MAC0499 - Trabalho Supervisionado de Formatura
Algoritmos de Aproximação e Problemas com Seqüências

Rafael Crivellari Saliba Schouery
Supervisora: Cristina Gomes Fernandes

Sumário

1 Introdução **5**

2 Conceitos estudados **7**
2.1 Otimização combinatória 7
2.2 Algoritmos de aproximação 7

3 Metodologia e atividades realizadas **9**
3.1 Metodologia 9
3.2 Atividades realizadas 9

4 Resultados obtidos **11**
4.1 Considerações iniciais 11
4.2 Um algoritmo inicial 13
4.3 Complexidade do problema 15
4.3.1 Seqüências curtas 20
4.4 Resultados de inaproximabilidade 21
4.5 Cobertura por circuitos 23
4.5.1 Algumas propriedades 25
4.6 Um segundo algoritmo guloso 26
4.7 Alcançando a 3-aproximação 28
4.8 Análise do algoritmo guloso 29

5 Conclusão **37**

6 Parte Subjetiva **39**
6.1 Desafios e frustrações 39
6.2 Disciplinas relevantes 40
6.3 Aplicação dos conceitos estudados 42
6.4 Possíveis passos para aprimorar os conhecimentos relevantes . . . 42

Capítulo 1

Introdução

Problemas envolvendo seqüências são uma classe importante dos problemas combinatórios, usualmente de simples exposição, mas muitas vezes intrigantemente difíceis. Estes problemas aparecem nas mais diversas áreas, como por exemplo em compressão de dados e em biologia molecular computacional [10, 18].

Alguns destes problemas estão bem resolvidos, no sentido de que existem algoritmos eficientes para solucioná-los, enquanto que outros são considerados computacionalmente difíceis. Quando os problemas em questão são problemas de otimização combinatória difíceis, é comum encontrar na literatura uma variedade de algoritmos de aproximação para eles, que surgem como uma possível forma de atacá-los.

Algoritmos de aproximação são algoritmos eficientes para um problema de otimização combinatória que produzem não necessariamente uma solução ótima para o problema, mas sim uma solução viável com uma certa garantia de qualidade [6, 11, 23]. Muitos destes algoritmos decorrem de ferramentas bem conhecidas de projeto de algoritmos, bem como de propriedades muitas vezes não-triviais do problema em questão.

O estudo de algoritmos de aproximação para problemas envolvendo seqüências inclui o estudo de uma série de conceitos, ferramentas, e problemas algorítmicos básicos, que são de interesse geral na área de projeto de algoritmos combinatórios.

Um dos problemas bem conhecidos envolvendo seqüências é o **problema da superseqüência comum mínima** (*shortest common supersequence*), denotado por SCS: dadas seqüências $s_1, \ldots, s_n$ (finitas) de símbolos sobre um alfabeto, encontrar uma seqüência de comprimento mínimo que seja superseqüência de cada s_i, para $i = 1, \ldots, n$.

O problema é discutido em diversos livros [10, 18, 23] e em uma série de artigos, alguns clássicos [4, 15, 21] e vários outros, alguns deles mais recentes [1, 2, 3, 5, 7, 12, 14, 19, 20, 24].

Sabe-se que o SCS é NP-difícil [8], e mesmo MAX SNP-difícil [4, 22], o que significa que mesmo a existência de um PTAS (esquema polinomial de aproximação) para ele é pouco provável (implicaria que P = NP).

Na literatura, há diversos algoritmos de aproximação para o SCS [2, 3, 4, 5, 7, 14, 15, 19, 20, 21] e há também uma conjectura clássica envolvendo um destes algoritmos, conhecido como algoritmo guloso (GREEDY) para o SCS.

Neste texto, abordamos alguns destes algoritmos mostrando suas respectivas razões de aproximação e ainda mostramos alguns resultados de complexidade sobre o SCS.

A motivação para o preparo desta monografia foi justamente a conjectura em relação ao algoritmo guloso. O estudo dos algoritmos para o problema da superseqüência comum mínima são importantes por si só, mas esta importante conjectura permanece em aberto há vários anos e isso motiva o estudo aprofundado dos resultados conhecidos para o problema.

Capítulo 2

Conceitos estudados

Apresentamos neste capítulo conceitos importantes para o conteúdo apresentado no capítulo 4. Abordaremos itens relacionados a otimização combinatória e sua sub-área chamada algoritmos de aproximação. A principal referência para este capítulo é o livro de Carvalho et al. [6].

2.1 Otimização combinatória

Um **problema de otimização** contém três itens: um c**onjunto de instâncias**, um conjunto finito $Sol(I)$ de **soluções viáveis** para cada instância I, e um função que atribui um número $val(S)$ para cada solução viável S (chamado **valor** de S). Para cada instância I associamos um número natural $\langle I \rangle$ que representa o **tamanho da instância**.

Em geral queremos encontrar uma solução viável S^* tal que $val(S^*)$ seja mínimo (ou máximo, dependendo do problema). Neste caso chamamos S^* de **solução ótima**. Denotamos por $opt(I)$ o valor de uma solução ótima, isto é, $opt(I) = val(S^*)$ para algum S^*, solução ótima do problema.

Como, para um determinado problema de otimização, o conjunto $Sol(I)$ é finito, um algoritmo possível para resolver tal problema é verificar o valor de val para cada solução viável da instância. Mas, em geral, existe uma quantidade exponencial em $\langle I \rangle$ de soluções viáveis e portanto teríamos um algoritmo ineficiente. Para alguns problemas de otimização combinatória, algoritmos polinomiais são conhecidos. Mas muitos outros problemas são NP-difíceis e não existe algoritmo eficiente para os mesmo exceto se P = NP.

2.2 Algoritmos de aproximação

Para problemas de otimização em que $val(S) \geq 0$ para toda solução viável S podemos definir o conceito de algoritmos de aproximação. Um algoritmo A é um **algoritmo de aproximação** para um problema de otimização se:

(1) A devolve uma solução viável $A(I)$ de I.

(2) Existe $\alpha = \alpha(< I >)$ tal que

$$val(A(I)) \leq \alpha opt(I)$$

se o problema é de minimização e

$$val(A(I)) \geq \alpha opt(I)$$

se o problema é de maximização.

(3) A é um algoritmo polinomial em $\langle I \rangle$.

Chamamos α de **razão de aproximação** e dizemos que A é uma **α-aproximação (polinomial)** para o problema. Vale notar que alguns autores (inclusive Carvalho et al. [6]) definem um algoritmo de aproximação sem exigir a condição (3). Neste texto estamos interessados apenas em aproximações polinomiais.

Um **esquema de aproximação polinomial** é uma família de algoritmos de aproximação para um determinado problema de otimização combinatória com uma característica bem específica. A família consiste em, para cada $\epsilon > 0$, uma $(1 + \epsilon)$-aproximação para o problema. Cada algoritmo na família é polinomial, mas pode não ser polinomial em $1/\epsilon$. Ou seja, podemos definir a taxa máxima de erro do algoritmo escolhendo ϵ. Assim, se quisermos que a solução encontrada seja no máximo 5% pior que a melhor solução, basta fixarmos $\epsilon = 0,05$.

Para vários problemas, a existência de um algoritmo de aproximação com determinada razão de aproximação implicaria que P = NP. Existem várias classes de problemas de otimização combinatória que capturam para representar a dificuldade em se ter bons algoritmos de aproximação para tais problemas. Em particular, a classe dos problemas **Max SNP-difíceis** é tal que a existência de um esquema de aproximação polinomial para um de seus problemas implica em P = NP.

Capítulo 3

Metodologia e atividades realizadas

3.1 Metodologia

A metodologia utilizada para a realização do trabalho foi o estudo acompanhado pela supervisora. O processo consistiu em entender os resultados apresentados através de vários livros e artigos, e rescrevê-los com uma notação única durante todo o texto. Através de reuniões semanais ou quinzenais, dependendo do ritmo do trabalho, as correções para o texto e outros resultados eram discutidos.

3.2 Atividades realizadas

Inicialmente estudei algumas noções gerais sobre algoritmos de aproximação pelo livro de Carvalho et al. [6]. Comecei então o estudo do SCS através de livros [10, 18, 23] que abordassem o assunto (principalmente os capítulos 2 e 7 do livro de Vazirani [23]). Isso possibilitou o entendimento do problema, dos métodos usados para sua resolução e dos algoritmos clássicos para a resolução do SCS. Estudei então uma parte do artigo de Blum et al. [4] referente aos algoritmos, deixando a análise do GREEDY para ser estudada posteriormente. Estudei nesta fase cinco algoritmos diferentes dos quais quatro foram escritos posteriormente e são apresentados no capítulo 4 (um foi omitido por não ser de grande interesse para nós). Posteriormente estudei dois resultados de complexidade (também detalhados no capítulo 4) para o SCS: a prova de que o mesmo é NP-difícil, dada por Gallant et al. [8], e de que o problema é MAX SNP-difícil. Por fim, fiz o estudo da prova de que o GREEDY é uma 4-aproximação, dada por Blum et al. [4].

As atividades realizadas podem ser vistas no texto apresentado no capítulo 4, já que a produção científica do trabalho foi escrever este texto com os resultados.

Capítulo 4

Resultados obtidos

O texto contido neste capítulo foi produzido durante o projeto. Ele contém os resultados estudados para o problema, utilizando uma linguagem mais concisa e fácil de entender comparada com as linguagens dos diversos livros e artigos estudados. O texto e outros produtos desta iniciação científica estão disponíveis em http://www.ime.usp.br/∼schouery/scs/.

4.1 Considerações iniciais

Considere um conjunto S de seqüências.

Definição 4.1.1 (Superseqüência comum)**.** *Uma seqüência $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ é uma **superseqüência comum** para S se, para todo $s \in S$, existem naturais i e j tais que $1 \leq i < j \leq n$ e $s = \sigma_i\sigma_{i+1}\ldots\sigma_j$.*

Definição 4.1.2 (Superseqüência comum mínima)**.** *Uma **superseqüência comum mínima** para S é uma superseqüência comum para S de comprimento mínimo. Denotamos o problema de encontrar tal seqüência por* $\textsc{scs}(S)$.

Definição 4.1.3. *Denotamos por $\textsc{scs}^*(S)$ uma solução ótima do $\textsc{scs}(S)$. Quando S estiver implícito no contexto escrevemos $\textsc{scs}^*$.*

Sejam $s_1, \ldots, s_n$ as seqüências dadas como entrada para o SCS. Sem perda de generalidade, podemos assumir que nenhum s_i é subseqüência de um s_j com $i \neq j$. De fato, se houver um s_i que seja subseqüência de um s_j com $i \neq j$, podemos remover s_i de S sem alterar a solução do problema.

Definição 4.1.4 (Livre de subseqüências)**.** *Um conjunto S de seqüências é **livre de subseqüências**[1] se nenhuma seqüência de S é subseqüência de alguma outra subseqüência de S.*

Portanto, a partir de agora, consideraremos que o conjunto S dado é livre de subseqüências.

Definição 4.1.5. *Seja C um conjunto qualquer e seja $f : C \to \mathbb{R}$. Para $C' \subseteq C$, seja $f(C') = \sum_{c \in C'} f(c')$.*

[1] *Substring free* no inglês.

Definição 4.1.6. *Para qualquer seqüência s, denotamos por $|s|$ o comprimento da seqüência s, denotamos por $\|S\|$ a soma dos comprimentos das seqüências em S, ou seja, $\|S\| = \sum_{s \in S} |s|$.*

Definição 4.1.7 (Sobreposição)**.** *Para duas seqüências s e t, uma **sobreposição entre** s **e** t é uma seqüência v tal que $s = uv$ e $t = vw$, onde ou u ou v não é vazia. Note que a seqüência v pode ser vazia e que s e t não são necessariamente diferentes.*

Definição 4.1.8 (Sobreposição máxima)**.** *Para duas seqüências s e t, a **sobreposição máxima**, denotada por $over(s,t)$[2], é a mais longa sobreposição entre s e t.*

Definição 4.1.9 (Prefixo)**.** *Para duas seqüências s e t, o **prefixo de** s **em relação a** t, denotado por $pref(s,t)$, é a seqüência u tal que $s = uv$ e $t = vw$ onde $v = over(s,t)$.*

Podemos portanto escrever $s = pref(s,t)over(s,t)$.

Definição 4.1.10 (Composição)**.** *Para duas seqüências s e t, a **composição de** s **e** t, denotada por $comp(s,t)$, é a seqüência uvw onde $s = uv$, $t = vw$ e $v = over(s,t)$.*

Por exemplo, a sobreposição máxima entre as seqüências $u =$ `abbbaab` e $v =$ `baababa` é `baab`, o prefixo de u em relação a v é `abb` e a composição de u e v é `abbbaababa`, que é justamente a superseqüência comum mínima de u e v.

Teorema 4.1.1 (Função objetivo)**.** Uma superseqüência comum mínima de $S = \{s_1, \ldots, s_n\}$ corresponde a uma permutação $(o_1, \ldots, o_n)$ de $\{1, \ldots, n\}$ tal que $\sum_{i=1}^{n-1} |over(s_{o_i}, s_{o_{i+1}})|$ é máximo e vice-versa. A superseqüência correspondente a uma permutação $(o_1, \ldots, o_n)$ é

$$pref(s_{o_1}, s_{o_2})pref(s_{o_2}, s_{o_3}) \cdots pref(s_{o_{n-1}}, s_{o_n})s_{o_n}.$$

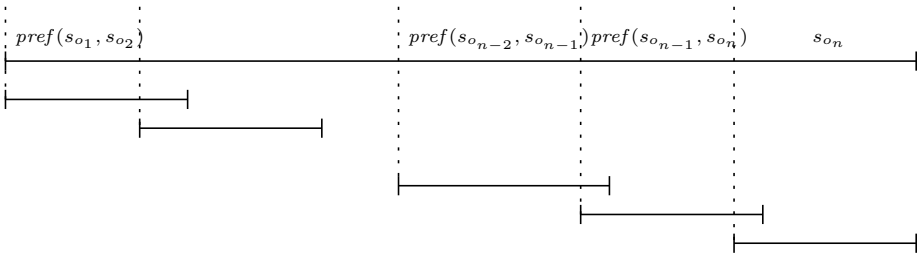

Figura 4.1: Estrutura de uma solução ótima.

Demonstração. Considere uma solução ótima SCS* do problema para a instância S. Sejam $o_1, \ldots, o_n$ os índices das seqüências de S na ordem em que ocorre a primeira aparição das seqüências de S em SCS*. Note que SCS* começa com a seqüência de s_{o_1}, pois caso não começasse poderíamos retirar algumas das

[2] Do inglês *overlap*.

primeiras letras de SCS* e ainda teríamos uma superseqüência de S mas de tamanho menor do que SCS*, contrariando o fato de que SCS* é mínima. Por outro lado, basicamente pela mesma razão, SCS* termina com s_{o_n}. Outro fato a notar é que em SCS* utilizamos a máxima sobreposição entre as seqüências s_{o_i} e $s_{o_{i+1}}$, para todo $i = 1, \ldots, n-1$, já que caso contrário poderíamos diminuir SCS* aumentando a sobreposição de duas seqüências consecutivas e isso contradiria o fato de SCS* ser mínima. Isso porque como nenhuma seqüência de S é subseqüência de outra seqüência, temos que se uma seqüência começa antes de outra em SCS* então ela também acaba antes. Temos então que $|\text{SCS}^*| = \|S\| - \sum_{i=1}^{n-1} |over(s_{o_i}, s_{o_{i+1}})|$. Como $\|S\|$ depende apenas da instância, encontrar SCS* é o mesmo que encontrar uma permutação $(o_1, \ldots, o_n)$ que maximize $\sum_{i=1}^{n-1} |over(s_{o_i}, s_{o_{i+1}})|$. □

4.2 Um algoritmo inicial

Um primeiro algoritmo guloso que se pode pensar ao conhecer o problema e a necessidade de se encontrar uma aproximação para o mesmo (já que o mesmo é NP-difícil) é da seguinte forma: para um conjunto S de seqüências, crie um conjunto $T = S$ e então escolha duas seqüências distintas s e t pertencentes a T tais que a sobreposição máxima entre elas seja a maior possível. Faça a composição c entre as duas e então remova s e t e insira c em T. Repita o processo até restar uma única seqüência em T. Após $n-1$ iterações, T conterá uma única seqüência que é uma superseqüência das seqüências em S.

Há na literatura [21] uma conjectura de que este algoritmo é uma 2-aproximação para o problema da superseqüência comum mínima. Blum et al. [4] provaram que ele é uma 4-aproximação e recentemente Kaplan e Shafrir [12] provaram que ele é uma 3,5-aproximação. Podemos ver facilmente que ele não é melhor do que uma 2-aproximação. Basta tomar $S = \{c(ab)^k, (ba)^k, (ab)^k c\}$ onde k é um inteiro positivo arbitrário. No primeiro passo, o algoritmo escolhe compor $c(ab)^k$ com $(ab)^k c$ e temos $T = \{c(ab)^k c, (ba)^k\}$. Então ele junta as duas seqüências gerando, por exemplo, a superseqüência $c(ab)^k c(ba)^k$ de comprimento $4k+2$. É fácil ver que a superseqüência comum mínima para a instância é $c(ab)^{k+1}c$, de comprimento $2k+4$.

Mostramos agora uma versão mais formal do algoritmo. Considere primeiro uma função que descobre o maior $over(s,t)$ entre seqüências do conjunto T:

```
MAXOVERLAP (T)
 1  over_max ← 0
 2  s ← t ← ∅
 3  para cada u ∈ T faça
 4      para cada v ∈ T faça
 5          se u ≠ v e over_max ≤ over(u, v) então
 6              over_max ← over(u, v)
 7              s ← u
 8              t ← v
 9  devolva (s, t)
```

Podemos agora considerar o algoritmo guloso mencionado acima, conhecido como GREEDY.

GREEDY (S)
1 $T \leftarrow S$
2 **enquanto** $|T| \neq 1$ **faça**
3 $(s,t) \leftarrow$ MAXOVERLAP(T)
4 $T \leftarrow T \setminus \{s,t\} \cup \{comp(s,t)\}$
5 seja t o único elemento de T
6 **devolva** t

Definição 4.2.1 (Função custo)**.** *Seja $G = (V, E)$ um grafo (digrafo) e $c : E \to \mathbb{Q}$. Vamos nos referir a c como uma função custo e, para cada e em E, $c(e)$ será o custo da aresta (arco) e. Se $S \subseteq E$, usaremos $c(S)$ para denotar $\sum_{e \in S} c(e)$. Se H é um subgrafo de G, então escrevemos $c(H)$ em vez de $c(E_H)$.*

Definição 4.2.2 (Digrafo associado)**.** *Para um conjunto S de seqüências, definimos o **digrafo associado** a S como um digrafo $G = (S, E)$, onde $E = \{(s_i, s_j) | s_1 \in S \text{ e } s_2 \in S\}$. Note que G inclui laços pela definição.*

Definição 4.2.3 (Digrafo de sobreposição máxima)**.** *Para um conjunto S de seqüências, definimos o **digrafo de sobreposição máxima** de S como o par (G, s) onde G é o digrafo associado a S e $s(u, v) = |over(u, v)|$. Denotamos (G, s) por G_s.*

Definição 4.2.4 (Digrafo de prefixo)**.** *Para um conjunto S de seqüências, definimos o **digrafo de prefixo** de S como o par (G, p) onde G é o digrafo associado a S e $p(u, v) = |pref(u, v)|$. Denotamos (G, p) por G_p.*

Podemos ver que o **GREEDY** escolhe, a cada iteração, o arco de maior peso em G_s que não forma circuito e que, considerando o digrafo gerador com apenas as arestas escolhidas, nenhum vértice tem grau maior que 1. Ou seja, não podemos concatenar o final de uma seqüência com mais de uma outra seqüência. Damos na Figura 4.2 um exemplo desse fato. Considere $S = \{s_1, s_2, s_3, s_4\}$ onde $s_1 = ababab$, $s_2 = bababa$, $s_3 = ababbb$ e $s_4 = bbbaba$.

Para resolver o SCS basta encontrarmos uma ordenação que pode ser entendida como um caminho hamiltoniano no digrafo associado a S. Portanto há uma relação do SCS com o problema do caixeiro viajante (TSP[3]).

Definição 4.2.5 (Caminho (circuito) hamiltoniano)**.** *Um caminho (circuito) hamiltoniano em um digrafo $G = (V, E)$ é um caminho (circuito) que passa por todos os vértices de G.*

Definição 4.2.6 (Problema do caixeiro viajante)**.** *Dado um digrafo $G = (V, E)$ e uma função custo $c : E \to \mathbb{Q}$, encontrar (se existir) um circuito hamiltoniano de custo mínimo em G, onde o custo de um circuito é a soma de $c(e)$ para todo arco e do circuito.*

Definição 4.2.7. *Para um conjunto S de seqüências denotamos por $\text{TSP}^*(S)$ um circuito hamiltoniano de custo mínimo para o digrafo de prefixos de S.*

[3] Do inglês *travelling salesman problem*.

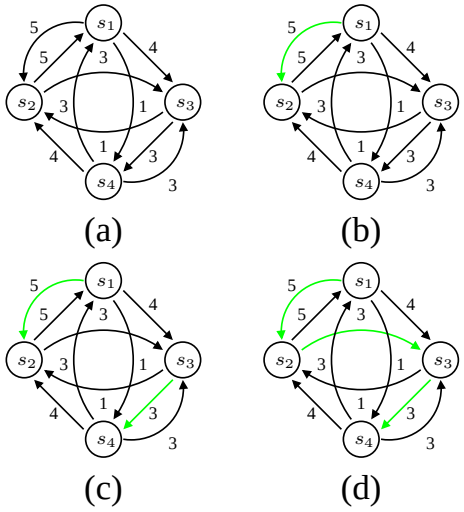

Figura 4.2: Execução do GREEDY no digrafo de sobreposição máxima. Na figura (a) vemos o digrafo de sobreposição máxima para S. Em (b) vemos que o GREEDY escolhe o arco de maior peso s_1s_2 (existem outras opções) ficando com três seqüências (*abababa*, *ababbb* e *bbbaba*). Em (c) o GREEDY escolhe s_3s_4 ficando com duas seqüências (*abababa* e *ababbbaba*), em (d) o GREEDY escolhe s_2s_3 gerando *abababababbbaba* uma superseqüência comum de S.

Seja $(o_1, \ldots, o_n)$ uma ordenação dos vértices em TSP*. Poderíamos pensar na seqüência $pref(s_{o_1}, s_{o_2}) \cdots pref(s_{o_{n-1}}, s_{o_n}) pref(s_{o_n}, s_{o_1})$ como o rótulo desse circuito hamiltoniano de custo mínimo no digrafo de prefixo de S. Se considerarmos $pref(s_{o_1}, s_{o_2}) \cdots pref(s_{o_{n-1}}, s_{o_n}) s_{o_n}$, temos uma superseqüência comum para S, de custo maior ou igual do que o do circuito pois $|pref(s,t)| \leq |s|$. Portanto temos:

$$\|\text{TSP}^*(S)\| \leq |\text{SCS}^*(S)| - \min\{|over(s,t)| : s,t \in S\} \leq |\text{SCS}^*(S)|.$$

O problema do caixeiro viajante é NP-difícil [9]. No entanto, uma aproximação para o mesmo poderia nos levar a uma aproximação para o SCS. Em geral, o TSP é difícil também de se aproximar, porém para o seu caso métrico existe uma $\frac{3}{2}$-aproximação. Só que infelizmente a instância do TSP obtida do SCS não é métrica (ela não satisfaz a desigualdade triangular necessariamente), logo não podemos derivar uma boa aproximação para o SCS desta maneira.

A análise do fator de aproximação do GREEDY é muito complexa e necessita de outros recursos ainda não apresentados, portanto ela será apresentada posteriormente na seção 4.8. Na seção 4.5 apresentaremos um problema fortemente relacionado com o TSP para o qual existem algoritmos eficientes.

4.3 Complexidade do problema

Nesta seção apresentaremos uma prova de que o SCS é NP-difícil [8]. Consideramos o problema de decisão relacionado ao SCS e então mostraremos uma

redução polinomial do problema da existência de um caminho hamiltoniano em um grafo à versão de decisão do SCS. Portanto, se existir um algoritmo polinomial para o SCS existirá também um para encontrar um caminho hamiltoniano num grafo genérico em tempo polinomial, implicando que P = NP.

Definição 4.3.1 (Versão de decisão do SCS)**.** *Dado um conjunto S de seqüências e um inteiro positivo k, descobrir se existe uma superseqüência comum de comprimento k para S.*

Inicialmente discutiremos uma variante do caminho hamiltoniano que será usada na redução.

Definição 4.3.2. *Para um digrafo $G = (V, E)$ e para $v \in V$, denotamos por $\delta^+(v)$ a quantidade de arcos de G saindo de v, ou seja, $\delta^+(v) = |\{w \in V : vw \in E\}|$. Do mesmo modo denotamos por $\delta^-(v)$ a quantidade de arcos de G entrando em v, ou seja, $\delta^-(v) = |\{w \in V : wv \in E\}|$.*

Definição 4.3.3 (Problema do caminho hamiltoniano direcionado restrito)**.** *Dado um digrafo $G = (V, E)$ e vértices s e t, onde $\delta^-(s) = 0$, $\delta^+(t) = 0$ e $\delta^+(v) \geq 2$ para todo $v \in V \setminus \{t\}$, decidir se existe um caminho hamiltoniano de s para t em G.*

Sabe-se que a versão sem restrições desse problema é NP-completo [13]. Começaremos mostrando que esta variante ainda é difícil.

Lema 4.3.1. O problema do caminho hamiltoniano direcionado restrito é NP-completo.

Demonstração. É fácil notar que o problema está em NP: um caminho hamiltoniano de s para t em G serve como certificado. Para a segunda parte da prova vamos reduzir o problema do circuito hamiltoniano direcionado à variante em questão do mesmo.

Seja $G = (V, E)$ uma instância do problema do circuito hamiltoniano direcionado. Seja $G' = (V', E')$ um novo digrafo com $V' = V \setminus \{v\} \cup \{s, t, a, b, t'\}$, onde $v \in V$. Para descrever E', considere $S = \{sw : w \in V, vw \in E\}$, $T = \{ut : u \in V, uv \in E\}$, $N = \{ta, tb, ab, ba, at', bt'\}$, $D = \{ut' : u \in V, \delta^+(u) < 2\}$ e $O = \{uw : u \in V', w \in V', uw \in E\}$. Temos que $E' = O \cup S \cup T \cup N$. Ou seja, dividimos o vértice v, que foi escolhido arbitrariamente, em dois vértices, s e t. Para todos os arcos que saem de v, há um arco que sai de s (conjunto S) e para todo arco que entra em v, há um arco que entra em t (conjunto T). Há arcos para garantir que o grafo tenha $\delta^+(u) \geq 2$, para todo $u \in V' \setminus \{t'\}$ (conjunto D), alguns novos arcos de interesse para o problema (conjunto N) e os arcos originais de G que não têm nenhuma ponta em v (conjunto O). Note que G tem um circuito hamiltoniano direcionado se e somente se G' tem um caminho hamiltoniano direcionado começando em s e terminando em t'. Veja o exemplo da figura 4.3. □

Definição 4.3.4. *Uma seqüência é **primitiva** se nenhum símbolo aparece mais de uma vez nela, ou seja, não há repetições.*

Teorema 4.3.1. O SCS é NP-difícil. Além disso, o SCS é NP-difícil mesmo quando a instância S é tal que, para algum $h \geq 3$, $|s| = h$ e s é primitiva para todo $s \in S$.

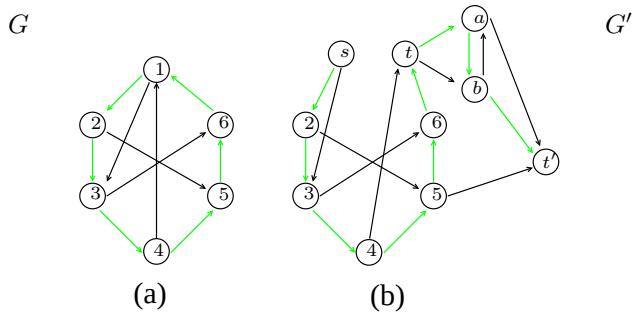

Figura 4.3: (a) Um digrafo G com um circuito hamiltoniano direcionado indicado pelos arcos claros. (b) O digrafo G' da redução do problema do circuito hamiltoniano ao problema do caminho hamiltoniano direcionado restrito para o digrafo G da figura (a). Note que $\delta^+(v) \geq 2$, para todo $v \neq t'$, que $\delta^-(s) = 0$ e $\delta^+(t') = 0$ e que um caminho hamiltoniano direcionado no digrafo G' que começa em s passa obrigatoriamente por todos os vértices do digrafo original G antes de chegar a t.

Demonstração. Inicialmente provaremos o teorema para $h = 3$ e seqüências não necessariamente primitivas. Então mostraremos como tornar as seqüências primitivas e estender a redução para $h \geq 3$.

Seja $G = (V, E)$ uma instância do problema do caminho hamiltoniano direcionado restrito. Podemos assumir que $V = \{1, \ldots, n\}$, $s = 1$ e $t = n$. Seja $m = |E|$. Iremos construir um conjunto S de seqüências para G sobre o alfabeto $\Sigma = V \cup B \cup X$, onde $B = \{\overline{v} : v \in V \setminus \{n\}\}$ e $X = \{\triangleright, \#, \$\}$. O conjunto S juntamente com um k fixado mais a frente será a instância correspondente a G da versão de decisão do SCS.

Para cada vértice $v \in V \setminus \{n\}$ criamos um conjunto A_v com $2\delta^+(v)$ seqüências da seguinte maneira. Seja $R_v = \{w : vw \in E\}$ o conjunto de vértices adjacentes a v em G. Estabeleça uma ordem arbitrária entre os elementos de R_v denotando-os por $w_0, \ldots, w_{\delta^+(v)-1}$ considerando os índices módulo $\delta^+(v)$. Seja $A_v = \{\overline{v}w\overline{v} : w \in R_v\} \cup \{w_i\overline{v}w_{i+1} : 0 \leq i < \delta^+(v)\}$.

Para cada $v \in V \setminus \{1, n\}$ crie um conjunto C_v contendo apenas a seqüência $v\#\overline{v}$, chamada de *conector*. Finalmente, seja $T = \{\triangleright\#\overline{1}, n\#\$\}$, o conjunto das chamadas seqüências *terminais*.

Seja S a união dos A_j, $1 \leq j < n$, dos C_i, $1 < i < n$, e de T. Tome $k = 2m + 3n$. Isso conclui a descrição da instância da versão de decisão do SCS correspondente a G. Note que é possível construir S e k em tempo polinomial em $n + m$.

Segue um pequeno exemplo para mostrar como a redução é feita. Considere o grafo $G = (V, E)$ da figura 4.4. Gostaríamos de descobrir se existe um caminho hamiltoniano direcionado em G de 1 a 5. Inicialmente construímos o conjunto C composto de conectores para todos os vértices menos o 1 e o 5, portanto $C = \{2\#\overline{2}, 3\#\overline{3}, 4\#\overline{4}\}$. Criamos o conjunto das seqüências terminais $T = \{\triangleright\#\overline{1}, 5\#\$\}$ e os conjuntos A_v para todo $v \in V$:

$$\begin{aligned} A_1 &= \{\overline{1}2\overline{1}, \overline{1}3\overline{1}, 2\overline{1}3, 3\overline{1}2\} \\ A_2 &= \{\overline{2}3\overline{2}, \overline{2}5\overline{2}, 3\overline{2}5, 5\overline{2}3\} \end{aligned}$$

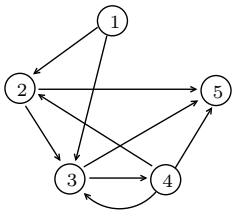

Figura 4.4: Um grafo $G = (V, E)$ restrito com $|V| = 5$ e $|E| = 9$.

$$\begin{aligned} A_3 &= \{\overline{3}5\overline{3}, \overline{3}4\overline{3}, 4\overline{3}5, 5\overline{3}4\} \\ A_4 &= \{\overline{4}2\overline{4}, \overline{4}3\overline{4}, \overline{4}5\overline{4}, 2\overline{4}3, 3\overline{4}5, 5\overline{4}2\}. \end{aligned}$$

Temos então que $S = C \cup T \cup A_1 \cup A_2 \cup A_3 \cup A_4$.

Lema 4.3.2. Se G tem um caminho hamiltoniano direcionado de 1 para n então S tem uma superseqüência comum de comprimento $k = 2m + 3n$.

Demonstração. Suponha que G tem um caminho hamiltoniano direcionado P de 1 a n. Seja (v, w_i) uma aresta de P. Primeiro crie a superseqüência comum de comprimento $2\delta^+(v) + 2$ para A_v da forma

$$\overline{v}w_i\overline{v}w_{i+1}\overline{v}w_{i+2}\overline{v}\cdots\overline{v}w_{\delta^+(v)-1}\overline{v}w_0\overline{v}\cdots\overline{v}w_i \tag{4.1}$$

chamada de *w_i-superseqüência* para A_v. Considerando-se os índices módulo $\delta^+(v)$, esta superseqüência é formada sobrepondo as seqüências de A_v na ordem $\overline{v}w_i\overline{v}, w_i\overline{v}w_{i+1}, \ldots, \overline{v}w_{i+\delta^+(v)-1}\overline{v}, w_{i+\delta^+(v)-1}\overline{v}w_i$ onde cada par sucessivo tem uma sobreposição de comprimento 2. Note que há uma bijeção entre o conjunto das w_i-superseqüências para A_v e o conjunto das rotações cíclicas dos inteiros de 0 até $\delta^+(v)-1$. Note também que as w_i-superseqüências para $i = 0, \ldots, \delta^+(v)-1$ são as únicas superseqüências de A_v de comprimento $2\delta^+(v) + 2$. Ademais elas têm o comprimento mínimo (ou seja, são superseqüências comuns mínimas para A_v). Seja $(u_1, u_2, \ldots, u_n)$ a seqüência dos vértices de P, sendo $u_1 = 1$ e $u_n = n$. Para $i = 1, \ldots, n-1$, denotamos a u_{i+1}-superseqüência para A_{u_i} como $STD(\overline{u_i}, u_{i+1})$. Podemos construir uma superseqüência para S fazendo sobreposições máximas para a seguinte ordem de seqüências:

$$\triangleright\#\overline{1}, STD(\overline{1}, u_2), u_2\#\overline{u_2}, STD(\overline{u_2}, u_3), u_3\#\overline{u_3}, \ldots, u_{n-1}\#\overline{u_{n-1}}, STD(\overline{u_{n-1}}, n), n\#\$$$

Esta superseqüência comum para S tem comprimento

$$\sum_{i=1}^{n-1}(2\delta^+(i) + 2) + (n - 2) + 4 = 2m + 3n$$

, já que temos $n-2$ #'s dos conectores e mais 4 símbolos das seqüências terminais. □

Note que, no nosso exemplo, a ordem $(1, 2, 3, 4, 5)$ nos fornece as seguintes w_i-superseqüências para A_v:

$$\begin{aligned} STD(\overline{1}, 2) &= \overline{1}2\overline{1}3\overline{1}2 \\ STD(\overline{2}, 3) &= \overline{2}3\overline{2}5\overline{2}3 \\ STD(\overline{3}, 4) &= \overline{3}4\overline{3}5\overline{3}4 \\ STD(\overline{4}, 5) &= \overline{4}5\overline{4}2\overline{4}3\overline{4}5. \end{aligned}$$

Podemos agora construir a seguinte superseqüência para S:

$$\triangleright\#\overline{1}2\overline{1}3\overline{1}2\#\overline{2}3\overline{2}5\overline{2}3\#\overline{3}4\overline{3}5\overline{3}4\#\overline{4}5\overline{4}2\overline{4}3\overline{4}5\#\$.$$

Esta superseqüência tem comprimento 33 que é exatamente $2|E| + 3|V|$. Como entre cada par de # temos uma w_i-superseqüências para A_v para algum v, vemos que a ordem $(1, 2, 3, 4, 5)$ representa um caminho hamiltoniano direcionado em G.

Lema 4.3.3. Se S tem uma superseqüência comum de comprimento $2m + 3n$ então G tem um caminho hamiltoniano direcionado de 1 até n.

Demonstração. Suponha que S tem uma superseqüência comum de comprimento $2m + 3n$. Vamos mostrar que G tem um caminho hamiltoniano direcionado de 1 para n.

Vamos mostrar que $2m + 3n$ é uma cota inferior para uma superseqüência comum para S e que só pode ser atingida através de uma superseqüência comum para S que codifica um caminho hamiltoniano direcionado de 1 para n em G.

Note que $|S| = 2m + n$ e que $\|S\| = 3(2m + n)$, já que toda seqüência em S tem comprimento 3. Seja σ uma superseqüência comum para S de comprimento $2m + 3n$. A maior compressão que poderíamos obter é uma ordenação de S em que todas as sobreposições de seqüências de S consecutivas tenham comprimento 2. Isso resultaria em uma superseqüência de comprimento $3 + 2m + n - 1 = 2m + n + 2$: a primeira seqüência contribui com 3 e as demais com 1, já que se sobrepõem com a anterior em 2 símbolos.

Mas como nenhuma seqüência começa ou termina com #, o melhor a ser feito com os $n-2$ conectores (que têm a forma $v\#v$) é obter uma sobreposição máxima de comprimento 1. Além disso, as seqüências terminais só têm sobreposições de comprimento no máximo 1 e em apenas um lado. Portanto, temos uma cota inferior melhor, de $(2m+n+2)+2(n-2)+2 = 2m+3n$ no comprimento de σ. Note também que, para atingir tal comprimento, σ obrigatoriamente teria que começar com $\triangleright\#\overline{1}$ e terminar com $n\#\$$.

Como σ é uma superseqüência comum mínima para S, temos n símbolos # em σ, um para cada conector. Sejam $u_1 = 1$, $u_n = n$ e $u_2, \ldots, u_{n-1}$ os símbolos anteriores aos # exceto o primeiro e o último, na ordem em que os # ocorrem em σ. Note que estes símbolos são todos distintos entre si. Considere agora duas ocorrências consecutivas do símbolo # em σ e seja x a seqüência entre estes dois #'s. Vamos mostrar que x é uma u_{i+1}-superseqüência de A_{u_i} para algum $1 \leq i \leq n-1$. Seja $(o_0, \ldots, o_{\delta^+(u_i)-1})$ uma ordenação dos vizinhos de u_i, e sem perda de generalidade considere que $o_0 = u_{i+1}$. O primeiro símbolo de x está em B e o último não está em B, já que são extremos de conectores ou terminais. Como não existem conectores em x, toda subseqüência de x exceto a primeira e a última precisam ter sobreposição de comprimento 2 nos dois lados (do contrário σ não teria comprimento $2m + 3n$). Portanto, a primeira seqüência precisa ser $\overline{v}u_{i+1}\overline{v}$. A seguinte deve ser $u_{i+1}\overline{v}u_{o_1}$ já que $u_{i+1} = o_0$. Além disso, toda seqüência em A_{u_i} exceto duas precisa ter sobreposição máxima de comprimento 2 em ambos os lados. Então toda seqüência em A_{u_i} exceto uma precisa suceder em ordem a única seqüência com a qual tem sobreposição igual a 2. Portanto, toda seqüência em A_{u_i} precisa ocorrer continuamente em ordem. Como x contém uma seqüência de A_{u_i} então ele precisa conter todas. Portanto, x é a u_{i+1}-superseqüência padrão para A_{u_i}. Como existe um arco de u_i para

u_{i+1} em G para $i = 1, \ldots, n-1$ onde $u_n = n$ e $u_1 = 1$, pois caso contrário σ não teria comprimento $2m + 3n$, e como $u_1, \ldots, u_n$ fornece uma permutação dos vértices de G, temos que esta ordenação dos vértices fornece um caminho hamiltoniano de 1 a n em G. □

Voltando ao nosso exemplo, considerando a superseqüência:

$$\triangleright\#\overline{1}3\overline{1}2\overline{1}3\#\overline{3}4\overline{3}5\overline{3}4\#\overline{4}2\overline{4}3\overline{4}5\overline{4}2\#\overline{2}5\overline{2}3\overline{2}5\#\$$$

de S, podemos ver que $(1, 2, 4, 2, 5)$ também é um caminho hamiltoniano em G. Note que o comprimento da seqüência também é, como no exemplo anterior, 33, exatamente $2|E| + 3|V|$.

Explicamos agora como restringir a que todas as seqüências sejam primitivas (que não possuem símbolos repetidos) e de tamanho exatamente h, para $h \geq 3$, mostrando como modificar as seqüências de S. Adicionamos ao alfabeto Σ o conjunto $\{\hat{a}|a \in V\}$. Para $h = 3$, precisamos apenas modificar A_v. Substitua as seqüências da forma $\overline{v}a\overline{v}$ pelas seqüências primitivas $\overline{v}a\hat{v}$, $a\hat{v}\hat{a}$ e $\hat{v}\hat{a}\overline{v}$. Substitua também as seqüências da forma $a\overline{v}b$ por $\hat{a}\overline{v}b$. Para $h \geq 4$, sejam y e y' seqüências primitivas de comprimento $h-4$ e $h-2$ sobre um alfabeto disjunto de Σ. Substitua o # em todos os conectores e terminais por y'. Para A_v, substitua as seqüências no S original da forma $\overline{v}a\overline{v}$ por $\overline{v}ay\hat{a}\hat{v}$, e aquelas da forma $a\overline{v}b$ por $\hat{a}\hat{v}y\overline{v}b$. Deixamos para o leitor a tarefa de determinar o valor de k que completa a redução neste caso. □

Na demonstração do teorema 4.3.1 o alfabeto utilizado não tem tamanho limitado por uma constante. Ou seja, seu tamanho cresce com o número de vértices de G. Gallant et at. [8] mostraram que o SCS ainda é NP-difícil quando o alfabeto tem tamanho 2.

4.3.1 Seqüências curtas

Na instância S descrita na prova do teorema 4.3.1, todas as seqüências têm comprimento igual e pelo menos 3. Quando as seqüências são mais curtas que isso, ou seja, para todo $s \in S$ temos que $|s| \leq 2$, existe um algoritmo linear para descobrir se existe uma superseqüência de tamanho k para S. Existe também um algoritmo linear para descobrir uma superseqüência comum mínima de S. Explicaremos agora a idéia geral deste segundo algoritmo.

Iremos analisar o que ocorre com as seqüências de S de comprimento 2, já que as seqüências de comprimento 1 podem ser concatenadas ao final da seqüência gerada pelo algoritmo. (Lembre-se que estamos considerando sempre S livre de subseqüências.)

O algoritmo de construção de uma superseqüência consiste em duas fases. A primeira fase se aplica enquanto existir uma letra a tal que há mais seqüências em S começando por a do que terminando por a. Caso não exista uma letra assim, vá para a segunda fase. Se existir, comece a superseqüência com a e encaixe tantas seqüências de S com sobreposição de tamanho 1 quanto possível, uma atrás da outra, não se importando que a reapareça. Ao fim deste passo, a seqüência corrente termina com uma letra que não é a primeira letra de nenhuma das seqüências remanescente de S. Caso tenhamos coberto todo o S, (ou seja, todas as seqüências de S foram usadas) teremos encontrado uma superseqüência comum mínima para S. Caso contrário, teremos uma seqüência

que cobre algumas seqüências de S de forma ótima. Esta seqüência começa e termina com letras diferentes porque por hipótese a está presente no começo de mais palavras do que no fim. Note, no entanto, que a diferença entre a quantidade de seqüências que começam com a e as seqüências que terminam com a foi diminuída de 1 se considerarmos apenas as seqüências ainda não cobertas. Podemos encontrar seqüências desta forma até reduzir essas diferenças de modo que, nas seqüências remanescentes de S, todas as letras estejam no começo em tantas seqüências quanto no fim. Isso conclui a primeira fase.

Infelizmente não podemos fazer nada melhor do que concatenar as seqüências obtidas nesta primeira fase, obtendo uma seqüência corrente. Na segunda fase, começamos por uma letra arbitrária e prosseguimos realizando sobreposições de tamanho 1 tanto quanto possível. Mas desta vez teremos que a seqüência encontrada obrigatoriamente começa e termina com a mesma letra. Esta seqüência é tal que qualquer "rotação cíclica" dela é superseqüência das mesmas seqüências de S que a originaram. Por "rotação cíclica", entenda a rotação cíclica da seqüência removido o último símbolo, e posteriormente adicionando-se o primeiro símbolo, após a rotação, ao final. (Por exemplo se temos *abcda*, removemos o a do final, fazemos a rotação para *cdab* e então inserimos c no final.) Portanto se a seqüência corrente contiver uma das letras desta seqüência, podemos inserir esta seqüência na mesma. Caso isso não ocorra, não podemos fazer nada melhor do que concatenar esta seqüência à seqüência corrente, que estamos formando para cobrir S. Podemos realizar o mesmo processo até cobrir todo o S e então teremos encontrado uma superseqüência comum mínima para S. (Isso requereria uma argumentação, mas decidimos apenas apresentar a descrição do algoritmo.)

4.4 Resultados de inaproximabilidade

Nesta seção, apresentamos um resultado importante que mostra que o SCS é difícil de aproximar. Neste caso temos que se existir um esquema de aproximação em tempo polinomial para o SCS então P = NP, ou seja, é improvável que exista um algoritmo que, para um ϵ fixo, seja uma $(1+\epsilon)$-aproximação.

Definição 4.4.1 (L-redução)**.** *Sejam A e B dois problemas de otimização. Dizemos que A L-reduz a B se existem dois algoritmos polinomiais f e g e constantes α e β, tais que para qualquer instância I de A:*

1. *O algoritmo f produz uma instância $I' = f(I)$ de B, tal que o custo ótimo de I e I', respectivamente denotados por $Opt_A(I)$ e $Opt_B(I')$, satisfazem $Opt_B(I') \leq \alpha \cdot Opt_A(I)$, e*

2. *Dada qualquer solução viável de I' com custo c', o algoritmo g produz uma solução de I com custo c tal que $|c - Opt_A(I)| \leq \beta \cdot |c' - Opt_B(I')|$.*

Definição 4.4.2. *Denotamos por* $\text{TSP}_4(1,2)$ *a seguinte variante do* TSP*: Dado um grafo completo G, vértices x e y distintos e $c(e) \in \{1,2\}$ para todo $e \in E(G)$, tal que o subgrafo H de G induzido pelas arestas de custo 1 tem grau máximo 4, encontrar um caminho hamiltoniano em G de x a y de custo mínimo.*

Papadimitriou e Yannakakis [17, Cor. 1] mostram que o $\text{TSP}_4(1,2)$ é Max SNP-difícil. Se substituirmos cada aresta de G por dois arcos, um em cada

direção, obtemos uma redução para a versão dirigida do problema, onde por grau máximo entende-se o grau de saída máximo. A partir de agora utilizamos a notação $\text{TSP}_4(1,2)$ para a versão dirigida do problema. Blum et al. [4] forneceram uma prova[4] curta disso que detalhamos mais a seguir. Adicionamos como hipótese que H tenha $\delta^+(v) \geq 1$ para todo $v \in V(G)$ e que $V = \{1, \ldots, n\}$, $x = 1$ e $y = n$.

Teorema 4.4.1. O SCS é Max SNP-difícil.

Demonstração. Iremos mostrar que o $\text{TSP}_4(1,2)$ L-reduz ao SCS. Utilizaremos a mesma construção do teorema 4.3.1 para o grafo H da definição 4.4.2. Como aqui permitimos que exista $v \in V(G)$ tal que $\delta^+(v) = 1$, caso isso ocorra, seja w o único vizinho de v. Criaremos um conjunto A_v degenerado da seguinte forma $A_v = \{\overline{v}w\overline{v}, w\overline{v}w\}$. Note que $|A_v| = 2\delta^+(v) = 2$ como anteriormente e que $|STD(\overline{v}, w)| = 2\delta^+(v) + 2 = 4$ como ocorria anteriormente. Portanto podemos usar os resultados anteriores sem nenhum problema.

Lema 4.4.1. $|\text{SCS}^*(S)| \leq 22 \cdot \|\text{TSP}_4^*(1,2)(I)\|$.

Demonstração. Seja $I = (G, c)$ a instância do $\text{TSP}_4(1,2)$ onde $G = (V, E)$ e H é o subgrafo de G induzido pelos arcos de custo 1. Seja S a instância do SCS construída a partir de H como indicado no teorema 4.3.1. Sejam $n = |V|$, $m = |E(H)|$ e k o número mínimo de arcos em G de custo 2 que precisam ser adicionados a H para que o mesmo tenha um caminho hamiltoniano. Observe que $\|\text{TSP}_4(1,2)^*(I)\| = n - 1 + k$ e que $m \leq 4n$ pois H tem grau (de saída) máximo 4. Mostraremos que o comprimento de uma superseqüência comum mínima para S é $2m + 3n + k$.

Se $k = 0$, o resultado segue do teorema 4.3.1. Caso $k > 0$, considere um caminho hamiltoniano P de custo mínimo em G. Se $ij \in P$ e $c(ij) = 2$ então como $ij \notin E(H)$ não temos como construir $STD(\overline{i}, j)$. Em vez de utilizar a seqüência $STD(\overline{i}, j)$, podemos usar $STD(\overline{i}, w)$, onde $iw \in E(H)$, e então concatenar (em vez de sobrepor) o conector $j\#\overline{j}$ e proceder normalmente. Seja σ a superseqüência comum para S obtida desta maneira. Note que $|\sigma| = 2m+3n+k$ já que em k arcos não pudemos usar $STD(\overline{i}, j)$. Vamos mostrar agora que σ é uma superseqüência comum mínima para S. Suponha que uma superseqüência comum mínima σ^* para S tenha comprimento menor do que $2m+3n+k$. Como visto anteriormente (teorema 4.3.1), podemos assumir que σ começa por $\triangleright\#\overline{1}$ e termina com $n\#\$$ e, para todo $v \in V \setminus \{n\}$, o conjunto A_v é coberto por algum $STD(\overline{v}, w)$, onde $vw \in E(H)$. Então, se $|\sigma^*| < |\sigma|$, existe um caminho hamiltoniano em G de 1 até n indicado pelos conectores e tal caminho utiliza menos de k arcos de custo 2, contrariando o fato que P é ótimo. Assim sendo $|\text{SCS}^*(S)| = 2m + 3n + k$.

Note que $2m + 3n + k \leq 2(4n) + 3n + k \leq 11n + k \leq 22(n - 1 + k)$ para $n \geq 2$, completando a prova do lema. □

Lema 4.4.2. Seja I uma instância do $\text{TSP}_4(1,2)$ e S a instância correspondente (pela redução) do SCS. Seja σ uma superseqüência comum para S de comprimento c. Então existe uma solução viável de I de custo c' tal que $|c' - \|\text{TSP}_4(1,2)^*(I)\|| \leq |c - |\text{SCS}(S)^*||$.

[4] Vale notar uma pequena discrepância entre nosso resultado e o dos autores. O comprimento da seqüência dos mesmos é maior uma unidade em relação ao nosso.

Demonstração. Primeiramente iremos transformar a superseqüência comum σ dada em outra num formato desejado, mostrando que a nova é tão curta ou mais do que a original. Nosso objetivo inicial é que apareça para todo $v \in V$ uma $STD(\overline{v}, w)$ na nova superseqüência. Para todo $v \in V$, sejam $\sigma_1, \ldots, \sigma_k$ as seqüências maximais formadas pelas sobreposições de tamanho 2 das primeiras ocorrências de cada seqüência de A_v. Claro que σ_i, com $i \in \{1, \ldots, k\}$, realiza sobreposição de tamanho no máximo 1 à esquerda e à direita na superseqüência. Note também que podemos sobrepor todas as seqüências σ_i com sobreposição de tamanho 2 e formaremos a $STD(\overline{v}, w)$ para algum vizinho w de v. Isso pode ser realizado desfazendo as sobreposições à esquerda de todas essas seqüências menos a primeira e todas à direita exceto a da última e concatenando essas seqüências intermediárias no fim da superseqüência. Note que a nova seqüência é uma superseqüência comum para S e que ela tem tamanho menor ou igual à original.

Temos agora uma seqüência formada por $STD(\overline{v}, w)$ para todo $v \in V$, conectores, terminais e possivelmente um resíduo no final da seqüência que não cobre exclusivamente nenhuma seqüência de S, e que pode portanto ser removido. Iremos agora organizar os conectores e terminais. Primeiramente, garantimos que o conector de um determinado vértice v seja sobreposto por $STD(\overline{v}, w)$. A nova seqüência gerada é uma superseqüência comum para S e tem comprimento menor ou igual a original, já que nem w-superseqüências e nem conectores realizam sobreposição entre si. Como o terminal $\triangleright\#\overline{1}$ não realiza sobreposição à esquerda com nenhuma outra seqüência, podemos colocá-lo no começo da nova seqüência e concatenar a o prefixo da seqüência anterior até $\triangleright\#\overline{1}$ no final desta nova seqüência. Do mesmo modo, como o terminal $n\#\$$ não realiza sobreposição à direita, podemos trocar o sufixo pelo mesmo.

Seja σ' a superseqüência comum para S gerada desta maneira. Note que $|\sigma| = 2m + 3n + r$, onde r é o número de conectores que foram concatenados (e não sobrepostos) ao final de cada w-superseqüência. Analisando a ordem dos vértices nos conectores temos um caminho P de custo $c' = n-1+r$. Portanto $c' - \|\textsc{tsp}_4(1,2)(I)\| = n-1+r-(n-1+k) = r-k = 2m+3n+r-(2m+3n+k) = |\sigma'| - |\textsc{scs}^*(S)| \leq c - |\textsc{scs}^*(S)|$ e portanto $|c' - \|\textsc{tsp}_4(1,2)^*(I)\|| \leq |c - |\textsc{scs}^*(S)||$. □

Concluímos a demonstração do teorema alegando que ambos os algoritmos são polinomiais no tamanho de I. □

4.5 Cobertura por circuitos

Como mencionado anteriormente, existe uma relação íntima entre o SCS e o TSP que não pode ser muito explorada. Mas utilizando esta informação podemos usar outras formas de encontrar informações pertinentes no digrafo de prefixos.

Definição 4.5.1 (Cobertura por circuitos)**.** *Uma* ***cobertura por circuitos*** *de um digrafo G é um conjunto de arcos tais que os mesmos formam circuitos direcionados em G e todo vértice de G pertence a um único destes circuitos direcionados.*

Definição 4.5.2 (Cobertura por circuitos de custo mínimo)**.** *Definimos uma* ***cobertura por circuitos de custo mínimo*** *de um digrafo $G = (V, E)$ e uma*

função de $c : E \to \mathbb{Q}$ como uma cobertura por circuitos $\text{CYC}^*(G, c)$ de G tal que para qualquer cobertura por circuitos Φ de G, $c(\text{CYC}^*) \leq c(\Phi)$. Denotamos $c(\text{CYC}^*)$ por $\|\text{CYC}^*(G, c)\|$. Quando G e c podem ser extraídos do contexto escrevemos CYC^*.

Definição 4.5.3 (Emparelhamento)**.** *Um **emparelhamento** em um grafo $G = (V, E)$ é um conjunto $M \subseteq E$ tal que para todo arco $uv \in M$, não existe $ij \in M$ e $ij \neq uv$ tal que $i \in \{u, v\}$ ou $j \in \{u, v\}$. Ou seja, em M nenhum arco compartilha uma ponta com outro arco.*

Definição 4.5.4 (Emparelhamento perfeito)**.** *Um emparelhamento é **perfeito** se para todo $v \in V$, existe ij em M tal que $v = i$ ou $v = j$. Ou seja, M^* cobre todos os vértices de G.*

Definição 4.5.5 (Emparelhamento perfeito de custo mínimo)**.** *Um **emparelhamento perfeito de custo mínimo** de um grafo $G = (V, E)$ com uma função custo $c : E \to \mathbb{Q}$ associada, é um emparelhamento perfeito M^* de G tal que para qualquer emparelhamento perfeito M de G, temos que $c(M^*) \leq c(M)$.*

O problema da cobertura por circuitos de custo mínimo pode ser resolvido por uma redução ao problema do emparelhamento perfeito de custo mínimo num grafo bipartido.

Podemos reduzir o problema da seguinte forma: dados um digrafo $G = (V, E)$ e uma função custo $c : E \to \mathbb{Q}$, descrevemos um grafo bipartido $H = (S \cup T, E')$ e uma função $c_H : E \to \mathbb{Q}$. Ambos S e T são cópias de V. Para cada vértice v em V, denotamos por v' e v'' suas cópias respectivamente em S e T. Para cada arco $e = uv$ em E, existe uma aresta $u'v''$ em E'. Ademais, $c_H(u'v'') = c(uv)$. Perceba que um emparelhamento perfeito fornece naturalmente uma cobertura por circuitos e que um emparelhamento perfeito de custo mínimo fornece uma cobertura por circuitos de custo mínimo. Como existe um algoritmo polinomial para encontrar um emparelhamento perfeito de custo mínimo [16], podemos encontrar uma cobertura por circuitos de custo mínimo de forma eficiente.

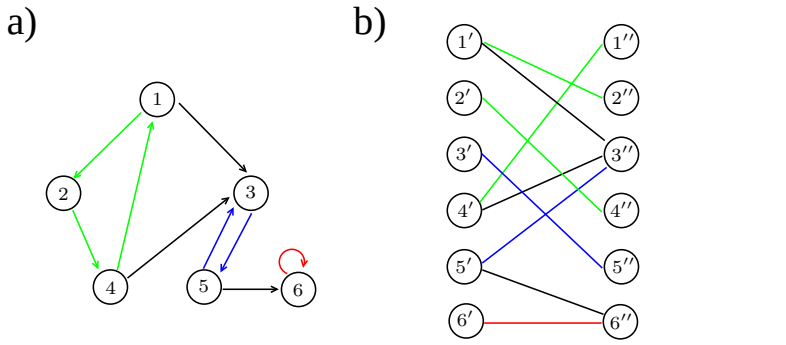

Figura 4.5: Redução da cobertura por circuitos de custo mínimo ao problema de emparelhamento perfeito de custo mínimo. Note como cada circuito em (a) aparece no emparelhamento perfeito em (b).

Definição 4.5.6. *Denotamos por* $\text{CYC}^*(S)$ *uma cobertura por circuitos de custo mínimo para o digrafo de prefixos* $G_p(S)$ *para um dado conjunto* S *de seqüências.*

Podemos encontrar uma cobertura por circuitos de custo mínimo em G_p e portanto achar superseqüências comuns para cada elemento de uma partição de S determinada pelos circuitos da cobertura. Concatenando as mesmas, obtemos uma superseqüência comum para S.

Como uma solução do TSP é também uma solução viável da cobertura por circuitos, temos que:

$$\|\text{CYC}^*(S)\| \leq \|\text{TSP}^*(S)\| \leq |\text{SCS}^*(S)|.$$

Considere um algoritmo, que chamaremos de CYCLECOVER, que resolva uma instância da cobertura por circuitos de custo mínimo, devolvendo uma lista de circuitos que são representados cada um por uma seqüência de vértices [4]. Podemos, então, encontrar uma aproximação para o SCS com o seguinte algoritmo:

```
CICLECOVER-APROX (S)
 1  Seja (G, p) o digrafo de prefixo de S
 2  Φ ← CICLECOVER(G_p, c)
 3  r ← ∅
 4  para i ← 1 até |Φ| faça
 5      φ_i ← Φ(i)
 6      σ_i ← ∅
 7      n ← |φ_i|
 8      para j ← 1 até n − 1 faça
 9          σ_i ← σ_i pref(φ_i(j), φ_i(j + 1))
10       σ_i ← σ_i φ_i(n)
11       r ← r σ_i
12   devolva r
```

4.5.1 Algumas propriedades

Apresentamos agora algumas definições e resultados que serão usados para provar o fator de aproximação do CICLECOVER-APROX.

Lema 4.5.1. Se cada seqüência em $S' \subseteq S$ é uma subseqüência de t^∞ para uma seqüência t, então existe um circuito de custo no máximo $|t|$ no digrafo de prefixos $G_p(S)$ cobrindo todos os vértices correspondendo às seqüências de S'.

Demonstração. Para cada seqüência de S' encontre o ponto inicial da primeira ocorrência em t^∞. Estes pontos serão distintos já que S é livre de subseqüências e estarão na primeira cópia de t. Considere as seqüências na ordem dada pelos seus pontos iniciais, e tome o circuito no digrafo de prefixos $G_p(S)$ que cobre todos os vértices nesta ordem. É claro que o custo deste circuito é no máximo $|t|$. □

Definição 4.5.7. *Dizemos que uma seqüência* s *é mapeada em uma seqüência cíclica* ϕ *se* s *é subseqüência de* ϕ^∞*.*

Lema 4.5.2. [Lema da Sobreposição Máxima] Sejam ϕ e ϕ' quaisquer dois circuitos em $\text{CYC}^*(S)$ e α e α' duas seqüências não necessariamente em S que são vértices de ϕ e ϕ' respectivamente. Temos que $|ov(\alpha, \alpha')| < |\phi| + |\phi'|$.

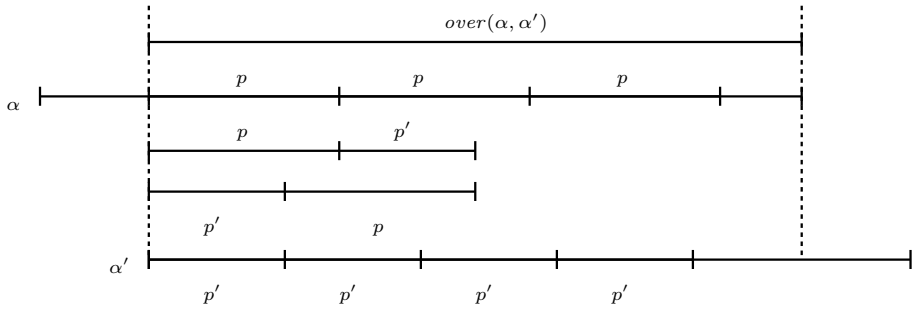

Figura 4.6: Sobreposição de α com α'. Note que, se $|over(\alpha, \alpha')| \geq |\phi| + |\phi'|$ então $pp' = p'p$.

Demonstração. Suponha, por contradição, que $|over(\alpha, \alpha')| \geq |\phi|+|\phi'|$. Denote por p (p') o prefixo de tamanho $|\phi|$ ($|\phi'|$ respectivamente) de $over(\alpha, \alpha')$. Note que $over(\alpha, \alpha')$ é prefixo tanto de p^∞ como de $(p')^\infty$. Portanto, p é prefixo de $(p')^\infty$ e p' é prefixo de p^∞. Como $|over(\alpha, \alpha')| \geq |\phi| + |\phi'|$, segue que p e p' comutam, ou seja, $pp' = p'p$. Mas com isso temos que $p^\infty = (p')^\infty$ já que para qualquer $k > 0$ temos que $p^k(p')^k = (p')^k p^k$. Então, para qualquer $N > 0$, o prefixo de tamanho N de p^∞ é o mesmo que o de $(p')^\infty$. Pelo lema anterior, existe um circuito de custo no máximo $|\phi|$ no digrafo de prefixos cobrindo todas as seqüências em S que são mapeadas em ϕ e ϕ', contradizendo o fato de que $\textsc{cyc}^*(S)$ é uma cobertura por circuitos de custo mínimo. □

Teorema 4.5.1. O algoritmo CICLECOVER-APROX é uma 4-aproximação para o problema da superseqüência comum mínima.

Demonstração. Note que a saída do algoritmo é $r = \sigma_1\sigma_2\cdots\sigma_{|\Phi|}$ e que $|\sigma_i| = \sum_{j=1}^{|\phi_i|-1} |pref(\phi_i(j), \phi_i(j+1))| + |\phi_i(|\phi_i|)|$. Seja $S_f = \{\phi_i(|\phi_i|)\}$ o conjunto das seqüências dos $\sigma's$ que foram escolhidas por última para fechar um circuito. Então $|r| = \|S_f\| + \sum_{\phi_i\in\Phi}\sum_{j=1}^{n-1} pref(\phi_i(j), \phi_i(j+1))$, e portanto $|r| \leq \|S_f\| + \sum_{\phi_i\in\Phi}|\phi_i| = \|P_f\| + \|\textsc{cyc}^*(S)\|$. Como $S_f \subseteq S$ temos que $\|\textsc{scs}^*(S_f)\| \leq \|\textsc{scs}^*(S)\|$ e como quaisquer duas seqüências α e α' em S_f são vértices de circuitos ϕ e ϕ' diferentes em $\textsc{cyc}^*(S)$ temos que $|over(\alpha, \alpha')| < |\phi| + |\phi'|$. Então $\sum_{i=1}^{|S_f|} over(s_{o_i}, s_{o_{i+1}}) \leq 2\|\textsc{cyc}^*(S)\|$ para qualquer permutação $(o_1, o_2, \ldots, o_{|S_f|})$ das seqüências de S_f. Portanto, $|\textsc{scs}^*(S)| \geq |\textsc{scs}^*(S_f)| \geq \|S_f\| - 2\|\textsc{cyc}^*(S)\|$. Como $\|\textsc{cyc}^*(S)\| \leq |\textsc{scs}^*(S)|$, temos que $\|S_f\| \leq 3\|\textsc{cyc}^*(S)\|$. Portanto, $|r| \leq \|S_f\| + \|\textsc{cyc}^*(S)\| \leq 4\|\textsc{cyc}^*(S)\|$. □

4.6 Um segundo algoritmo guloso

Quando descrevemos o algoritmo GREEDY, criamos a restrição que, a cada passo, as seqüências s e t escolhidas fossem distintas. Isso garantia que não haveriam circuitos sendo formados pelos arcos escolhidos no digrafo de sobreposição máxima. Considere agora o seguinte algoritmo: crie um conjunto T inicialmente vazio e a cada passo escolha duas seqüências s e t de S que tenham sobreposição máxima. Se $s \neq t$, então troque s e t pela sua composição em

S. Caso contrário, remova s de S e insira em T. Quando S estiver vazio, o conjunto T terá várias seqüências que cobrem o S original. Basta concatená-las para obter a superseqüência comum para S. Veja o algoritmo abaixo:

MAXOVERLAP-LAÇOS (T)
1 $over_{max} \leftarrow 0$
2 $s \leftarrow t \leftarrow \emptyset$
3 **para cada** $u \in T$ **faça**
4 **para cada** $t \in T$ **faça**
5 **se** $over_{max} \leq over(u,t)$ **então**
6 $over_{max} \leftarrow over(u,t)$
7 $s \leftarrow u$
8 $t \leftarrow t$
9 **devolva** (s,t)

MGREEDY-SEQÜÊNCIAS (S)
1 $T \leftarrow \emptyset$
2 $r \leftarrow \emptyset$
3 **enquanto** $|S| \neq \emptyset$ **faça**
4 $(s,t) \leftarrow$MAXOVERLAP-LAÇOS(S)
5 **se** $s = t$ **então**
6 $S \leftarrow S \setminus \{s\}$
7 $T \leftarrow T \cup \{s\}$
8 **senão**
9 $S \leftarrow S \setminus \{s,t\} \cup \{comp(s,t)\}$
10 **devolva** T

MGREEDY (S)
1 $T \leftarrow$ MGREEDY-SEQÜÊNCIAS(S)
2 $r \leftarrow \emptyset$
3 **para cada** $t \in T$ **faça**
4 $r \leftarrow rt$
5 **devolva** r

Este é um algoritmo tão fácil de implementar quanto o GREEDY. Mas de fato, provaremos agora que ele fornece uma cobertura por circuitos de custo mínimo, sendo assim uma 4-aproximação para o SCS potencialmente mais fácil de implementar do que o algoritmo CICLECOVER-APROX. Enunciamos a seguir o seguinte lema provado por Turner [21] que será usado no teorema a seguir.

Lema 4.6.1 (Condições de Monges)**.** Sejam u, u^+, v^- e v seqüências, não necessariamente diferentes, tais que $|over(u,v)| \geq \max\{|over(u,u^+)|, |over(v^-,v)|\}$. Então, $|over(u,v)| + |over(v^-,u^+)| \geq |over(u,u^+)| + |over(v^-,v)|$ e $|pref(u,v)| + |pref(v^-,u^+)| \leq |pref(u,u^+)| + |pref(v^-,v)|$.

Teorema 4.6.1. O MGREEDY é uma 4-aproximação para o SCS.

Demonstração. Seja G o digrafo associado a S e seja M o conjunto de arcos de G escolhidos pelo algoritmo MGREEDY. Note que, para um determinado arco st de G temos que $|over(s,t)| + |pref(s,t)| = |s|$. Portanto, uma cobertura por circuitos de custo mínimo para G_p é uma cobertura por circuitos de custo máximo para G_s. Considere agora uma cobertura por circuitos de custo máximo

N para G_s que contenha o maior número de arcos em comum com M. Vamos mostrar que $M = N$.

Suponha que isto não ocorra. Como M e N tem o mesmo número de arcos, existe um arco em $M \setminus N$. Seja $e = (u, v)$ o primeiro arco de M na ordem induzida pelo MGREEDY que não está em N. Então existe (v^-, v) e (u, u^+) em N. Note que (v^-, v) e (u, u^+) não pertencem a M. Como $|over(u, v)| \geq \max\{|over(v^-, v)|, |over(u, u^+)|\}$, pelo lema 4.6.1, $|over(v^-, v)| + |over(u, u^+)| \leq |over(u, v)| + |over(v^-, u^+)|$. Portanto, podemos substituir (v^-, v) e (u, u^+) em N por (u, v) e (v^-, u^+) e teríamos uma cobertura por circuitos de custo não menor que o custo de N e que tem mais arcos em comum com M do que N, contrariando a escolha de N. □

4.7 Alcançando a 3-aproximação

No MGREEDY encontramos seqüências relacionadas a uma cobertura por circuitos e as concatenamos. Poderíamos considerar agora o conjunto de seqüências devolvido pelo MGREEDY-SEQÜÊNCIAS e tentar agora encontrar uma superseqüência comum mínima para o mesmo. O algoritmo TGREEDY, apresentado a seguir, utiliza esta idéia para encontrar uma aproximação melhor para o SCS.

TGREEDY (S)
1 $T \leftarrow$ MGREEDY-SEQÜÊNCIAS(S)
2 $t \leftarrow$ GREEDY(T)
3 **devolva** t

Veremos futuramente que, para uma instância S do SCS o algoritmo GREEDY garante que a compressão obtida em S é maior ou igual a metade da compressão ótima. Neste ponto, mostraremos apenas que existe um algoritmo com esta característica.

Lema 4.7.1. Existe um algoritmo para o SCS tal que a compressão obtida para uma instância S é maior ou igual a metade da compressão ótima de S.

Demonstração. O seguinte algoritmo é apresentado por Vazirani [23]. Seja G o digrafo de sobreposição de S sem laços. Encontre agora uma cobertura por circuitos de valor máximo para G, ou seja, uma cobertura C tal que $over(C)$ seja máximo. Isso pode ser feito através da mesma redução apresentada anteriormente para encontrar uma cobertura por circuitos de custo mínimo, e portanto, podemos calcular esta cobertura em tempo polinomial. Note agora que, como G não contém laços, todo circuito tem pelo menos dois arcos. Para cada circuito remova o arco de menor valor, encontrando assim uma série de caminhos disjuntos. Note que o valor da compressão ótima para S é igual ao valor de um caminho hamiltoniano de comprimento máximo em G, que por sua vez tem valor menor ou igual ao da cobertura por circuitos de valor máximo para G. Como removemos o arco de menor valor para cada circuito, temos que a soma dos valores destes caminhos é pelo menos metade do valor da cobertura por circuitos e portanto é pelo menos metade da compressão ótima. Considere agora a sobreposição máxima entre as seqüências dos caminhos e então concatene as seqüências resultantes. A superseqüência comum assim obtida para S atinge pelo menos metade da compressão ótima. □

Portanto podemos modificar a linha 2 do TGREEDY para executar o algoritmo mencionado neste lema. No teorema a seguir usamos apenas o fato de

que o algoritmo executado na linha 2 garante uma boa compressão.

Teorema 4.7.1. O algoritmo TGREEDY é uma 3-aproximação para o SCS.

Demonstração. Considere uma instância $S = \{s_1, s_2, \ldots, s_n\}$, e sejam $T = \{t_1, \ldots, t_m\}$ o conjunto devolvido por MGREEDY-SEQÜÊNCIAS(S) e t a superseqüência comum para T devolvida por TGREEDY(S).

Seja $w : T \to \mathbb{N}_+$ a função que leva $r \in T$ ao tamanho do circuito associado em G_s.

Pelo lema da sobreposição máxima (lema 4.5.2) temos que, para $u, v \in T$, $|over(u, v)| < |w(u)| + |w(v)|$ e T é uma cobertura por circuitos de custo mínimo (pelo teorema 4.6.1). Seja $(o_1, \ldots, o_m)$ uma ordenação de T que maximize $\sum_{i=1}^{n-1} |over(t_{o_i}, t_{o_{i+1}})|$. Seja t^* a superseqüência comum mínima obtida por esta ordenação. Portanto temos que $|t^*| = \|T\| - \sum_{i=1}^{n-1} |over(t_{o_i}, t_{o_{i+1}})|$. Então

$$|t^*| = \|T\| - \sum_{i=1}^{n-1} |over(t_{o_i}, t_{o_{i+1}})| > \|T\| - 2w(T) \Rightarrow \|T\| - |t^*| < 2w(T).$$

Usando o lema 4.7.1 temos:

$$\|T\| - |t| \geq \frac{\|T\| - |t^*|}{2} = \|T\| - |t^*| - \frac{\|T\| - |t^*|}{2} \geq \|T\| - |t^*| - w(T)$$

e portanto $|t| \leq |t^*| + w(T)$.

Seja $R = \{r_1, r_2, \ldots, r_m\}$, onde r_i é a seqüência inicial de t_i, isto é, a seqüência que foi escolhida primeiro pelo algoritmo para formar t_i. Seja $T' = \{t'_1, t'_2, \ldots, t'_m\}$, onde $t'_i = t_i \circ r_i$, e portanto t'_i começa e termina com r_i. Seja t'^* uma superseqüência comum mínima para T'. Note que $|t^*| \leq |t'^*|$ e que à compressão máxima em T' é maior ou igual a compressão máxima de R (já que em T' toda seqüência t_i começa e termina por r_i) e seja r^* uma superseqüência comum mínima para R. Temos que $\|T'\| - |t'^*| \geq \|R\| - |r^*|$. Lembre-se que s^* é uma superseqüência comum mínima para S. Temos que $|r^*| \leq |s^*|$ já que $R \subseteq S$. Por fim, note que $\|T'\| = \|R\| + w(T)$. Portanto temos

$$\|T'\| - |t'^*| \geq \|R\| - |r^*| \Rightarrow \|R\| + w(T) - |t'^*| \geq \|R\| - |r^*| \Rightarrow |t'^*| - w(T) \leq |r^*|$$

e como, $|r^*| \leq |s^*|$, temos que $|t'^*| \leq |s^*| + w(T)$. Como $|t| \leq |t^*| + w(T)$, temos que $|t| \leq |s^*| + 2w(T)$, e portanto, $|t| \leq 3|s^*|$ já que $|s^*| \geq w(T)$, concluindo a demonstração. □

4.8 Análise do algoritmo guloso

Nesta seção apresentaremos uma análise da razão de aproximação do GREEDY [4]. Em particular, provaremos que o GREEDY é uma 4-aproximação para o SCS. A análise será feita comparando as escolhas de arcos do GREEDY com as escolhas realizadas pelo MGREEDY e no final, com uma análise amortizada, teremos o resultado desejado.

Podemos pensar no GREEDY como um algoritmo que considera um a um cada arco do digrafo G_s, com os arcos ordenados pelo seu custo em G_s. Vamos então escrever $e \prec f$ para dois arcos de G_s se e é considerado antes que f pelo GREEDY.

Os arcos escolhidos pelo GREEDY descrevem um caminho hamiltoniano P em G_s. Sejam $s_1, \ldots, s_n$ as seqüências de S na ordem em que elas aparecem em P. Para descarregar a notação, abusaremos dela usando $1, \ldots, n$ para nos referirmos às seqüências $s_1, \ldots, s_n$.

Vamos descrever uma partição específica de $\{1, \ldots, n\}$ em três conjuntos. Para tanto, precisamos de algumas definições.

Definição 4.8.1. *Durante a execução do GREEDY, o conjunto de arcos escolhidos determina uma coleção $\mathcal{P}$ de caminhos disjuntos. Um arco (j, i) é* ***de retorno*** *se, quando é considerado pelo GREEDY, há um caminho de i a j em $\mathcal{P}$. Ou seja, a escolha de (j, i) fecharia um circuito em $\mathcal{P}$. Note que, se (j, i) é um arco de retorno, então $i \leq j$.*

Definição 4.8.2. *Seja $f = (j, i)$ um arco de retorno. Dizemos que f* ***gera*** *o intervalo fechado $I_f = [i, j]$.*

Lema 4.8.1. A família $\mathcal{F} = \{I_e : e \text{ é arco de retorno}\}$ é laminar. Ademais, se e e f são arcos de retorno tais que $I_f \subset I_e$ então $f \prec e$.

Demonstração. Note que, se $f = (j, i)$ é um arco de retorno, quando o GREEDY considerou f, os arcos $(i - 1, i)$ e $(j, j + 1)$ ainda não tinham sido escolhidos. (Caso contrário $\mathcal{P}$ não teria nesse momento um caminho de i a j.)

Suponha que dois arcos de retorno e e f são tais que $I_e = [a, c]$ e $I_f = [b, d]$ com $a < b < c < d$. Se $e \prec f$ então pela afirmação acima, temos que f compartilha a cabeça com o arco que já foi escolhido entrando em b contrariando o fato de f ser um arco de retorno. Se $e \prec f$, e compartilha a cauda com o arco que já foi escolhido saindo de c e temos outra contradição. Portanto não existem dois intervalos que se cruzam, formando assim uma família laminar.

Suponha agora que existem arcos de retorno e e f tais que $I_f \subset I_e$ mas $e \prec f$. Mas, como visto, todos os arcos no caminho do intervalo fechado I_e já foram escolhidos e portanto f compartilha a cabeça e a cauda com arcos já escolhidos, contradizendo o fato de que f é um arco de retorno. □

Sejam $[i_1, j_1], \ldots, [i_k, j_k]$ os intervalos minimais de $\mathcal{F}$, com $j_\ell < i_{\ell+1}$, para $\ell = 1, \ldots, k - 1$. Seja f_ℓ o maior arco na ordem $\prec$ no subcaminho de P de j_ℓ e $i_{\ell+1}$. Seja h_ℓ a cabeça de f_ℓ e t_ℓ a cauda de f_ℓ, para $\ell = 1, \ldots, k - 1$. Ademais seja $t_0 = 1$ e $h_k = n$. Veja a figura 4.7.

Chamamos os arcos $f_1, \ldots, f_k$ de **fracos**. Note que tais arcos são aqueles que "emendam" caminhos de $\mathcal{P}$ correspondentes a intervalos minimais de $\mathcal{F}$. Eles são escolhidos "por último", ou seja, são os de menor custo em G_s — por isso são chamados de fracos.

A partição então consiste nos seguintes três conjuntos:

$$S_E = \cup_{\ell=1}^{k}[t_{\ell-1}, i_\ell),\ S_M = \cup_{\ell=1}^{k}[i_\ell, j_\ell] \text{ e } S_D = \cup_{\ell=1}^{k}(j_\ell, h_\ell].$$

Sejam $L_E = \sum_{\ell \in S_E} |pref(l, l + 1)|$, $L_M = \sum_{\ell=1}^{k} |comp(i_\ell, \ldots, j_\ell)|$ e $L_D = \sum_{\ell \in S_D} |pref((\ell + 1)^R, \ell^R)|$, onde $comp(s_1, \ldots, s_m)$ é a composição das seqüências de s_1 até s_m e s^R denota o reverso da seqüência s. Seja $O_W = \sum_{\ell=1}^{k} |over(h_\ell, t_\ell)|$. Note que o comprimento da seqüência gerada pelo GREEDY é

$$L_E + L_M + L_D - O_W$$

Queremos mostrar que essa soma é no máximo $4|s^*|$. Para tanto, precisamos introduzir mais alguns elementos.

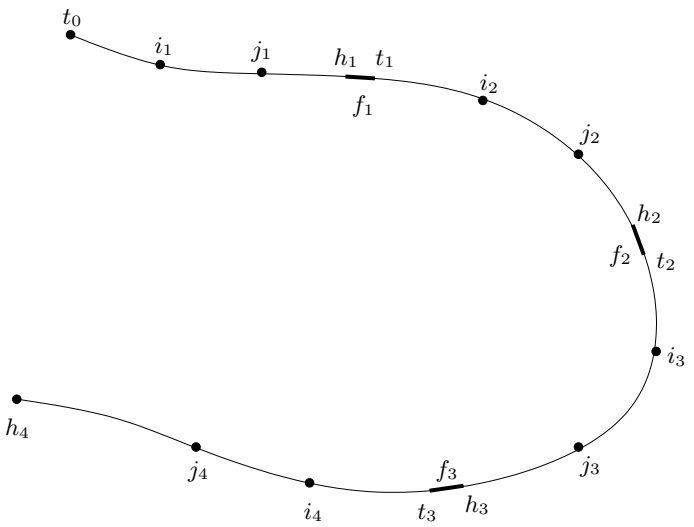

Figura 4.7: Localização dos intervalos minimais e dos arcos fracos.

Definição 4.8.3. *Seja $V_E = S_E \cup S_M$ e A_E o conjunto de arcos de P entre vértices de V_E que não sejam arcos fracos. De forma similar, definimos V_D e A_D.*

Definição 4.8.4. *Seja M_E uma cobertura por circuitos de V_E de custo máximo em $G_s[V_E]$ (o subgrafo de G_s induzido por V_E). Analogamente definimos M_D.*

Definição 4.8.5. *Seja E um conjunto de arcos. Denotamos por $over(E)$ a soma dos comprimentos das sobreposições máximas de E, ou seja, $over(E) = \sum_{e \in E} |over(e)|$.*

Lema 4.8.2. $L_E + L_M + L_D \leq 2|\textsc{scs}^*| + over(M_E) - over(A_E) + over(M_D) - over(A_D) - L_M$.

Demonstração. Para um conjunto V de seqüências e um conjunto de A de arcos sobre V, definimos $\textsc{cost}(A) = \|V\| - over(A)$. Como $S_E \subseteq S$ e $S_D \subseteq S$ e ambos M_E e M_D são coberturas por circuito de custo máximo de $G_s[V_E]$ e $G_s[V_D]$ respectivamente, então $\textsc{cost}(M_E) \leq |\textsc{scs}^*|$ e $\textsc{cost}(M_D) \leq |\textsc{scs}^*|$. Ademais temos que $\textsc{cost}(A_E) = L_E + L_M$ e $\textsc{cost}(A_D) = L_D + L_M$. Portanto, temos que:

$$\begin{aligned}
L_E + L_M + L_D &= (L_E + L_M) + (L_M + L_D) - L_M \\
&= \textsc{cost}(A_E) + \textsc{cost}(A_D) - L_M \\
&= \|V_E\| - over(A_E) + \|V_D\| - over(A_D) - L_M \\
&= \textsc{cost}(M_E) + over(M_E) - over(A_E) \\
&\quad + \textsc{cost}(M_D) + over(M_D) - over(E_D) - L_M \\
&\leq 2|\textsc{scs}^*| + over(M_E) - over(A_E) \\
&\quad + over(M_D) - over(A_D) - L_M.
\end{aligned}$$

□

Definição 4.8.6. *Denotamos por V a união disjunta de V_E e V_D (ou seja, mantemos duas cópias para elementos da intersecção de V_E e V_D). Seja W o conjunto dos arcos entre V_D e V_E correspondentes aos arcos fracos (ou seja, para cada arco fraco $f\_ell = (h\_ell, t\_ell)$, há em W um arco da cópia de h_ell na união disjunta V correspondente a V_D para a cópia de t_ell em V correspondente a V_E). Em geral na verdade h_ell e t_ell aparecem uma única vez em V, mas podem aparecer duas vezes quando $h\_ell = j\_ell$ ou $t\_ell = i_{\ell+1}$). Note que $O_W = over(W)$. Seja A'_E o conjunto dos arcos em V correspondentes aos arcos de A_E, e A'_D definido similarmente. (Ou seja, $A'_E \cap A'_D = \emptyset$.) Finalmente, seja $A = A'_E \cup A'_D \cup W$. De maneira análoga, definimos M como a "união" de M_E e M_D.*

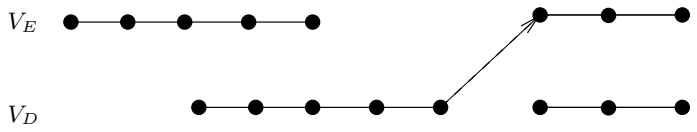

Figura 4.8: O arco tracejado indica um arco de W, enquanto que os vértices e arcos da linha superior indicam os elementos vindos de V_E e A'_E, e os vértices e arcos da linha inferior indicam os elementos vindos de V_D e A'_D.

Note que $over(A) = over(A_E) + over(A_D) + O_W$ e que M é uma cobertura por circuitos em G com $over(M) = over(M_E) + over(M_D)$.

Observe que, para completarmos a prova, resta mostrarmos que

$$over(M) - over(A) - L_M \leq 2|\textsc{scs}^*|.$$

Seja H o digrafo completo em V. Os lemas 4.8.3 e 4.8.5 serão usado na prova do lema 4.8.6.

Lema 4.8.3. Seja N uma cobertura por circuitos em H tal que $N \setminus A$ não contém nenhum arco em $V_D \times V_E$. Existe uma cobertura por circuitos N' em A satisfazendo as seguintes propriedades:

(1) $N' \setminus A$ não contém arcos em $V_D \times V_E$

(2) $over(N') \geq over(N)$

(3) para cada arco e em $A \setminus N'$, existe um arco $f \prec e$ em N' que compartilha cabeça ou cauda com e.

Demonstração. Se N satisfaz as três propriedades, então tome $N' = N$. Caso contrário, vamos mostrar como criar uma cobertura por circuitos C a partir de N tal que a propriedade (1) é mantida, mas C tem mais arcos em comum com A do que N e tem custo pelo menos o custo de N. Podemos, portanto, iterar o método até que (3) seja satisfeita.

Seja $e = (k, j)$ em $A \setminus N$ que viole a condição (3). Considere os arcos $f = (i, j)$ e $g = (k, l)$ de N. Então temos que $e \prec f$ e $e \prec g$ e observe que, como $e \in A$, os arcos f e g não pertencem a A. Pelo lema 4.6.1 (de Monges), podemos trocar f e g por e e (i, l) em N para produzir uma nova cobertura por circuitos C tal que $over(C) \geq over(N)$. A cobertura C tem mais arcos em comum com

A do que N. Resta mostrar que $(i, l) \notin V_D \times V_E$ pois $e \in C \cap A$ e portanto (1) não se aplica a e.

Suponha que $i \in V_D$, portanto $j \in V_D$ já que $f \in N$. Temos então que $k \in V_D$ já que $e \in A$ (lembre-se que A não contém arcos com a cauda em V_E e a cabeça em V_D), e como $g \in N$ temos que $l \in V_D$ (pois $g \notin V_D \times V_E$), concluindo que se $i \in V_D$ então $l \in V_D$. A figura 4.8 ajuda a visualizar este fato. □

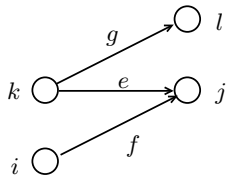

Figura 4.9: Representação dos arcos e, f e g.

O seguinte lema será usado na prova do lema 4.8.5.

Lema 4.8.4. Seja $e = (j, i)$ um arco de retorno. O vértice i pertence a S_E ou então é o primeiro vértice de um intervalo minimal em $\mathcal{F}$. Da mesma forma, o vértice j pertence a S_D ou então é o último vértice de um intervalo minimal em $\mathcal{F}$.

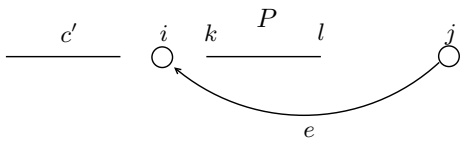

Figura 4.10: Esquema para o lema 4.8.4.

Demonstração. Seja $P = [k, l]$ o intervalo minimal mais a esquerda em I_f. Suponha que $i < k$ e i está em S_D (do contrário, ou $i \in S_E$, ou $i \in S_C$ e é o primeiro do intervalo P, assim não há mais nada a provar). Seja c' o intervalo minimal imediatamente à esquerda de i. Então, I_e inclui o arco fraco entre c' e c, que é pior, por definição, que todos os arcos no caminho entre c' e c, incluindo o arco $(i - 1, i)$ que ainda não foi escolhido. Isso contradiz a afirmação inicial da prova do lema 4.8.1. Portanto ou $i < k$ e $i \in S_E$ ou então $i = k$ e portanto é o primeiro vértice de um intervalo minimal em $\mathcal{F}$. A prova para j é análoga. □

Lema 4.8.5. Seja N uma cobertura por circuitos em H. Seja e um arco de $N \setminus A$ que não está em $V_D \times V_E$. Então e compartilha cabeça ou cauda com um arco que o precede em $\prec$ pertencente a uma das categorias abaixo:

(1) arcos de A, ou

(2) arcos de retorno de um intervalo minimal em $\mathcal{F}$ com o qual compartilha a cabeça, que está em V_D, ou

(3) arcos de retorno de um intervalo minimal em $\mathcal{F}$ com o qual compartilha a cauda, que está em V_E.

Demonstração. Como $e \notin A$, temos que e não foi escolhido pelo GREEDY. Isso significa que e é um arco de retorno ou vale (1), pois e compartilha cabeça ou cauda com um arco que o precede em $\prec$ que foi escolhido pelo GREEDY e que portanto está em A.

Se $e = (j, i)$ é um arco de retorno, pelo lema 4.8.4, então i é um vértice de S_E ou i é o primeiro vértice de um intervalo minimal em $\mathcal{F}$ e j é um vértice de S_D ou j é o último vértice de um intervalo minimal em $\mathcal{F}$. Como e não está em $V_D \times V_E$, então ou $i \notin S_E$ ou $j \notin S_D$. Portanto e compartilha a cabeça (ou seja, vale (2)) ou a cauda (ou seja, vale (3)) com o arco de retorno f de um intervalo minimal em $\mathcal{F}$ e $f \prec e$. □

Lema 4.8.6. Seja O_M a soma dos custos dos arcos de retorno. Temos que $over(M) \leq over(A) + 2O_M$.

Demonstração. Aplique o lema 4.8.3 a M, obtendo uma cobertura por circuitos M'. Como M é de custo máximo, por (2) do lema 4.8.3, M' também tem custo máximo. Portanto, basta agora mostrar que $over(M') \leq over(A) + 2O_M$. Na verdade, basta provar que $over\ (\ M' \setminus A\ ) \leq over\ (\ A \setminus M'\ ) + 2O_M$, já que os arcos que pertençam a $M' \cap A$ podem ser adicionados nos dois lados da desigualdade sem problemas. O que faremos agora é associar cada arco e de $M' \setminus A$ com um arco $f \prec e$, tal que f pertence a $A \setminus M'$ ou f é um arco de retorno. Note agora que todo arco de retorno pode ser cobrado no máximo 2 vezes (pelos itens (2) e (3) do lema 4.8.5) e que cada arco de $A \setminus M'$ pode ser cobrado apenas uma vez, já que ele próprio é precedido por um de seus arcos adjacentes de $M' \setminus A$ (pelo item (3) do lema 4.8.3). Portanto temos que $over(M' \setminus A) \leq over(A \setminus M') + 2O_M$ e portanto $over(M') \leq over(A) + 2O_M$. □

Definição 4.8.7. *Denotamos por C_M a cobertura por circuitos do conjunto S_M que contém um circuito para cada intervalo minimal em $\mathcal{F}$ composto dos arcos escolhidos pelo GREEDY mais o arco de retorno de cada intervalo minimal em $\mathcal{F}$.*

Lema 4.8.7. C_M é uma cobertura por circuitos de custo mínimo no digrafo $G_s[S_M]$.

Demonstração. Note que a execução do algoritmo MGREEDY no conjunto S_M nos fornece exatamente C_M. □

Teorema 4.8.1. O GREEDY é uma 4-aproximação polinomial para o SCS.

Demonstração. Seja W_M a soma dos custos (dados por $\overline{(\cdot,\cdot)}$) de C_M. Temos que $L_M = W_M + O_M$ e $W_M \leq |\text{SCS}^*|$ (já que, se e é um arco de retorno, então $over(e)$ é menor que a seqüência mais longa do intervalo minimal em $\mathcal{F}$). Considere as mais longas seqüências de cada circuito de C_M. Pelo lema 4.5.2, temos que a soma das sobreposições máximas numa superseqüência comum mínima para este conjunto é no máximo $2W_M$ e portanto, temos que $|\text{SCS}^*| \geq O_M - 2W_M$. Assim sendo,

$$\begin{aligned}
L_E + L_M + L_R - O_W &\leq 2|\text{SCS}^*| + over(M_E) - over(A_E) + over(M_D) \\
&\quad - over(A_D) - L_M - O_W \\
&\leq 2|\text{SCS}^*| + over(M) - over(E) - L_M \\
&\leq 2|\text{SCS}^*| + 2O_M - L_M \\
&= 2|\text{SCS}^*| + O_M - W_M \\
&\leq 3|\text{SCS}^*| + W_M \\
&\leq 4|\text{SCS}^*|.
\end{aligned}$$

□

Capítulo 5

Conclusão

Alguns novos resultados são de interesse no estudo do SCS. Sweedyk [19] criou a melhor aproximação conhecida para o SCS, mostrando que existe um algoritmo que é uma $2\frac{1}{2}$-aproximação para o problema. Recentemente, Kaplan et al. [12] mostraram que o GREEDY é melhor do que uma 4-aproximação, mostrando que ele é na verdade uma $3\frac{1}{2}$-aproximação para o SCS.

A conjectura que diz que o GREEDY é uma 2-aproximação é de 1994, quase 15 anos atrás. Ainda assim, a melhor razão provada é de $3\frac{1}{2}$, mostrando a dificuldade dessa conjectura e do real entendimento desse algoritmo tão simples. Caso a conjectura seja verdadeira, o GREEDY será a melhor aproximação conhecida para o SCS. Portanto um aprofundamento na análise do GREEDY e do MGREEDY (já que a sua análise está ligada com a do GREEDY) é essencial na busca da resolução desta conjectura.

Concluindo, apresentamos nesta monografia um texto que mostra os resultados mais importantes para o estudo do SCS. Ele contém o essencial para entender o problema, seus resultados de complexidade e inaproximabilidade e a análise de quatro algoritmos para esse problema. O texto também serve como exemplo da estrutura utilizada em provas para determinar razões de aproximação.

Por fim, acreditamos que este texto pode facilitar o entendimento do SCS e proporcionar um início mais rápido no estudo do SCS se comparado a análise dos artigos originais que os expõem.

Capítulo 6

Parte Subjetiva

Esta parte contém informações sobre desafios e dificuldades encontradas neste trabalho, sobre disciplinas do Bacharelado em Ciência da Computação do IME-USP relevantes ao estudo aqui apresentado destes algoritmos de aproximação e, por fim, sobre possíveis caminhos que podem ser tomados na continuação deste estudo.

6.1 Desafios e frustrações

O primeiro desafio que enfrentei ao começar este projeto foi o contato inicial com algoritmos de aproximação. Fui convidado pela professora Cristina para fazer uma iniciação cientifica após o curso Desafios de Programação. Acabei preferindo este tema ao invés dos outros, mas na época não tinha cursado ainda a disciplina Algoritmos de Aproximação. Estudei então o livro de Carvalho et al. [6] para suprir as necessidades que eu tinha para entender o problema de interesse.

Um desafio constante durante o projeto foi o formalismo matemático. Apesar de sempre ter gostado das matérias mais teóricas do curso, quando comecei o projeto ainda não tinha capacidade de escrever um texto matemático formal e de fácil entendimento. A professora Cristina me ajudou muito neste ponto, sempre sugerindo muitas correções ao texto. Acredito ter evoluído bastante neste aspecto.

Todos os algoritmos foram estudados em várias fontes diferentes [10, 18, 23, 4], principalmente no começo onde era importante conhecer melhor cada um deles. Infelizmente cada fonte utilizava notações diferentes, enunciava teoremas com variações e apresentava provas com modificações. Precisávamos de uma notação concisa e fácil de entender, e em conversa com a minha orientadora acredito que chegamos a este resultado.

Nesta iniciação científica, foi a primeira vez que estudei uma redução polinomial para mostrar que um problema é computacionalmente difícil apesar do assunto já ter sido apresentado em uma aula de análise de algoritmos (eu não tinha cursado Algoritmos e Complexidade de Computação na época). Por causa disso, demorei um bom tempo para entender completamente a prova e conseguir rescrevê-la.

O mesmo ocorreu com o outro resultado de complexidade que mostra que

o SCS é MAX SNP-difícil [4]. Como nunca tinha estudado L-reduções antes, tive muitas dificuldades em entender todo o processo. O texto original da prova é vago em muitos aspectos, principalmente pela semelhança com a redução de Gallant et al. [8] mencionada acima. Isso dificultou muito o processo de entender e rescrever a prova, mas isso foi possível graças a ajuda da professora Cristina que me ajudou a provar o resultado de forma independente. Apesar de nos basearmos na prova do artigo original, uma parte considerável do mesmo foi ignorada para escrever esta versão.

O maior desafio deste projeto foi estudar a análise do GREEDY. Inicialmente acreditávamos que o entendimento desta análise não levaria muito tempo, mas na verdade a análise é muito mais complicada do que podíamos imaginar. A prova utiliza várias definições e lemas por si só complicados e que também dependem do entendimento aprofundado do algoritmo MGREEDY, já que a análise de certa forma compara os dois algoritmos. O grande problema desta análise é que ela é a única não mencionada em nenhuma das outras fontes que pesquisamos e portanto não pudemos ver a explicação por um segundo autor. Assim mesmo, conseguimos concluir o estudo deste intrincado algoritmo. Por fim, reestruturamos toda a prova agrupando resultados relacionados e modificando muitas notações para simplificar a prova deste teorema de grande importância para o resultado de interesse. Com isso acreditamos que agora esta prova é muito mais fácil de compreender do que a original.

Na etapa final do projeto foi gasto um tempo considerável para apresentar o projeto ao público. Primeiramente participei do Simpósio de Iniciação Científica do IME, realizando uma apresentação oral. Foram necessárias várias alterações no meu texto por causa da limitação de espaço e formato imposto pelos organizadores. Achei muito inconveniente ter que remover partes de itens que eu estudei para apresentar em um simpósio desta natureza. Afinal, por ser um Simpósio de Iniciação Cientifica, não acho que os organizadores deveriam impedir os alunos de exporem tudo o que aprenderam. Posteriormente, participei das Jornadas de Iniciação Científica 2008 do IMPA, onde novamente fui escolhido para realizar uma apresentação oral. Todos estes eventos consumiram bastante tempo no último mês do projeto, dado que todos precisavam de um certo preparo de material e tive também de viajar ao Rio de Janeiro para apresentar o trabalho de IC no IMPA.

A minha maior frustração nesta iniciação científica foi a grande quantidade de tempo investido para estudar a análise do GREEDY. Haviam vários artigos que eu gostaria de ter estudado, incluindo uma melhoria na aproximação do GREEDY e resultados de outras áreas. A análise gastou muito tempo e foi particularmente desmotivante, o que causou um atraso grande no projeto. Vários planos, que incluíam até a implementação dos algoritmos e a realização de estudos empíricos, foram deixados de lado pela falta de tempo. Mas acredito que consegui superar diversos desafios que foram importantes para a minha formação, apesar de algumas frustrações que surgiram.

6.2 Disciplinas relevantes

Todas as disciplinas do curso foram muito importantes para a minha formação, mas algumas disciplinas tiveram um papel muito importante neste estudo:

- **MAT0138 - Álgebra I para Computação:** Este foi o primeiro con-

tato com teoremas e provas. Aprendi neste curso ferramentas que utilizo como freqüência, como a indução por exemplo.

- **MAC0315 - Programação Linear:** Esta disciplina foi meu primeiro contato com problemas de otimização. Ela é muito importante para a teoria de algoritmos de aproximação e foi muito utilizada em MAC0325, MAC0450 e em Combinatória Poliédrica e o Método dos Planos de Cortes (uma matéria da pós-graduação que cursei como aluno especial).

- **MAC0328 - Algoritmos em Grafos:** Esta foi a primeira disciplina sobre teoria de elementos combinatórios e foi o que despertou meu interesse para esta área. A matéria abordou conceitos teóricos interessantes indo além dos algoritmos de busca.

- **MAC0338 - Análise de Algoritmos:** O primeiro contato com a dificuldade de resolver alguns problema computacionais (os problemas NP-difíceis) ocorre no final dessa disciplina. Nesta disciplina ocorreu também o meu primeiro contato com algoritmos de aproximação após a explicação sobre a dificuldade de resolver certos problemas.

- **MAC0325 - Otimização Combinatória:** Após ter estudado MAC0328, tive interesse em continuar estudando estruturas combinatórias. Esta matéria utilizou muitos conceitos de Programação Linear para descrever problemas de otimização de conjuntos discretos. Ela, juntamente com o estudo da complexidade de computação, formam a motivação para Algoritmos de Aproximação.

- **MAC0310 - Matemática Concreta:** Esta foi uma das matérias mais teóricas que cursei, apresentando o começo da teoria de grafos. Tive a oportunidade de explorar melhor esta teoria e aprofundar conceitos vistos em MAC0328.

- **MAC0414 - Linguagens Formais e Autômatos:** Aprendi nessa disciplina os conceitos básicos para lidar com linguagens sobre um alfabeto. Isso foi muito importante, já que este é um trabalho sobre seqüências. Aprendi também os conceitos básicos para cursar MAC0430 que foi fundamental para a parte de análise de complexidade do meu trabalho.

- **MAC0430 - Algoritmos e Complexidade de Computação:** O estudo da computabilidade e da complexidade computacional me levou a entender melhor a dificuldade em resolver de forma exata o problema que estudei. Aprendi com maior formalismo do que em MAC0338 as classes P e NP e o conceito de problema NP-completo.

- **MAC0450 - Algoritmos de Aproximação:** Por fim, esta matéria apresentou várias formas de criar algoritmos de aproximação para um problema de otimização combinatória difícil. Aprendi também sobre a inaproximabilidade de alguns problemas, resultado que estudei para o problema da superseqüência comum mínima.

Como este trabalho tem uma natureza teórica, outras disciplinas que não foram utilizadas diretamente, mas que merecem ser citadas aqui são: **MAC0110**

- Introdução à Computação, **MAC0122 - Princípios de Desenvolvimento de Algoritmos**, **MAC0211 - Laboratório de Programação I**, **MAC0316 - Conceitos Fundamentais de Linguagens de Programação**, **MAC0323 - Estruturas de Dados**, **MAC0327 - Desafios de Programação**. Todas elas contribuíram muito na minha formação, tanto teórica quanto prática. Devo a estas matérias e aos professores que a ministraram meu conhecimento em programação, linguagens e estruturas de dados.

6.3 Aplicação dos conceitos estudados

Neste trabalho utilizei diversos conceitos estudados nas disciplinas do curso. A teoria de grafos apresentada em MAC0328/MAC0310 e os problemas de otimização combinatória estudados em MAC0325 foram utilizados pela forte relação entre o problema e um grafo orientado completo com custos nos arcos e também pela importância do estudo da cobertura por circuitos de custo mínimo e sua redução ao problema do emparelhamento perfeito de custo mínimo.

Os conceitos aprendidos relacionados a complexidade computacional em MAC0430 e os relacionados a razões de aproximação e inaproximabilidade de MAC0450 foram importantes para entender a demonstração os principais teoremas deste estudo e os conceitos sobre seqüências apresentado em MAC0414 foram importantes para ganhar uma desenvoltura com o problema.

6.4 Possíveis passos para aprimorar os conhecimentos relevantes

O passo natural a seguir neste estudo é analisar resultados mais novos para o problema da superseqüência comum mínima. Existem resultados que mostram melhores razões de aproximação para os algoritmos aqui apresentados e também resultados sobre algoritmos novos que são aproximações melhores para o problema.

Uma opção interessante seria estudar também alguns resultados empíricos para o problema ou até mesmo gerar resultados neste sentido. Alguns artigos mostram a qualidade dos algoritmos quando a quantidade de seqüências tende ao infinito, uma abordagem diferente da estudada aqui mas que apresenta resultados muito interessantes.

Referências Bibliográficas

[1] C. Armen and C. Stein. Improved length bounds for the shortest superstring problem. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 494–505. Springer, 1995.

[2] C. Armen and C. Stein. Short superstrings and the structure of overlapping strings. *Journal of Computational Biology*, (2):307–333, 1995.

[3] C. Armen and C. Stein. A $2\frac{2}{3}$ superstring approximation algorithm. *Discrete Applied Mathematics*, 88(1-3):29–57, 1998.

[4] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the Association for Computing Machinery*, 41(4):630–647, 1994.

[5] D. Bongartz. On the approximation ratio of the Group-Merge algorithm for the shortest common superstring problem. *Computing and Informatics*, 20(4):325–357, 2001.

[6] M.H. Carvalho, M.R. Cerioli, R. Dahab, P. Feofiloff, C.G. Fernandes, C.E. Ferreira, F.K. Miyazawa, J.C. de Pina, J. Soares, and Y. Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemáticas do IMPA, 2001.

[7] A. Czumaj, L. Gąsieniec, M. Piotrów, and W. Rytter. Sequential and parallel approximation of shortest superstrings. *Journal of Algorithms*, 23(1):74–100, 1997.

[8] J. Gallant, D. Maier, and J.A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[10] D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, Cambridge, 1997. Computer Science and Computational Biology.

[11] D.S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, 1997.

[12] H. Kaplan and N. Shafrir. The greedy algorithm for shortest superstrings. *Information Processing Letters*, 93(1):13–17, 2005.

[13] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.M. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum, 1972.

[14] R. Kosaraju, J. Park, and C. Stein. Long tours and short superstrings. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 125–134, 1994.

[15] M. Li. Towards a DNA sequencing theory. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 125–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990.

[16] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[17] C.H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18:1–11, 1993.

[18] J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, 1997.

[19] Z. Sweedyk. A $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29(3):954–986 (electronic), 2000.

[20] S.H. Teng and F.F. Yao. Approximating shortest superstrings. *SIAM Journal on Computing*, 26(2):410–417, 1997.

[21] J.S. Turner. Approximation algorithms for the shortest common superstring problem. *Information and Computation*, 83(1):1–20, 1989.

[22] V. Vassilevska. Explicit inapproximability bounds for the shortest superstring problem. In *Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 793–800. Springer, Berlin, 2005.

[23] V.V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[24] M. Weinard and G. Schnitger. On the greedy superstring conjecture. *SIAM Journal on Discrete Mathematics*, 20(2):502–522, 2006.