# Resumo do artigo "SAGA: sequence alignment by genetic algorithm"

Rodney Rick
email: rodneyrick@gmail.com

***Resumo*—Neste artigo apresenta uma nova abordagem para alinhamento de sequências múltiplas usando algoritmos genéticos e um pacote de software associado chamado SAGA[1] (*sequence alignment by genetic algorithm*). Este método envolve a evolução de uma população de alinhamentos de uma forma quase evolutiva e melhorar gradativamente a aptidão da população medida por uma função objetiva (conforme metodologia de Algoritmos Genéticos) capaz de medir a qualidade do alinhamento múltiplo. Com uma série de operadores para combinação de alinhamentos ou mutações entre as gerações, tenta otimizar a função objetivo para encontrar os melhores pares genéticos entre alinhamentos.**

## I. Introdução

O alinhamento simultâneo de muitas sequências de ácidos nucleicos ou aminoácidos é uma das técnicas mais necessárias para análise de sequências. Alinhamentos múltiplos são usados para ajudar a prever estruturas semelhantes em novas sequências e possivelmente em famílias existentes e mesmo para sugerir pontos iniciadores para PCR (*Polymerase chain reaction* ou Reação em cadeia da polimerase) e para a reconstrução filogenética. Conforme Feng e Doolittle[2], a grande maioria dos alinhamentos múltiplos são utilizados de modo "progressivo".

Os autores exploram a vantagem de velocidade e simplicidade combinada com sensibilidade (razoável) admitindo estruturas terciárias conhecidas. Ao realizar esta abordagem, ganha-se um mínima vantagem nos alinhamentos, mas depara-se com o problema de "mínimo local"que decorre na utilização algoritmos gulosos. Com possíveis alinhamentos intermediários, podendo ser tratado qual parâmetros de qualidade mínima aceitável.

Há duas formas mais utilizadas para alinhamento progressivo:

(A) Modelos Ocultos de Markov (HMMs - *Hidden Markov Models*), o qual tentam encontrar simultaneamente um alinhamento e um modelo de probabilidade de substituições, inserções e deleções que é mais autoconsistente, mas é limitada de muitas sequências (acima de 100);

(B) Outra opção é usar funções objetivas (FOs) que medem a qualidade de alinhamento múltiplo e encontrar o melhor alinhamento de pontuação. Caso a FO seja bem definida, pode-se obter uma qualidade adequada, ganhando a vantagem de que se pode ter certeza de que o alinhamento resultante é realmente o melhor por algum critério escolhido. No entanto, nesta segunda forma, o limitante está no comprimento razoável.

Existem duas soluções para este problema. O programa MSA (*Multiple Sequence Alignment*), tenta diminuir o espaço da solução para uma área relativamente menor, onde o melhor alinhamento provavelmente ocorrerá, reduzindo o alinhamento para este espaço. Esta escolha reduz drasticamente para cerca de sete ou oito sequências no máximo. Uma segunda abordagem é usar métodos de otimização estocástica, como simulação de recozimento (Simulated Annealing - SA) ou algoritmos genéticos (AGs). Para SA utiliza-se numerosas ocasiões para alinhamento múltiplo, mas pode ser muito lento e geralmente só funciona bem como um otimizador de alinhamentos, ou seja, quando o método é dado um alinhamento que já está perto de ideal, escapando do mínimo local. A amostragem de Gibbs[3] tem sido muito bem aplicada ao problema de encontrar o melhor bloco de alinhamento múltiplo local sem falhas, porém nào é trivial. Aplicando assim uma tentativa de usar AGs neste contexto. Aqui eles usaram um programa híbrido iterativo de programação dinâmica (PD) e GA.

A opção pela estratégia de GA e SAGA parece capaz de encontrar alinhamentos múltiplos ótimos globalmente (ou próximos a ele) em tempo razoável, a partir de sequências completamente desalinhadas. Pode encontrar soluções que são tão boas ou melhores do que MSA ou CLUSTAL W[4]. Biologicamente, para esta aplicação ser bem sucedida de métodos de otimização para este problema, depende criticamente do FO. Caso FO não

seja um bom descritor de qualidade de alinhamento múltiplo, então os alinhamentos não serão necessariamente melhores em qualquer sentido real, uma vez que não é possível otimizá-la.

## II. Métodos

Nesta seção concentra-se a explicação da funcão objetivo e a metodologia SAGA utilizada para melhoria das soluções.

### A. Função Objetiva - FO

A avaliação dos alinhamentos é feita utilizando uma FO que é simplesmente uma medida de qualidade de alinhamento múltiplo. Para isso, considera-se duas FOs relacionadas às somas ponderadas de pares com penalidades de *gap*. De início considera-se um custo para cada par de resíduos alinhados em cada coluna do alinhamento (custo de substituição), e outro custo para as lacunas (custo da *gaps*). Estes são adicionados para dar o custo global do alinhamento. Além disso, cada par de sequências é dado um peso relacionado com a sua semelhança com os outros pares.

$$CustoDeAlinhamento(A) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} W_{i,j} Cost(A_i, A_j) \quad (1)$$

Na equação **??**, $Cost$ é a pontuação de alinhamento entre duas sequências alinhadas ($A_i$ e $A_j$) e $W_{i,j}$ é o seu peso. Na função $Cost$ inclui penalidades para abertura e extensão de *gaps*. Do pacote SAGA, dois métodos diferentes: (i) penalidades de *gaps* naturais e (ii) penalidades de *gaps* quase naturais. Estes métodos diferem na forma como tratam as lacunas aninhadas. Em ambos os casos, as posições em que nulos aparecem, são removidos. As lacunas de terminais são penalizadas para a extensão, mas não para a abertura.

Os pesos de sequência são tentativas de minimizar informação redundante, com base na relação das sequências. Em MSA, um peso para cada par de sequências é derivado de uma árvore filogenética conectando as seqüências. No caso de CLUSTAL W, calcula-se um peso para cada sequência eo peso do par ($W_{i,j}$) para duas sequências é simplesmente o seu produto. Estes pesos diferem em detalhe embora ambos sejam projetados para uma finalidade similar.

### B. SAGA

Os autores criaram um método de alinhamento de sequências múltiplas chamado SAGA. Este método deriva do algoritmo genético simples descrito por Goldberg[5]. Assim como em qualquer AG, uma população de soluções que evoluem por meio da seleção natural. A estrutura geral da SAGA é mostrada na Figura 1 do artigo (veja artigo original). A população que consideramos é feita de alinhamentos. Inicialmente, uma geração zero ($G_0$) é criada aleatoriamente. O tamanho da população é mantido constante. Para ir de uma geração para a seguinte, os filhos são derivados de pais que são escolhidos por algum tipo de seleção natural, com base em sua aptidão medida pelo FO. Para criar uma filho, um operador é selecionado que pode ser um *crossover* (misturando o conteúdo dos dois pais) ou uma mutação (modificação de um único pai). Cada operador tem uma probabilidade de ser escolhido que é otimizado dinamicamente durante a execução.

Estas etapas são repetidas iterativamente, geração após geração. Durante estes ciclos, novas peças de alinhamento aparecem por causa das mutações ou das combinadas pelos *crossovers*. A seleção assegura que as boas peças sobreviveram e a configuração dinâmica dos operadores ajuda a população a melhorar criando as crianças de que precisa.

Este processo é feito até que a aptidão da população é aumentada, sendo que não mais melhorias possam ser feitas. Todas estas etapas, mostradas na Figura 1 do artigo, podem ser resumidas pelo seguinte pseudo-código:

**Algorithm 1:** Avaliação da população de geração n $G_n$

```
Result: Geração G_i
Criação de G_0
if populacão estável then
    return G_n
end
while Avaliação < Resultado da Função Objetivo
 do
    Seleção dos indivíduos
    Escolha dos pais
    Seleção de Operadores
    Geração de Filhos (G_{n+1})
    n = n+1
end
```

Inicialização. O primeiro passo do algoritmo é a criação de uma população aleatória. Esta geração zero

representa um conjunto de alinhamentos contendo apenas *gaps* terminais. Para criar um destes alinhamentos, é escolhido um desvio aleatório para todas as sequências e cada sequência é movida para a direita, de acordo com o seu deslocamento. As sequências são então preenchidas com sinais nulos para ter o mesmo comprimento. Os alinhamentos da geração zero serão os pais dos filhos usados para preencher a geração um.

Avaliação. Para uma nova geração, o primeiro passo é a avaliação da aptidão de cada indivíduo, utilizando a função objetivo. Assim, quanto melhor o alinhamento, melhor sua pontuação e, portanto, maior a sua aptidão. Se o objetivo é minimizar a FO, então as pontuações são invertidas para dar a aptidão. A descendência esperada (DE) de um alinhamento é derivada da aptidão. A derivação usada é conhecido como amostragem estocástica de resto sem reposição, gerando um intervalo aceitável.

Apenas uma parte (por exemplo 50%) da população deve ser substituída durante cada geração. Esta técnica, conhecida como superposição de geração, significa que metade dos alinhamentos sobreviverão inalterados, a outra metade será substituída pelos filhos. SAGA mantém apenas os melhores indivíduos e substituir os outros. Na prática, todos os indivíduos são classificados de acordo com seu resultado de aptidão, e os mais fracos são substituídos por novos filhos na próxima geração. Somente os mais aptos sobreviverão.

Reprodução. Primeiro, a nova geração é preenchida diretamente com os indivíduos mais aptos da geração anterior (tipicamente 50%). Em seguida, os outros 50% de indivíduos na nova geração são criados por selecionar os pais e modificá-los. Durante a reprodução, o DE é usado como uma probabilidade para cada indivíduo a ser escolhido como um pai. Uma roda é girada onde cada pai potencial tem um número de slots igual ao seu DE. Esta seleção de roda ponderada é realizada até que todos os pais tenham sido escolhidos, evitando uma possível substituição.

Para modificar um pai, um operador tem que ser escolhido. Um operador nada mais é que pequeno programa que irá modificar um alinhamento (por exemplo, adição de um *gap*), e cria-se vários operadores. Cada um deles tem uma probabilidade específica de ser usado. Para criar um filho, um operador é escolhido de acordo com essa probabilidade (girando outra roda ponderada). O operador escolhido é então aplicado ao pai selecionado.

Uma característica a ser considerada da estrutura da população SAGA é a restrição pela a ausência de duplicatas. Na mesma geração, todos os alinhamentos têm que ser diferentes. Esta técnica ajuda a manter um alto nível de diversidade em uma população de pequeno porte. Sendo isso, cada novo filho é verificado para garantir que não é idênticao ao seu ancestral direto. Se não for, será adicionado na nova geração. Caso contrário, ele simplesmente será descartado junto com seu pai e o operador, a fim de evitar problemas repetidos. Este processo é aplicado em um parte dos filhos. O processo de Avaliação / Reprodução será realizado até que a decisão seja tomada, e parar a busca.

Finalização. Não há nenhuma prova válida de que um GA deve atingir o ótimo global, mesmo em uma quantidade infinita de tempo, como há para *Simulated Annealing*. Assim, a decisão de parar a busca tem que ser uma escolha arbitrária, utilizando critérios heurísticos. Critério de parada: SAGA é interrompido quando a pesquisa tem sido incapaz de melhorar para um número especificado de gerações. Esta condição é a mais utilizada quando se trabalha numa população para evitar duplicados.

### *C. Operadores*

Conforme nomenclatura de algoritmos genéticos, dois tipos de operadores estão representados na SAGA: os cruzamentos (*crossover*) e as mutações. Estes programas executam modificações (mutação) ou fusão de alinhamentos de pai (crossover). Em SAGA não existe qualquer distinção entre estes dois tipos no que se refere à forma de aplicação. São concebidos como programas independentes que introduzem um ou dois alinhamentos (os pais) e produzem um alinhamento (o descendente). Cada operador requer um ou mais parâmetros que especificam onde a operação deve ser realizada. Por exemplo, um operador que insere uma nova fenda deve ser informado onde (em que posição no alinhamento) e em que sequências a lacuna deve ser inserido.

Sobre os parâmetros de um operador podem ser escolhidos completamente aleatoriamente, caso um operador é utilizado de forma estocástica. Exemplos de operadores que podem ser selecionados:

- *Crossover*;
- Inserção de *gaps*;
- Troca de blocos em mais pontos de cortes;
- Aplicação de busca de blocos específicos;
- Rearranjo do ótimo local ou sublocal.

## III. Discussão do Artigo

Os autores descrevem métodos mais específicos para utilização como escolha de conjunto de mutação a serem aplicadas ou uma pré-agendamento dos operadores selecionados (adicionando, caso necessário, sequenciamento dos mesmos).

Os métodos descritos anteriormente, e os resultados apresentados no artigo, revelam que existe um considerável ajuste na função objetivos, aplicação de diversos operadores, sendo considerado grande quantidade de combinação dos mesmos ou conforme a necessidade. Outro ponto que destaca-se no artigo é, dado os resultados de testes em cada etapa, pode-se ponderar a aplicação de novos valores para as probabilidades de mutação, de *crossover* ou no uso dos operadores, sempre visando uma melhoria da função objetivo. Este processo, os autores descrevem como *tuning* de habilidade.

Algo que deve ser considerado é que na utilização de algoritmos genéticos, para alcançar uma solução, enfrenta-se muitos problemas técnicos de processamento e tempo de execução, logo, deve-se utilizar de forma eficiente, visando sempre a otimização máxima dos conjuntos de parâmetros, operadores, quantidade gerações e tamanho da população.

## Referências


[1] Cédric Notredame and Desmond G Higgins. Saga: sequence alignment by genetic algorithm. *Nucleic acids research*, 24(8):1515–1524, 1996.

[2] Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of molecular evolution*, 25(4):351–360, 1987.

[3] Charles E Lawrence, Stephen F Altschul, Mark S Boguski, Jun S Liu, Andrew F Neuwald, John C Wootton, et al. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *SCIENCE-NEW YORK THEN WASHINGTON-*, 262:208–208, 1993.

[4] Julie D Thompson, Desmond G Higgins, and Toby J Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680, 1994.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.