UM ALGORITMO GENÉTICO PARA O PROBLEMA DA SEQUÊNCIA MAIS PRÓXIMA

Stênio Sã Rosário Furtado Soares
Universidade Federal de Juiz de Fora (UFJF)
Rua José Lourenço Kelmer, s/n - Campus Universitário, Juiz de Fora/MG.
ssoares@ice.ufjf.br

Luciana Brugiolo Gonçalves
Universidade Federal de Viçosa (UFV)
Av. Peter Henry Rolfs, s/n. Campus Universitário, Viçosa/MG.
lbrugiolo@ufv.br

Adria Ramos de Lyra
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
Av. Governador Amaral Peixoto, s/n. Centro, Nova Iguaçu/ RJ. Brasil.
adrialyra@ufrrj.br

RESUMO

Este trabalho aborda o Problema da Sequência mais Próxima (PSMP), cujo objetivo é encontrar uma sequência em que a distância de Hamming, para um dado conjunto de entrada, seja mínima. Esse problema possui aplicações nas áreas de Bioinformática e Teoria da Codificação. Neste trabalho, foi proposto um novo algoritmo para encontrar boas soluções, mas não necessariamente ótimas, em um tempo computacional razoável. O algoritmo proposto é baseado em Algoritmos Genéticos e estratégias de busca local e Reconexão por Caminhos. A fim de avaliar a qualidade do trabalho, as soluções do algoritmo proposto são comparadas com as obtidas em um trabalho recente da literatura inspirado na metaheurística GRASP. Como conclusão, mostra-se que esta proposta conseguiu encontrar soluções de melhor qualidade em um tempo computacional razoável para a maioria das instâncias testadas.

PALAVARAS CHAVE. Otimização Combinatória, Biologia Computacional, Algoritmo Genético

Otimização Combinatória (OC), Outras Aplicações em Pesquisa Operacional (OA).

ABSTRACT

This work deal with the Closest String Problem (CSP), that aims to find a sequence whose Hamming distance from the member of a given set is miminal. The CSP has applications in Bioinformatics and Coding Theory. In this work, is proposed a new algorithm to get good quality solutions, within a reasonable time. The proposed algorithm is based on Genetic Algorithms and strategies as local search and path relinking. In order to evaluate the quality of this work, the solutions of the proposed algorithm are compared with those obtained in a recent work of literature inspired in the metaheuristic GRASP. As conclusion, it is shown that our proposal was able to return high quality solutions within a reasonable time, for almost all the instances tested.

KEYWORDS. Combinatorial Optimization. Computational Biology. Genetic Algorithm.

Combinatorial Optimization, Other Applications in Operational Research.

1. Introdução

É comum na área de Biologia Molecular problemas onde deseja-se comparar e encontrar uma nova sequência de DNA, RNA ou proteína a partir de um conjunto de sequências de DNA, RNA ou proteínas, que já estejam bem estudadas e anotadas. É comum que sequências semelhantes tenham a mesma função, ou ainda, quando duas sequências de organismos diferentes são similares pode haver uma sequência ancestral comum, (Li, 2002; Liew,2005).

Outro problema com apelo prático para a área de Biologia é a descoberta de padrões comuns para um dado conjunto de entrada de sequencias de DNA. Quando se deseja produzir uma droga genética com uma estrutura similar a um conjunto de sequências de RNA, é comum a necessidade de descoberta de padrões, Lanctot *et al.* (2003). Esses problemas possuem várias aplicações como a busca de regiões conservadas em sequências não alinhadas, a identificação de drogas genéticas, a formulação de sondas genéticas, entre outras (Ben-dor, 1997; Deng, 2003; Gramm, 2002; Lanctot, 2003; Hertz, 1995; Li, 2002; Stormo, 1990; Wang, 2005).

Na área de Biologia Computacional, um dos problemas que envolvem estes tipos de tarefas é chamado de Problema da Sequência Mais Próxima (PSMP), também conhecido na literatura como *Closest String Problem* (CSP). No PSMP, deseja-se determinar a sequência que mais se aproxima, segundo alguma métrica, de um dado conjunto de sequências. Este problema também possui aplicações na área de Teoria da Codificação (Roman, 1992). Em geral, a métrica utilizada no PSMP é a distância de Hamming.

A principal motivação para a escolha do PSMP como tema deste trabalho se deve ao grande número de aplicações do mesmo e ao fato deste ser classificado como um problema da classe NP-difícil (Frances, 1997).

Neste trabalho é proposta uma abordagem heurística baseada em algoritmos evolucionários, que combina estratégias de busca local e reconexão por caminhos, para o Problema da Sequência Mais Próxima. Os resultados são comparados com aqueles obtidos por um algoritmo baseado na metaheurística GRASP (Lyra, 2012).

O restante do trabalho está assim organizado: primeiramente, na Seção 2, é apresentada a definição do problema e alguns conceitos básicos. Posteriormente, na Seção 3, apresentam-se alguns dos trabalhos existentes na literatura para o PSMP. Na Seção 4 é descrito o algoritmo genético proposto neste trabalho. Na Seção 5 são discutidos os resultados computacionais obtidos e finalmente, na Seção 6, as conclusões e trabalhos futuros.

2. Definição do Problema

Considere Σ um alfabeto com k caracteres, sendo que Σ_m representa todas as possíveis sequências de tamanho m usando os caracteres de Σ. Para duas sequências s e t pertencentes a Σ_m, a distância de Hamming entre estas sequências é definida como o número de posições diferentes entre s e t, representada por $d_H(s,t)$. Ou seja, $d_H(s, t) = \sum_{i=1}^{m} \delta(s_i,t_i)$, onde s_i e t_i representam, respetivamente, o i-ésimo caractere das sequências s e t, sendo que $\delta(s_i, t_i) = 0$ se $s_i = t_i$; ou $\delta(s_i, t_i) = 1$, caso contrário ($s_i \neq t_i$).

Para o Problema da Sequência Mais Próxima, dado um conjunto $S_c = \{s_1, s_2, ..., s_n\}$ contendo n sequências onde $S_c \subseteq \Sigma_m$, o objetivo é determinar uma sequência x de tamanho m sobre o alfabeto Σ que possua a menor diferença em relação às cadeias de S_c, ou seja, obter x tal que $argmin_{x\in\Sigma_m} \max_{s^i\in S_c} d_H(x, s^i)$.

Por exemplo, dado o conjunto S_c = {ACGTA, TTACA, CCGCA, GGGGA}, onde Σ ={CTGA} e m=5. Para uma sequência x = TCGCA, considerando cada sequência $s^i \in S_c$, tem-se que $d_H(x,s^1) = 2$, $d_H(x,s^2) = 2$, $d_H(x,s^3) = 1$, $d_H(x,s^4) = 3$. Logo, a distância máxima entre a sequência x e as cadeias de S_c é definida por $d_c = \max_{i=1..4}\{d_H(x, s^i)\} = 3$.

3. Trabalhos Relacionados

Ben-dor *et al.* (1997) apresentam um algoritmo aproximativo utilizando a técnica de arredondamento randômico, com razão de desempenho próximo do valor ótimo para *d* suficientemente grande, onde *d* representa a maior distância a ser minimizada. Este trabalho sugere ainda uma técnica de *derandomização* para o algoritmo proposto. A formulação apresentada a seguir foi usada nesse trabalho da literatura com objetivo de resolver o modelo de relaxação linear e utilizar suas componentes como probabilidades em seu algoritmo de arredondamento randômico.

$$\text{minimizar } d \tag{3.1}$$

Sujeito a

$$\sum_{\sigma\in\sum} x_{j\sigma} = 1, \qquad j = 1\dots m \tag{3.2}$$

$$\sum_{j=1}^{m}\sum_{\sigma\in\sum} x_{j\sigma} \times d_H\left(\sigma, s^i[j]\right) \le d, \qquad i = 1 \dots n \tag{3.3}$$

$$x_{j\sigma} \in \{0,1\}, \qquad j = 1 \dots m, \sigma \in \sum \tag{3.4}$$

Neste modelo, a Equação (3.1) representa a função objetivo a ser minimizada. Considere as variáveis binárias $x_{j\sigma}$ que assumem valor 1, caso o símbolo σ seja associado a posição *j* na sequência, e 0, caso contrário. As restrições (3.2) asseguram que somente um símbolo σ pertencente ao alfabeto será selecionado em cada posição da solução ótima. As desigualdades (3.3) especificam que a distância entre cada sequência do conjunto de entrada *S* e a solução ótima deve ser menor ou igual a *d*. As desigualdades (3.4) indicam o domínio das variáveis do modelo.

Em Lanctot *et al.* (2003) é apresentado outro algoritmo aproximativo, obtido através de pequenas adaptações ao algoritmo de Ben-dor (1997). Para uma visão mais geral dos algoritmos aproximativos existentes na literatura, ver Yamamoto (2004), onde é apresentado um levantamento sobre estes algoritmos e ainda a estratégia de *derandomização* sugerida em Ben-dor (1997) é desenvolvida.

Em Meneses (2004) são propostos três formulações de programação inteira e uma heurística, que é utilizada para gerar limites superiores para a solução ótima. Ainda são apresentados os resultados computacionais de um algoritmo *branch-and-bound* baseado em uma das formulações e na heurística apresentada, executado sobre um conjunto de instâncias geradas aleatoriamente.

Em Gomes (2004) é descrito e implementado um algoritmo paralelo para encontrar soluções aproximadas para o PSMP. Os resultados foram comparados com o ótimo comprovando a qualidade das soluções encontradas.

Nos últimos anos, o PSMP tem recebido bastante atenção e diversas abordagens usando diferentes metaheurísticas tem sido propostas na literatura. Lyra (2012) implementou e analisou, para instâncias existentes na literatura, três heurísticas de construção, uma de busca local baseada na metaheurística VNS e uma heurística de reconexão de caminhos. Em Faro (2010) é proposto um novo algoritmo baseado na metaheurística de Colônia de Formigas. Em Mousavi (2012) um algoritmo baseado na metaheurística GRASP e uma nova função heurística probabilística usada para comparar e avaliar soluções candidatas são propostos.

4. Algoritmos propostos para o PSMP

O algoritmo proposto neste trabalho para o PSMP é uma abordagem baseada na combinação de estratégias de Busca Local e de Reconexão por Caminhos com Algoritmos Genéticos. A proposta inicial deste trabalho era o uso de Algoritmos Genéticos puros, entretanto, dada a natureza computacional do problema, para abreviar a convergência do algoritmo para soluções de boa qualidade, optou-se pelo desenvolvimento da variação de algoritmo evolucionário do tipo Algoritmo Memético.

Algoritmos Genéticos consistem em uma abordagem pertencente à classe dos algoritmos evolucionários, baseada no processo de seleção natural e evolução de uma população (Goldber (1989)). Neste tipo de abordagem, cada solução do problema é referenciada como um indivíduo, que é codificado por uma estrutura composta por elementos associados ao problema. O algoritmo opera sobre um conjunto de indivíduos que compõem a população. Através de operadores de seleção, cruzamento e mutação, o algoritmo simula o processo de seleção natural, pelo qual, indivíduos mais aptos tendem a transferirem suas características para outras gerações através de cruzamentos e, possivelmente por mutação dos seus descendentes.

Para representar uma solução, o cromossomo foi codificado através de um vetor de caracteres, representando uma sequência pertencente a Σ_m. Assim, nenhum processo de decodificação é necessário, uma vez que o próprio cromossomo é uma sequência definida sobre o alfabeto de entrada do PSMP.

4.1 Geração da população inicial

Para gerar os indivíduos da população inicial, foi utilizada uma estratégia não determinística baseada na frequência em que os caracteres ocorrem em cada posição do conjunto de cadeias da entrada S_c. O Algoritmo 1, que recebe como parâmetros a população *pop* onde os novos indivíduos devem ser inseridos e também o número de indivíduos a inserir *numIndividuos*, descreve o funcionamento desta heurística de geração de soluções.

Algoritmo 1 - geraPopulacaoPorRoleta (*pop, numIndividuos*)

```
1:  matFrequencia ← montarMatrizFrequencia( );
2:  contador ← 0;
3:  enquanto (contador < numIndividuos) faça
4:    s ← {};
5:    para (i ← 1 … m) faça
6:      s_i ← runRoletaMatriz(matFrequencia, i);
7:    fim para
8:    pop ← pop ∪ {s}
9:    contador ← contador + 1;
10: fim enquanto
```

Na linha 1 é gerada a matriz auxiliar *matFrequencia*, utilizada para registrar, para cada uma das *m* posições, o número que vezes que cada caractere do alfabeto aparece nas sequências de S_c. Enquanto o número desejado de soluções não for atendido, a cada execução da estrutura de repetição iniciada na linha 3, uma nova solução *s* é gerada. Para definir cada um dos *m* caracteres que compõem a sequência, para cada posição *i* de *s*, representada por s_i, a função *runRoletaMatriz*, na linha 6, sorteia um caractere do alfabeto usando a estratégia da roleta, sendo dada mais maior probabilidade de escolha àqueles caracteres que ocorrem com mais frequência na posição *i*, informação esta registrada em *matFrequencia*. Após a construção da solução *s*, esta é incluída na população *pop* (linha 8).

4.2 Operadores de Seleção, Cruzamento e Mutação

O operador de seleção usado neste trabalho foi o método da roleta. Por este método, a cada indivíduo é associada uma probabilidade deste ser selecionado para cruzamento, probabilidade esta proporcional à sua aptidão.

Quanto ao operador de cruzamento, foi adotado o operador de dois pontos de corte, já que este permite trocas cruzadas de características presentes nos cromossomos. Esta escolha foi influenciada pelo fato de termos uma representação direta do individuo.

O operador de mutação tem como finalidade inserir na população características que não podem ser transmitidas pelos pais aos descendentes, mas que, possivelmente, podem vir a trazer melhora na aptidão destes.

Pelo operador de mutação usado neste trabalho, para cada posição do cromossomo, verifica-se a probabilidade deste de sofrer mutação. Caso a probabilidade seja atendida, os dois filhos sofrerão alteração nesta posição. Esta alteração consiste em escolher, dentre os caracteres mais frequentes no conjunto de sequências da entrada nesta posição, um novo caracter a ser inserido no cromossomo do indivíduo recém-obtido do cruzamento. Na escolha deste caracter, utilizou-se o método da roleta, de forma que os caracteres mais frequentes numa dada posição têm mais chances na escolha.

4.3 Intensificação por troca de janela

Com objetivo de inserir algumas características observadas nas melhores soluções nas demais soluções da população, um processo de intensificação é proposto.

O procedimento é baseado no conceito de consenso entre um conjunto de sequências. Para determinar a sequência consenso *w* associada a um conjunto de sequências *S*, deve-se associar, para cada posição de *w*, aquele caractere do alfabeto que possui maior ocorrência entre as sequências de *S*. Na intensificação *trocaJanelas* é computada uma subsequência consenso *w'*, ou seja, determina-se os componentes de *w* apenas para um intervalo da sequência.

Na Figura 1 pode-se observar um conjunto de quatro soluções compondo o conjunto *S*, a sequência consenso *w* associada a este exemplo e uma subsequência consenso *w'*.

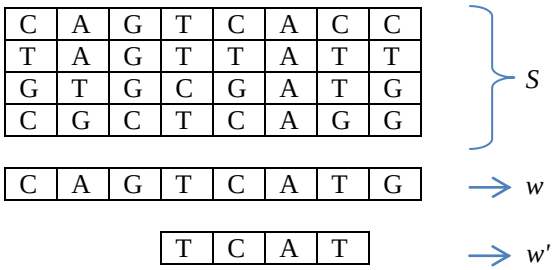

Figura 1: Sequência e subsequência consenso

O Algoritmo 2 descreve a heurística *trocaJanelas*, onde o parâmetro *populacao* contém as soluções pertencentes a população atual, *numIteracoes* indica o número de subsequências consenso que serão avaliadas e o parâmetro *percentualElite* representa o percentual das melhores soluções da população que será considerado para se obter a subsequência consenso.

Nas duas primeiras instruções do algoritmo a população atual é dividida. O parâmetro *percentualElite* define a quantidade das melhores soluções serão atribuídas ao conjunto das soluções elites *popElite*, enquanto as demais soluções são inseridas no conjunto *popNaoElte*. Em cada uma das iterações do algoritmo, controladas pela estrutura apresentada na linha 3, uma subsequência consenso é determinada. Na linha 4 são selecionadas as posições p_1 e p_2 que limitam a janela em que será computada a subsequência consenso. O consenso é obtido entre as soluções pertencentes ao conjunto *popElite*, linha 5.

Para definir as soluções que serão submetidas ao processo de intensificação, na linha 6 é selecionada uma posição *index* da *popNaoElite*. Todas as soluções localizadas após a posição *index* passarão pelo processo de intensificação.

Algoritmo 2 – trocaJanela (*populacao*, *numIteracoes*, *percentualElite*)

```
1.  popElite ← melhoresSoluções(populacao, percentualElite);
2.  popNaoElite ← populacao \ popElite;
3.  para i ← 1 até numIteracoes faça
4.    p1, p2 ← limitesJanela( );
5.    janela ← substringConsenso( popElite, p1, p2);
6.    index ← rand( ) mod |popNaoElite|;
7.    para j ← index até |popNaoElite| faça
8.      s ← popNaoElite[j];
9.      s ← inserirJanela (s, janela);
10.     se (s.fitness < popNaoElite[j].fitness) então
11.       popNaoElite[j] ← s;
12.     fim-se;
13.   fim-para;
14. fim-para;
15. retorne populacao ← popElite ∪ popNaoElite;
```

Na linha 8 do Algoritmo 2 é feita uma cópia, na variável *s*, de uma das soluções não elite. Na linha 9, os caracteres de *s*, localizados entre as posições p_1 e p_2, são substituídos pela subsequência consenso *janela*. Na linha 10 é verificado se a alteração melhorou o custo da solução original. Caso a alteração tenha reduzido o custo da solução, *s* substituirá a *j*-ésima solução de *popNaoElite*, linha 11.

A população resultante deste processo de intensificação é redefinida na linha 15, sendo a união de *popElite* e da nova *popNaoElite*.

4.4 Busca Local

Como forma de intensificar a exploração da região do espaço de busca próxima às soluções obtidas pelo Algoritmo Genético, foi desenvolvida uma estratégia de busca local que faz uso de janelas deslizantes ao longo da cadeia que codifica o indivíduo.

O primeiro passo do algoritmo consiste em obter a cadeia consenso considerando as *k* melhores soluções da população, denominada sequência consenso *w*, como apresentado na Seção 4.3.

Para uma solução *s*, a estratégia visa substituir, de forma iterativa, os caracteres de um intervalo de *s* pelos caracteres equivalentes encontrados em *w*. O tamanho *t* deste intervalo é obtido de forma aleatória no conjunto {1, 2, ..., *m*}.

Dado o valor de *t*, o algoritmo avalia a troca, na sequência da solução atual, a partir da posição inicial até a posição *m*-*t*, dos *t* próximos caracteres. Por se tratar de uma estratégia baseada na busca melhor aprimorante, após avaliar todas as possíveis trocas, aquela que levar a melhor solução é executada, a solução incumbente é atualizada e o processo é reiniciado. Caso nenhuma troca resulte em melhora, a busca local termina.

4.5 Reconexão de caminho

Com o propósito de obter soluções a partir da combinação de características oriundas de diferentes soluções de boa qualidade, é utilizada a abordagem Reconexão de Caminhos (RC). Esta heurística, proposta originalmente por Glover (1996), inicia com duas soluções, sendo explorado o caminho no espaço de vizinhança entre estas em busca de soluções ainda melhores. Para tanto, definida as soluções origem s_{base} e destino s_{guia}, para gerar os caminhos, movimentos são selecionados de forma a introduzir na solução atual (inicialmente s_{base}) atributos presentes na solução destino (s_{guia}). O Algoritmo 3 descreve a estrutura básica deste método que recebe como parâmetro as duas soluções que participarão do procedimento.

Algoritmo 3 – Reconexão de caminhos (s_1, s_2)

1: $s_{base} \leftarrow \text{argmin}_{s\in\{s_1,s_2\}} f(s)$;
2: $s_{guia} \leftarrow \text{argmax}_{s\in\{s_1,s_2\}} f(s)$;
3: $s^* \leftarrow s_{base}$;
4: $\Delta \leftarrow \{j=1..m : s_{base}[j] \neq s_{guia}[j]\}$;
5: **enquanto** $\Delta \neq \emptyset$ **faça**
6: $v^* \leftarrow \text{argmin}_{v\in\Delta} f(s_{base} \oplus v)$;
7: $s_{base} \leftarrow s_{base} \oplus v^*$;
8: $\Delta \leftarrow \Delta \setminus \{ v^* \}$;
9: **se (** $f(s_{base}) < f(s^*)$ **) faça**
10: $s^* \leftarrow s_{base}$;
11: **fim-se;**
12: **fim-enquanto;**
13: **retorne** s^*;

Como visto em Aiex *et al.* (2005), a estratégia de RC tende a gerar melhor resultado quando a solução de partida (s_{base}) é a melhor entre as soluções que participam do processo (RC *backward*). Desta forma, este foi o critério para definir quais seriam as soluções s_{base} e s_{guia}, linhas 1 e 2 do algoritmo. Para registrar a melhor solução encontrada durante a trajetória, a variável s^* é inicializada na linha 3. O conjunto Δ armazena todas as posições em que as soluções s_{base} e s_{guia} diferem. Então, enquanto este conjunto for diferente de vazio (linha 5), ainda haverá movimentos a serem realizados com o objetivo de transformar a solução s_{base}, alterada a cada iteração, na solução destino s_{guia}. Para cada posição $v \in \Delta$, define-se como $s \oplus v$ a solução obtida, a partir de s, pela troca do conteúdo da v-ésima posição pelo conteúdo observado de s_{guia}. Na linha 6, determina-se a posição v^* de Δ de tal forma que $s_{base} \oplus v^*$ resulte na melhor solução, entre os vizinhos avaliados. Este movimento é realizado na linha 7 e esta posição é removida de Δ na linha 8. Se esta alteração tornar a solução base melhor que a solução s^*, esta última é atualizada na linha 10. A melhor solução obtida durante o processo é retornada na última linha do algoritmo.

4.6 Estratégia de Renovação da População

Para gerar a população da iteração $t+1$, são preservadas as melhores soluções da população t. A Figura 2 apresenta a estratégia usada no processo de renovação da população a cada nova geração. Para garantir diversidade na população, as piores soluções da iteração atual são substituídas por novas soluções geradas da mesma forma que as soluções da população inicial. O restante da população é obtido a partir dos operadores de cruzamento e mutação.

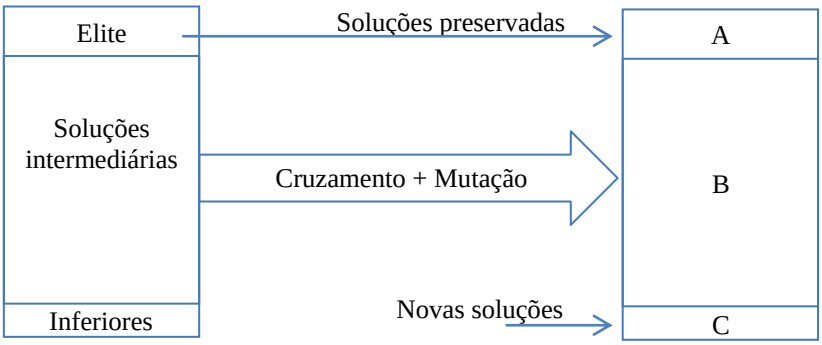

Figura 2: Esquema da estratégia de renovação da população

O Algoritmo 4 descreve o funcionamento do Algoritmo Genético proposto para o PSMP. Como parâmetros, o algoritmo recebe o tamanho da população (*tamPop*), o número máximo de gerações (*numGer*) e as probabilidades associadas ao cruzamento e mutação (*tCros* e *tMut*).

Algoritmo 4 –Algoritmo Genético (*tamPop, numGer, tCros, tMut*)

```
1:  t ← 0;
2:  geraPopulacaoPorRoleta (P_0, tamPop)
3:  faça
4:      t ← t+1;
5:      Gera(P_t);
6:      para i ← 1 até tCros× tamPop faça
7:          pai1, pai2 ← Roleta( P_t-1, fitness);
8:          f1, f1 ←Cruzamento(pai1, pai2);
9:          Mutacao (f1, f2, tMut);
10:         P_t ← f1, f2;
11:     fim-para
12:     geraPopulacaoPorRoleta (P_t, 0,05×tamPop)
13:     P_t ← (P_t-1).Elite ∪ P_t;
14:     BuscaLocal(Best);
15:     Reconexão de Caminhos(Best,Elite);
16:     trocaJanela(Pt,nIter,pConsenso);
17: enquanto ((t < numGer) e (numItSemMelhora < 20));
18: retorne melhorSolucao;
```

5. Experimentos Computacionais

Os algoritmos foram testados para o conjunto de 39 instâncias apresentado em Lyra (2012). O alfabeto utilizado nestas instâncias consiste em Σ = {A, C, T, G}. Os algoritmos foram desenvolvidos na Linguagem C++ e cada heurística foi executada dez vezes em uma máquina Intel® Core™ 2 Quad, com velocidade de 3.20 GHz e 8GB de memória RAM em um sistema operacional Linux.

O Algoritmo Genético foi executado com os seguintes parâmetros, obtidos após testes preliminares: 1000 (mil) indivíduos na população; número máximo de 100 gerações, sendo 20 gerações sem melhora o segundo critério de parada; taxa de cruzamento de 80% e taxa de mutação de 5%.

A Tabela 1 mostra os resultados do algoritmo para o conjunto de instâncias testadas. A primeira coluna mostra as instâncias, cujos nomes seguem a forma nXmYt*, onde X indica o número de sequências e Y indica o numero de caracteres de cada sequência. As demais colunas da tabela mostram a melhor solução obtida pelo algoritmo, a média da geração da melhor solução, a qualidade media dentre as dez execuções, o tempo médio para se obter a melhor solução e o tempo total médio do algoritmo.

Pode-se observar a escalabilidade do algoritmo ao se comparar os tempos gastos nas instâncias pequenas com as maiores instâncias. Além disso, observa-se que, embora tenha-se dado 100 gerações para o algoritmo, em média, o algoritmo encontrou a melhor solução antes da metade deste valor.

Observa-se ainda, pelas duas últimas colunas da tabela, que a convergência do algoritmo não é precoce, já que a diferença entre o tempo médio em que se obteve o melhor indivíduo e o tempo total do algoritmo é razoável.

Tabela 1 - Resultados computacionais

Instancia	Melhor Solução	Média da Geração	Qualidade Média	Tempo (seg)	
				Média Tempo Melhor	Média Tempo Total
n10m1000tai1	589	30,4	591,80	23,22	36,50
n10m1000tai2	583	27,3	585,20	24,01	35,60
n10m1000tai3	588	22,2	591,56	18,15	32,14
n10m2000tai1	1173	28,4	1182,10	44,31	68,93
n10m2000tai2	1180	39,8	1189,00	58,15	82,46
n10m2000tai3	1166	32,7	1171,70	48,62	72,54
n10m3000tai1	1751	24,0	1755,60	61,66	98,74
n10m3000tai2	1762	22,4	1766,40	56,60	94,00
n10m3000tai3	1749	9,0	1753,60	40,06	71,50
n10m300tai1	178	32,3	179,60	15,01	17,85
n10m300tai2	179	29,5	180,30	15,81	17,63
n10m300tai3	177	18,1	179,00	9,26	17,12
n10m4000tai1	2338	29,3	2345,50	102,94	154,78
n10m4000tai2	2335	26,0	2342,00	102,95	147,81
n10m4000tai3	2340	28,3	2352,40	109,91	153,81
n10m400tai1	236	30,1	237,30	15,80	21,72
n10m400tai2	235	32,5	236,60	14,79	21,83
n10m400tai3	238	35,7	241,80	18,59	22,94
n10m5000tai1	2917	26,2	2932,50	136,10	208,10
n10m5000tai2	2916	38,7	2928,20	173,46	242,85
n10m5000tai3	2912	30,0	2920,30	141,05	211,84
n10m500tai1	296	22,1	297,40	15,24	20,40
n10m500tai2	291	15,1	292,10	6,64	18,72
n10m500tai3	291	25,1	293,10	15,79	21,49
n20m1000tai1	641	26,1	643,00	24,28	40,52
n20m1000tai2	643	39,1	646,10	34,08	50,42
n20m1000tai3	644	25,0	647,00	31,50	43,45
n20m2000tai1	1275	19,8	1278,50	43,29	72,20
n20m2000tai2	1278	42,7	1281,70	76,87	108,06
n20m2000tai3	1286	29,8	1288,70	60,77	87,64
n20m3000tai1	1912	22,7	1915,00	76,83	120,06
n20m3000tai2	1910	43,7	1912,50	119,08	166,46
n20m3000tai3	1926	28,1	1931,50	83,27	130,51
n20m4000tai1	2545	27,0	2548,10	117,23	183,50
n20m4000tai2	2551	19,5	2558,00	91,73	162,66
n20m4000tai3	2554	24,8	2565,00	112,91	181,67
n20m5000tai1	3175	23,4	3185,00	143,65	232,38
n20m5000tai2	3178	23,5	3183,10	144,56	231,90
n20m5000tai3	3191	29,0	3200,30	167,94	257,08

A Tabela 2 mostra a comparação entre os melhores resultados obtidos pela abordagem proposta e aqueles apresentados em Lyra (2012), em que a autora apresenta uma abordagem baseada em GRASP com Reconexão de Caminhos. Os valores destacados em negrito indicam as instâncias onde cada abordagem obteve a melhor solução.

Pode-se observa que, das 39 instâncias, apenas em oito o Algoritmo Genético proposto não obteve solução melhor ou igual à apresentada pelo algoritmo da literatura. Além disso, pode-se verificar que o tempo médio em que a melhor solução foi obtida é quase cinco vezes menor que a abordagem GRASP da literatura.

Tabela 2 - Comparação com a literatura

Instancia	AG		GRASP	
	Melhor Solução	Tempo (seg)	Melhor Solução	Tempo (seg)
n10m1000tai1	**589**	23,22	592	3,83
n10m1000tai2	**583**	24,01	584	7,18
n10m1000tai3	**588**	18,15	592	6,15
n10m2000tai1	**1173**	44,31	1180	18,38
n10m2000tai2	**1180**	58,15	1187	21,62
n10m2000tai3	**1166**	48,62	1174	24,63
n10m3000tai1	**1751**	61,66	1756	46,86
n10m3000tai2	**1762**	56,60	**1762**	39,97
n10m3000tai3	**1749**	40,06	1757	55,02
n10m300tai1	**178**	15,01	184	1,15
n10m300tai2	**179**	15,81	180	1,15
n10m300tai3	**177**	9,26	180	0,7
n10m4000tai1	2338	102,94	**2328**	95,3
n10m4000tai2	2335	102,95	**2332**	208,66
n10m4000tai3	**2340**	109,91	2343	111,32
n10m400tai1	**236**	15,80	240	0,94
n10m400tai2	**235**	14,79	236	0,85
n10m400tai3	**238**	18,59	243	1,58
n10m5000tai1	**2917**	136,10	2932	236,91
n10m5000tai2	2916	173,46	**2909**	291,75
n10m5000tai3	**2912**	141,05	2919	286,89
n10m500tai1	**296**	15,24	297	1,28
n10m500tai2	**291**	6,64	293	1,07
n10m500tai3	**291**	15,79	293	1,32
n20m1000tai1	**641**	24,28	642	28,1
n20m1000tai2	**643**	34,08	644	57,22
n20m1000tai3	**644**	31,50	**644**	119,08
n20m2000tai1	**1275**	43,29	1276	166,91
n20m2000tai2	**1278**	76,87	1279	229,48
n20m2000tai3	1286	60,77	**1279**	229,48
n20m3000tai1	**1912**	76,83	**1912**	649,3
n20m3000tai2	1910	119,08	**1904**	974,42
n20m3000tai3	1926	83,27	**1921**	640,53
n20m4000tai1	2545	117,23	**2540**	725,21
n20m4000tai2	2551	91,73	**2547**	969,66
n20m4000tai3	**2554**	112,91	2557	1202,49
n20m5000tai1	**3175**	143,65	3181	1611,01
n20m5000tai2	**3178**	144,56	3775	1793,84
n20m5000tai3	**3191**	167,94	3781	2284,85
Média tempo		66,57		337,08

6. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem evolucionária para o PSMP baseada em Algoritmo Genético combinado com Reconexão de Caminhos e Busca Local utilizando uma estratégia denominada janela deslizante.

Os resultados foram comparados com os obtidos por uma abordagem GRASP, onde pode-se observar a eficiência do algoritmo proposto. Verificou-se que os tempos demandados pela abordagem desenvolvida são bastante competitivos.

Como trabalhos futuros, sugere-se um estudo de ajuste dos parâmetros do Algoritmo Genético, bem como dos parâmetros referentes à busca local, troca de janelas e Reconexão de Caminhos. Além disto, é interessante analisar os resultados utilizando outras métricas de comparação, além de média e melhor solução. Por tratar-se de problema da área de Biologia Computacional, é proposta de trabalho futuro testar a abordagem apresentada para um universo maior de instâncias e comparar com outros trabalhos da literatura.

Agradecimentos

Ao CNPq e a FAPERJ pelo suporte financeiro que possibilitou a realização desse trabalho de pesquisa.

Referências

Aiex, **R.M., Pardalos, P.M., Resende, M.G.C. e Toraldo, G.** *GRASP with path-relinking for three-index assignment*. INFORMS J. on Computing, 17:224–247, 2005.

Ben-dor, A., Lancia, G., Perone, J. e Ravi, R. Banishing Bias from Consensus Sequences, Combinatorial Pattern Matching. *8th Annual Symposium*, Springer-Verlag, Berlin, 1997.

Deng, X., Li, G., Li, Z., Ma, B. e Wang, L. Genetic Design of Drugs without Side-Effects.SIAM Journal on Computing, 32: 1073–1090, 2003.

Faro, S. e Pappalardo, E. Ant-CSP: An Ant Colony Optimization Algorithm for the Closest String Problem. In SOFSEM 2010: Theory and Practice of Computer Science. Lecture Notes in Computer Science, vol 5901, pages 370-381, 2010.

Frances, M. e Litman, A. On covering problems of codes, *Theor. Comput. Systems*, 30:113-119, 1997.

Glover, F. Tabu search and adaptive memory programing - Advance, applications and challenges. In: . : R. S. Barr, R. V. Helgason and J. L. Kennington, p. 1_75, 1996.

Goldberg, **E.** Genetic Algorithms in Search, Optimization, and Machine Learning. EUA: Addison-Wesley, 1989.

Gomes, F., Meneses, C., Pardalos, P. e Vianna, V. Parallelal Algorithm for the Closest-String Problem. *INFORMS Journal on Computing*, Vol. 16, No. 4, 419-429, 2004.

Gramm, J., **Huffner, F. e Niedermeier, R.** Closest Strings, Primer Design, and Motif Search. Currents in Computational Molecular Biology, poster abstracts of RECOMB 2002: 74–75. pp. 74–75, 2002.

Hansen, P. e Mladenovic, N. Variable Neighborhood Search for the p-median, *Location Science*, vol. 5, 207-226, 1997.

Hertz, G. e Stormo, G. Identification of Consensus Patterns in Unaligned DNA and Protein Sequences: A Large-Deviation Statistical for Basis Penalizing Gaps, *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research*, 201-216, 1995.

Lanctot K, Li M, Ma B, Wang S e Zhang L. Distinguishing String Selection Problems. (2003).Information and Computation, 185: 41–55. A preliminary version appeared in Proceedings of 10th ACM-SIAM Symposium on Discrete Algorithms, 633–642.

Li, M., Ma, B. e Wang, L. On the Closest string and substring problems. *Journal of the ACM*, 49(2), 157-171, 2002.

Liew, A., Yan, H. e Yang, M. Pattern recognition techniques for the emerging field of bioinformatics: A review. Pattern Recognition, 38: 2055–2073, 2005.

Lyra, A. Métodos Exatos e Heurísticos para o Problema da Sequência Mais Próxima. *Proceedings of the XVI Latin-Ibero American Conference on Operations Research and XLIV Brazilian Symposium on Operations Research*, 2012.

Ma, B. e Sun, X. More efficient algorithms for closest string and substring problems. *12th Annual International Conference on Research in Computational Molecular Biology* (RECOMB'08). Springer-Verlag, Berlin, 396-409, 2008.

Meneses, C., Lu, Z., Oliveira, C. e Pardalos, P. Optimal Solutions for the Closest-String Problem via Integer Programming. *INFORMS Journal on Computing*, vol. 16, No. 4, 419-429, 2004.

Michalewicsj, Z. *Genetic Algorithms + Data Structures = Evolution Programs* , Springer, 1996.

Mousavi, S. e Esfahani, N. A GRASP algorithm for the Closest String Problem using a prabability-based heuristic. Computers & Operations Research, vol. 39, pages 238-248, 2012.

Roman, S. Coding and Information Theory. Graduate Texts in Mathematics. Springer-Verlag, vol 134, 1992.

Stormo, G.D., Consensus patterns in DNA, in Doolittle, R. F., ed., *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences, Methods in Enzymology*, vol. 183, Academic Press, 211-221, 1990.

Wang, L. e Dong, L. Randomized Algorithms for Motif Detection. Journal of Bioinformatics and Computational Biology, 3: 1038–1052, 2005.

Yamamoto, K., Arredondamento Randômico e o Problema da Sequência mais Próxima, *Tese de Mestrado*, IC/UFF, Universidade Federal Fluminense. (2004).