Algoritmo Evolutivo com Método de Correção de Infactibilidade para o Problema de Estruturas de Proteínas.

Guilherme Pacheco de Oliveira[1], **Christiane Regina Soares Brasil**[1]

[1]Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Uberlândia– MG – Brasil

{guilhermepo2,christiane}@ufu.br

Abstract. *The protein structures prediction is one of the most thought-provoking combinatorial problems in the real world, being classified computationally as a NP problem. The objective of this work was to analyze an Evolutionary Algorithm for prediction of 3D structures of proteins with backtracking correction of infeasible solutions, using the HP-3D Model to represent the structures. The analysis showed that the results were coherent to the literature, proving to be competitive.*

Resumo. *A predição de estruturas de proteínas é um dos problemas combinatórios mais instigantes do mundo real, sendo classificado computacionalmente como um problema NP. O objetivo desse trabalho foi analisar um Algoritmo Evolutivo para predição de estruturas 3D de proteínas com correção por* backtracking *de soluções infactíveis, usando o Modelo HP-3D para representar as estruturas. A análise mostrou que os resultados foram coerentes à literatura, revelando-se competitivo.*

1. Introdução

Na grande área de Ciência da Computação, há uma subárea que trabalha com problemas combinatórios complexos, que se refere a uma classe de problemas NP, isto é, com "tempo polinomial não determinístico". Neste contexto, temos os seguintes problemas computacionais: do caixeiro viajante, da mochila, torre de Hanoi, algoritmos em grafos como cobertura de vértices, ciclo hamiltoniano, entre outros [Cormen et al. 2010].

No mundo real, um famoso desafio da ciência há décadas classificado como NP é o Problema de Predição de Estruturas de Proteínas (PSP, do inglês *Protein Structure Prediction*) [Crescenzi et al. 1998] [Cox and Doudna 2012]. Sabe-se que a estrutura proteica fornece valiosas informações quanto à sua função em um organismo vivo. Deste modo, determinar sua funcionalidade pode colaborar efetivamente na investigação de doenças, bem como na criação de novos fármacos. Portanto, o processo de predizer uma estrutura é fundamental para o avanço da ciência. No entanto, a maioria das proteínas não possui estrutura conhecida, pois os processos convencionais de predição ainda são ineficientes (ressonância nuclear magnética e cristalografia) por serem caros, lentos e apresentarem limitações em relação ao tamanho da proteína.

Deste modo, diversas técnicas de otimização computacional são aplicadas ao problema de PSP [Simons et al. 1999] [Holley and Karplus 1999] [Custódio et al. 2004] [Shmygelska and Hoos 2005] [Lin and Su 2011] [Brasil et al. 2013] . Neste trabalho,

foram abordados os Algoritmos Evolutivos (AEs) [Goldberg 1989] que são bastante utilizados, pela simplicidade da implementação e pelos bons resultados.

O objetivo deste trabalho foi desenvolver um AE para o problema de PSP, chamado AE_{BACK}, com um modelo simplificado HP-3D [Lau and Dill 1989] aplicando um método de correção de soluções infactíveis. Há muitos pesquisadores que têm trabalhado com procedimentos com estes fins [Cotta 2003] [Johnson and Katikireddy 2006][Gabriel et al. 2012], seja por *backtracking*, penalidade ou outra técnica, reduzindo, desta maneira, o espaço de busca para somente regiões com soluções válidas. Os resultados obtidos foram comparados com trabalhos relevantes da literatura [Cotta 2003][Lin and Su 2011].

2. Algoritmos Evolutivos

2.1. Breve introdução sobre Algoritmos Evolutivos

Os algoritmos evolutivos (AEs) são técnicas de otimização computacional criadas por Goldberg [Goldberg 1989], as quais foram baseadas na Teoria da Evolução de Darwin [Darwin 1859]. Segundo essa teoria, uma população é capaz de evoluir a partir da sobrevivência dos indivíduos mais aptos. No contexto computacional, sua implementação é simples e de fácil implementação, além de apresentar ao final não apenas uma solução, mas um conjunto de soluções.

Esses algoritmos possuem uma função objetivo, que deve ser minimizada ou maximizada, dependendo do problema. Primeiramente, é gerado um conjunto aleatório de soluções, que corresponde à população inicial, onde se aplica a função objetivo à cada solução, a fim de medir quão boa é, atribuindo-lhes um valor de aptidão chamado *fitness*. Com base no *fitness*, as melhores soluções são escolhidas para a geração seguinte. Por meio de operadores de reprodução, novas soluções são geradas e competem com as antigas. Ao final de cada geração, as soluções com os melhores *fitness* permanecem na população, ocasionando uma evolução. Esse processo é repetido até que a condição de parada seja satisfeita (número de gerações ou *fitness* alcançado). Uma ilustração desse processo pode ser visto na Figura 1.

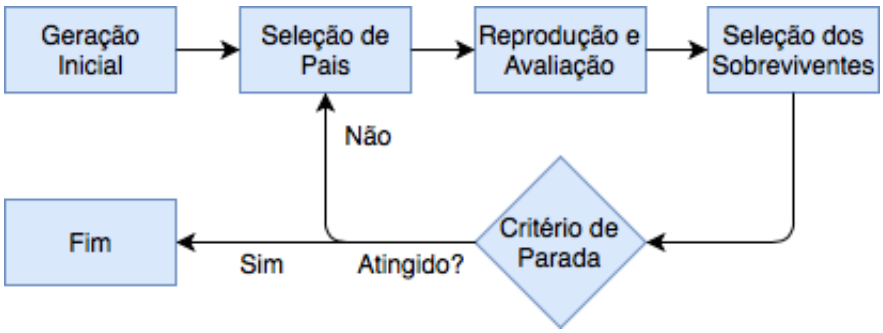

Figure 1. Fluxograma do Algoritmo Evolutivo.

O operador de recombinação (*crossover*) gera novos indivíduos pela troca de informações de dois ou mais indivíduos [Gabriel and Delbem 2008], havendo, portanto, a combinação de características dos pais. O tipo de recombinação mais simples é a recombinação de um ponto. Nesta, seleciona-se um ponto aleatório igual para os dois

pais, então são gerados dois filhos, cada um com a parte à esquerda de um pai e à direita de outro pai. A recombinação de um ponto pode ser generalizada para n pontos, sorteando n cortes aleatórios nos pais e criando seções nos indivíduos.

O operador de mutação modifica aleatoriamente um ou mais posições de um indivíduo (lembrando que este pode ser representado por um vetor). Um exemplo de mutação ocorre com a seleção de uma posição de modo aleatório no indivíduo e, em seguida, a troca deste pelo conteúdo de outra posição. Por exemplo, usando uma codificação binária, caso um gene fosse 0, passaria a ser 1. Essa operação acontece com uma probabilidade chamada de taxa de mutação, que geralmente é baixa.

A seguir, será descrito o AE com método de correção de soluções infactíveis.

2.2. Algoritmo Evolutivo com Correção por *Backtracking*

2.2.1. O Modelo HP e o cálculo do *Fitness*

No algoritmo evolutivo desenvolvido, a representação das soluções é feita com uma malha (lattice) tridimensional chamada de modelo simplificado HP-3D [Lau and Dill 1989]. Este modelo é baseado no efeito hidrofóbico dos aminoácidos, ou seja, na capacidade que o aminoácido tem de repelir (ou não) a água. Neste contexto, os aminoácidos são classificados em hidrofóbicos (H) e hidrofílicos (P), também conhecidos como polares, como pode ser observado na Figura 2.

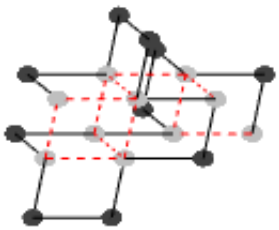

Figure 2. Uma proteína representada no modelo HP-3D, obtida pela execução do programa implementado. Em tons escuros são aminoácidos polares (P), em tons claros, hidrofóbicos (H). As linhas pretas são as ligações entre aminoácidos conectados, e as vermelhas tracejadas são as interações hidrofóbicas.

O Modelo HP possui uma fórmula específica para calcular o *fitness* (aptidão) de uma dada proteína representada neste modelo. Essa energia é inversamente proporcional à quantidade de interações H-H, uma vez que a função objetivo do PSP busca a menor energia livre possível. A energia de conformação é dada pela seguinte fórmula:

$$E = \beta_{i,j} \sum \delta(r_i, rj)$$

Onde:

- β é 1 se os aminoácidos forem do tipo H, e 0, caso contrário.
- A função δ assume 1 se os aminoácidos r_i e r_j são vizinhos não conectados, e 0, caso contrário.

Existem possibilidades de representação de um indivíduo no modelo HP, como coordenadas cartesianas [Unger and Moult 1993] ou internas [Cotta 2003]. Mais especificamente, a representação por coordenadas internas pode ser absoluta ou relativa.

A codificação absoluta adota um sistema de referência absoluta e os indivíduos são representados como movimentos em relação a esse sistema. Por exemplo, em um *lattice* cúbico com esse sistema, existem seis movimentos possíveis: norte, sul, leste, oeste, cima e baixo. Em contrapartida, na codificação relativa o sistema adotado não é fixo, logo, ele depende do último movimento realizado. Como exemplo, novamente em um *lattice* cúbico, com o sistema de coordenadas relativas existem cinco movimentos possíveis: frente, direita, esquerda, cima e baixo.

Neste trabalho foi adotada a representação por coordenadas internas relativas.

2.2.2. O Algoritmo de correção para soluções infactíveis

Quando uma solução está sendo gerada por um algoritmo, pode ocorrer uma colisão, isto é, quando um aminoácido ocupa a mesma posição de outro. Há duas formas intuitivas de lidar com o problema de colisões: uma é trabalhar com formas de penalidade [Gabriel and Delbem 2008] e a outra com algoritmos de correção [Cotta 2003] [Johnson and Katikireddy 2006] [Gabriel et al. 2012].

Nesse estudo utilizou-se um algoritmo de correção por *backtracking* baseado no trabalho do Cotta [Cotta 2003]. O algoritmo de referência cria uma solução nova (estrutura nova), com a ajuda de operadores genéticos, que efetuam a correção das soluções infactíveis. O algoritmo de correção em nosso trabalho trata de um candidato inválido que tem seus movimentos alterados gradualmente enquanto ainda há colisão, até que se torne factível. A principal diferença é que este não usa operadores genéticos para a correções.

O procedimento de correção é descrito a seguir: primeiramente, é verificado se a nova solução sendo construída possui ou não colisões. Caso não haja colisões, é verificado se possui o mesmo tamanho da sequência inválida original; caso seja positivo, significa que o algoritmo de correção chegou ao final e a estrutura é uma solução factível. Se houver colisão, o último movimento é trocado pelo próximo movimento da lista de movimentos possíveis (é trocado o último pois da forma como o algoritmo foi implementado a colisão sempre será introduzida no último aminoácido que foi inserido). Deste modo, o algoritmo é executado recursivamente. Caso não haja mais movimentos possíveis, não há solução para aquele indivíduo específico, e o algoritmo termina sua execução, sendo esta solução imprópria eliminada da população. Caso o tamanho não seja o mesmo, o próximo movimento é adicionado à solução sendo construída, e em seguida, a lista de movimentos possíveis é atualizada, para conter todos os movimentos possíveis, exceto aquele que foi adicionado (diminuindo, portanto, gradativamente as possibilidades de movimentos), e o algoritmo volta a ser executado recursivamente. A validade do resultado é verificada e caso seja uma solução factível, esta toma o lugar da antiga.

3. Resultados e Discussões

Neste trabalho, os algoritmos foram implementados em linguagem C, no sistema operacional macOS X 10.11.6 e em um processador Intel Core i5 2.5GHz com 4GB de memória

1600MHz DDR3. Os experimentos foram realizados com os dados advindos dos trabalhos de referência [Cotta 2003][Lin and Su 2011].

O algoritmo desenvolvido neste trabalho, chamado AE_{BACK}, foi baseado no artigo de Cotta pelo fato de também utilizar um método de correção com *backtracking*, com a diferença de que o dele utiliza operadores genéticos para correção e o AE_{BACK} não aplica esses operadores. No entanto, ambos utilizam coordenadas internas relativas.

Foram realizadas 10 execuções para cada sequência nas configurações mostradas na Tabela 1.

Configuração	**AE_{BACK}**	**Cotta**
Sobrevivência	Elitismo	Elitismo
População	100	100
Taxa *Crossover*	90%	90%
Taxa Mutação	1%	1%
Iterações	100000	100000
Operadores *Crossover*	*Crossover* de um ponto (50%) e dois pontos (50%)	*Crossover* de um ponto
Operadores Mutação	Um ponto aleatório (50%) e dois pontos aleatórios (50%)	Mutação por *backtracking*
Correção	Utilizado para correção no *crossover* e na mutação	Utilizado para geração de indivíduos aleatórios, para mutação e para correção após *crossover*

Tabela 1: Configuração do primeiro experimento.

Tamanho	**Sequência**	**E***	**AE_{BACK}**	**Cotta**
20	$(HP)_2PH(HP)_2(PH)_2HP(PH)_2$	-11	-11	-11
24	$H_2P_2(HP_2)_6H_2$	-13	-13	-13
25	$P_2HP_2(H_2P_4)_3H_2$	-9	-9	-9
36	$P(P_2H_2)_2P_5H_5(H_2P_2)_2P_2H(HP_2)_2$	-18	-16	-16
48	$P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$	-29	-26	-26
50	$H_2(PH)_3PH_4PH(P_3H)_2P_4(HP_3)_2HPH_4(PH)_3PH_2$	-26	-24	-24
60	$P(PH_3)H_5P_3H_{10}PHP_3H_{12}P_4H_6PH_2PHP$	-49	-44	-43

Tabela 2: Resultados do primeiro experimento, onde E* é a energia mínima da sequência correspondente, e na quarta e quinta colunas são apresentadas as menores energias encontradas por AE_{BACK} e pelo AE do Cotta [Cotta 2003], respectivamente.

Na Tabela 2, pode-se observar que no primeiro experimento o AE_{BACK} e o AE do Cotta obtêm resultados equivalentes, com exceção para proteína maior (60 aminoácidos). Para melhorar estes resultados, foi executado outro conjunto de testes com uma nova configuração, uma vez que foi verificado que o AE_{BACK} convergiu antes das 100.000 iterações. Deste modo, a quantidade de iterações foi para 50.000 e o tamanho na população foi 200, para que se obtivesse maior diversidade na população. A fim de evitar a estagnação em ótimos locais, a taxa de mutação foi aumentada para 5% e a taxa de *crossover* foi reduzida para 80% para que pudessem ser preservadas mais soluções boas, o que foi observado empiricamente. Os novos resultados foram mostrados na Tabela 3.

Tamanho	**E***	**AE$_{BACK}$**	**Cotta**	**Cheng-Jian Lin e Shih-Chieh Su**
20	-11	-11	-11	-11
24	-13	-13	-13	-13
25	-9	-9	-9	-9
36	-18	-18	-16	-18
48	-29	-27	-26	-29
50	-26	-25	-24	-26
60	-49	-47	-43	-49

Tabela 3: Resultados do segundo experimento.

Com essa nova configuração, o AE$_{BACK}$ alcançou resultados mais promissores que Cotta para as proteínas maiores que 25 aminoácidos, mostrando que com um ajuste adequado de parâmetros é possível melhorar o desempenho de um AE de modo significativo [Goldberg 1989]. Na Tabela 3 também foi mostrado os resultados obtidos no trabalho de Cheng-Jian Lin e Shih-Chieh Su [Lin and Su 2011], o qual apresenta uma abordagem híbrida de um AE com PSO (*Particle Swarm Optimization*). Neste sentido, notam-se resultados animadores do AE$_{BACK}$, uma vez que, diferente de um AE híbrido [Lin and Su 2011], este não tem nenhuma técnica de otimização computacional combinada ao AE convencional.

Por fim, foi realizado um terceiro experimento a fim de comparar o AE com correção por *backtracking* (AE$_{BACK}$) com o AE sem *backtracking* (AE$_{NOBACK}$), apenas com uma função de penalidade atribuída ao candidato com colisão. Os parâmetros dos testes foram os mesmos utilizados na Tabela 1. A função de *fitness* do candidato com a penalidade é baseada no trabalho de P. Gabriel [Gabriel and Delbem 2008], sendo: *FitnessP* = $colisoes$, onde $colisoes$ é a quantidade de colisões encontradas no indivíduo.

As execuções foram realizadas em duas sequências (com 20 e 36 aminoácidos) da Tabela 2, denominadas S20 e S36 respectivamente. Foram escolhidas essas proteínas pois representam uma instância pequena e outra mediana do conjunto de teste usado, para dar uma ideia geral do comportamento dos AEs implementados. Os resultados da execução com AE$_{BACK}$ e AE$_{NOBACK}$ estão nas Tabelas 4 e 5. A coluna Média do Melhor *Fitness* apresenta a média da menor energia encontrada pelo AE em dez execuções, a coluna Desvio Padrão apresenta o desvio padrão das melhores energias de cada execução, a coluna de Menor Energia indica a menor energia encontrada em todas as execuções e a coluna Média do Tempo mostra a média da quantidade de segundos despendidos nas dez execuções.

Algoritmo	**Média do Melhor *Fitness***	**Desvio Padrão**	**Menor Energia**	**Média do Tempo (s)**
AE$_{BACK}$	-10.5	0.7	-11	5745.150
AE$_{NOBACK}$	-9.5	0.7	-11	3431.332

Tabela 4: Resultados do terceiro experimento para a sequência S20.

Algoritmo	**Média do Melhor *Fitness***	**Desvio Padrão**	**Menor Energia**	**Média do Tempo (s)**
AE$_{BACK}$	-14.8	1.03	-16	35279.8
AE$_{NOBACK}$	-11.4	1.26	-13	15518.1

Tabela 5: Resultados do terceiro experimento para a sequência S36.

De acordo com as tabelas 4 e 5, observamos que o AE sem *backtracking* é mais rápido que o AE com *backtracking*. Sobre os valores das energias, o AE_{BACK} obteve resultados melhores, no sentido de que o desvio padrão é igual ou menor, indicando uma menor diversidade nas melhores energias obtidas, e pela média, mostrou-se que esses valores são mais próximos do resultado ótimo.

4. Conclusões

O principal objetivo deste trabalho foi analisar o efeito de um procedimento de correção por *backtracking* em um AE aplicado para PSP, usando o modelo HP-3D. Comparando o AE_{BACK} à literatura [Cotta 2003], no primeiro experimento os resultados foram equivalentes, exceto pela proteína maior cuja energia mínima obtida foi melhor com o AE_{BACK}.

No segundo experimento houve um ajuste de parâmetros, conforme observações empíricas, tornando-o mais eficiente. Isso confirma o fato que ajustar adequadamente os parâmetros em AEs é uma etapa fundamental para a obtenção de melhores resultados, uma vez que evidências mostram que 10% do tempo dedicado está no desenvolvimento de um Algoritmo Evolutivo, e os 90% restantes são gastos no processo de ajuste de parâmetros adequados [Coy and et al. 2000] [Gallagher and Yuan 2007]. Deste modo, foi possível superar os resultados de Cotta [Cotta 2003] para as quatro maiores proteínas sob o ponto de vista das energias. Neste experimento também pode-se verificar que o AE_{BACK} alcançou valores de energia promissores, por se tratar de um AE puro, ou seja, sem acréscimo de nenhuma outra técnica de otimização, em comparação ao trabalho de Cheng-Jian Lin e Shih-Chieh Su [Lin and Su 2011], que é um AE híbrido.

No terceiro experimento, o método de correção com *backtracking* no AE_{BACK} mostrou-se eficaz em relação à robustez do algoritmo e às menores energias alcançadas, apesar de ser mais custoso computacionalmente do que o AE_{NOBACK}. Deste modo, tem-se claro uma questão inerente aos algoritmos evolutivos, onde se deve ponderar o quanto se almeja alcançar o melhor *fitness* possível, e o quanto se deseja obter resultados satisfatórios, mesmo que não sejam os ótimos, mas em um tempo viável.

Baseado nessas limitações, novos passos são considerados para trabalhos futuros, como: (i) implementar um procedimento para ajuste automático de parâmetros; (ii) desenvolver um método de correção que alterne o *backtracking* e a penalidade, tendo em vista que o *backtracking* prejudicou o tempo computacional do AE; (iii) alterar o algoritmo de correção para uma solução semelhante que não utiliza de recursão e estudar o impacto no custo computacional e (iv) considerar a inserção de características positivas de outras técnicas computacionais ao AE, tais como ACO [Shmygelska and Hoos 2005], PSO [Lin and Su 2011], ou outras.

References

Brasil, C. R. S., Delbem, A. C. B., and Da Silva, F. L. B. (2013). Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction. *Journal of Computational Chemistry*, 34:1719–1734.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., editors (2010). *Introduction to Algorithms*. MIT Press and McGraw-Hill., Cambridge, MA, USA.

Cotta, C. (2003). Protein structure prediction using evolutionary algorithms hybridized with backtracking. *Lecture Notes in Computer Science*, 2687:321–328.

Cox, M. M. and Doudna, J. A., editors (2012). *Biologia Molecular: Princípios e Técnicas*. Artmed Editora.

Coy, S. and et al. (2000). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, pages 77–97.

Crescenzi, P., Golman, D., Papadimitriou, C. H., Piccolboni, A., and Yannakakis, M. (1998). On the complexity of protein folding. *Journal of Computational Biology*, 50:423–466.

Custódio, F. L., Barbosa, H. J. C., and Dardenne, L. E. (2004). Investigation of the three-dimensional lattice hp protein folding model using a genetic algorithm. *Genetics and Molecular Biology (Impresso)*, 27(4):611–615.

Darwin, C. R. (1859). On the origin of species.

Gabriel, P. H. R., de Melo, V. V., and Delbem, A. C. B. (2012). Algoritmos evolutivos e modelo hp para predição de estruturas de proteínas. *Revista Controle & Automação*, 23(1).

Gabriel, P. H. R. and Delbem, A. C. B., editors (2008). *Fundamentos de Algoritmos Evolutivos*. Instituto de Ciências Matemáticas e de Computação, Universidade Estadual de São Paulo (USP).

Gallagher, M. and Yuan, B. (2007). Combining meta-eas and racing for difficult ea parameter tuning tasks. In *LOBO, Fernando G.; LIMA, Cláudio F.; MICHALEWICZ, Zbigniew. Parameter Setting in Evolutionary Algorithms*, volume 54, pages 121–142, Berlin. Springer.

Goldberg, D. E., editor (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc.

Holley, L. H. and Karplus, M. (1999). Secondary structure prediction with a neural network. *Proc. Natl. Acad. Sci. USA*, 86:152–156.

Johnson, C. M. and Katikireddy, A. (2006). A genetic algorithm with backtracking for protein structure prediction. *GECCO 2006*, 2:299–300.

Lau, K. and Dill, K. (1989). A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macro-molecules*, 22(10):3986–3997.

Lin, C.-J. and Su, S.-C. (2011). Protein 3d hp model folding simulation using a hybrid of genetic algorithm and particle swarm optimization. *International Journal of Fuzzy Systems*, 13(2).

Shmygelska, A. and Hoos, H. H. (2005). An ant colony optimization algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinf.*, 6(30):1–22.

Simons, K. T., Bonneau, R., Ruckzinski, I., and Baker, D. (1999). Ab initio protein structure prediction of casp iii targets using rosetta. *PROTEINS: Structure, Functions and Genetics Suppl 3*, pages 171–176.

Unger, R. and Moult, J. (1993). A genetic algorithm for 3d protein folding simulations. *Proceedings of ICGA*, pages 581–588.