COMPUTAÇÃO DESTAQUES

Algoritmos Genéticos

By **Rodrigo Kato**

9 de abril de 2021

Algoritmos Genéticos

Rodrigo Kato, Vinícius Paiva, Sandro Izidoro

Revisão: Diego Mariano

BIOINFO – Revista Brasileira de Bioinformática. Edição #01. Julho, 2021.

DOI: 10.51780/978-6-599-275326-13

Algoritmos Genéticos (AGs) são métodos meta-heurísticos baseados na teoria de seleção natural de Charles Darwin e foram inicialmente propostos por J. H. Holland em 1992 [1]. A Figura 1 ilustra o funcionamento de um AG padrão. AGs são procedimentos iterativos que evoluem uma população de indivíduos, onde cada indivíduo representa uma solução candidata para o problema em questão. A cada iteração, denominada geração, os melhores indivíduos são selecionados com base em uma função de aptidão (fitness). Operadores genéticos (cruzamento e mutação) são aplicados aos indivíduos selecionados, visando produzir novos indivíduos a partir do material genético de seus pais a partir de uma probabilidade para ser realizada a operação (pm – probabilidade de mutação e pc – probabilidade de cruzamento). Esse processo é repetido até que uma condição de parada seja satisfeita, podendo ser um número definido de gerações, uma detecção de convergência ou tempo de execução do AG [2; 3]. A Figura 1 apresenta um AG padrão com seus procedimentos.

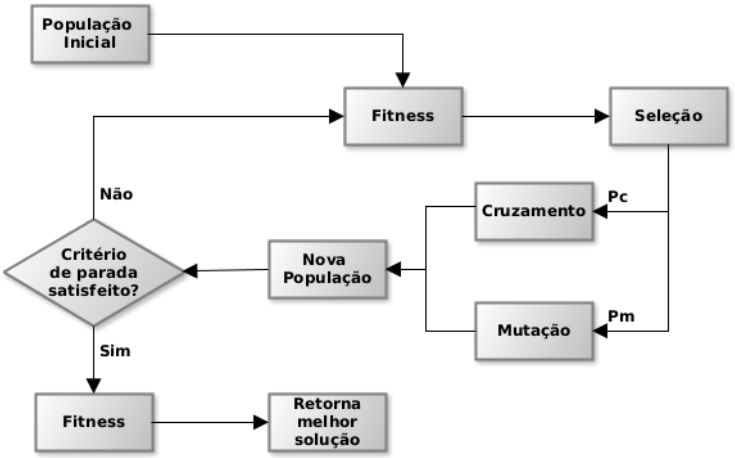

Figura 1 – Funcionamento de um Algoritmos Genético padrão.
Fonte: Izidoro et al. (2014

A seguir, são descritos com mais detalhes cada componente de um AG, bem como a sua execução. Os tópicos abordados são: representação do indivíduo, população, função de avaliação *(fitness)*, métodos de seleção, operadores genéticos, parâmetros, condição de parada e a abordagem paralela de AGs.

Representação do indivíduo

A representação de um indivíduo corresponde à primeira etapa da modelagem de um algoritmo genético. Um indivíduo de um AG é uma abstração de um indivíduo do mundo real. A definição de um indivíduo envolve simplificar aspectos do mundo real e representa uma possível solução para o problema em questão. A modelagem dos indivíduos deve ser realizada de forma que eles possam ser avaliados, selecionados e manipulados pelos operadores genéticos, e geralmente são definidos por especialistas na área [3].

Um indivíduo, ou solução candidata, pode ser representado de acordo com alguns modelos descritos por Eiben e Smith (2007) [3]: binário, inteiro, real e permutação. A representação binária é a mais simples, onde um indivíduo consiste em uma simples *string* binária de dígitos. O tamanho da *string* vai depender do contexto do problema e como se dará o mapeamento do indivíduo do mundo real para o indivíduo do AG. Um problema dessa abordagem é que diferentes bits têm diferentes significados e uma simples alteração em um dos bits (por exemplo, através de uma mutação) pode trazer resultados muito variados.

A representação do tipo inteiro é uma maneira de definir indivíduos de um AG quando o problema naturalmente mapeia diferentes genes (características de um indivíduo) em um elemento de um conjunto.

Uma outra maneira de representar indivíduos de um AG é através de valores reais ou de ponto flutuante. Essa forma consiste em usar números reais para compor a *string* e é utilizada para representar genes com valores contínuos, e não mais discretos como na representação do tipo inteiro. É útil para descrever, por exemplo, valores de distâncias, alturas ou pesos.

A representação do tipo permutação é útil para problemas que envolvem ordenação, como ordenação de tarefas ou problemas de otimização (por exemplo, o problema do caixeiro viajante). Nessa representação, cada indivíduo é formado por uma *string* de números que representam a sequência para a solução do problema.

População

Um conjunto de indivíduos forma a população de um algoritmo genético. Essa população contém possíveis soluções para o problema e pode ser gerada de maneira aleatória ou através de sementes (*seeds*). Diaz-Gomez e

Hougen (2007) [4] citam uma série de fatores a serem levados em consideração ao definir uma população inicial gerada aleatoriamente: o espaço de busca, função *fitness*, diversidade, dificuldade do problema, seleção e o número de indivíduos. As populações iniciadas através de sementes geralmente são criadas a partir de pré-processamento feito com indivíduos aleatórios, onde os mesmos são avaliados de acordo com a função *fitness*. Aqueles mais bem avaliados (melhor pontuação) irão compor a população inicial [5].

Função de avaliação (*fitness*)

Uma função de avaliação ou função *fitness* deve ser capaz de representar os requisitos necessários a que uma população deve se adaptar a fim de avançar para a geração seguinte [3]. Todos os indivíduos da população de cada geração do AG são avaliados por essa função.

É importante que a função *fitness* seja representativa e possa diferenciar com precisão os indivíduos (soluções) bons dos ruins. Uma função *fitness* não ajustada na avaliação de indivíduos pode acabar descartando um indivíduo promissor, que poderia ajudar a encontrar soluções melhores para o problema, além do fato de consumir recursos em indivíduos que agregam pouco no desenvolvimento do AG.

Funções *fitness* multi-objetivas levam em consideração diversos aspectos do problema ao avaliar indivíduos, e podem tratá-los de maneira igualitária ou dando pesos diferentes para cada um.

Métodos de seleção

A operação de seleção implica em como deverá ser feita a escolha de indivíduos que formarão descendentes para a próxima geração [6]. O termo pressão seletiva é muitas vezes usado para mostrar quanto um método de seleção considera o valor de avaliação de indivíduos [7]. O objetivo da seleção de um AG é destacar indivíduos mais aptos na população para que possam gerar descendentes ainda melhores.

Eiben e Smith (2007) [3] citam diversas maneiras de realizar a seleção de indivíduos de um AG, como: o método da roleta, a seleção baseada no valor absoluto de *fitness*, a seleção por torneio e a seleção por *ranking*. Dentre esses, o mais usado é a seleção por torneio.

Seleção por roleta

O método da roleta é uma maneira comum de seleção de indivíduos a partir de seu valor esperado de *fitness*. Para cada indivíduo é assegurado uma área da roleta, onde o tamanho dessa área é proporcional ao valor *fitness* do indivíduo. A roleta então é girada *N* vezes, onde *N* é a quantidade de

indivíduos da população, e ao final de cada vez em que a roleta é girada, o indivíduo marcado é selecionado para ser um pai na próxima geração [6].

Seleção por fitness

O princípio da seleção por *fitness* baseia-se em indivíduos que são selecionados apenas de acordo com o valor absoluto avaliado pela função *fitness*. Indivíduos mais aptos tendem a ocupar toda a população de maneira muito rápida, fazendo com que o processo de busca seja mais focado em uma região do espaço de busca específica. Dessa forma fica mais difícil para o AG cobrir todas as possíveis soluções do problema. Esse fenômeno é conhecido como **convergência prematura** [3].

Seleção por torneio

A seleção pelo método de torneio requer que um número N de indivíduos aleatórios sejam selecionados. Depois, um número aleatório R é escolhido entre 0 e 1. Se $R < K$ (onde K é um parâmetro, como por exemplo 0.75), o indivíduo mais apto é selecionado, caso contrário o menos apto [6]. Quanto maior é o valor de K, maior será a pressão seletiva imposta à população.

Seleção por ranking

A seleção feita através do método de ranking classifica os indivíduos com base em sua *fitness* e depois são alocadas probabilidades de seleção de acordo com o ranqueamento (e não em relação ao valor de *fitness*) [3]. Essa abordagem evita que a maior parte da seleção seja feita por indivíduos mais aptos, reduzindo a pressão seletiva. É uma alternativa para evitar a convergência prematura [6].

Operadores genéticos

Após a seleção de indivíduos, dois operadores genéticos (cruzamento e mutação) são utilizados para gerar uma nova população (próxima geração do AG). Esses operadores genéticos têm como finalidade refinar e espalhar a busca, respectivamente, trazendo também mais variabilidade genética.

O operador de cruzamento utiliza a combinação entre dois indivíduos (aqui definidos como pais) para gerar indivíduos descendentes (definidos como filhos) [8].

Basicamente, a operação de cruzamento acontece quando dois indivíduos são selecionados e partes aleatórias destes são trocadas entre eles, formando assim novos indivíduos. Pode-se citar três principais formas de cruzamento: **ponto simples** (ou um ponto), **multiponto** (ou k-pontos) e **uniforme**.

Cruzamento ponto simples

O cruzamento do tipo **ponto simples** seleciona dois indivíduos pais para o cruzamento e seleciona aleatoriamente um ponto P_i nesses indivíduos (onde $i >= 0$ e $i < n$, sendo n o tamanho do indivíduo). Então, dois indivíduos filhos são criados pela combinação das partes criadas pela divisão dos indivíduos pais pelo ponto P_i [9]. A Figura 2 mostra um exemplo do cruzamento do tipo um ponto.

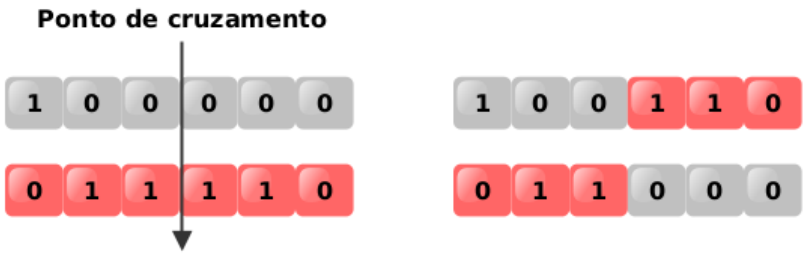

Figura 2 – Funcionamento do cruzamento do tipo um ponto. Fonte: próprio autor

Cruzamento multiponto

O cruzamento **multiponto** atua de forma bem similar ao cruzamento de ponto simples, porém, nesse caso mais de um ponto é criado. O método seleciona dois indivíduos pais e também seleciona aleatoriamente um valor de K, que determina os pontos P_{1i} a P_{k-1i} (onde $i >= 0$ e $i < n$, sendo n o tamanho do indivíduo) que serão os locais onde haverá o cruzamento [10]. A Figura 3 ilustra o cruzamento multiponto com K = 2.

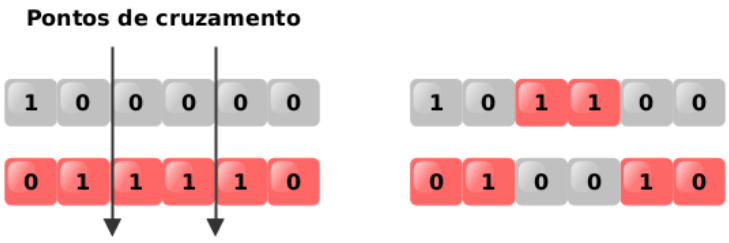

Figura 3 – Funcionamento do cruzamento multiponto com K = 2.
Fonte: Próprio autor

Cruzamento uniforme

O cruzamento **uniforme** (Figura 4) usa uma proporção fixa para determinar a contribuição de cada pai, e essa contribuição ocorre no nível do gene, e não

no nível do segmento. Durante a operação de cruzamento, uma máscara aleatória de 0 e 1 é gerada conforme a taxa de cruzamento. Para uma taxa de cruzamento de 0,5, metade dos genes nos filhos seria herdada do pai 1, enquanto a outra metade seria herdada do pai 2. Os genes que correspondem ao bit 1 são retirados do pai 1, enquanto os correspondentes a 0 são retirados do pai 2 [11].

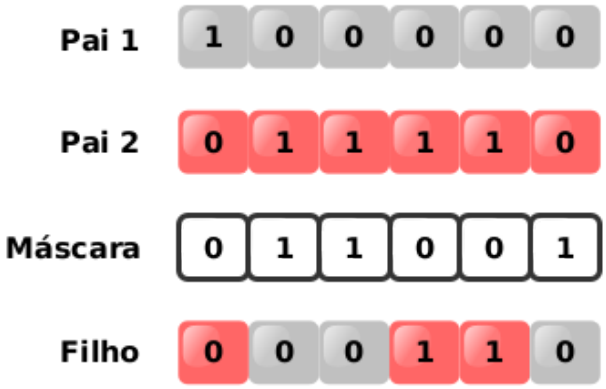

Figura 4 – Funcionamento do cruzamento uniforme.
Fonte: Adaptado de Chaudhry e Usman (2017).

Mutação

O operador de mutação ocorre alterando aleatoriamente algumas características genéticas de certos indivíduos que foram selecionados por um critério probabilístico [11]. A mutação é uma operação que utiliza apenas o indivíduo pai para criar o indivíduo filho, aplicando algum tipo de modificação aleatória em sua representação [3]. Diversos tipos de mutação são descritos por Soni e Kumar (2014) [12], como: a **mutação de inserção**, **de inversão** e **uniforme**.

Mutação de inserção

A **mutação de inserção** (Figura 5) seleciona dois genes aleatórios do indivíduo e então move o primeiro gene para seguir o segundo, movendo todos os outros genes de acordo. Esse tipo de mutação não modifica muito a ordem em que os genes aparecem e é utilizada em problemas de permutação.

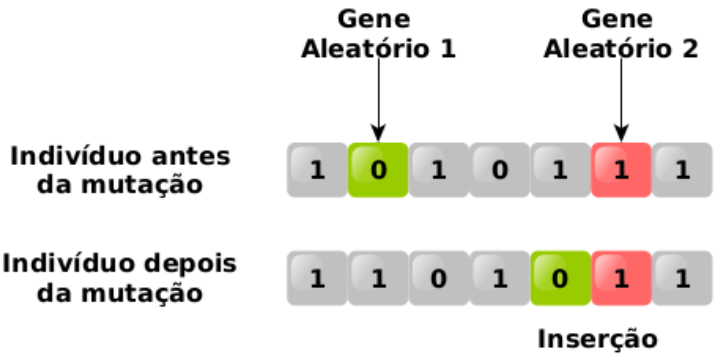

Figura 5 – Funcionamento da mutação de inserção. Fonte: Próprio autor.

Mutação de inversão

Na **mutação de inversão** (Figura 6), dois genes aleatórios são escolhidos e realiza-se a inversão de todos os genes. Isso faz com que seja preservada a informação adjacente entre os genes, porém, perde-se informação de ordem. Também é utilizado em problemas de permutação.

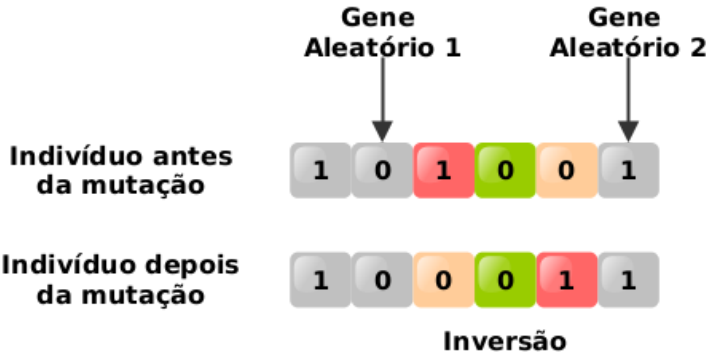

Figura 6 – Funcionamento da mutação de inversão. Fonte: Próprio autor.

Mutação uniforme

Já a **mutação uniforme** (Figura 7) realiza a mudança de um gene aleatório de acordo com um valor específico em que esse gene pode assumir. Ou seja, um gene G escolhido para sofrer a mutação pode receber um valor *i*, onde *i* corresponde a um elemento do conjunto de valores que *G* pode assumir. Esse tipo de mutação é usado em casos de representação do tipo real e inteira de indivíduos.

Figura 7 – Funcionamento da mutação uniforme. Fonte: Próprio autor.

Parâmetros

Os algoritmos genéticos possuem alguns parâmetros que impactam diretamente em seu funcionamento. Apesar de haver na literatura valores padrão recomendados, a configuração de parâmetros é particular no contexto em que o AG está inserido [2]. Alguns dos parâmetros utilizados nos AGs são:

- **Número de gerações:** é um dos critérios de parada de um AG. Ao executar um número muito pequeno de gerações, um AG pode não encontrar uma resposta satisfatória. Por outro lado, um número de gerações muito grande pode impactar negativamente no tempo computacional gasto.
- **Tamanho da população:** é a quantidade de indivíduos presentes em cada geração do AG. Pode ser estática, se mantendo a mesma durante toda a execução do algoritmo, ou pode sofrer alterações em seu tamanho de acordo com a execução. Populações maiores tendem a consumir mais tempo de execução, enquanto populações menores podem acabar não cobrindo todo o espaço de busca do problema.
- **Probabilidade de Cruzamento:** é um percentual que indica a chance de um indivíduo trocar material genético com outro dentro da população a fim de gerar indivíduos descendentes. Tem como finalidade refinar a busca de indivíduos. AGs com taxas de cruzamento altas tendem a inserir novas características mais rapidamente à população, porém pode-se acabar perdendo bons indivíduos que possam ser substituídos. O uso de operadores de cruzamento com probabilidade alta ou baixa depende do contexto do problema.
- **Probabilidade de Mutação:** determina a chance de um indivíduo sofrer alterações em suas características. Tem como finalidade evitar com que o AG fique preso em mínimos locais, sendo responsável por inserir diversidade à população. Geralmente, possui taxas baixas, mas como acontece com o cruzamento, a utilização de taxas altas ou baixas depende do contexto.
- **Tamanho do Torneio:** é um parâmetro da seleção por torneio que controla a pressão seletiva, evitando uma convergência precoce do AG. O operador de torneio trabalha selecionando aleatoriamente *N* indivíduos da população e selecionando a melhor solução entre eles para seguir para a geração seguinte.

Critérios de parada

Existem duas principais formas para o término da execução de um algoritmo genético [3]. A primeira é em relação às características dos indivíduos que compõem a solução do problema. Quando é possível identificar um padrão ótimo em relação aos indivíduos da população, não existe mais a necessidade de se continuar executando o AG, podendo assim encerrar sua execução.

A segunda forma de condição de parada de um AG acontece quando não se sabe identificar um padrão ótimo dos indivíduos. Pode-se citar alguns fatores que podem fazer com que um AG termine sua execução:

- Tempo máximo de execução do algoritmo ou número de gerações é excedido;
- Número total de avaliações feitas pela função *fitness* é alcançado;
- Melhorias em indivíduos feitas através de operadores genéticos e seleção já alcançaram um certo limite, não havendo mais mudanças.

Geralmente, um AG tem sua execução terminada quando se satisfaz uma das formas descritas anteriormente: quando um certo valor ótimo (ou satisfatório) é alcançado pelos indivíduos ou quando uma condição de parada é satisfeita.

Algoritmos genéticos paralelos

Algoritmos genéticos são uma importante abordagem computacional para resolver diversos problemas de busca e otimização. Problemas que muitas vezes estão inseridos em contextos amplos e que por isso acabam requerendo uma grande quantidade de recursos computacionais. Existem casos em que executar AGs em máquinas em série pode levar dias ou até semanas até completar sua execução, e uma abordagem paralela pode trazer ganhos consideráveis em tempo de execução e utilização de recursos [13].

Computação paralela diz respeito a vários processos que trabalham simultaneamente a fim de resolver um determinado problema. O paralelismo funciona decompondo a carga de trabalho, ou tarefas, entre os vários recursos computacionais disponíveis, a fim de ter ganhos em relação a tempo e/ou melhora nos resultados. Abordagens de problemas que utilizam o paralelismo devem levar em consideração a comunicação entre os processos, pois muitas vezes apenas ajustar um problema serial não garante a melhor abordagem paralela [14].

Algoritmos Genéticos têm como característica implícita uma busca naturalmente paralela por uma solução. Isto se evidencia ao notar-se que cada indivíduo dentro de uma população busca por si só otimizar sua *fitness*

[15]. É esta propriedade que permite em uma mesma população, com todos os indivíduos expostos aos mesmos operadores, o surgimento de soluções boas diversas. Este fato, atrela o conceito de Algoritmos Genéticos ao conceito de paralelização, indicando intuitivamente a ideia de um AG paralelo.

Tendo isso em mente, AGs paralelos trabalham, por exemplo, com problemas de multi-população, onde vários processos diferentes trabalham de maneira independente com suas respectivas populações e AG. Ao final de cada execução paralela ou até mesmo após algumas gerações, processos podem trocar mensagens entre si, compartilhando e integrando soluções [16].

Embora AGs sequenciais têm mostrado sucesso em diversos contextos e problemas diferentes, existem casos em que essa abordagem não é o suficiente, sendo necessário partir para uma implementação paralela do AG. Pode-se citar como casos em que AGs paralelos são mais vantajosos [17]:

- Quando a população é demasiadamente grande;
- Quando a função *fitness* consome muitos recursos (tempo e/ou memória);
- Quando AGs sequenciais caem em regiões subótimas do espaço de busca.

Existem três classes gerais de AGs paralelos: primário-secundário, granulado-grosso (ilha) e granulado-fino (célula) [19]. Os AGs paralelos do tipo primário-secundário utilizam a mesma ideia de um AG sequencial e aplicam o paralelismo de maneira simples, onde não se altera nem restringe nenhum dos operadores genéticos. No modelo primário-secundário geralmente apenas a avaliação de indivíduos pela função *fitness* é feita de maneira paralela. Todo o controle de gerações, seleção e operadores de mutação e cruzamento são feitos de forma serial. Esse tipo de implementação é utilizado principalmente quando a análise da *fitness* é complexa e consome muita carga computacional. Uma desvantagem desse método é que muitas vezes o processo mestre fica ocioso, esperando pelos outros processos. A Figura 8 ilustra a topologia de um AG paralelo do tipo primário-secundário [19].

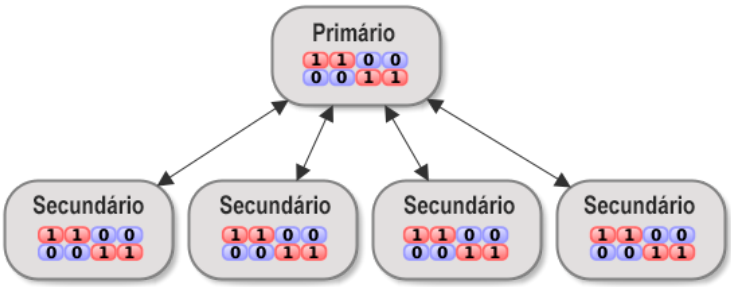

Figura 8 – Exemplo de um AG paralelo do tipo primário-secundário (antigamente denominado como master-slave). Fonte: Adaptado de Fauzi Mohd Johar et al. (2013).

Um outro tipo de abordagem paralela em AGs é a de granulado-grosso. Essa forma apresenta uma diferença significativa na modelagem de um AG, pois trata-se da paralelização de populações, dividindo em subgrupos que são direcionados a processos diferentes (ilhas). Cada ilha tem seu próprio processo evolutivo, com suas respectivas gerações e operadores de seleção, cruzamento e mutação. Os processos trabalham como um AG sequencial, separadamente, e durante as execuções dos processos, indivíduos de uma ilha migram para outra a fim de gerar uma variabilidade genética maior [14].

Por último, tem-se os AGs paralelos do tipo granulado-fino ou celular, onde indivíduos são distribuídos em processos diferentes e as operações de cruzamento ficam restritas a processos ou células vizinhas. AGs paralelos do tipo granulado-fino requerem uma topologia bem definida e geralmente decaem em performance à medida que a população aumenta [19]. Essa é uma alternativa viável principalmente quando implementada sobre um dispositivo de processamento SIMD (*Single Instruction Multiple Data*).

Algoritmos genéticos na bioinformática

Algoritmos genéticos (AGs) são técnicas de otimização estocástica que não garantem a solução ótima do problema todas as vezes em que são executados, se tratando assim de um processo não-determinístico. Apesar disso, existem diversos contextos em que AGs podem ser utilizados, e a bioinformática é um exemplo.

Deaven e Ho (1995) [20], por exemplo, propuseram um método usando AGs para determinar a estrutura de menor energia de um *cluster* atômico. Wild e Willett (1996) [21], por sua vez, fizeram uma pesquisa de similaridade em bases de dados de estruturas tridimensionais de produtos químicos, representados pelos seus campos potenciais eletrostáticos moleculares. Os autores utilizaram um AG para alinhar os campos moleculares, maximizando a sua sobreposição.

Jones et al. (1997) [22] descreveram o *Genetic Optimisation for Ligand Docking* (GOLD) um método de *docking* automático de ligantes que utiliza um AG para explorar toda a gama de flexibilidade conformacional do ligante com a flexibilidade parcial da proteína.

No ano 2000, Szustakowski e Weng (2000) [23] modelaram um AG para alinhar estruturas de proteínas. Dado que as estruturas das proteínas são mais conservadas no núcleo do que nos loops e alças (com exceção dos loops e alças envolvidos em sítios ativos), a estratégia utilizada foi a de alinhar os núcleos das proteínas, representados por seus elementos da estrutura secundária (SSE). Para essa tarefa, um AG foi utilizado para encontrar a menor diferença entre as matrizes de distância. Quatro anos mais tarde, de Magalhães et al. (2004) [24] utilizaram AGs para tratar do problema de *docking* para proteína-ligante. Nele, o AG trabalha com uma população de indivíduos, onde cada indivíduo é a posição do ligante com relação a proteína. Desta forma, a conformação do ligante é representada por um cromossomo constituído de genes de valores reais representando os graus de liberdade de orientação e conformação. A função de fitness foi baseada na interação total de energia entre a proteína e a molécula ligante. Os resultados mostraram que a distribuição da população inicial pode ser relevante para a performance do algoritmo.

Ainda em 2004, Unger (2004) [25] apresentou um trabalho discutindo o uso e os resultados dos AGs em problemas de predição e alinhamento de estruturas de proteínas. Na predição de estruturas, são tratadas questões como representação dos indivíduos, função de *fitness* e operadores genéticos. Entre as questões discutidas está o problema de colisão entre os átomos quando um indivíduo é definido como sendo os valores dos ângulos φ e ψ ao longo da cadeia principal. Dessa forma, as aplicações que usam essa

abordagem devem incluir, de alguma maneira, um procedimento para detectar essas colisões.

Liu e Tao (2008) [26] propuseram a utilização de AG para a predição de estrutura de proteínas baseando-se em sua sequência. Os autores utilizaram valores hidrofóbicos de proteínas como um modelo de otimização matemático e um AG foi utilizado para resolver o problema de otimização.

Em Fober et al. (2009) [27], os AGs foram usados na construção de alinhamentos de múltiplos grafos (MGA) para a análise estrutural de biomoléculas. No trabalho proposto, cada MGA corresponde a uma solução candidata (indivíduo).

Kernytsky e Rost (2009) [28] utilizaram AGs para o problema de predição de função com uma abordagem diferente. O indivíduo do AG nessa abordagem é codificado a partir de informações de resíduo, estrutura secundária, acessibilidade do solvente, hélice transmembrana e conservação obtidos através de alinhamentos múltiplos de sequências. O AG seleciona os indivíduos mais aptos para a próxima geração a partir da avaliação feita por um algoritmo de aprendizagem baseado em redes neurais.

Kato et al. (2015) [29] implementaram um AG para refinar parâmetros de campo de força com o objetivo de determinar a energia de RNA. Nessa abordagem, os nucleotídeos uracila, adenina, guanina ou citosina são utilizados como referências para os cálculos de mecânica quântica e clássica, onde as energias de torção (diedro) e a eletrostática são reparametrizadas.

Otovic et al. (2020) [30] desenvolveram um AG para busca em espaços químicos de pequenos peptídeos. O trabalho permitiu a definição de bibliotecas de peptídeos capazes de cobrir uma grande área do espaço de pesquisa de novos peptídeos ativos.

Genetic active site search (GASS)

Esta seção apresenta com detalhes uma metodologia para a busca de sítios catalíticos similares através de informações estruturais de proteínas. O método *Genetic Active Site Search* (GASS) [2] utiliza um algoritmo genético (AG) para a busca de sítios catalíticos baseados em *templates*.

Izidoro et al. (2014) [2] definem o problema de busca baseada em *templates* da seguinte forma. Dado um conjunto de *N* aminoácidos que compõe o sítio catalítico *A* de uma enzima de função conhecida (*template*), e uma proteína hipotética B com *M* aminoácidos de função desconhecida, o método procura o padrão A em B (Figura 9).

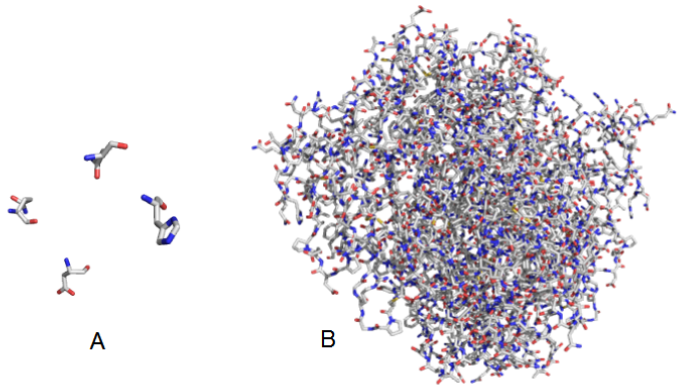

Figura 9 – Sítio catalítico de função conhecida (A) – Proteína de função desconhecida (B). Fonte: Izidoro et al. (2014)

A Figura 10 ilustra o funcionamento do GASS. Proteínas e *templates* são selecionados pelo usuário para a etapa de pré-processamento. Nesta etapa é criado um repositório de proteínas com informações provenientes do *Protein Data Bank* (PDB; https://www.rcsb.org) e do M-CSA (https://www.ebi.ac.uk/thornton-srv/m-csa/), que serão acessados pelo GASS para criar sua população inicial do AG. Em seguida, o AG executa uma busca heurística para encontrar os sítios catalíticos similares nas proteínas selecionadas, produzindo um ou mais sítios catalíticos candidatos. A fim de lidar com a mutação conservativa, o AG também tem a opção de consultar uma matriz de substituição de resíduos. A seguir serão apresentados em maiores detalhes a modelagem e configuração do AG empregado no método.

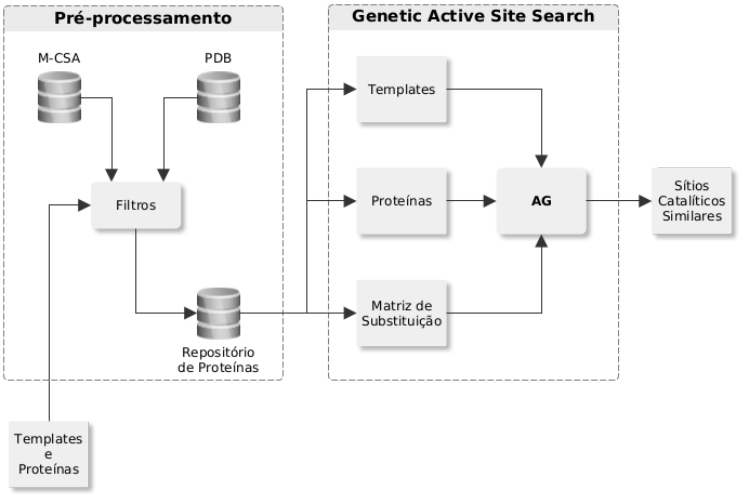

Figura 10 – Metodologia para a busca de sítios catalíticos similares utilizando AGs. Fonte: Izidoro et al. (2014).

Representação do indivíduo e inicialização da população

A representação de um indivíduo é um ponto muito importante na modelagem de um AG, e depende muito do conhecimento disponível sobre o problema a ser resolvido. Para o problema em questão, um indivíduo representa um grupo de aminoácidos, o qual é um candidato a sítio catalítico de uma enzima. O indivíduo é codificado como um vetor, onde cada posição recebe dados sobre um aminoácido, obtidos a partir do repositório de proteínas criado na fase de pré-processamento.

Assim, para cada aminoácido que pode fazer parte de um sítio catalítico, é armazenado o seu nome, o nome do último átomo mais pesado na cadeia lateral (*Last Heavy Atom* – LHA) e sua posição (x, y, z), a posição do aminoácido na sequência da enzima e sua cadeia. A Figura 11 mostra um exemplo de um indivíduo formado por 4 aminoácidos.

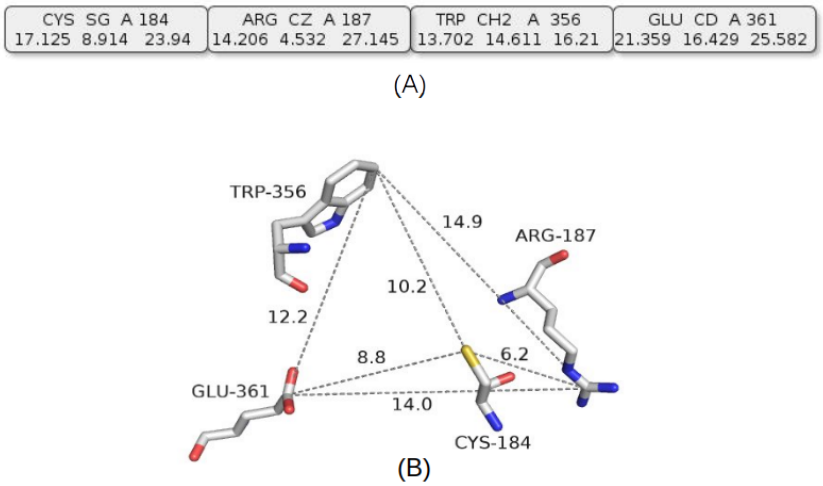

Figura 11 – Representação de um indivíduo candidato a sítio catalítico – (A) Indivíduo do GASS – (B) Sítio catalítico da enzima 3NOS (Human Endothelial Nitric Oxide Synthase with Arginine Substrate – EC Number: 1.14.13.39) com as distâncias (em Angstroms) entre os LHAs de cada resíduo. Fonte: Izidoro et al. (2014).

A população inicial é gerada a partir do repositório de dados obtidos na etapa de pré-processamento. Cada indivíduo é formado por *n* aminoácidos que são aleatoriamente escolhidos do repositório, sempre respeitando seus tipos conforme o *template* dado, por exemplo, se a primeira posição requerida é um glutamato, apenas aminoácidos desse tipo poderão ser selecionados para tal posição.

Função de avaliação (*fitness*)

Tendo a população inicial, o próximo passo do AG é avaliar os indivíduos. Na metodologia implementada, a distância entre as coordenadas dos LHAs representadas por um vetor de coordenadas 3D é calculada para cada par de resíduos do *template* (*v*), e as coordenadas de cada par de resíduos do sítio ativo candidato encontrado pelo GASS (*w*), de acordo com a Equação 1, onde *n* é igual ao número de resíduos no *template* e no indivíduo. Quanto mais próximo de zero o valor do *fitness*, maior a similaridade, em termos de distância, entre o *template* e o sítio candidato.

$$Fit(\mathbf{v}, \mathbf{w}) = \sqrt{\sum_{i=1}^{(n^2-n)/2} \|v_i - w_i\|^2}$$

Equação 1 – Função de avaliação (fitness).

Seleção e operadores genéticos

Após a avaliação dos indivíduos segue-se a fase de seleção. Essa fase é crucial para a evolução da população, pois dá uma maior chance de sobrevivência aos melhores indivíduos, por exemplo, aqueles com melhor *fitness*. Aqui foi utilizado a seleção por torneio, onde um subconjunto de *k* indivíduos é sorteado aleatoriamente da população, e o melhor indivíduo desse subconjunto de acordo com a *fitness* é selecionado.

Uma vez feita a seleção, dois operadores genéticos são usados para gerar uma nova população: cruzamento de um ponto e mutação de um ponto (Figura 12). No caso da mutação de um ponto, apenas o ponto escolhido é substituído por um resíduo aleatório, que pode ser do mesmo tipo a partir da enzima selecionada (TRP 356 trocado pelo TRP 190 – em vermelho na Figura 8), ou por um tipo diferente de resíduo (mutação conservativa), indicado pela matriz de substituição de resíduos da mesma enzima (GLU 361 trocado pelo ASP 369 – em azul na Figura 12).

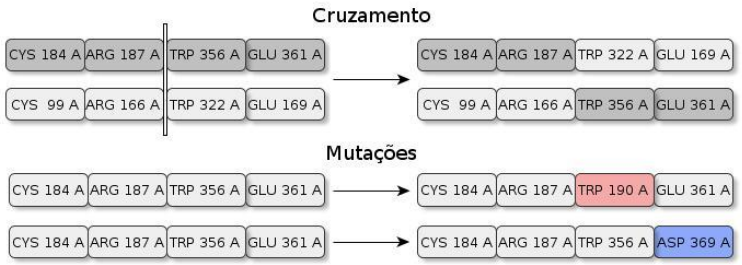

Figura 12 – Representação dos operadores de cruzamento e mutação. Na representação resumida do indivíduo tem-se o nome do resíduo, a posição na sequência e a cadeia. Fonte: Adaptado de Izidoro et al. (2014)

Parâmetros

Um AG possui um conjunto de parâmetros que influencia diretamente o seu comportamento, e cada problema requer uma configuração particular a partir de testes e análises de resultados preliminares. Para isso, são utilizados valores padrões como ponto de partida até a obtenção dos valores finais para os parâmetros. Os parâmetros do AG – GASS (Tabela 1) foram ajustados de forma empírica.

Parâmetros	**Valores**
Tamanho da População	400
Número de Gerações	100
Taxa de Cruzamento	90%

Parâmetros	Valores
Taxa de Mutação	30%
Taxa de Mutação	10%
Tamanho do Ranking	10
Tamanho do Torneio	2

Tabela 1 – Valores dos parâmetros utilizados no AG – GASS.

Resultados

O GASS foi testado contra os 17 métodos participantes do *Critical Assessment of protein Structure Prediction* (CASP 10), na categoria *Function Prediction* (FN) [2]. No experimento, o GASS aparece em quarto lugar geral, com valor médio de MCC (*Matthew Correlation Coefficient*) de 0,63 (Figura 13). Se comparado apenas aos métodos automáticos, o GASS aparece em terceiro lugar.

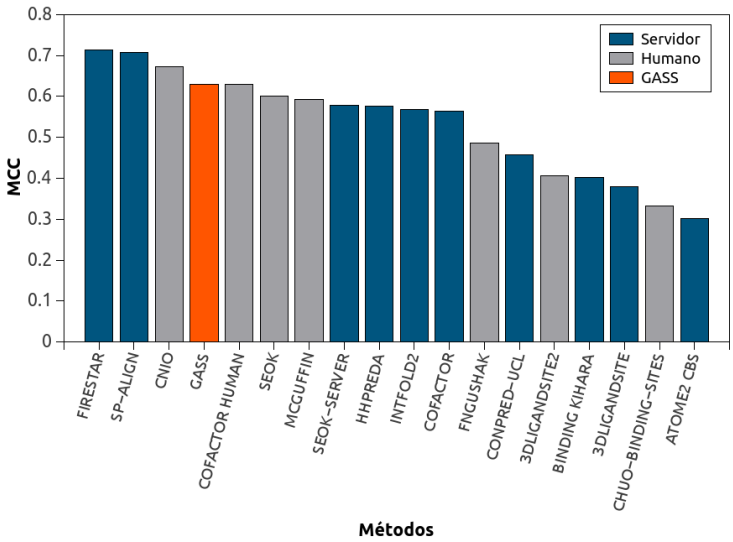

Figura 13 – Grupos participantes do CASP 10 (categoria FN) classificados em ordem decrescente pelo valor médio de MCC juntamente com GASS. Preditores humanos são mostrados em cinza, preditores baseados em servidores em azul e o GASS em laranja. Fonte: Izidoro et al. (2014).

Considerações finais

Os Algoritmos Genéticos (AGs) não são tão simples quanto parecem, e o grande desafio está na modelagem do seu problema e no ajuste de seus

parâmetros. Porém eles são considerados muito úteis para resolver problemas de otimização e/ou problemas onde o espaço de busca é muito grande.

Neste trabalho foram apresentados os conceitos básicos dos algoritmos genéticos (padrão e o multiobjetivo), além de diversas aplicações na área de bioinformática. O que procuramos mostrar aqui é que na área de bioinformática existem problemas tão complexos que seria proibitivo resolvê-los em tempo hábil e esse é um dos motivos do surgimento de algoritmos evolutivos, sendo o AG um deles. Podemos concluir que apesar de algumas limitações, os AGs são ferramentas alternativas utilizadas para resolver problemas complexos.

Referências

1- Katoch, S.; Chauhan, S. S. e Kumar, V. (2020). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications.

2- Izidoro, S. C.; de Melo-Minardi, R. C. e Pappa, G. L. (2014). GASS: identifying enzyme active sites with genetic algorithms. Bioinformatics, 31(6):864–870.

3- Eiben, A. E. e Smith, J. E. (2007). Introduction to Evolutionary Computing . Springer Verlag.

4- Diaz-Gomez, P. A. e Hougen, D. F. (2007). Initial population for genetic algorithms: A metric approach. In Arabnia, H. R.; Yang, J. Y. e Yang, M. Q., editores, Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, GEM 2007, June 25-28, 2007, Las Vegas, Nevada, USA, pp. 43–49. CSREA Press.

5- Meadows, B.; Riddle, P.; Skinner, C. e Barley, M. M. (2013). Evaluating the seeding genetic algorithm. In Cranefield, S. e Nayak, A., editores, AI 2013: Advances in Artificial Intelligence, pp. 221–227, Cham. Springer International Publishing.

6- Mitchell, M. (1998). An Introduction to Genetic Algorithms. A Bradford book. Bradford Books.

7- Camargo, G. d. M. (2006). Controle da pressão seletiva em algoritmo genético aplicado a otimização de demanda em infra-estrutura aeronáutica. Master's thesis, Escola Politécnica, Universidade de São Paulo.

8- Dréo, J.; Chatterjee, A.; Pétrowski, A.; Siarry, P. e Taillard, E. (2006). Metaheuristics for Hard Optimization: Methods and Case Studies. Springer Berlin Heidelberg.

10- Umbarkar, A. J. , P. D. S. (2015). Crossover operators in genetic algorithms:a review. ICTACT Journal on Soft Computing, 6(1):1083–1092.

11- Chaudhry, I. A., e Usman, M. (2017). Integrated process planning and scheduling using genetic algorithms, *Tehnički vjesnik*, 24(5), pp. 1401-1409. https://doi.org/10.17559/TV-20151121212910.

12- Soni, N. e Kumar, T. (2014). Study of various mutation operators in genetic algorithms. volume 5, pp. 4519–4521.

13- Goldberg, D.; David Edward, G.; Goldberg, D. e Goldberg, V. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Artificial Intelligence. Addison-Wesley Publishing Company.

14- Umbarkar, A. J. e Joshi, M. S. (2013). Review of parallel genetic algorithm based on computing paradigm and diversity in search space. ICTACT Journal on Soft Computing, 3(4):615–622.

15- Madhuri e Deep, K. (2009). A state-of-the-art review of population-based parallel meta-heuristics. In 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pp. 1604–1607.

16- Cantu-Paz, E. (1998). A survey of parallel genetic algorithms. CALCULATEURS PARALLELES, 10.

17- Majd, A.; Lotfi, S. e Sahebi, G. (2013). Review on parallel evolutionary computing and introduce three general framework to parallelize all ec algorithms. The 5th Conference on Information and Knowledge Technology, pp. 61–66.

18- Nowostawski, M. e Poli, R. (1999). Parallel genetic algorithm taxonomy. In 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410), pp. 88–92.

19- Fauzi Mohd Johar; Farah Ayuni Azmin; Mohamad Kadim Suaidi; Shibghatullah, A. S.; Badrul Hisham Ahmad; Siti Nadzirah Salleh; Mohamad Zoinol Abidin Abd Aziz e Shukor, M. M. (2013). A review of genetic algorithms and parallel genetic algorithms on graphics processing unit (gpu). In 2013 IEEE International Conference on Control System, Computing and Engineering, pp. 264–269.

20- Deaven, D. M. e Ho, K. M. (1995). Molecular Geometry Optimization with a Genetic Algorithm. Phys. Rev. Lett., 75:288-291.

21- Wild, D. J. e Willett, P. (1996). Similarity Searching in Files of Three-Dimensional Chemical Structures. Alignment of Molecular Electrostatic Potential Fields with a Genetic Algorithm. J. Chem. Inf. Comput. Sci. , 36 (2):159-167.

22- Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R. e Taylor, R. (1997). Development and validation of a genetical gorithm for exible docking. Journal of Molecular Biology, 267:727-748.

23- Szustakowski, J. D. e Weng, Z. (2000). Protein Structure Alignment Using a Genetic Algorithm. Proteins: Structure, Function, and Genetics, 38:428-440.

24- de Magalhães, C. S.; Barbosa, H. J. C. e Dardenne, L. E. (2004). A genetic algorithm for the ligand-protein docking problem. Genetics and Molecular Biology, 27:605-610.

25- Unger, R. (2004). The Genetic Algorithm Approach to Protein Structure Prediction. Structure and Bonding , 110:153-175.

26- Liu, Y. e Tao, L. (2008). Protein structure prediction based on an improved genetic algorithm. In 2008 2nd International Conference on Bioinformatics and Biomedical Engineering, pp. 577–580.

27- Fober, T.; Mernberger, M.; Klebe, G. e Hüllermeier, E. (2009). Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. Bioinformatics, 25(16):i2110-i2117.

28- Kernytsky, A. e Rost, B. (2009). Using genetic algorithms to select most predictive protein features. Proteins-Structure Function and Bioinformatics , 75(1):75-88.

29- Kato, R. B.; Silva, F. T.; Pappa, G. L. e Belchior, J. C. (2015). Genetic algorithms coupled with quantum mechanics for refinement of force fields for RNA simulation: a case study of glycosidic torsions in the canonical ribonucleosides. Phys. Chem. Chem. Phys., 17:2703-2714.

30- Otovic, E.; Njirjak, M.; Zuzic, I.; Kalafatovic, D. e Mausa, G. (2020). Genetic algorithm parametrization for informed exploration of short peptides chemical space. In 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–3.