MÉTODOS EXATOS E HEURÍSTICOS PARA O PROBLEMA DA SEQUÊNCIA MAIS PRÓXIMA

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

adrialyra@ufrrj.br

RESUMO

Neste trabalho lidamos com o Problema da Sequência mais Próxima (PSMP). Foram implementadas e analisadas, para instâncias existentes na literatura, três heurísticas de construção, uma de busca local baseada na metaheurística VNS e uma heurística de reconexão de caminhos. Além disso, são feitas algumas comparações das soluções obtidas pelo algoritmo exato de *Branch and Cut* com as das heurísticas propostas, com objetivo de avaliar a qualidade das soluções obtidas. Como conclusão obtemos que as melhores soluções heurísticas foram obtidas pelo método não-determinístico guloso, seguido de uma busca local baseada no método VNS e com a aplicação da heurística de religação de caminhos, ambas adaptadas para o PSMP. Comparando as soluções obtidas por essa combinação de métodos com o método exato, vimos que as soluções possuem excelente qualidade, com erros de no máximo 0,03%.

PALAVARAS CHAVE. Otimização Combinatória, Biologia Computacional, Heurísticas.

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

ABSTRACT

In this work we deal with the Closest String Problem (CSP). We implement and analyze three construction heuristics, one local search based on the VNS metaheuristic and a path relinking method, for existent instances in the literature. Besides, we compare exact solutions with the ones obtained by the methods proposed to evaluate the quality of our solutions. We conclude that the best solutions were obtained by the greedy non-deterministic heuristic, followed by the local search based on the metaheuristic VNS and the application of the Path Relinking heuristic. When comparing the solutions obtained by this combination of methods with the exact one, we realize that our methods return high quality solutions, with maximum errors of 0.03%.

KEYWORDS. Combinatorial Optimization. Computational Biology. Heuristics.

Combinatorial Optimization, Other Applications in Operational Research.

1. Introdução

Em muitos problemas de biologia molecular deseja-se comparar e encontrar regiões comuns ou que sejam próximas a um conjunto de sequências de DNA, RNA ou proteínas. Esses problemas possuem várias aplicações como: busca de regiões conservadas em sequências não alinhadas, identificação de drogas genéticas, formulação de sondas (probes) genéticas, entre outras, veja: Hertz (1995), Stormo (1990). No Problema da Sequência Mais Próxima (PSMP), também conhecido na literatura como *Closest String Problem* (CSP), deseja-se determinar a sequência que mais se aproxima, segundo alguma métrica, de um dado conjunto de sequências.

A principal motivação para a escolha do PSMP como tema deste trabalho se deve ao fato dele ser classificado como NP-difícil. Em Lanctot (1999) foi mostrado inicialmente, que o Problema da 3-SAT (NP-completo) é redutível ao Problema da Sequência Mais Distante na versão de decisão (PSMD - decisão) e que o PSMD - decisão é redutível ao PSMP - decisão.

Nesse trabalho serão apresentados primeiramente a definição do problema e alguns conceitos básicos necessários, seção 2. Posteriormente, na seção 3, apresenta-se alguns dos trabalhos existentes na literatura para o PSMP. Na seção 4, serão apresentadas algumas heurísticas existentes na literatura e as propostas neste trabalhos, que foram implementadas para o PSMP. Dentre elas, temos heurísticas de construção, de busca local e uma de reconexão de caminhos. Além disso, também apresentamos a formulação matemática usada para obtenção das soluções exatas. Na seção 5, apresentamos os resultados computacionais obtidos e finalmente, na seção 6, as conclusões e trabalhos futuros

2. Definição do Problema e Conceitos Básicos

Sejam s e t duas sequências de mesmo tamanho, *i.e.*, $|s| = |t|$ e $d_H(s, t)$ a distância de Hamming entre elas, onde $d_H(s, t)$ é definido como o número de posições diferentes entre s e t. Por exemplo, se s = ACT e t = CCA, então $d_H(s, t) = 2$.

Como entrada de dados para o PSMP tem-se: um conjunto finito $S_c = \{ s^1, s^2, ..., s^n \}$ de n sequências, todas de tamanho m sobre um alfabeto Σ. Como objetivo deseja-se encontrar uma sequência x de tamanho m sobre Σ minimizando d_c de modo que para toda sequência $s^i \in S_c$, $d_H(x, s^i) \leq d_c$.

Exemplo: Seja S_c = {ACGT, TTAC, CCGC, GGGG}. Logo, x = TCGC é a sequência mais próxima à cada uma das sequências de S_c e o menor valor de d_c é 3.

3. Trabalhos Relacionados

Em Ben-dor (1997) é apresentado um algoritmo aproximativo, utilizando a técnica de arredondamento randômico, com razão de performance próximo do valor ótimo para d suficientemente grande, onde d representa a maior distância a ser minimizada. Ainda, neste trabalho, é sugerida uma técnica de *derandomização* para o algoritmo proposto.

Em Lanctot (1999) é apresentado outro algoritmo aproximativo, obtido através de pequenas adaptações ao algoritmo de Ben-dor (1997). Os autores exibem um fator de aproximação constante independente de n, m e d_c. Basicamente, na estratégia adotada pelos autores, considera-se as k posições não coincidentes ($k \leq n$) entre as duas piores sequência de S. Esse problema ainda possui um esquema de aproximação polinomial, que foi mostrado em Li (2002).

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 Ma (2008) são desenvolvidos algoritmos de parâmetros fixos, cuja complexidade é $O\left(n|\Sigma|^{o(d)}\right)$, onde raio d é o parâmetro e Σ é o alfabeto. Como resultado tem-se um algoritmo de tempo polinomial para d = O(log n) e Σ de tamanho constante. Além disso, em Ma (2008) também é apresentado um Esquema de Aproximação de Tempo Polinomial, cuja complexidade é $O(n^{O(\varepsilon^{-2})})$.

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. Os resultados mostraram que não é possível resolver exatamente instâncias de tamanho moderado.

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.

4. Métodos propostos para o PSMP

Nesta seção serão apresentadas três heurísticas de construção, uma heurística de busca local e o algoritmo de reconexão de caminhos propostos neste trabalho. Além da formulação usada para gerar as soluções exatas.

Essas três técnicas são aplicadas da seguinte forma: primeiramente, uma solução viável é construída, por um algoritmo de construção. Em seguida, aplica-se o algoritmo de busca local para tentar melhorar a qualidade da solução obtida na fase anterior. Se a primeira fase é não-determinística, pode-se gerar várias soluções iniciais e aplicar a busca local após cada uma delas, com a expectativa de que em alguma iteração uma solução melhor seja encontrada. Como as soluções geradas de maneira não-determinística são diferentes umas das outras, faz sentido guardar um *pool* com as melhores soluções e aplicar um algoritmo de reconexão de caminhos em algumas iterações. Dessa maneira, espera-se encontrar uma solução de melhor qualidade entre uma solução que saiu da busca local e uma pertencente ao *pool* de soluções.

4.1. Algoritmos de Construção

Nesta seção serão apresentados os três algoritmos de construção para o PSMP utilizados neste trabalho. Esses algoritmos são utilizados na construção de uma solução viável.

4.1.1. Algoritmo 2-aproximado natural

Esse algoritmos foi proposto em Yamamoto (2004). Considere um conjunto $S = \{s^1, s^2, ..., s^n\}$ de sequências, todas de tamanho m, sobre um alfabeto Σ. Seja $s^{opt} \in \Sigma^m$ uma solução ótima e d_{opt} a distância ótima para o PSMP.

Nessa heurística, seleciona-se uma sequência s qualquer do conjunto de entrada S como solução do problema. Desta forma, pode-se garantir que $d_H(s, s^i) \leq 2d_{opt}$ para $i = 1 .. n$. Esse resultado pode ser provado pela desigualdade triangular: $d_H(s, s^i) \leq d_H(s, s^{opt}) + d_H(s^i, s^{opt})$. Como para qualquer $s^i \in S$, $d_H(s^i, s^{opt}) \leq d_{opt}$ pode se garantir que para qualquer sequência $s^i \in S$, $d_H(s, s^i) \leq 2d_{opt}$, sendo $s \in S$ uma sequência escolhida arbitrariamente como solução do problema.

A complexidade deste algoritmo é O(1) para escolher a sequência e O(n) para carrega-la na memória. Logo, a complexidade total do algoritmo é de O(n).

4.1.2. Algoritmo proposto

Nesta seção, é apresentado o algoritmo de construção proposto neste trabalho.

Inicialmente, foi usada a estratégia de verificar cada uma das posições nas sequências de entrada e atribuir à mesma posição da sequência de saída o caractere que tivesse mais incidências nas respectivas posições das sequências do conjunto de entrada.

Posteriormente, um segundo algoritmo foi gerado, baseado no anterior. Neste algoritmo, ao invés de se escolher deterministicamente o caractere mais popular de determinada posição, ele apenas recebe uma chance maior de ser escolhido. Um esquema de escolha aleatória foi criado com o objetivo de que todos os caracteres tenham chance de serem escolhidos, entretanto, o mais popular naquela posição vai ter uma chance maior de entrar. Esse esquema é baseado no sistema de roleta russa dos Algoritmos Genéticos, ver Michalewicsj (1996). O parâmetro associado a chance do caractere mais popular ser escolhido ainda deve passar por mais testes.

A complexidade tanto do algoritmo determinístico quanto do não determinístico é O(mn). O(m) para percorrer cada uma das n colunas do conjunto S.

4.2. Algoritmos de Busca Local propostos

Os métodos de busca local visam encontrar soluções melhores na vizinhança das soluções geradas pelas heurísticas de construção. Em outras palavras, visam encontrar ótimos locais nessas vizinhanças.

4.2.1. Busca Local VNS

Baseado na metaheurística VNS (*Variable Neighbourhood Search*) proposta por Hansen (1997), que consiste basicamente em uma sistemática troca de vizinhança associada a um algoritmo randômico de busca local, desenvolvemos nosso algoritmo de busca.

Contrariamente a outras metaheurísticas baseadas em métodos de busca local, a VNS não segue uma trajetória, mas explora incrementalmente vizinhanças distantes da solução corrente. O VNS utiliza um conjunto pré-definido de estruturas de vizinhança N_k, (k = 1, ..., k_{max}), sendo N_k o conjunto de soluções na k-ésima vizinhança de s, onde s é solução na qual a busca local esta sendo realizada.

A cada iteração da busca local, o algoritmo vai checar se houve melhora na qualidade da solução. Em caso afirmativo, o algoritmo continuará explorando a mesma vizinhança, caso contrário, ele passará a explorar a próxima vizinhança, até que o critério de parada esteja satisfeito.

No caso desse problema especificamente, foram utilizadas 20 estruturas de vizinhança, ou seja, é possível se trocar até 20 posições diferentes de uma solução inicial. Nesse caso, as posições são escolhidas de maneira aleatória, assim como o símbolo do alfabeto que irá substituir o caractere corrente.

Veja o Algoritmo 1 onde é apresentado um algoritmo básico da busca local VNS.

1. **Algoritmo BL-VNS:** Heurística de Busca Local VNS
2. $s = s^0$ // s é a solução corrente
3. $s^* = s^0$ // s é a melhor solução até o momento
4. $f(s^*) = f(s^0)$ // $f(s^*)$ é custo de s^*
5. **Enquanto** (critério de parada) **Faça**
6. $k = 1$
7. **Enquanto** ($k \leq k_{max}$) **Faça**
8. Gerar aleatoriamente uma solução s^1 na vizinhança de $N_k(s^*)$
9. Aplicar busca local a partir de s^1 obtendo a solução s^2
10. **Se** ($f(s^2)$ for melhor do que $f(s^*)$) **então**
11. $s^* = s^2$
12. $s = s^2$
13. $f(s^*) = f(s^2)$
14. $k = 1$
15. **Senão**
16. $k = k + 1$

Algoritmo 1. Algoritmo de Busca Local baseado na Metaheurística VNS.

4.2.2. Reconexão de Caminhos (RC)

A Reconexão de Caminhos (RC) é um método que vem sendo cada vez mais aplicado na literatura, cuja ideia básica é que entre duas soluções de boa qualidade pode haver uma melhor ainda.

A técnica consiste em explorar trajetórias que conectam soluções de alta qualidade, começando de uma *solução base* e gerando um caminho na vizinhança dessa solução na direção de outra solução, chamada de *solução alvo*, ver Figura 1. Esse caminho é gerado selecionando-se movimentos que introduzam atributos da solução alvo na solução base. A cada passo, todos os movimentos que incorporam atributos da solução alvo são analisados e o melhor movimento é escolhido.

Maiores detalhes desta técnica podem ser encontrados em Glover (2000) onde é apresentado um levantamento sobre reconexão de caminhos.

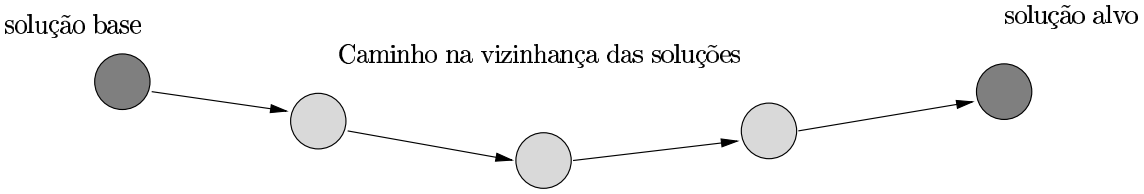

Figura 1. Reconexão de Caminhos: exploração de trajetórias que conectam soluções de alta qualidade.

No Algoritmo 2, exibimos um esquema de uma RC para um problema de minimização. Note, na linha 2 as definições das solução base (x_s) e da solução alvo (x_t). Na linha 3, temos a definição do conjunto $\Delta(x_s, x_t)$, o conjunto dos movimentos necessários para se atingir x_t partindo de x_s. Na linha 4, definimos o melhor valor da função objetivo (f^*) como sendo o menor valor entre os valores das funções objetivos de x_s e x_t. O algoritmo vai seguir executando as linhas 7 a 12 até que não haja mais movimentos possíveis em $\Delta(x_s, x_t)$. Na linha 7, o melhor movimento de $\Delta(x_s, x_t)$ é selecionado. Na linha 8, retiramos de $\Delta(x_s, x_t)$ o movimento selecionado. Na linha 9, realizamos o movimento na solução parcial. Na linha 10, verificamos se este movimento gera alguma melhora no valor da função objetivo, em caso positivo, nas linhas 11 e 12, atualizamos os valores da função objetivo e da melhor solução, respectivamente.

1. **Algoritmo PR:** HEURÍSTICA DE RECONEXÃO DE CAMINHOS
2. Sejam x_s a solução base e x_t a solução alvo
3. Seja $\Delta(x_s, x_t)$ o conjunto dos movimentos necessários para se atingir x_t partindo de x_s
4. $f^* = min\{f(x_s), f(x_t)\}$
5. $x = x_s$
6. **enquanto** $\Delta(x, xt) \neq \emptyset$ **faça**
7. m^* = selecione o melhor movimento de $\Delta(x_s, x_t)$;
8. $\Delta(x_s, x_t) = \Delta(x_s, x_t) \backslash \{m^*\}$
9. realiza o movimento m^* em x
10. **se** $f(x) < f^*$ **então**
11. $f^* = f(x)$;
12. $x^* = x$;

Algoritmo 2. Algoritmo de Reconexão de Caminhos.

Neste trabalho, a reconexão de caminhos é feita sempre da melhor solução (solução base) para a solução de pior qualidade (solução alvo). Uma vez que, esta estratégia é a que tem se mostrado mais atraente na literatura. Uma solução sempre vem da busca local e a outra solução pertence ao *pool* de soluções. A cada iteração que o método julga ser conveniente a aplicação da RC, a reconexão vai ser feita entre uma solução que veio da busca local e todas as demais pertencentes ao *pool*. É claro que a reconexão só tem sentido de ser aplicada ser for em soluções com um certo percentual de distinção, em soluções muito parecidas ela não surte o efeito esperado.

Além disso, a RC não é realizada nas primeiras iterações dos métodos propostos, pois nesses casos é provável que *pool* ainda não esteja com soluções de muito boa qualidade. Logo, não há justificativa para o emprego da RC.

4.3. Composição dos métodos analisados

Para realizar os testes computacionais, as heurísticas de construção, busca local e RC, foram combinadas resultando nos seguintes métodos.

No Algoritmo 3, apresenta-se o método composto pela Heurística 2-aproximada e pela Busca Local VNS. O critério de parada utilizado foi o número de iterações.

No Algoritmo 4, tem-se o método 2, composto pela heurística determinística e pela busca local VNS. Observe que este método é executado apenas uma vez.

No Algoritmo 5, observa-se o método 3, que consiste da heurística não - determinística seguida da busca local VNS. Como as soluções geradas na fase de construção são distintas umas das outras, este procedimento se repete enquanto o critério de parada não for satisfeito, que nesse caso, é baseado no número de iterações.

O método 4, é obtido acrescentando-se o módulo de reconexão de caminhos, após a fase de busca local do método 3. É importante lembrar que o módulo de RC não é aplicado em todas as iterações.

1. **Método 1:** HEURÍSTICA 2-APROXIMADA + BUSCA LOCAL
2. **Enquanto** (critério de parada) **faça**
3. Escolher uma seqüência aleatóriamente do conjunto de entrada S
4. Realizar a busca local VNS sobre a seqüência escolhida
5. Guardar a melhor solução;

Algoritmo 3. Método 1: Heurística 2-aproximada + Busca Local VNS

1. **Método 2:** HEURÍSTICA DETERMINÍSTICO + BUSCA LOCAL
2. Determinar a seqüência como no algoritmo determinístico proposto
3. Realizar a busca local VNS sobre a seqüência determinada
4. Retonar a seqüência de saída;

Algoritmo 4. Método 2: Heurística Determinística + Busca Local VNS

1. **Método 3:** HEURÍSTICA NÃO DETERMINÍSTICA PROPOSTA + BUSCA LOCAL
2. **Enquanto** (critério de parada) **faça**
3. Montar uma seqüência da maneira proposta
4. Realizar a busca local VNS sobre a seqüência montada
5. Guardar a melhor solução;

Algoritmo 5. Método 3: Heurística não Determinística + Busca Local VNS

4.4. Formulação matemática

Nessa subseção, apresentamos a formulação usada no algoritmos de *Branch and Cut* implementado. Essa formulação foi usada em Ben-dor (1997), com objetivo de resolver o modelo de relaxação linear e utilizar suas componentes como probabilidades em seu algoritmo de arredondamento randômico.

$$d_{opt} = \min d \qquad (4.1)$$

$$s.a. \begin{cases} \sum_{\sigma \in \sum} x_{j,\sigma} = 1, j = 1..m & (4.2) \\ \sum_{j=1}^{m} \sum_{\sigma \in \sum} x_{j,\sigma} d_H(\sigma, s_i[j]) \le d, i = 1..n & (4.3) \\ x_{j,\sigma} \in \{0,1\}, j = 1..m; \sigma \in \sum & (4.4) \end{cases}$$

No modelo anterior, (4.1) representa a função objetivo que se deseja minimizar. As restrições (4.2) asseguram que somente um símbolo $\sigma \in \Sigma$ será selecionado em cada posição da solução ótima. As desigualdades (4.3) especificam que a distância entre cada sequência de S do conjunto de entrada e a solução ótima, deve ser menor ou igual a d_{opt}. As desigualdades (4.4) indicam que somente valores 0 ou 1 podem ser atribuídos às variáveis $x_{j,\sigma}$. Foi utilizado o *software* XPRESS-MP, da Dash Optimization, para implementar o algoritmo de *Branch and Cut*.

5. Experimentos Computacionais

Os algoritmos foram testados para um conjunto de 39 instâncias. O alfabeto utilizado foi o mesmo para todas elas: A, C, T, G. Cada heurística foi executada três vezes, inclusive a com a fase de construção determinística, que por ser seguida de uma busca local não determinística pode gerar resultados diferentes a cada execução.

A Tabela 1 mostra as médias dos resultados obtidos após três execuções de cada um dos métodos (1, 2, 3 e 4). As colunas Instância, Dist., Temp. e Iter. representam as instâncias testadas, a distância de Hamming, tempo de execução em segundos e a iteração onde a melhor solução foi encontrada, respectivamente.

Para gerar as instâncias de n10m300tai1 à n10m500tai1 foi utilizado o gerador e o script disponíveis em *http://grove.ufl.edu/~claudio*. O mesmo utilizado para gerar as instâncias em Meneses (2004). Já as instâncias de n10m1000tai1 à n20m5000tai3 são instâncias existentes na literatura que formam gentilmente cedidas pelo professor Valdísio (www.lia.ufc.br/ ~valdisio).

Para efeito de testes o número de execuções dos 4 métodos foi de 1.000 iterações. Como estes métodos são heurísticos, a expectativa é de que aumentando-se o número de iterações a qualidade das soluções também sejam melhores.

Na Tabela 1 o algoritmo que retornou as melhores soluções foi o quarto método proposto. Exceto na instância n10m400tai2, em que ele empata com o algoritmo 3. Note também que o método do algoritmo 4 é mais caro computacionalmente e isto se deve ao módulo de Reconexão de Caminhos (RC), que investiga todas as trocas de bases possíveis antes de tomar uma decisão. Por outro lado, repare que as médias das iterações onde este algoritmo encontra sua melhor solução é menor do que as médias das iterações dos demais algoritmos, com exceção do método 2, que tem somente uma iteração devido a sua fase de construção ser determinística.

Repare nos Gráficos 1, 2 e 3 os contrastes entre a qualidade das soluções geradas pelos algoritmos propostos e os seus respectivos tempos de execução. No gráfico da Figura 1.A, por exemplo, tem-se uma comparação com as médias das soluções obtidas para a instância n10m1000tai1. O Algoritmo 1 gera as soluções de pior qualidade. No Gráfico 1.B pode se observar o comportamento em relação ao tempo dos algoritmos estudados, para a instância n10m1000tai1. Naturalmente, o algoritmo que demanda menor tempo computacional é o exibido no Algoritmo 2, que por ter uma fase de construção determinística é executado uma única vez. Em contra partida, o algoritmo mais lento e também o que gera as melhores soluções é o Algoritmo 4. Esse algoritmo é, sem dúvida, o mais lento devido ao módulo de Reconexão de Caminhos. Um comportamento parecido ocorre para as demais instâncias.

Na Tabela 2 apresenta-se os melhores resultados encontrados pelos algoritmos testados. Essa tabela possui resultados ligeiramente melhores do que a tabela anterior (Tabela 1) pois a anterior se comparam as médias de três soluções obtidas pelos algoritmos.

	2APRO + BL			PROPD+ BL			PROPND+ BL			PROPND+ BL + PR		
Instância	Dist.	Temp	Iter.	Dist.	Temp	Iter.	Dist.	Temp	Iter.	Dist.	Temp	Iter.
n10m300tai1	221	1	419	190	0	1	186	1	218	**184**	**0**	**136**
n10m300tai2	220	1	433	191	0	1	184	1	465	**180**	**1**	**290**
n10m300tai3	221	1	684	188	0	1	183	1	264	**180**	**0**	**336**
n10m400tai1	296	2	239	247	0	1	241	1	606	**240**	**0**	**166**
n10m400tai2	295	2	255	238	0	1	**236**	**1**	**152**	**236**	**0**	**93**
n10m400tai3	296	2	156	255	0	1	250	1	518	**245**	**1**	**173**
n10m500tai1	370	2	405	302	0	1	299	1	218	**297**	**1**	**190**
n10m500tai2	369	2	583	300	0	1	296	1	430	**293**	**1**	**203**
n10m500tai3	369	2	142	299	0	1	295	1	482	**293**	**1**	**186**
n10m1000tai1	743	5	201	604	0	1	600	3	329	**593**	**4**	**166**
n10m1000tai2	743	4	379	603	0	1	594	3	406	**585**	**4**	**200**
n10m1000tai3	740	4	444	610	0	1	602	3	356	**592**	**6**	**416**
n10m2000tai1	1491	9	791	1192	0	1	1189	6	489	**1180**	**17**	**280**
n10m2000tai2	1489	11	707	1203	0	1	1197	6	440	**1187**	**24**	**273**
n10m2000tai3	1489	9	410	1195	0	1	1186	6	490	**1174**	**24**	**110**
n10m3000tai1	2235	14	659	1766	0	1	1764	9	409	**1756**	**39**	**253**
n10m3000tai2	2243	16	675	1777	0	1	1772	9	444	**1762**	**45**	**263**
n10m3000tai3	2237	13	624	1772	0	1	1768	8	401	**1757**	**62**	**303**
n10m4000tai1	2991	18	126	2348	0	1	2341	8	637	**2328**	**127**	**196**
n10m4000tai2	2982	22	277	2375	0	1	2361	11	285	**2333**	**204**	**230**
n10m4000tai3	2991	21	485	2368	0	1	2361	11	600	**2344**	**130**	**233**
n10m5000tai1	3736	23	177	2969	0	1	2959	11	606	**2933**	**257**	**250**
n10m5000tai2	3743	25	350	2931	0	1	2927	14	542	**2909**	**228**	**206**
n10m5000tai3	3741	26	443	2938	0	1	2933	14	601	**2920**	**261**	**216**
n20m1000tai1	749	8	574	656	0	1	650	5	418	**643**	**26**	**193**
n20m1000tai2	753	9	363	666	0	1	656	5	524	**646**	**52**	**196**
n20m1000tai3	752	9	185	674	0	1	662	5	190	**645**	**122**	**296**
n20m2000tai1	1502	20	336	1295	0	1	1290	11	210	**1279**	**150**	**320**
n20m2000tai2	1500	20	436	1304	0	1	1292	11	790	**1280**	**204**	**343**
n20m2000tai3	1506	18	255	1295	0	1	1289	11	306	**1280**	**131**	**370**
n20m3000tai1	2251	28	524	1935	0	1	1930	17	498	**1913**	**519**	**156**
n20m3000tai2	2250	32	664	1955	0	1	1940	17	127	**1906**	**893**	**196**
n20m3000tai3	2255	27	3002	1968	0	1	1954	17	389	**1922**	**2540**	**246**
n20m4000tai1	3003	37	311	2560	0	1	2555	22	2572	**2540**	**819**	**256**
n20m4000tai2	3002	40	302	2584	0	1	2572	22	689	**2548**	**1162**	**300**
n20m4000tai3	3007	41	472	2588	0	1	2582	22	319	**2561**	**1084**	**183**
n20m5000tai1	3755	47	337	3217	0	1	3205	28	500	**3182**	**1833**	**210**
n20m5000tai2	3762	46	743	3193	0	1	3190	28	549	**3176**	**1853**	**203**

Tabela 1: Resultados gerados pelos algoritmos implementados nesse trabalho.

A tabela 3 mostra a comparação entre os melhores valores obtidos com os algoritmos propostos e o valor da solução ótima, para as instâncias cujo ótimo é conhecido. O algoritmo utilizado para encontrar a solução ótima foi o *branch-and-bound* do Xpress-MP versão 2004-c. Na primeira coluna tem-se o nome da instância. Na segunda coluna tem-se o *Best Bound* encontrado, *i.e.*, o valor dado pela heurística. Na terceira coluna é apresentado o valor da solução ótima e finalmente, na última coluna tem-se a distância entre as soluções heurística e ótima, respectivamente. Observe que a tabela 3 não possui resultados para todas as 39 instâncias, pois o algoritmo exato não foi capaz de encontra-los. Por esse motivo em trabalhos futuros novas formulações deverão ser testadas.

	Dist.	Temp	Iter.
n10m300tai1	184	1.15	90
n10m300tai2	180	1.15	330
n10m300tai3	180	0.7	470
n10m400tai1	240	0.94	140
n10m400tai2	236	0.85	100
n10m400tai3	243	1.58	250
n10m500tai1	297	1.28	300
n10m500tai2	293	1.07	340
n10m500tai3	293	1.32	280
n10m1000tai1	592	3.83	480
n10m1000tai2	584	7.18	270
n10m1000tai3	592	6.15	490
n10m2000tai1	1180	18.38	260
n10m2000tai2	1187	21.62	350
n10m2000tai3	1174	24.63	120
n10m3000tai1	1756	46.86	260
n10m3000tai2	1762	39.97	270
n10m3000tai3	1757	55.02	380
n10m4000tai1	2328	95.3	90
n10m4000tai2	2332	208.66	310
n10m4000tai3	2343	111.32	250
n10m5000tai1	2932	236.91	440
n10m5000tai2	2909	291.75	140
n10m5000tai3	2919	286.89	420
n20m1000tai1	642	28.1	310
n20m1000tai2	644	57.22	200
n20m1000tai3	644	119.08	310
n20m2000tai1	1276	166.91	400
n20m2000tai2	1279	229.48	220
n20m2000tai3	1279	229.48	440
n20m3000tai1	1912	649.3	270
n20m3000tai2	1904	974.42	210
n20m3000tai3	1921	640.53	330
n20m4000tai1	2540	725.21	240
n20m4000tai2	2547	969.66	490
n20m4000tai3	2557	1202.49	180
n20m5000tai1	3181	1611.01	230
n20m5000tai2	3175	1793.84	270
n20m5000tai3	3181	2284.85	210

Tabela 2: Melhores resultados encontrados, tempo de execução e iteração do melhor resultado.

Instância	BB	Ótimo	%
n10m400tai1	240	233	0,03
n10m400tai3	243	236	0,03
n10m500tai1	297	292	0,02
n10m500tai2	293	289	0,01

Tabela 3: Comparando a melhor solução obtida pela heurística com o ótimo global obtido pelo Software Xperss.

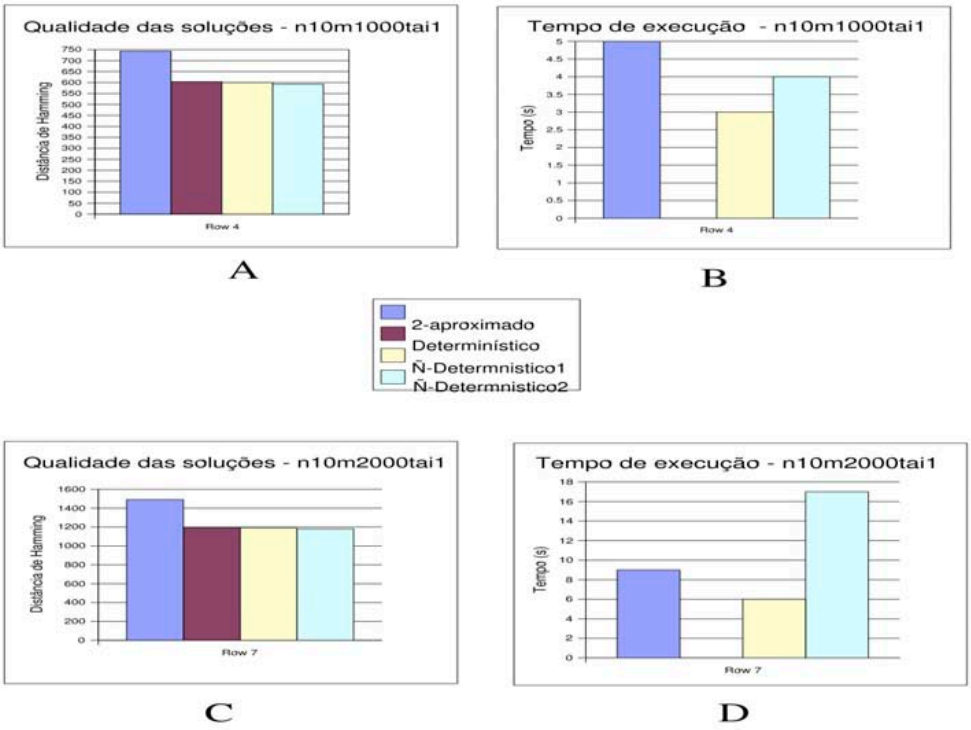

Gráfico 1: Os gráficos A e B (C e D) mostram os contrastes entre a qualidade das soluções obtidas e o tempo de execução obtidos pelos algoritmos propostos para a instância n10m1000tai1 (n10m2000tai1).

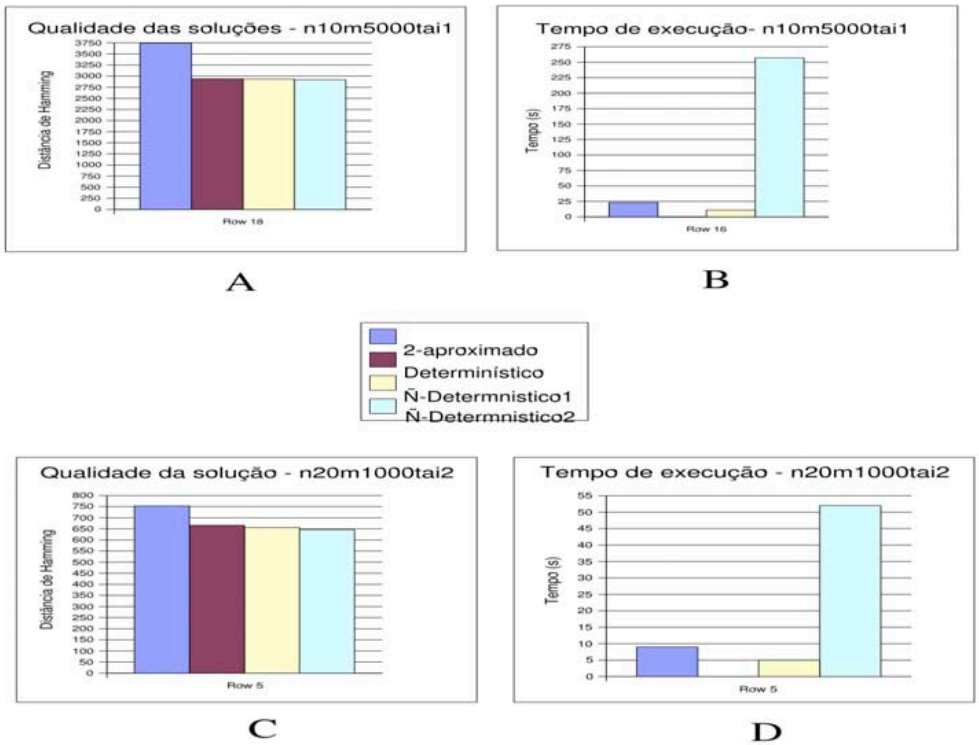

Gráfico 2: Os gráficos A e B (C e D) mostram os contrastes entre a qualidade das soluções obtidas e o tempo de execução obtidos pelos algoritmos propostos para a instância n10m5000tai1 (n20m2000tai2).

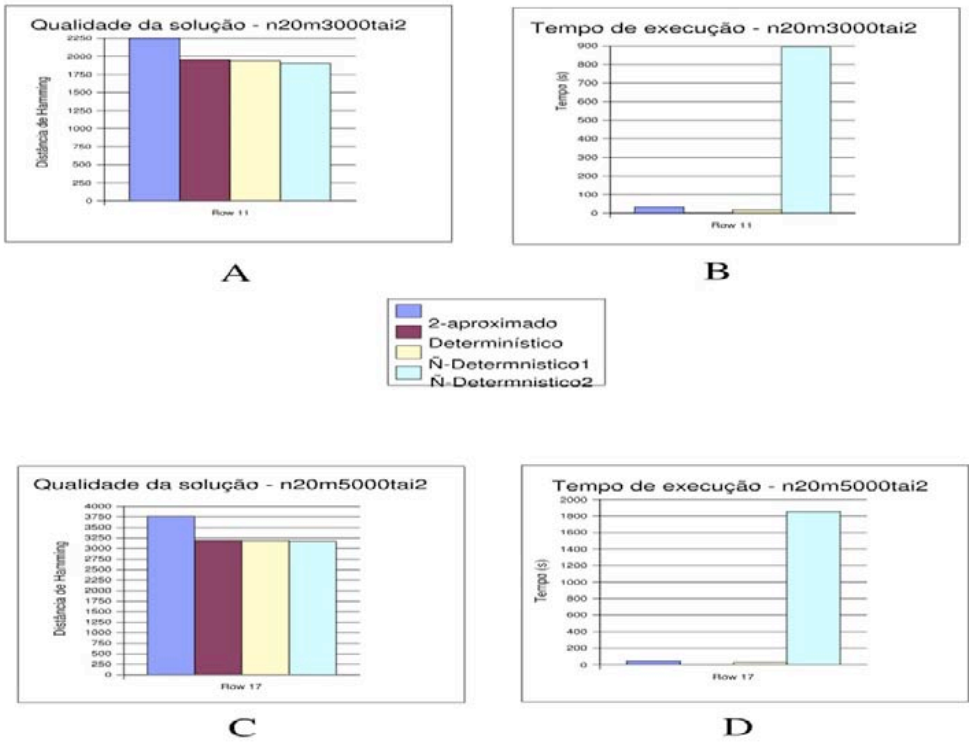

Gráfico 3: Os gráficos A e B (C e D) mostram os contrastes entre as qualidades das soluções obtidas e o tempo de execução obtidas pelos algoritmos propostos para a instância n20m3000tai2 (n20m5000tai3).

6. Conclusões e Trabalhos Futuros

Através dos resultados computacionais realizados observa-se que a heurística 2-aproximada não gerou bons resultados perdendo sempre para as propostas. Dentre as propostas a que gerou melhor resultado foi a não-determinística. Pode-se observar também que o procedimento de busca local sempre melhora as soluções construídas e que além disso a aplicação da RC gerou resultados ainda melhores. Dessa maneira, a heurística que gerou o melhores resultados foi o método 4 (não-determinística, com busca local e reconexão de caminhos). Pela comparação dos resultados heurísticos com os valores ótimos, das instâncias em que o ótimo é conhecido, podemos concluir que ao menos para as instâncias de médio porte, a melhor heurística proposta neste trabalho encontra resultados muito próximos dos valores ótimos.

Como trabalhos futuros existem o melhor ajuste dos parâmetros das heurísticas (número de iterações dos algoritmos, tamanho do conjunto elite da RC, direção da RC, tamanho e estrutura das vizinhanças da busca local, a "chance" de cada símbolo ser escolhido no algoritmo de construção proposto, entre outros) que deve ser feito através de testes, a paralelização dos algoritmos propostos, desenvolvimento de outras formulações matemáticas, implementação de algoritmos exatos para o PSMP, como *Branch and Bound*, *Branch and Cut* ou *Relax and Cut*. Além de implementação dos algoritmos aproximativos existentes na literatura, que ainda não foram testados na prática.

Agradecimentos

A FAPERJ (Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro) pelo suporte financeiro.

Referências

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.

Ma, B., 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.

Glover, F., Laguna,M., Martí, R.. Fundamentals of scatter search and path relinking. *Technical Report*, Graduate School of Business and Administration, University of Colorado, Boulder, CO 80309-0419, 2000.

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

Hansen, P. and Mladenovi`c, N. Variable Neighborhood Search for the p-median, *Location Science*, vol. 5, 207-226, 1997.

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

Lanctot, K., Li, M., Ma, B., Wang, S. and Zhang, L. Distinguishig string selection problems, *Proc. 10th ACM-SIAM Symp. On Discrete Algorithms*, 633-642, 1999.

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

Meneses, C., Lu, Z., Oliveira, C., 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 Estructures = Evolution Programs* , Springer, 1996.

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.

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