparação. No caso da instância 4, exceto pelo algoritmo M_NSGAII, todos os outros atingiram o valor de -14. Já no caso da instância 5 o algoritmo M_IBEA e 4 algoritmos mono-objetivo obtiveram o valor ótimo de 23. Entretanto, M_NSGAII e os demais algoritmos obtiveram um valor inferior de -22.

Já no caso da instância 6 todos os algoritmos obtiveram o valor ótimo de -21. Para a instância 7 o algoritmo M_IBEA obteve -35 da mesma maneira que o EDA. Entretanto o valor ótimo para sequência 7 é -36 e foi obtido pelo NewACO e PERM. Para a instância 8 o algoritmo M_IBEA obteve o valor ótimo -42 mesmo valor que o EDA e NewACO. Os demais algoritmos obtiveram valores inferiores. No caso das instâncias 9,10 e 11, as maiores, apenas o algoritmo PERM conseguiu obter os melhores resultados. Estas instâncias são as mais complexas devido ao seu tamanho. Dessa maneira, os algoritmos tendem a ficar presos em mínimos locais. Consequentemente, nestes casos os algoritmos necessitam de mecanismos inteligentes para escapar de mínimos locais e guiar a busca para regiões promissoras.

Após analisar os resultados foi possível constatar que apenas um AEMO, o M_IBEA conseguiu encontrar os melhores resultados para 7 instâncias de 11. Entretanto, nas instâncias mais complexas seu desempenho é degradado. Já os outros AEMOs propostos não obtiveram resultados expressivos.

6.1.2 Conclusão dos experimentos utilizando os AEMOs

AEMOs são algoritmos evolucionários que visam a otimização de múltiplos objetivos em paralelo. Estes apresentam bons resultados quando aplicados a problemas de várias áreas da ciência. Nestes experimentos dois AEMOs foram aplicados ao problema PDP utilizando o modelo HP-2D. Duas abordagens multi-objetivas foram apresentadas: a primeira utiliza as versões padrão dos algoritmos IBEA e NSGAII; a segunda abordagem consistiu em modificar o

IBEA e NSGAII adicionando a inicialização com *backtrack* e o *pool* de operadores, com objetivo de melhorar os resultados. Dado os resultados se tornou claro que as versões modificadas dos algoritmos tem uma maior habilidade em explorar o espaço de busca quando comparados com as versões padrão.

Também foi possível verificar que a abordagem multi-objetiva utilizando o NSGAII e o M_NSGAII também não apresentou resultados satisfatórios, tanto em termos de *hypervolume* como valores de energia, quando comparado com o IBEA e a versão modificada. Mesmo a versão padrão de IBEA obteve melhores resultados do que ambas as versões do NSGAII. Porém, a versão modificada M_IBEA apresentou os melhores resultados dentre todos AEMOs propostos. Isto sugere que apenas uma formulação multi-objetiva não é suficiente para atingir bons resultados em termos de energia das estruturas. Com a inicialização via *backtracking*, *pool* de operadores e a maneira sofisticada de explorar o espaço de busca multi-objetivo do IBEA possibilitaram resultados promissores.

Dado os resultados é indiscutível que o algoritmo M_IBEA foi capaz de escapar dos mínimos locais em vários os casos, exceto para as instâncias 7,9,10 e 11. Estas instâncias são as mais complexas e muitas propostas não conseguem obter os melhores resultados. Dentre os trabalhos relacionados apenas o PERM consegue explorar o espaço de busca o suficiente para encontrar os melhores resultados.

Também vale mencionar que os parâmetros para os algoritmos não foram *tunados* e que existe uma chance de obterem melhores resultados caso um procedimento de *tunning* seja feito.

Os resultados obtidos pelo M_IBEA abrem um amplo campo de possibilidades para aprofundar formulações multi-objetivas para o problema PDP com o modelo HP-2D ou até mesmo o HP-3D. As descobertas desta dissertação motivam pesquisas adicionais utilizando abordagens multi-objetivas e adição de um *pool* de operadores com objetivo de aprimorar a habilidade de escapar de mínimos locais. No caso de formulações multi-objetiva é possível mencionar que o *design* de abordagens inovadoras como o uso de métricas para avaliar o grau de compactação das conformações de proteínas ou outros métodos que considerem diferentes fatores, podem aprimorar os AEMOs e possibilitar que melhores resultados sejam obtidos.

6.2 Resultados obtidos com EGHYPDP

Nesta seção, serão discutidos os experimentos conduzidos com objetivo de avaliar o *design* automático de heurísticas de alto nível para um *framework* hiper-heurístico. Este *framework* foi aplicado ao PDP para avaliar seu desempenho. A implementação do EGHYPDP ocorreu durante o período de desenvolvimento desta dissertação.

Três grupos de experimentos foram desenvolvidos e executados. O primeiro grupo de experimentos tratou apenas de gerar mecanismos de seleção. O segundo grupo foi desenvolvido com objetivo de gerar apenas critérios de aceitação. E por último o terceiro grupo foi desenvolvido com intuito de gerar em paralelo tanto mecanismos de seleção como critérios de aceitação.

O tempo de execução total destes 3 grupos de experimentos foi de 28 dias. Após a execução dos experimentos os resultados obtidos por cada grupo foram comparados entre si. Porém esta comparação não foi suficiente para avaliar a capacidade das heurísticas de alto nível. [Misir, 2012] nos forneceu o código fonte utilizado em seu estudo e isto possibilitou a execução do GIHH (Generic Intelligent Hyper-heuristic) no contexto do PDP com o modelo HP-2D. O GIHH obteve resultados impressionantes em outros problemas de *benchmark*. Portanto, a comparação dos resultados em relação aos obtidos por uma estratégia hiper heurística (do estado da arte) desenvolvida por um pesquisador foi realizada. .

O primeiro grupo de experimentos denominado, EGHYPDP-1, apenas gerou mecanismos de seleção. O critério de aceitação foi configurado e mantido fixo com uma estratégia que aceita apenas soluções melhores ou iguais [Burke et al., 2013]. O objetivo deste grupo é avaliar a habilidade da abordagem proposta em gerar mecanismos de seleção utilizando um critério de aceitação fixo.

O segundo grupo de experimentos, EGHYPDP-2, consistiu em gerar apenas critérios de aceitação. O mecanismo de seleção foi configurado e mantido fixo utilizando o melhor mecanismos de seleção gerado pelo experimento EGHYPDP-1. Consequentemente, este experimento depende da saída do primeiro grupo. O objetivo deste experimento é avaliar a qualidade dos critérios de aceitação gerados, separadamente do mecanismo de seleção o qual foi mantido fixo.

O terceiro grupo de experimentos, EGHYPDP-3, foi desenvolvido para gerar tanto mecanismos de seleção e critérios de aceitação ao mesmo tempo. Este grupo avalia a habilidade em gerar mecanismos de seleção em conjunto com critérios de aceitação. Diferentemente do grupo EGHYPDP-2 , este grupo não depende de nenhuma saída dos experimentos anteriores.

Para cada grupo de experimentos a fase de treino consistiu em executar o processo da EG utilizando a função de *fitness* descrita no Capítulo 5. Na fase de validação o melhor mecanismo de seleção e o melhor critério de aceitação obtidos foram executados utilizando todas as 11 instâncias descritas na Tabela 6.1.

6.2.1 Resultados obtidos com EGHYPDP-1

O melhor indivíduo (mecanismo de seleção gerado e critério de aceitação melhor ou igual) encontrado no experimento EGHYPSP-1 é apresentado na Equação 6.1

$$RC * C_{current} * C_{ava} - Cr. \quad (6.1)$$

Este mecanismo de seleção, juntamente com o critério de aceitação (melhor ou igual) foi executado novamente por 30 vezes com um tempo limite de 30 minutos em cada execução, porém utilizando todas as 11 instâncias disponíveis. A Tabela 6.5 apresenta os resultados de média, desvio padrão, mínimo e máximo das execuções do melhor indivíduo apresentado na Equação 6.1. A última linha denota o melhor valor de energia conhecido para cada instância. Os valores destacados com negrito representam os melhores valor conhecido para determinada

Tabela 6.5: Resultados da execução do melhor indivíduo encontrado pelo grupo de experimento EGHyPDP-1

Instância	1	2	3	4	5	6	7	8	9	10	11
Média	-8.1	-7.6	-6.7	-11.9	-17.4	-16	-30	-28.3	-40.1	-35.6	-35.9
Desvio Padrão	0.3	0.5	0.5	0.7	1	1.4	1.7	2	2.7	2.1	2.9
Máximo	-8	-7	-5	-11	-15	-13	-25	-23	-34	-32	-27
Mínimo	-9	-9	-8	-14	-23	-21	-33	-42	-46	-40	-41
Melhor Valor de Energia	-9	-9	-8	-14	-23	-21	-36	-42	-53	-48	-50

instância. É possível notar analisando as duas últimas linhas, que exceto para as instâncias 7,9,10 e 11, todas as outras instâncias, o indivíduo gerado conseguiu obter os melhores valores conhecidos. Dessa maneira, é possível observar que quando o tamanho das instâncias aumentam a qualidade dos resultados obtidos degrada.

6.2.2 Resultados obtidos com EGHyPDP-2

O melhor indivíduo (mecanismo de seleção fixo e critério de aceitação gerado) encontrado no grupo de experimento EGHyPDP-2 é apresentado na Equação 6.2. O mecanismo de seleção utilizado foi o mesmo apresentado na Equação 6.1. A razão por usar este mecanismo de seleção foi: os resultados obtidos pelo melhor indivíduo da execução do EGHyPDP-1 se apresentaram promissores e o objetivo era gerar um um bom critério de aceitação combinado com um mecanismo de seleção que funcione bem.

$$((TI/Delta)/((Delta * ((TI/Delta)/CI) * Delta/Delta * TI) - CI)) \quad (6.2)$$

Este critério de aceitação, juntamente com o melhor mecanismo de seleção gerado pelo EGHyPDP-1, foram executados 30 vezes com um tempo máximo de 30 minutos utilizando as 11 instâncias disponíveis. A Tabela 6.6 apresenta os resultados obtidos. Novamente, observando as duas ultimas linhas é possível verificar que em apenas em duas instâncias os melhores valores foram encontrados. Dessa maneira, é fácil notar que os resultados apresentados na Subseção 6.2.1 obtiveram êxito em encontrar, para mais instâncias, os melhores valores conhecidos. Vale mencionar, que o melhor mecanismo de seleção encontrado combinado com um critério de aceitação melhor ou igual apresentou melhores resultados do que quando combinado com o melhor critério de aceitação gerado.

Tabela 6.6: Resultados da execução do melhor indivíduo encontrado no grupo de experimento EGHyPDP-2

Instância	1	2	3	4	5	6	7	8	9	10	11
Média	-8	-7.6	-6.7	-10.1	-14.8	-14.7	-27	-25.7	-38.3	-32.8	-30
Desvio Padrão	0.6	0.4	0.6	0.7	1.5	1.4	2.0	2.5	3.3	3.7	3.4
Máximo	-7	-7	-5	-8	-12	-12	-23	-22	-31	-26	-24
Mínimo	-9	-8	-8	-11	-17	-18	-31	-31	-44	-40	-37
Melhor Valor de Energia	-9	-9	-8	-14	-23	-21	-36	-42	-53	-48	-50

Tabela 6.7: Resultados da execução do melhor indivíduo encontrado no grupo de experimento EGHyPDP-3

Instância	1	2	3	4	5	6	7	8	9	10	11
Média	-7.6	-7.0	-5.7	-9.7	-13.8	-12.7	-24	-24.2	-31.6	-27	-26.4
Desvio Padrão	0.6	0.7	0.8	0.9	1.3	0.9	1.1	1.6	1.7	1.7	2
Mínimo	-7	-6	-4	-7	-12	-11	-21	-21	-29	-24	-24
Máximo	-9	-8	-8	-11	-18	-15	-26	-28	-37	-31	-31
Melhor Valor de Energia	-9	-9	-8	-14	-23	-21	-36	-42	-53	-48	-50

6.2.3 Resultados obtidos com EGHyPDP-3

O melhor indivíduo gerado pelo grupo de experimento EGHyPDP-3 é composto por um mecanismo de seleção e um critério de aceitação. As Equações 6.3 e 6.4 apresentam respectivamente o mecanismo de seleção e o critério de aceitação gerados.

$$((((Caccept/RC) * Cr/Caccept)/RC * Cr)/Caccept/RC) * Cr)/Caccept \quad (6.3)$$

$$((((CI/PF) * Delta/CI)/PF * Delta)/CI/PF) * Delta)/CI \quad (6.4)$$

A Tabela 6.7 apresenta os resultados da execução do mecanismo de seleção e do critério de aceitação gerados pelo grupo de experimentos EGHyPDP-3. Da mesma maneira, consistiram em 30 execuções com tempo máximo de 30 minutos. Os valores destacados com negrito são os casos que o indivíduo alcançou os melhores valores conhecidos. É possível observar que apenas em dois casos isto ocorreu nas menores instâncias, entre 20 a 25 aminoácidos. Estas sequências são extremamente triviais levando apenas minutos para um algoritmo genético encontrar as melhores soluções conhecidas. Dessa forma, é possível concluir que a geração de ambas as estratégias neste contexto não apresentou bons resultados.

Após estes experimentos foi concluído que apenas a geração de mecanismos de seleção apresentou algum potencial. Os melhores resultados conhecidos foram obtidos em 7 instâncias de um total de 11, utilizando o melhor mecanismo de seleção encontrado. No caso dos critérios de aceitação os resultados não foram como esperados. A hipótese inicial era que combinadas as estratégias de seleção e aceitação esta se complementariam. Dessa maneira, possibilitando que o

Tabela 6.8: Os melhores resultados conhecidos e resultados encontrados com o EGHyPDP e GIHH.

Instância	1	2	3	4	5	6	7	8	9	10	11
EGHyPDP	-9	-9	-8	-14	-23	-21	-33	-42	-46	-40	-41
GIHH	-9	-9	-8	-14	-22	-21	-35	-37	-49	-43	-45
Melhor Valor de Energia	-9	-9	-8	-14	-23	-21	-36	-42	-53	-48	-50

framework hiper heurístico pudesse explorar de maneira adequada o espaço de busca. Apesar do resultado promissor obtido, gerando apenas mecanismos de seleção, é possível notar a perda de desempenho em instâncias maiores (64 a 100 aminoácidos). É questionável: se a falta de um critério de aceitação, melhor do que aceitar apenas soluções melhores ou iguais, pudesse guiar a busca afim de escapar dos mínimos locais.

6.2.4 Comparação com *framework* hiper heurístico GIHH

Com objetivo de comparar a melhor heurística de alto nível obtida pelo grupo de experimentos EGHyPDP-1 uma comparação com o *framework* GIHH proposta por [Misir, 2012] foi realizada. O GIHH obteve os melhores resultados para 6 domínios distintos de problemas contidos no *framework* HyFlex [Ochoa et al., 2012]. Misir disponibilizou o código fonte do GIHH e em seguida foram executados experimentos utilizando o problema PDP com o modelo HP-2D.

A Tabela 6.8 apresenta os melhores resultados encontrados pelos experimentos realizados com EGHyPDP, os resultados da execução do GIHH e por fim o melhor valor de energia conhecido. Os valores destacados com negrito são os melhores conhecidos. Tanto o EGHyPDP e o GIHH foram executados por 30 minutos. É possível verificar que em 7 instâncias de um total de 11 o EGHyPDP obteve os melhores valores conhecidos. Já o GIHH conseguiu obter os melhores valores apenas em 5 instâncias. Entretanto, se observamos as instâncias 7,9,10 e 11 o GIHH obteve valores mais próximos dos melhores do que o EGHyPDP. Dessa maneira, é possível afirmar que nas instâncias mais complexas o GIHH consegue chegar mais próximo dos melhores resultados conhecidos. No entanto, o EGHyPDP conseguiu encontrar os melhores valores em mais instâncias do que o GIHH.

6.2.5 Discussão

O EGHyPDP obteve êxito em gerar um mecanismo de seleção, que combinado com um critério de aceitação que aceita soluções melhores ou iguais, obteve os melhores resultados em 7 instâncias. Entretanto, nas instâncias mais complexas os resultados obtidos são bem inferiores em relação aos melhores valores conhecidos. Também foi constatado que o EGHyPDP conseguiu gerar pelo menos um bom mecanismo de seleção, já no caso dos critérios de aceitação gerados nenhum apresentou bons resultados. Um novo experimento foi realizado para avaliar o

comportamento dos critérios de aceitação. Este experimento consistiu em selecionar de maneira aleatória 10 indivíduos gerados pelo experimento EGHyPDP-2 e executá-los novamente para analisar o comportamento de cada um dos 10 critérios de aceitação. De 10 indivíduos 7, estavam aceitando apenas soluções melhores ou iguais, da mesma maneira que o critério fixo que foi utilizado no primeiro experimento EGHyPDP-1. A única diferença entre os indivíduos gerados e o critério fixo: é que os indivíduos são mais lentos que o fixo pois é necessário executar uma expressão aritmética enquanto com o critério fixo, um simples *if* é avaliado. Mas com respeito ao comportamento os critérios de aceitação são iguais. Outros 2 indivíduos dos 10 selecionados estavam aceitando qualquer solução pior do que a solução corrente da mesma maneira que o critério "todos movimentos" descrito por [Burke et al., 2013]. Finalmente, um indivíduo não estava aceitando nunca soluções piores da mesma maneira que o critério "apenas melhorias" [Burke et al., 2013].

Estes experimentos mostraram que o EGHyPDP conseguiu várias vezes gerar um critério de aceitação com exatamente o mesmo comportamento que critérios de aceitação desenvolvidos por pesquisadores. Entretanto, estes critérios são muito simples e não são suficientes para possibilitar que a busca não fique travada em mínimos locais. Levando em consideração que os experimentos envolvendo geração de critérios de aceitação (EGHyPDP-2 e EGHyPDP-3) não obtiveram resultados competitivos.

6.2.6 Conclusão dos experimentos utilizando o EGHyPDP

Esta seção apresentou os experimentos realizados com EGHyPDP, uma ferramenta automática para gerar heurísticas de alto nível para um *framework* hiper heurístico para resolver o problema PDP com o modelo HP-2D. O PDP é um problema muito desafiador com uma grande quantidade de mínimos locais e um espaço de busca complexo.

Muitos estudos exploraram o PDP com modelos simplificados, tais como modelo HP-2D, utilizando métodos heurísticos. Embora, é comum que estratégias heurísticas não consigam obter os melhores resultados conhecidos no caso das maiores instâncias. Geralmente, os *frameworks* hiper heurísticos se adaptam bem a este tipo de contexto complexo. Consequentemente, o objetivo desta abordagem foi gerar mecanismos de seleção e critérios de aceitação, através de um processo de evolução gramatical, de um *framework* hiper heurístico. Tal *framework* foi utilizado para resolver o PDP utilizando o modelo HP-2D. O desempenho e comportamento das heurísticas de alto nível geradas foi avaliado executando o *framework* utilizando um conjunto de 11 instâncias. Três grupos de experimentos foram desenvolvidos: o primeiro grupo apenas gerou mecanismos de seleção e utilizou um critério de aceitação "melhor ou igual"; o segundo gerou apenas critérios de aceitação. E no caso do mecanismo de seleção foi utilizado o melhor gerado pelo primeiro grupo; O terceiro e último grupo gerou tanto mecanismos de seleção e critérios de aceitação em paralelo. Os resultados obtidos pelos experimentos demonstram que a melhor heurística de alto nível foi encontrada no primeiro grupo. Já os resultados obtidos com

o segundo e terceiro grupo não são expressivos. Analisando, o melhor resultado obtido pelo primeiro experimento (um mecanismo de seleção e o critério de aceitação "melhor ou igual") é possível notar que de 11 instâncias em 7 casos foram encontrados os melhores resultados conhecidos. Entretanto, nas maiores instâncias (7,9,10 e 11) os resultados foram inferiores aos melhores conhecidos. O melhor indivíduo gerado também foi comparado com o GIHH, um *framework* hiper heurístico do estado da arte, os resultados obtidos por ambos são ligeiramente próximos. Em 5 instâncias o GIHH conseguiu obter os melhores resultados conhecidos. No caso das instâncias 7,9,10 e 11 o GIHH não conseguiu encontrar os melhores resultados, mas encontrou resultados superiores ao EGHYPDP. Isto demonstra que é possível automatizar o *design* de heurísticas de alto nível. E obter resultados próximos aos de uma estratégia hiper heurística com capacidade de obter bons resultados, em outros domínios de problema.

Outra constatação deste estudo foi o comportamento dos critérios de aceitação gerados. Foi notado que uma parte dos critérios de aceitação gerados se comportavam da mesma maneira que o critério "melhor ou igual". Também foram gerados critérios que sempre aceitam e que nunca aceitam soluções piores, da mesma maneira que os critérios "todos os movimentos" e "apenas melhorias" respectivamente. Dessa forma, foi possível constatar que os critérios gerados se comportam da mesma maneira que alguns critérios simples, mas que foram desenvolvidos por cientistas. Entretanto, o EGHYPDP não foi capaz de gerar critérios de aceitação que possibilitassem escapar dos mínimos locais de um espaço de busca complexo como o do PDP.

Capítulo 7

Conclusão

Nesta dissertação foram apresentadas duas abordagens heurísticas para o problema PDP utilizando o modelo simplificado HP-2D.

A primeira abordagem consistiu em uma formulação multi objetiva para o PDP. Um novo objetivo foi adicionado o qual avalia o grau de compactação das estruturas de proteínas. Dessa maneira, considerando o valor da energia e o quão compacta uma dada estrutura é. AEMOs foram utilizados para explorar o espaço de busca no contexto multi objetivo. Os algoritmos utilizados foram o NSGAI [Deb et al., 2002a] e o IBEA [Zitzler e Künzli, 2004]. Uma versão modificada desses algoritmos, com objetivo de obter melhores resultados, também foi proposta. Os algoritmos M_NSGAI e o M_IBEA são versões adaptadas utilizando uma inicialização via *backtracking* e um *pool* de operadores. Portanto, 4 versões de AEMOs foram utilizadas para tentar resolver o PDP com o modelo HP-2D. Apenas a versão M_IBEA conseguiu obter resultados expressivos em 7 instâncias de 11. Isto, demonstra que apenas a formulação multi objetiva não é suficiente para explorar o espaço de busca de maneira adequada.

A segunda abordagem, denominada EGHYPDP, consistiu em gerar as heurísticas de alto nível para um *framework* hiper heurístico, para o PDP utilizando o modelo HP-2D, através da evolução gramatical. Três grupos de experimentos foram definidos o primeiro gerando apenas mecanismos de seleção; o segundo gerando apenas critérios de aceitação e o terceiro gerando ambos em paralelo. Analisando os resultados obtidos por cada grupo de experimento foi possível perceber que apenas o primeiro grupo de experimentos conseguiu obter resultados expressivos para 7 instâncias de 11. Os outros dois grupos de experimentos conseguiram obter bons resultados apenas para duas instâncias. Dessa maneira, demonstrando que os critérios de aceitação gerados não são suficientes para guiar a busca para regiões promissoras.

Infelizmente nenhuma das abordagens propostas nesta dissertação conseguiu obter os melhores resultados conhecidos para as maiores instâncias. Contudo, ambas as abordagens conseguiram obter bons resultados para 7 instâncias. Tanto o algoritmo M_IBEA e a melhor heurística de alto nível gerada pelo EGHYPDP obtiveram os melhores resultados nas mesmas instâncias. Já nas instâncias mais complexas, nas quais não foi possível encontrar o melhor

resultado conhecido, o M_IBEA obteve resultados superiores ao EGHyPDP. Porém com diferença estatística segundo o teste de Kruskal-Wallis [McKight e Najab, 2010] em apenas uma instância.

O PDP é um problema, em aberto, extremamente desafiador por conta da enorme quantidade de possíveis soluções contidas no espaço de busca. Dessa maneira, motivando muitos pesquisadores buscarem métodos mais elegantes e robustos para que estratégias heurísticas consigam explorar o espaço de busca de maneira apropriada.

Referências Bibliográficas

- [Anfinsen, 1972] Anfinsen, C. B. (1972). Studies on the principles that govern the folding of protein chains.
- [Anfinsen et al., 1961] Anfinsen, C. B., Haber, E., Sela, M. e White Jr, F. (1961). The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences of the United States of America*, 47(9):1309.
- [Atkins e Hart, 1999] Atkins, J. e Hart, W. E. (1999). On the intractability of protein folding with a finite alphabet of amino acids. *Algorithmica*, 25(2-3):279–294.
- [Barr, 1998] Barr, N. A. (1998). *The economics of the welfare state*. Stanford University Press.
- [Bastolla et al., 1997] Bastolla, U., Frauenkron, H., Gerstner, E., Grassberger, P. e Nadler, W. (1997). Testing a new monte carlo algorithm for protein folding. *arXiv preprint cond-mat/9710030*.
- [Baudrillart, 1872] Baudrillart, H. J. L. (1872). *Manuel d'économie politique*. Guillaumin et cie.
- [Bell et al., 2002] Bell, S., Klein, C., Müller, L., Hansen, S. e Buchner, J. (2002). p53 contains large unstructured regions in its native state. *Journal of molecular biology*, 322(5):917–927.
- [Benítez, 2010] Benítez, C. M. V. (2010). Um algoritmo genético paralelo para o problema de dobramento de proteínas utilizando o modelo 3dhp com cadeia lateral.
- [Berger e Leighton, 1998] Berger, B. e Leighton, T. (1998). Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete. *Journal of Computational Biology*, 5(1):27–40.
- [Blum et al., 2011] Blum, C., Puchinger, J., Raidl, G. R. e Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151.
- [Bornberg-Bauer, 1997] Bornberg-Bauer, E. (1997). Chain growth algorithms for hp-type lattice proteins. Em *Proceedings of the first annual international conference on Computational molecular biology*, páginas 47–55. ACM.

- [Branden et al., 1999] Branden, C. I. et al. (1999). *Introduction to protein structure*. Garland Science.
- [Burke et al., 2013] Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. e Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.
- [Burke et al., 2009] Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E. e Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. Em *Computational intelligence*, páginas 177–201. Springer.
- [Chakhlevitch e Cowling, 2008] Chakhlevitch, K. e Cowling, P. (2008). Hyperheuristics: recent developments. Em *Adaptive and multilevel metaheuristics*, páginas 3–29. Springer.
- [Cox et al., 2013] Cox, M. M. et al. (2013). *Lehninger principles of biochemistry*. Freeman.
- [Crescenzi et al., 1998] Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A. e Yannakakis, M. (1998). On the complexity of protein folding. *Journal of computational biology*, 5(3):423–465.
- [Custódio et al., 2004] Custódio, F. L., Barbosa, H. J. e Dardenne, L. E. (2004). Investigation of the three-dimensional lattice hp protein folding model using a genetic algorithm. *Genetics and Molecular Biology*, 27(4):611–615.
- [Custódio et al., 2014] Custódio, F. L., Barbosa, H. J. e Dardenne, L. E. (2014). A multiple minima genetic algorithm for protein structure prediction. *Applied Soft Computing*, 15:88–99.
- [Dawson et al., 2003] Dawson, R., Müller, L., Dehner, A., Klein, C., Kessler, H. e Buchner, J. (2003). The n-terminal domain of p53 is natively unfolded. *Journal of molecular biology*, 332(5):1131–1141.
- [Deb et al., 2002a] Deb, K., Pratap, A., Agarwal, S. e Meyarivan, T. (2002a). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- [Deb et al., 2002b] Deb, K., Pratap, A., Agarwal, S. e Meyarivan, T. (2002b). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.
- [Devlin e Toma, 1998] Devlin, T. M. e Toma, L. (1998). *Manual de bioquímica: com correlações clínicas*.
- [Eswar et al., 2003] Eswar, N., John, B., Mirkovic, N., Fiser, A., Ilyin, V. A., Pieper, U., Stuart, A. C., Marti-Renom, M. A., Madhusudhan, M. S., Yerkovich, B. et al. (2003). Tools for

- comparative protein structure modeling and analysis. *Nucleic acids research*, 31(13):3375–3380.
- [Ferreira, 2006] Ferreira, C. (2006). *Gene expression programming: mathematical modeling by an artificial intelligence*, volume 21. Springer.
- [Figueiredo et al., 2013] Figueiredo, E. M. d. N., Ludermir, T. B. O. e Bastos Filho, C. J. A. C. (2013). Algoritmo baseado em enxame de partículas para otimização de problemas com muitos objetivos. *Repositório Institucional da UFPE*.
- [Fonseca et al., 2005] Fonseca, C. M., Knowles, J. D., Thiele, L. e Zitzler, E. (2005). A tutorial on the performance assessment of stochastic multiobjective optimizers. Em *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 216, página 240.
- [Gabriel et al., 2012] Gabriel, P. H., Melo, V. V. d., Delbem, A. C. et al. (2012). Algoritmos evolutivos e modelo hp para predição de estruturas de proteínas. *Revista de Controle e Automação*, 23(1):25–37.
- [Garza-Fabre et al., 2012] Garza-Fabre, M., Toscano-Pulido, G. e Rodriguez-Tello, E. (2012). Locality-based multiobjectivization for the hp model of protein structure prediction. Em *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, páginas 473–480. ACM.
- [Göbl e Tjandra, 2012] Göbl, C. e Tjandra, N. (2012). Application of solution nmr spectroscopy to study protein dynamics. *Entropy*, 14(3):581–598.
- [Grantcharova et al., 2001] Grantcharova, V., Alm, E. J., Baker, D. e Horwich, A. L. (2001). Mechanisms of protein folding. *Current opinion in structural biology*, 11(1):70–82.
- [Hamming, 1950] Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160.
- [Hart e Istrail, 1997] Hart, W. E. e Istrail, S. (1997). Lattice and off-lattice side chain models of protein folding: linear time structure prediction better than 86% of optimal. *Journal of Computational Biology*, 4(3):241–259.
- [Hsu et al., 2003] Hsu, H.-P., Mehra, V., Nadler, W. e Grassberger, P. (2003). Growth algorithms for lattice heteropolymers at low temperatures. *The Journal of chemical physics*, 118(1):444–451.
- [Hutton et al., 2001] Hutton, M., Lewis, J., Dickson, D., Yen, S.-H. e McGowan, E. (2001). Analysis of tauopathies with transgenic mice. *Trends in molecular medicine*, 7(10):467–470.

- [Ilari e Savino, 2008] Ilari, A. e Savino, C. (2008). Protein structure determination by x-ray crystallography. Em *Bioinformatics*, páginas 63–87. Springer.
- [Ishibuchi et al., 2008] Ishibuchi, H., Tsukamoto, N. e Nojima, Y. (2008). Evolutionary many-objective optimization. Em *Genetic and Evolving Systems, 2008. GEFS 2008. 3rd International Workshop on*, páginas 47–52. IEEE.
- [Ishimaru et al., 2003] Ishimaru, D., Andrade, L. R., Teixeira, L. S., Quesado, P. A., Maiolino, L. M., Lopez, P. M., Cordeiro, Y., Costa, L. T., Heckl, W. M., Weissmüller, G. et al. (2003). Fibrillar aggregates of the tumor suppressor p53 core domain. *Biochemistry*, 42(30):9022–9027.
- [Krasnogor et al., 2002] Krasnogor, N., Blackburne, B., Burke, E. K. e Hirst, J. D. (2002). Multimeme algorithms for protein structure prediction. Em *Parallel Problem Solving from Nature PPSN VII*, páginas 769–778. Springer.
- [Krasnogor et al., 1999] Krasnogor, N., Hart, W., Smith, J. e Pelta, D. (1999). Protein structure prediction with evolutionary algorithms.
- [Krieger et al., 2009] Krieger, E., Joo, K., Lee, J., Lee, J., Raman, S., Thompson, J., Tyka, M., Baker, D. e Karplus, K. (2009). Improving physical realism, stereochemistry, and side-chain accuracy in homology modeling: four approaches that performed well in casp8. *Proteins: Structure, Function, and Bioinformatics*, 77(S9):114–122.
- [Lau e Dill, 1989] Lau, K. F. e Dill, K. A. (1989). A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997.
- [Lin e Su, 2011] Lin, C.-J. e Su, S.-C. (2011). Protein 3 d hp model folding simulation using a hybrid of genetic algorithm and particle swarm optimization. *International Journal of Fuzzy Systems*, 13(2):140–147.
- [Lodish et al., 2000] Lodish, H. F., Berk, A., Zipursky, S. L., Matsudaira, P., Baltimore, D., Darnell, J. et al. (2000). *Molecular cell biology*, volume 4. Citeseer.
- [Lopes, 2008] Lopes, H. S. (2008). Evolutionary algorithms for the protein folding problem: A review and current trends. Em *Computational intelligence in biomedicine and bioinformatics*, páginas 297–315. Springer.
- [Lourenço et al., 2012] Lourenço, N., Pereira, F. e Costa, E. (2012). Evolving evolutionary algorithms. Em *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, páginas 51–58. ACM.
- [McKight e Najab, 2010] McKight, P. E. e Najab, J. (2010). Kruskal-Wallis test. *Corsini Encyclopedia of Psychology*.

- [McNaught et al., 2001] McNaught, K. S. P., Olanow, C. W., Halliwell, B., Isacson, O. e Jenner, P. (2001). Failure of the ubiquitin–proteasome system in parkinson’s disease. *Nature Reviews Neuroscience*, 2(8):589–594.
- [Misir, 2012] Misir, M. (2012). Intelligent hyper-heuristics: a tool for solving generic optimisation problems.
- [Ngo et al., 1994] Ngo, J., Marks, J. e Karplus, M. (1994). The protein folding problem and tertiary structure prediction. *Merz and LeGrand, ed. Birkhauser, Boston*.
- [Nicosia e Stracquadanio, 2008] Nicosia, G. e Stracquadanio, G. (2008). Generalized pattern search algorithm for peptide structure prediction. *Biophysical journal*, 95(10):4988–4999.
- [Ochoa et al., 2012] Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A. J., Petrovic, S. et al. (2012). Hyflex: A benchmark framework for cross-domain heuristic search. Em *Evolutionary computation in combinatorial optimization*, páginas 136–147. Springer.
- [Pacheco, 1999] Pacheco, M. A. C. (1999). Algoritmos genéticos: princípios e aplicações. *ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro. Fonte desconhecida*.
- [Pedersen, 2000] Pedersen, C. N. (2000). *Algorithms in computational biology*. BRICS.
- [Qian et al., 2004] Qian, B., Ortiz, A. R. e Baker, D. (2004). Improvement of comparative model accuracy by free-energy optimization along principal components of natural structural variation. *Proceedings of the National Academy of Sciences of the United States of America*, 101(43):15346–15351.
- [Röthlisberger et al., 2008] Röthlisberger, D., Khersonsky, O., Wollacott, A. M., Jiang, L., DeChancie, J., Betker, J., Gallaher, J. L., Althoff, E. A., Zanghellini, A., Dym, O. et al. (2008). Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195.
- [Ryan et al., 1998] Ryan, C., Collins, J. e Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. Em *Genetic Programming*, páginas 83–96. Springer.
- [Sabar et al., 2015] Sabar, N. R., Ayob, M., Kendall, G. e Qu, R. (2015). Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 19(3):309–325.
- [Santana et al., 2008] Santana, R., Larrañaga, P., Lozano, J. et al. (2008). Protein folding in simplified models with estimation of distribution algorithms. *Evolutionary Computation, IEEE Transactions on*, 12(4):418–438.

- [Sant'Anna, 2008] Sant'Anna, C. N. (2008). *On the Modularity of Aspect-Oriented Design : A Concern-Driven Measurement Approach*. Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ.
- [Sela et al., 1957] Sela, M., White Jr, F. H. e Anfinsen, C. B. (1957). Reductive cleavage of disulfide bridges in ribonuclease. *Science*, 125(3250):691–692.
- [Selkoe, 2001] Selkoe, D. J. (2001). Clearing the brain's amyloid cobwebs. *Neuron*, 32(2):177–180.
- [Shakhnovich e Gutin, 1993] Shakhnovich, E. I. e Gutin, A. M. (1993). Engineering of stable and fast-folding sequences of model proteins. *Proceedings of the National Academy of Sciences*, 90(15):7195–7199.
- [Shen et al., 2009] Shen, Y., Vernon, R., Baker, D. e Bax, A. (2009). De novo protein structure generation from incomplete chemical shift assignments. *Journal of biomolecular NMR*, 43(2):63–78.
- [Shmygelska et al., 2002] Shmygelska, A., Aguirre-Hernandez, R. e Hoos, H. H. (2002). An ant colony optimization algorithm for the 2d hp protein folding problem. Em *Ant Algorithms*, páginas 40–52. Springer.
- [Shmygelska e Hoos, 2003] Shmygelska, A. e Hoos, H. H. (2003). An improved ant colony optimisation algorithm for the 2d hp protein folding problem. Em *Advances in Artificial Intelligence*, páginas 400–417. Springer.
- [Stillinger et al., 1993] Stillinger, F. H., Head-Gordon, T. e Hirshfeld, C. L. (1993). Toy model for protein folding. *Physical review E*, 48(2):1469.
- [Suzuki et al., 1986] Suzuki, D. T., Griffiths, A. J., Miller, J. H., Lewontin, R. C. et al. (1986). *An introduction to genetic analysis*. Número Ed. 3. WH Freeman and Company.
- [Talbi e Bachelet, 2006] Talbi, E.-G. e Bachelet, V. (2006). Cosearch: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22.
- [Thomas e Dill, 1993] Thomas, P. D. e Dill, K. A. (1993). Local and nonlocal interactions in globular proteins and mechanisms of alcohol denaturation. *Protein Science*, 2(12):2050–2065.
- [Thomas et al., 1992] Thomas, P. J., Ko, Y. H. e Pedersen, P. L. (1992). Altered protein folding may be the molecular basis of most cases of cystic fibrosis. *FEBS letters*, 312(1):7–9.
- [Unger e Moult, 1993a] Unger, R. e Moult, J. (1993a). Finding the lowest free energy conformation of a protein is an np-hard problem: proof and implications. *Bulletin of Mathematical Biology*, 55(6):1183–1198.

- [Unger e Moult, 1993b] Unger, R. e Moult, J. (1993b). Genetic algorithms for protein folding simulations. *Journal of molecular biology*, 231(1):75–81.
- [Ursini et al., 2002] Ursini, F., Davies, K. J., Maiorino, M., Parasassi, T. e Sevanian, A. (2002). Atherosclerosis: another protein misfolding disease? *Trends in Molecular Medicine*, 8(8):370–374.
- [Vendruscolo et al., 2000] Vendruscolo, M., Najmanovich, R. e Domany, E. (2000). Can a pairwise contact potential stabilize native protein folds against decoys obtained by threading? *Proteins: Structure, Function, and Bioinformatics*, 38(2):134–148.
- [Wang et al., 2012] Wang, L., Althoff, E. A., Bolduc, J., Jiang, L., Moody, J., Lassila, J. K., Giger, L., Hilvert, D., Stoddard, B. e Baker, D. (2012). Structural analyses of covalent enzyme–substrate analog complexes reveal strengths and limitations of de novo enzyme design. *Journal of molecular biology*, 415(3):615–625.
- [Zitzler e Künzli, 2004] Zitzler, E. e Künzli, S. (2004). Indicator-based selection in multiobjective search. Em *Parallel Problem Solving from Nature-PPSN VIII*, páginas 832–842. Springer.
- [Zitzler et al., 2003] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. e Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132.