SMNE UFPR
Simpósio de Métodos Numéricos em Engenharia
25 a 27 de outubro, 2017

Inserção de Bound externo ao método de resolução em árvore aplicado ao TSP

Alexandre Checoli Choueiri [a,1]
Cassius Tadeu Scarpin[a,b,2]
Gustavo Valentim Loch[a,b,3]
Nathália Cristina Ortiz da Silva[a,4]
Cleder Marcos Schenekemberg[a,5]
Deidson Vitorio Kurpel[a,6]
[a] Programa de Pós-Graduação em Métodos Numéricos em Engenharia
[b] Departamento de Administração Geral e Aplicada
Universidade Federal do Paraná
Curitiba, Brasil
[1]alexandrechecoli@gmail.com, [2]cassiusts@gmail.com, [3]gustavo.gvalentim@gmail.com, [4]ncosilva2@gmail.com, [5]cledercms@hotmail.com, [6]kurpeld@gmail.com

Resumo - O Problema do Caixeiro Viajante (PCV) trata da ordenação/sequenciamento de um dado conjunto de pontos de tal forma que um caminho passando por todos os pontos, uma única vez em cada ponto, e voltando ao ponto inicial seja mínimo. Embora exista uma vasta literatura a respeito do PCV, uma pequena parte desta trata da relação do PCV com o Problema de designação PD junto a estrutura de resolução exata de adição de restrições de sub-rotas por meio de arvore de decisão. O objetivo deste trabalho é verificar a eficiência de se realizar uma inserção de um Bound para o problema, posteriormente ao seu início, dessa forma, eliminado ramos da arvore a serem exploradas pelo algoritmo. Os resultados mostraram que de fato existe uma melhora ao se partir de um Bound externo à arvore, em relação aqueles produzidos pela própria arvore.

Palavras-chave— Problema do caixeiro viajante; Busca em arvores; Problema de designação; Metodo exato.

I. INTRODUÇÃO

O problema do caixeiro viajante (PCV) possui, talvez, a mais simples enunciação dentre os problemas de análise combinatória. Porém, traz em seu bojo um nível de dificuldade e complexidade intrínsecas tais que, mesmo 50 anos após sua primeira aparição formal, no artigo "*Solution to a large-scale traveling salesman problem"* [5] pesquisadores ainda se interessam pela busca de melhores soluções para o problema. Sua descrição pode ser feita da seguinte forma: encontrar a menor rota (*tour t*) para um vendedor, começando em uma dada cidade e visitando todas, de um grupo especifico de cidades, para então retornar ao ponto inicial. Genericamente, dados *n* pontos no espaço e $C = (c_{ij})$ *n x n*, a matriz de distancias, em que que c_{ij} representa a distância do ponto I ao ponto J, o problema consiste em ordenar os pontos de forma cíclica de tal maneira que $\sum c_{ij}$ seja mínimo.

Diversos algoritmos foram propostos para a resolução do PCV (Problema do caixeiro viajante), tanto exatos quanto

heurísticos [8]. Um dos métodos exatos proposto por Eastman, parte do resultado de que toda solução do PCV é também solução factível para o problema de designação equivalente [2]. O algoritmo realiza uma busca em árvore tomando como solução inicial a solução do problema de designação (PD) associado à matriz D e segue, sucessivamente, inserindo restrições de eliminação de sub-rotas até que um caminho hamiltoniano seja encontrado. O custo deste caminho se torna então um Bound para resolução, ou seja, nós que possuírem custos superiores ao Bound atual não são explorados. A busca continua, armazenando esses caminhos, até que não existam mais nós a serem explorados. A solução do PCV é então o caminho Hamiltoniano com menor custo da arvore completa.

Sabendo que a busca continua até o encontro do primeiro caminho hamiltoniano, visitando todos os nós de acordo com o tipo de busca empregado, pode-se inserir um Bound inicial antes mesmo do início da exploração da arvore. O objetivo deste trabalho é então realizar testes e verificar a praticabilidade da inserção deste Bound na arvore de resolução, avaliando a distância que este pode estar da solução ótima, e ainda trazer reduções no número de nós explorados.

O trabalho está organizado da seguinte forma: Na próxima seção é apresentada a relação existente entre o problema do caixeiro viajante e o problema de designação, bem como a estruturação da arvore de decisão utilizada na resolução do PCV. Na seção 3 o pseudocódigo para a resolução do problema de designação é apresentado. A seção 4 mostra os tipos de busca em arvore que foram utilizados.

Na seção 5 a ideia de inserção do Bound externo é apresentada, seguida dos testes computacionais, a conclusão está presente na seção 7.

II. Relação do PCV com o PD e estrutura de resolução em arvore

O problema da designação se resume à tarefa de encontrar a designação ótima de *n* trabalhadores à *n* tarefas, visto que a cada tarefa existe um peso indicando o quão bem o trabalhador realiza a mesma. Uma designação ótima é aquela em que faz a soma de pesos à máxima possível (no caso de pesos como preferências do tipo "maior é melhor"), ou ainda, uma que faça a soma a menor possível (no caso que os pesos sejam o custo de designação) e para esta, o modelo matemático de Programação Linear Binária é o que segue: [9].

$$Min\ Z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}X_{ij} \qquad (1)$$

Sujeito a:

$$\sum_{i=1}^{n} X_{ij} = \sum_{i=1}^{n} X_{ji} = 1 \qquad \forall\, j = 1,2 \ldots n\ e\ \ X_{ij} = 0\ ou\ 1 \ (2)$$

Em que X_{ij} é igual a 1, se o trabalhador i é designado à tarefa j, e igual a 0, caso contrário. Estudos de base teórica sobre o PCV demonstram o resultado de que todo tour *t* do PCV é uma solução factível para o PD e o custo deste é justamente o custo total de *t* [2]. Porém, nem toda solução do PD é um tour do PCV, podendo ser consideradas coleções de sub-rotas do PCV. As restrições do modelo de designação simplesmente garantem que a solução seja cíclica, dessa forma, se uma nova restrição for adicionada garantindo que o ciclo formado seja único (Hamiltoniano), tem-se um modelo matemático para o PCV.

Ainda, se as restrições $C_{ii} = \infty$ fossem adicionadas ao modelo, poderiam ser interpretadas como restrições que eliminam a possibilidade de sub-rotas com cardinalidade 1 [3].

O processo de sucessivamente realizar inserções de $C_{ij} = \infty$ na matriz de custos (ou pesos) C, de modo a eliminar todos os subciclos gerado na solução do PD original, resulta em um caminho Hamiltoniano, o qual é solução para o PCV. Para organizar a estruturação da resolução em forma de árvore de decisão, o PD inicial (P0) é considerado como o nó raiz. Por meio de inserções de $C_{ij} = \infty$, de acordo com algum critério, novos subproblemas (P1, P2, etc.) são gerados a partir de P0 (ramificações). Este processo é repetido para todos os subproblemas P_K (onde K = 1, .., k) até que o melhor caminho Hamiltoniano seja encontrado.

As regras de inserção do valor $C_{ij} = \infty$ podem variar. A regra utilizada neste estudo segue a seguinte lógica:

1. Encontrar a menor sub-rota de um Problema P_k; Por exemplo (2,4,3)
2. Escolher um ponto da sub-rota; Por exemplo (2);
3. Formar todas combinações de arcos dos pontos escolhidos com todos os outros da sub-rota; exemplo (2,3) (2,4).
4. Para todos os arcos (i, j), encontrados no passo 3, criar um novo subproblema, alterando a matriz do problema de origem do recém criado, com $C_{ij} = \infty$.

Nota-se que ao executar o passo 2, a escolha do ponto que será combinado com todos os outros gera o subproblema "filho", dessa forma, existem tantas possibilidades de ramificação quantos forem os nós daquela sub-rota.

A Fig.1 exemplifica uma estrutura de solução. Em cada nó estão as informações das sub-rotas geradas e o valor da função objetivo. Os nós acinzentados apresentam caminhos hamiltonianos, ou seja, são Bounds factíveis para o PCV. Como a matriz de custos utilizada para a resolução do PD em qualquer nó filho (ex P2), é herdada do nó pai (ex P0) com as inserções de $C_{ij} = \infty$ para evitar alguma sub-rota, necessariamente $Z(P2) \geq Z(P0)$. Assim, obtém-se um critério de parada de ramificação da árvore. Uma vez que um Bound do PCV é encontrado e armazenado, qualquer outro nó com $Z \geq Bound$ deve ser eliminado e não deve ser ramificado, finalizando assim a exploração daquele ramo.

III. Implementação do método hungaro

O método Húngaro é talvez o algoritmo mais conhecido para a resolução do PD. Primeiramente foi desenvolvido por [7] e posteriormente aprimorado por [9]. Existem diversas implementações para o método [3] e [10] e, dessa forma, pode acontecer que se um determinado problema é passível de múltiplas soluções ótimas, as implementações podem divergir na solução final. Isto afeta a quantidade de ramificações da árvore, uma vez que a mesma é ramificada

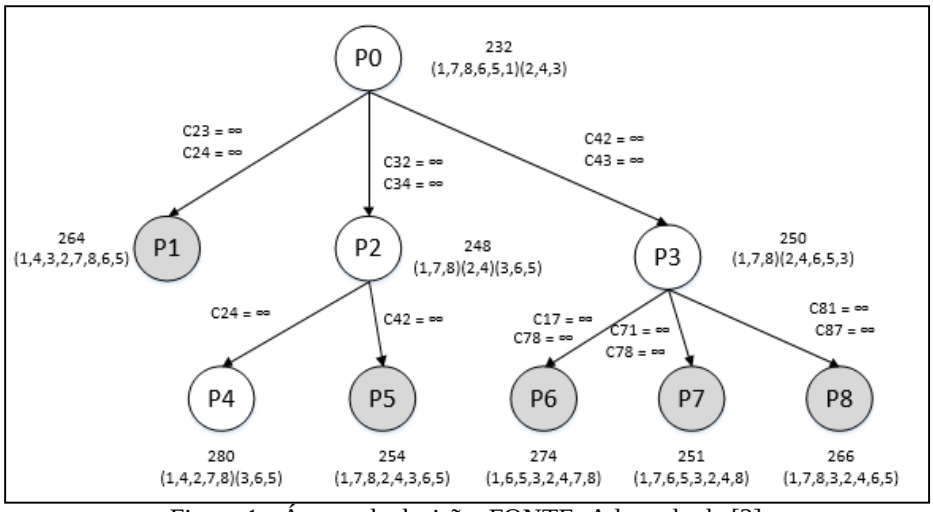

Figura 1 - Árvore de decisão. FONTE: Adaptado de [3]

de acordo com as sub-rotas geradas na solução. O algoritmo usado para resolver os subproblemas de designação nesse trabalho foi o proposto por Christofides;

A. Implementação de Christofides para o método Húngaro

A implementação de Christofides [3] segue a ideia do algoritmo aprimorado de Munkres [9].

O algoritmo de Christofides segue os seguintes passos:

I. Para cada linha da matriz, encontre o menor elemento e subtraia-o de todos os elementos da referida linha. Armazene todos os mínimos em um vetor π_i.

II. Para cada coluna da matriz encontre o menor elemento e o subtraia de todos os elementos da referida coluna. Armazene todos os mínimos em um vetor π_k

III. Marque o máximo número de zeros na matriz, de tal forma que nenhum zero seja marcado mais de uma vez em alguma linha ou coluna.

IV. Encontre uma linha *s* sem zero marcado e inicialize um vetor $p_s = 0$. Se tal linha não existir, a solução atual é ótima.

V. Começando em $p_s = 0$, com todas as outras linhas e colunas não marcadas:

VI. Marque toda coluna não marcada k que contenha um 0 não marcado em uma linha marcada, com $p_k = i$

VII. Marque toda linha *i* não marcada, que tenha uma 0 marcado em uma coluna marcada, com $p_i = k$

VIII. Repita as marcações VI e VII até que uma coluna *t* sem marcações seja marcada (A), ou até que nenhuma marcação seja possível (B).

IX. (A) Um caminho melhorado $t, p_{t'}, p_{t''}, \dots, s$ foi encontrado, em que $t' = p_t$, $t'' = p_{t'}, etc.$ Inverter os zeros marcados e não marcados nas posições $(t, p_t), (p_{t'}, p_{t''}), \dots, (., s)$, dessa forma um elemento na solução atual. Os valores de p_i são apagados e o processo se repete até que o número de zeros = *n*.

X. (B) Defina $I^+ \, e \, K^+$ como o conjunto de todas as linhas e colunas marcadas, e $I^- \, e \, K^-$ as não marcadas e calcule: $\Delta = \min_{\substack{i \in I^+ \\ k \in K^-}} [C'_{ik}]$ e atualize $\pi_i = \pi_i + \Delta$ para $i \in I^+$ e $\pi_k = \pi_k - \Delta$ para $k \in K^+$. Repita VI e VII.

IV. ALGORITMOS DE BUSCA EM ÁRVORE

A. Busca em largura

O algoritmo de busca em Largura é o mais simples para pesquisa em grafos e serve de base para outros algoritmos de busca, tais como: a busca pelo caminho mínimo de Djikstra e o algoritmo de árvore espalhada mínima de Prim. [4].

Dado um grafo $G = (V, A)$ e uma origem para o início da busca (*s*), o algoritmo explora todos os vértices acessíveis a partir da origem e, assim sucessivamente, a partir de cada vértice encontrado. Desta forma, explora todos os novos vértices acessíveis a cada vértice encontrado e a busca elimina todas as possibilidades de ramificação, nível a nível da árvore. Ou seja, o algoritmo explora todos os vértices à distância *k* de *s* antes de explorar os vértices com distancia *k +1*. Um exemplo de busca em largura no problema da Fig.1 1, exploraria os nós na seguinte ordem: (P0, P1, P2, P3, P4, P5, P6, P7, P8).

B. Busca em Profundidade

A busca em profundidade, como o nome indica, expande seus nós para níveis mais baixos sempre que possível, chegando em um objetivo. O próximo nó a ser explorado é o nó mais profundo que ainda possa ser ramificado. Uma busca em profundidade aplicada ao problema da Fig. 1 exploraria os nós com a seguinte ordem: (P0, P1, P2, P4, P5, P3, P6, P7, P8).

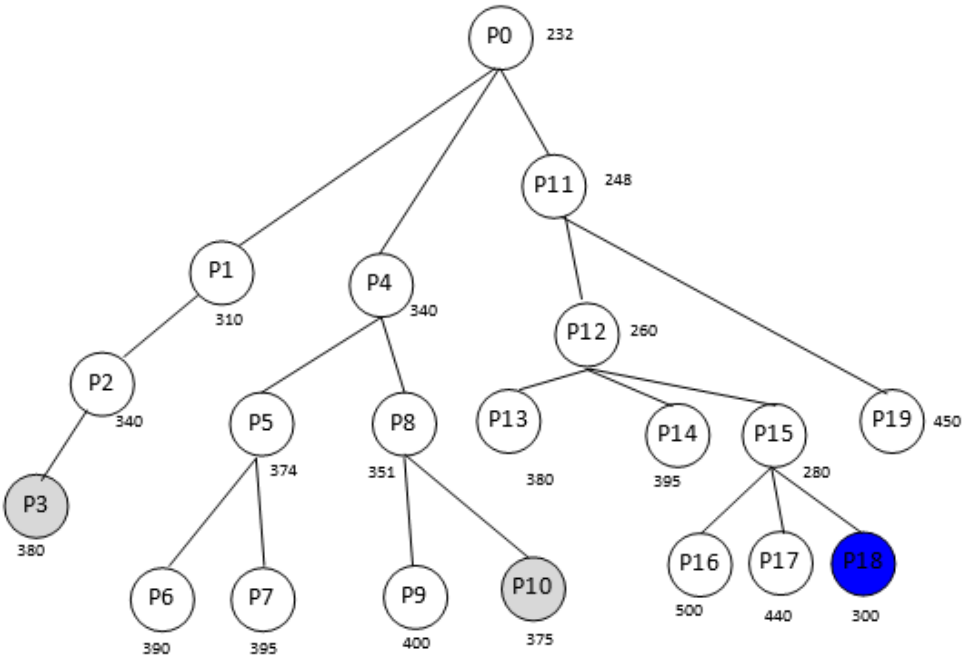

Figura 2 - Nós explorados com novo Bound. FONTE: O autor

V. INSERÇÃO DE BOUND EXTERNO

A base para uma eficiente exploração dos nós da arvore de decisão se pauta basicamente em 3 aspectos:

1. A velocidade de resolução de cada problema de designação no nó.
2. O tipo de busca utilizado para explorar a arvore.
3. Os Bounds fornecidos pela própria arvore.

Para problemas com muitos pontos, pode ser demorado até a própria arvore fornecer o primeiro Bound, ou seja, um caminho Hamiltoniano a partir das solução de um problema de designação. Dessa forma, um Bound externo pode ser inserido à arvore posteriormente ao início das buscas, de forma que agora, existe a possibilidade de se obter um menor número de nós explorados ao final do algoritmo.

A Fig.2 ilustra uma busca em arvore convencional por profundidade, o primeiro Bound é encontrado no nó 3, com valor Z = 380, o segundo ao nó 10 com Z = 375 , e finalmente o terceiro , que é o resultado ótimo da arvore no nó 18, com Z = 300, dessa forma, o método convencional precisou explorar 19 nós até encontrar o valor ótimo.

Se um Bound externo fosse fornecido com o valor de Z = 340, o caminho da exploração seria (com a mesma numeração de nós da Fig.2 (P0, P1, P2, P4, P11, P12, P13, P14, P15, P16, P17, P18, P19), dessa forma explorando um total de 13 nós, 5 a menos do que anteriormente.

VI. RESULTADOS COMPUTACIONAIS

Todos os algoritmos foram programados em linguagem VB.net e os testes realizados em computadores com processador I7 e 8GB de memória RAM. Todos os testes foram realizados até o ótimo ser atingido.

Para se verificar o padrão da exploração de nós em relação a solução inicial fornecida como Bound, os seguinte passos foram computados programados para matrizes com 50 e 70 pontos respectivamente, para coletar as informações supracitadas:

1 Resolver o problema sem inserção de Bound inicial,
2 Enquanto incremento < 100
3 Incremento = Incremento + 1
4 Bound Inicial = solução_inicial + incremento
5 Resolver o problema com o Bound inicial

Dessa forma, uma vez resolvido o problema original, o mesmo é resolvido novamente 100 vezes, a cada vez com um incremento unitário na solução ótima, por exemplo, o custo otimizado Z_0 do problema é de 100, o mesmo é resolvido novamente com um Bound inicial de 101, em seguida 102, e assim por diante, até a que o Bound inicial tenha sido incrementado em 100 unidades. A cada resolução com Bound

diferente, o número de nós explorados pela árvore, até que a solução ótima seja encontrada, foi armazenado.

Nos gráficos das Fig.3, Fig.4 e Fig.5, a distância que o Bound está para a função ótima é exibida no eixo x (em unidades), e o número de nós explorados até o que o ótimo seja encontrado no eixo y.

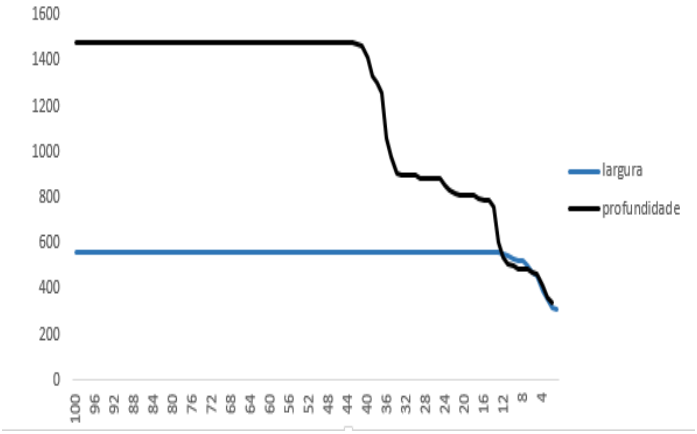

Figura 3 – Distancia do Bound x Numero de nós explorados (50 pts.)

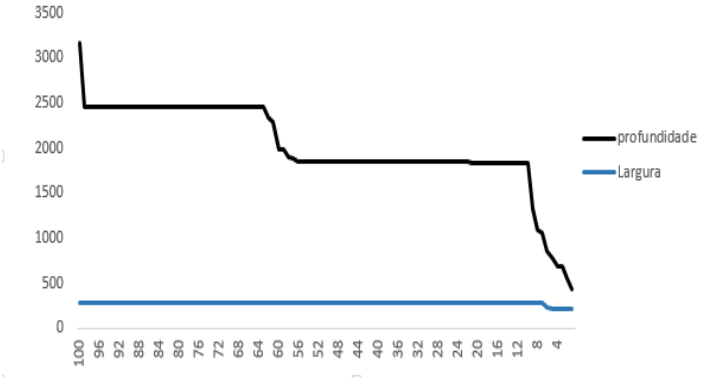

Figura 4 - Distancia do Bound x Numero de nós explorados (70 pts.)

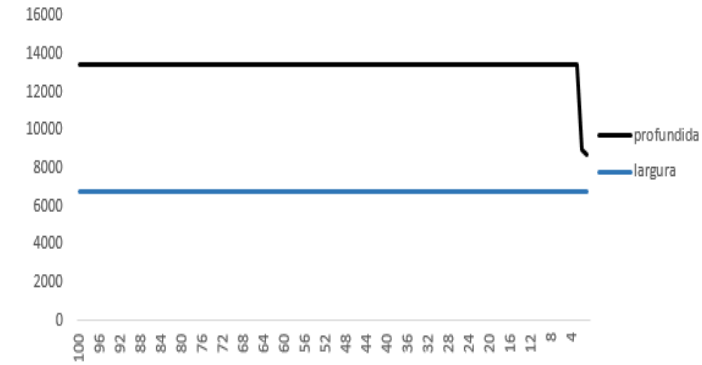

Figura 5 - Distancia do Bound x Numero de nós explorados (100 pts.)

A Tabela 1 apresenta os valores ótimos para os conjuntos testados.

TABELA I: SOLUÇÕES ÓTIMAS

Pontos	**Z ótimo**
50	3270
70	1479
100	3865

O padrão de decrescimento no número de nós explorados é muito acentuado, a medida em que o Bound inicial se aproxima da solução ótima. Na Fig. 2, referente ao problema com 50 nós, com uma solução inicial 60 unidades acima da solução ótima (3270 + 60 = 3330), o número de nós explorados fica o mesmo comparado ao início da buscasem um Bound inicial (1473 nós) , já com um incremento de somente 20 unidade da solução ótima (2370 + 20 = 2390), o número de nós explorados é reduzido para 800, quase metade do número explorado sem a inserção do Bound incial.

VII. CONCLUSÕES

O presente trabalho teve por objetivo avaliar a praticabilidade de se inserir um Bound externo à arvore de decisões que resolve o problema do caixeiro viajante, via sucessivas resoluções de problemas de designação.

Os resultados se mostraram satisfatórios, de modo que a inserção de um limitante externo de fato reduz o número de nós a serem explorados, porém, a eficiência desta nova abordagem depende da qualidade da solução fornecida inicialmente, se ela não estiver próxima o suficiente da solução ótima, o número de nós explorados permanecerá constante.

Em trabalhos futuros, uma abordagem de algoritmo multi-threading pode ser testada, na medida em que a exploração da arvore acontece em uma thread, uma segunda é inicializada com algoritmos de melhoria de rota em um caminho Hamiltoniano encontrado, ao final da atualização, se o Bound do caminho melhorado for inferior ao Bound atual, o mesmo é substituído, e a Thread é novamente inicializada com um novo caminho Hamiltoniano.

REFERÊNCIAS

[1] Bellmore, M,. (1966) The traveling salesman problem: a survey. Operations Research, v. 16, n. 3, p. 538–558.

[2] Bellmore, M. e Malone, J. C. (1971) Pathology of Traveling-Salesman Subtour-Elimination Algorithms. Operations Research

[3] Christofides, N. (1974) Graph Theory - An Algorithmic Approach K. Elissa, “Title of paper if known,” unpublished.

[4] Cormen , T. H. Introduction to algorithms. [s.l: s.n.]. v. 1

[5] Dantzig, G. B. e Fulkerson, D. R e Johnson. (1954) S. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operations Research Society of America, v. 2, n. 4, p. 393–410

[6] Ford, L. R, e D. R. F. Flows in networks. [s.l: s.n.]. v. 275

[7] Kuhn, H. W. (1950) The hungarian method for the Assigment problem

[8] Laporte, G. (1992) The traveling salesman problem: an overview 9of exact and approximate algorithms. European Journal of Operational Research, v. 59, p. 231–47–231–47

[9] Munkres (1957) , J. Algorithms for the assigment and transportation problems. Journal of the society for indusrtial and applied Mathematics, v. 5, n. 1, p. 32–38

[10] Papadimitriou, C. e Steiglitz. (1982) K. Combinatorial Optimization. [s.l.] Dover Publications, INC.

[11] Eastman, W. L. (1958) "Linear Programming with Pattern Constraints," Ph.D. Dissertation, Harvard, 1958