Um Algoritmo Branch-and-Bound para o Problema da Mínima Latência

Gabriel A. C. Mattar
João F. M. Sarubbi
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte-MG, Brasil
gabrielmattar41@gmail.com
joaosarubbi@gmail.com

Gustavo C. Menezes
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte-MG, Brasil
gustavo@contagem.cefetmg.br

RESUMO

O Problema da Mínima Latência (PML) é uma variação do clássico Problema do Caixeiro Viajante. Dado um nó de partida, o problema consiste em encontrar uma rota que passe por todos os demais nós, minimizando o somatório dos custos de espera de cada um deles. Neste trabalho, é proposto um algoritmo Branch-and-Bound (B&B) para resolver esse problema. Sua eficiência é avaliada utilizando três limites inferiores, calculados sem o auxílio de um resolvedor comercial. Além das estratégias exatas também é proposta uma heurística baseada no algoritmo Branch-and-Bound. O algoritmo exato foi testado para dois conjuntos de instâncias: (i) instâncias da TSPLIB; (ii) instâncias aleatórias que obedecem a desigualdade triangular. O algoritmo foi capaz de provar a otimalidade para instâncias de até 48 nós, com tempo computacional menor que o do resolvedor CPLEX *stand-alone*.

PALAVRAS CHAVE. Branch-and-Bound. Problema de Mínima Latência. Otimização Combinatória.

ABSTRACT

The Minimum Latency Problem, also known by Traveling Repairman Problem, the Deliveryman Problem and the Traveling Salesman Problem with Cumulative Costs, is a variant of the Traveling Salesman Problem in which a start node of a tour is given and the goal is to minimize the sum of the arrival times at all the other nodes. In this work, we propose three new lower bounds for the MLP and one heuristic, each of them with their respective benefits. We implemented then, a Branch-and-Bound (B&B) algorithm to compare the three lower bounds with a mixed-integer linear program formulation. Our algorithm was tested for two sets of instances: (i) TSPLIB instances; (ii) random generated instances that obey triangle inequality. With our method, we achieve optimal solutions for TSPLIB instances up to 48 nodes faster than CPLEX.

KEYWORDS. Branch-and-Bound. Minimum Latency Problem. Combinatorial Optimization.

1. Introdução

O Problema da Mínima Latência (PML) é um problema clássico de otimização combinatória no qual um entregador deve visitar todos os nós de um dado grafo, de maneira a minimizar o somatório de custos de espera de todos os consumidores. O PML foi introduzido por Conway et al. [1967] como um problema de sequenciamento, entretanto os próprios autores já o classificaram como uma variante do Problema do Caixeiro Viajante (PCV) [Dantzig et al., 1954]. De acordo com Eijl [1995], o PML pode ser interpretado como um problema de sequenciamento com uma única máquina, com tempo de processamento dependendo da sequência e cujo objetivo é minimizar a soma dos tempos das tarefas. De acordo com Goemans e Kleinberg [1998], apesar da óbvia similaridade com o Problema do Caixeiro Viajante, o PML aparenta ser bem menos comportado de um ponto de vista computacional. O PML possui maior complexidade que o PCV porque incorpora os objetivos conflitantes dos consumidores, ao invés de trabalhar com um único tempo de viagem. Na literatura o PML também é conhecido como Traveling Repairman Problem (TRP), Deliveryman Problem [Lucena, 1990] e Traveling Salesman Problem with Cumulative Costs [Fischetti et al., 1993].

Neste trabalho, é proposto um algoritmo Branch-and-Bound (B&B) para resolver o PML com três estratégias distintas para calcular o limite inferior: (i) Menores Arcos; (ii) Sem Ciclos; e, (iii) Caminhos Parciais. Com a primeira estratégia, Menores Arcos, obtém-se um limite inferior em cada nó da árvore de Branch-and-Bound através de uma seleção de menores arcos do grafo original (respeitando algumas restrições). A segunda estratégia, Sem Ciclos, é equivalente a primeira estratégia com o acréscimo de uma restrição que não permite sub-ciclos na construção do limite inferior. Na última estratégia, Caminhos Parciais, o limite inferior é calculado através da combinação de diversos caminhos parciais, obtidos a partir da estratégia Menores Arcos. Todas as estratégias propostas foram implementadas utilizando a linguagem de programação C++, sem o auxílio de quaisquer resolvedores comerciais. Além das estratégias exatas, também é proposta uma heurística para encontrar limites superiores para o problema. Essa heurística é similar ao método Sem Ciclos, porém, todo nó pertencente ao limite deve possuir grau menor ou igual a dois.

Neste trabalho é realizada uma comparação entre o algoritmo B&B e uma formulação de programação linear mista. Os resultados obtidos demonstram que:

- Para instâncias com tamanho inferior a 40 nós, as duas primeiras estratégias são mais rápidas que a estratégia Caminhos Parciais.
- Em muitos casos, a primeira solução encontrada pela estratégia Caminhos Parciais é a solução ótima para o problema.
- O algoritmo B&B apresenta solução exata para instâncias da TSPLIB de tamanho até 48 nós.
- A heurística desenvolvida apresenta solução exata para todas instâncias da biblioteca TSPLIB testadas.

O trabalho foi organizado da seguinte forma: Seção 2 apresenta os trabalhos relacionados. Seção 3 apresenta uma formulação de programação linear inteira mista para o problema. Seção 4 apresenta o algoritmo de Branch-and-Bound, as três estratégias de cálculo do limite inferior e a heurística. Seção 6 apresenta os resultados computacionais. Seção 7 conclui o trabalho.

2. Trabalhos Relacionados

Algoritmos de otimização para o PML já foram propostos por diversos autores: Lucena [1990] propôs um algoritmo enumerativo baseado em programação não linear inteira, na qual os limites inferiores foram obtidos através de relaxação Lagrangeana. O autor apresentou soluções exatas para instâncias de tamanho de até 30 nós. Simchi-Levi e Berman [1991] descreveram um algoritmo Branch-and-Bound baseado em uma relaxação usando árvore geradora mínima. Fischetti et al. [1993] resolveram instâncias de tamanho até 60 nós usando um algoritmo Branch-and-Bound baseado em uma formulação de programação linear e matroides cumulativas. Wu et al. [2004] conseguiram resolver de maneira exata problemas de tamanho até 23 nós, usando uma estratégia de programação dinâmica e podas. Sarubbi et al. [2007] apresentaram uma formulação de programação linear para o PML baseado no Problema Quadrático de Atribuição (PQA) [Koopmans e Beckmann, 1957]. Nesse mesmo trabalho foi utilizado um método heurístico especializado (GRASP) para gerar limites superiores. Esses limites superiores foram utilizados para ajudar a encontrar a solução ótima. Soluções aproximadas para instâncias de tamanho até 60 nós foram obtidas em tempo razoável.

Em 2008, Méndez-Díaz et al. [2008] propuseram uma nova formulação para o PML. Eles implementaram um algoritmo de planos de corte, que utiliza desigualdades válidas associadas com sua formulação. Eles então comparam seus limites inferiores com os obtidos por Fischetti et al. [1993] e com Eijl [1995]. Os autores obtiveram soluções exatas para instâncias de tamanho até 40 nós, em tempo próximo a duas horas. João F. M. Sarubbi [2008] propôs uma formulação de fluxos para o PML baseado no Problema de Caminho Mínimo com restrições adicionais. O grafo original foi transformado em um grafo multinível, no qual os nós representam a posição de cada cliente em uma rota. Com essa formulação, o autor obteve resultados exatos para instâncias assimétricas que respeitam a desigualdade triangular de tamanho de até 80 nós. Angel-Bello et al. [2013] desenvolveram duas formulações de fluxo para o PML. Uma delas foi baseada no trabalho de João F. M. Sarubbi [2008] e mostra resultados exatos para instâncias de tamanho até 40. Ban et al. [2013] implementaram um algoritmo Branch-and-Bound para resolver o PML. Eles utilizaram uma estratégia interessante para computar o limite inferior na qual o limite é composto pelos menores arcos do grafo original. Eles também desenvolveram cortes válidos e mostraram resultados exatos para instâncias de até 40 nós.

3. Formulação de Programação Linear Mista

Neste artigo, será apresentado o modelo de programação linear inteira mista na qual o PML é modelado como um Problema de Caminho Mínimo com restrições adicionais em um grafo multinível. Nesse modelo, os nós do grafo multinível representam também a posição que o cliente é visitado na rota. Essa formulação foi baseada na de Picard e Queyranne [1978] e foi apresentada por João F. M. Sarubbi [2008]. Godinho et al. [2014] também apresentaram uma formulação para o PML baseada no Problema de Caminho Mínimo. Sua formulação também foi baseada no trabalho de Picard e Queyranne, mas difere da usada neste artigo já que os autores utilizaram variáveis de quatro índices enquanto neste trabalho são usadas variáveis de três índices.

Considere um grafo dirigido conexo $G(V, E)$, no qual V denota um conjunto de vértices e E um conjunto de arcos. Ao invés de trabalhar com o grafo G, uma transformação é aplicada a ele e então é obtido um grafo multinível $G' = (V', E')$. Para cada $|V| - 1$ vértices do grafo original, são criados $|V| - 1$ novos vértices no grafo G' para cada nível k. Além disso, para o vértice origem é criado um vértice espelhado, conectado com os vértices do último nível, k.

O grafo G' possui $|V|$ níveis. Cada nível representa a posição do determinado vértice na sequência da rota. Como no PML o custo depende da ordem que o vértice aparece na solução, então, para cada arco desse novo grafo o custo vai ser igual ao associado do grafo original, multiplicado por um $fator\_multiplicador$. O fator multiplicador é calculado da seguinte forma: $fator\_multiplicador = |V| - k + 1$, no qual k é a posição do vértice na rota. As figuras (1) e (2) representam a transformação de um grafo de quatro vértices $G = (V, E)$ em um multinível

$G' = (V', E')$. Se for garantido que todo nível será visitado uma e somente uma vez o problema se restringe a encontrar um caminho mínimo da origem 1 para a origem espelhada $1'$. Esse caminho mínimo será a solução ótima para o PML. Sendo assim, ao invés de resolver o Problema da Mínima Latência no grafo original $G = (V, E)$ é utilizado uma modelagem do Problema do Caminho Mínimo com restrições adicionais em um grafo multinível $G' = (V', E')$.

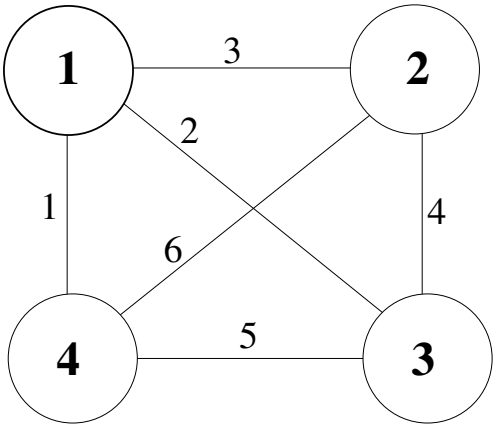


Figura 1: Grafo Exemplo

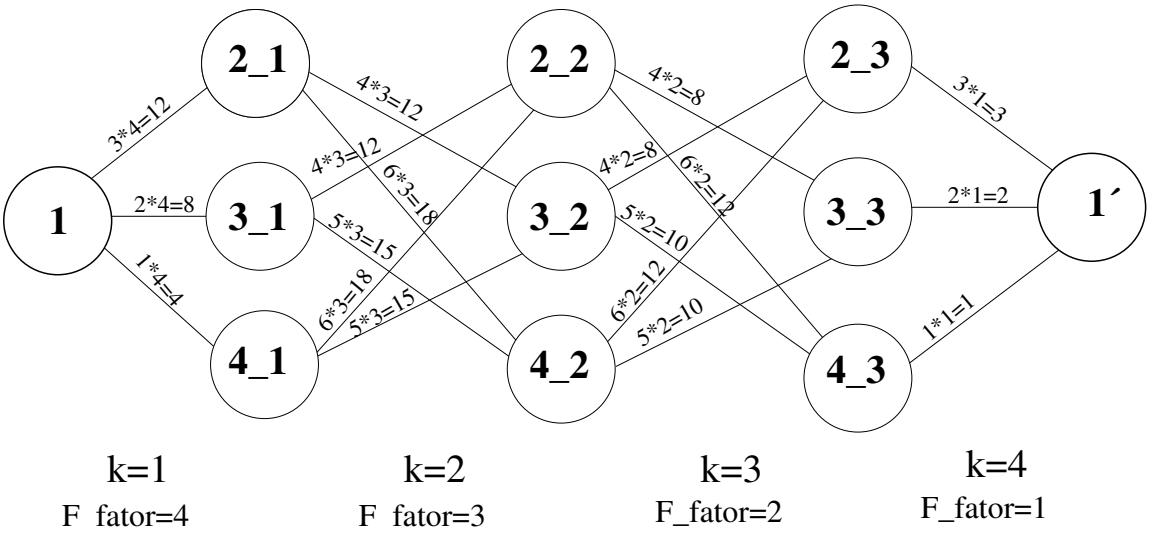


Figura 2: Grafo Multinível G'

Para esse problema é possível definir uma formulação de programação linear inteira mista contendo o seguinte conjunto de variáveis:

$$f_{ijk} = \begin{cases} 1 & \text{se o arco } (i,j) \text{ é visitado na ordem } k \\ 0 & \text{caso contrário.} \end{cases}$$

e o seguinte parâmetro:

c_{ijk}: custo fixo pago pelo entregador ao atravessar o arco $(i,j) \times (n-k+1)$. A formulação matemática é:

$$\text{minimize} \quad \sum_{j|j>1} c_{1j1} f_{1j1} + \sum_{k=2}^{n-1} \sum_{i=2}^{n} \sum_{j=2|j\neq i}^{n} c_{ijk} f_{ijk} + \sum_{i=2}^{n} c_{i1n} f_{i1n}, \tag{1}$$

sujeito a:

$$\sum_{j=2}^{n} f_{1j1} = 1 \tag{2}$$

$$\sum_{j=2}^{n} f_{j1n} = 1 \tag{3}$$

$$\sum_{j=2|i\neq j}^{n} f_{ij2} = f_{1i1} \quad i = 2,..,n \tag{4}$$

$$\sum_{j=2|i\neq j}^{n} (f_{jik} - f_{ij,k+1}) = 0 \; k = 2,..,n-2 \quad i = 2,..,n \tag{5}$$

$$\sum_{i=2|i\neq j}^{n} f_{ij,n-1} = f_{j1n} \quad j = 2,..,n \tag{6}$$

$$\sum_{i=2|i\neq j}^{n} \sum_{k=2}^{n-1} f_{ijk} + f_{1j1} = 1 \quad j = 2,..,n \tag{7}$$

$$f_{1j1} \in \{0,1\} \quad j = 2,..,n \tag{8}$$

$$f_{ijk} \in \{0,1\} \quad i = 2,..,n \quad j = 2,..,n \quad k = 2,..,n-1 \tag{9}$$

$$f_{i1n} \in \{0,1\} \quad i = 2,..,n \tag{10}$$

A função objetivo (1) soma os custos de todos os arcos do caminho. É importante enfatizar que a origem e a origem espelhada possuem apenas $|V|-1$ arcos incidentes. As restrições (4), (5) e (6) são restrições de conservação de fluxo. Restrições (4) são relacionadas com os vértices que possuem associações com o vértice origem, enquanto as restrições (6) estão relacionadas com os vértices que são associados a origem espelhada. As restrições (5) estão relacionadas com os demais vértices. Restrições (7) são as que transformam o problema de caminho mínimo no PML. Essas restrições garantem que somente um arco é incidente em cada conjunto de vértices para todos os níveis k. Restrições (8), (9) e (10) garantem que toda variável de fluxo será binária. É possível observar que para um grafo completo o modelo (1)-(10) possui $2*(n-1)+(n-2)*(n-1)*(n-2)$ nós, isto é, G' é um grafo de ordem n^3.

4. Algoritmos de Branch-and-Bound

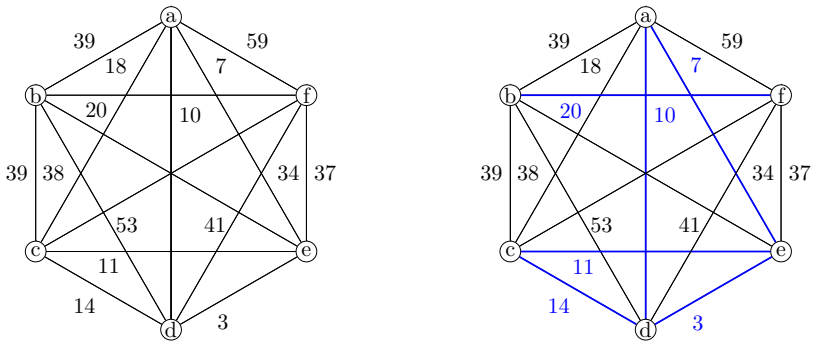
Nesta Seção, serão apresentadas respectivamente, três estratégias exatas diferentes para resolver o PML: (i) Menores Arcos; (ii) Sem Ciclos; e, (iii) Caminhos Parciais. Nas duas primeiras estratégias, Menores Arcos e Sem Ciclos, um algoritmo padrão de Branch-and-Bound foi implementado para gerar limites superiores e inferiores. Já na estratégia Caminhos Parciais, foi utilizado uma abordagem de *Full Strong Branching* [Achterberg et al., 2004], na qual cada possível nó de branch é avaliado e o mais promissor é escolhido. A estratégia de avaliação será explicada a fundo na sua respectiva subseção.

4.1. Estratégia de Limite Menores Arcos

Nesta Seção, será apresentada a primeira estratégia de cálculo de limite inferior chamada Menores Arcos. Essa estratégia funciona como um algoritmo convencional de B&B. No entanto, ao invés de calcular o limite utilizando o método Simplex ou o método Dual, utilizou-se uma abordagem simples de complexidade de tempo $O(n^2)$. Em cada nó da árvore B&B, a abordagem iterativamente seleciona arcos com menor custo e que não retornem para algum nó pai do nó atual. Além disso, o último nó da construção do limite deve ser incidente ao nó origem.

A figura 3b mostra um exemplo dessa abordagem. Supondo um grafo representado na figura 3a contendo 6 vértices e que a origem é o vértice a. Ordenando os arcos de maneira crescente tem-se a seguinte ordem: $(e,d) = 3$, $(a,e) = 7$, $(a,d) = 10$, $(c,e) = 11$, $(c,d) = 14$, $(a,c) = 18$, $(b,f) = 20$, $(a,e) = 34$, $(e,f) = 37$, $(a,c) = 38$, $(a,b) = 39$, $(b,c) = 39$, $(d,f) = 41$, $(b,d) = 53$ e $(a,f) = 59$. Caso a estratégia Menores Arcos seja escolhida, primeiramente o arco (a,e) é selecionado, pois é o de menor custo incidente ao nó origem a. Em seguida, o arco (e,d) é selecionado pois é o de menor custo no grafo. O próximo seria o arco (a,d) mas não é selecionado pois incide ao nó origem. Seleciona-se então o próximo arco que é (c,e) e depois o arco (c,d). O próximo $((a,c))$ levaria também para a origem e portanto não é selecionado. O arco (b,f) é escolhido e então, por último, deve-se escolher o arco de menor custo que incida a origem, arco (a,d).

O limite inferior obtido então é: $(7\times6)+(3\times5)+(11\times4)+(14\times3)+(20\times2)+(10\times1) = 193$.



(a) Grafo Original

(b) Estratégia Menores Arcos

Figura 3: A Figura 3a representa um grafo com seis nós. A Figura 3b representa os arcos selecionados pela estratégia Menores Arcos para formar o limite inferior no nó raiz. O limite inferior obtido foi: (7x6) + (3x5) + (11x4) + (14x3) + (20x2) + (10x1) = 193.

A seguir é apresentado um pseudo algoritmo para a construção do limite inferior. O conjunto A' representa as arestas do grafo ordenadas de maneira ascendente. O procedimento *isValid* verifica se a aresta em questão retornaria para algum vértice já fixado na árvore de B&B, o parâmetro $nivel$ indica quantos vértices já foram fixos no caminho da árvore de B&B.

Algoritmo 1: $MenoresArestas$

Data: $origem, nivel$

```
1  limite = 0 ;
2  for aresta ∈ A' do
3      if isValid(aresta) then
4          if (nivel = 1 and aresta.dest = origem) or
5          (nivel! = 1 and aresta.dest! = origem) then
6              limite = limite + nivel × aresta.custo ;
7              nivel = nivel − 1 ;
8          end
9      end
10 end
11 return limite ;
```

4.2. Estratégia de Limite Sem Ciclos

Nesta Seção, será apresentada a segunda estratégia de cálculo do limite inferior, chamada: Sem Ciclos. Esse novo algoritmo funciona de maneira similar a estratégia Menores Arcos, selecionando arcos de menor custo de maneira iterativa em um dado nó da árvore de B&B de forma a compor o limite inferior. Sua diferença, no entanto, consiste em uma restrição adicional. O algoritmo assim como o seu nome sugere, não permite a geração de sub-ciclos no limite inferior. Sempre que um arco for selecionado é verificado se a sua inclusão gera algum ciclo. Caso afirmativo, esse arco é descartado e o algoritmo tenta inserir o próximo arco. Essa nova abordagem tende a gerar limites inferiores maiores, que a priori diminuem os nós da árvore de B&B. Por outro lado, o cálculo do limite inferior é mais custoso que o da estratégia Menores Arcos.

A Figura 4a mostra um exemplo dessa abordagem. Assim como o método Menores Arcos, esse seleciona primeiramente os seguintes arcos: (a,e), (e,d) e (c,e). No entanto, para o quarto arco, ao invés de selecionar (c,d) com custo 14, o algoritmo o descarta pois sua inclusão no limite causaria ciclo. Em seguida, utilizando a mesma estratégia que o algoritmo Menores Arcos, são selecionados os arcos: $(b,f) = 20$ e $(a,e) = 34$. Por último, incidindo de volta à origem, o arco (a,d) é inserido na solução.

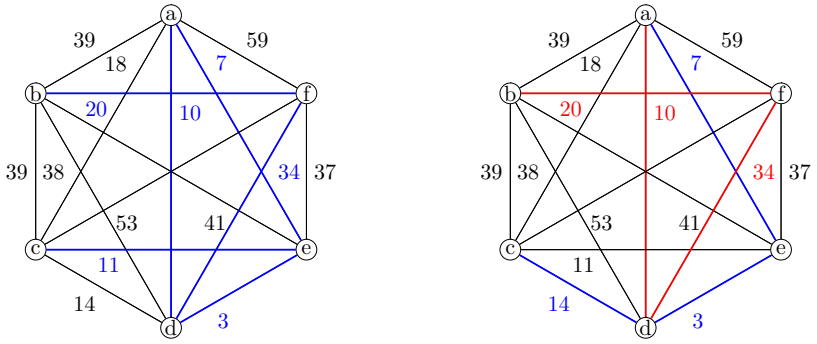
O limite obtido então é: $(7\times6)+(3\times5)+(11\times4)+(20\times3)+(34\times2)+(10\times1) = 239$.

4.3. Estratégia dos Caminhos Parciais

Nesta Seção, será apresentada a terceira estratégia para calcular o limite inferior, chamada Caminhos Parciais. Em um dado nó da árvore de B&B, ao invés de calcular o limite inferior através da seleção dos menores arcos, como nos métodos Menores Arcos e Sem Ciclos, o limite inferior é obtido através da combinação de vários caminhos parciais ótimos de tamanho máximo igual a D, no qual $D \leq N$ e N é o número de nós da instância. Cada caminho parcial é obtido através do método Menores Arcos, descrito previamente na seção 4.1, mas ao invés de percorrer a árvore inteira de B&B, é necessário apenas ir até o nível de profundidade D.

Por exemplo, suponha-se que: (i) a estratégia Caminhos Parciais está sendo utilizada; (ii) o algoritmo está no nó inicial da árvore de uma instância de tamanho 40; e, (iii) o parâmetro $D = 15$. Ao invés de selecionar 40 arcos usando os métodos Menores Arcos ou Sem Ciclos para computar o limite inferior, é utilizado o algoritmo Menores Arcos para computar um caminho ótimo de tamanho 15. Esse caminho parcial obtido é o ótimo para o PML supondo que fosse necessário visitar somente 15 nós e não retornar a origem. Depois disso, ainda é necessário selecionar mais 25 arcos. Para isso, é chamado novamente o método Menores Arcos para encontrar outro caminho parcial de tamanho 15. Esse novo caminho pode começar de qualquer um dos 40 nós do grafo, exceto o origem. No entanto, se o algoritmo estiver em um nó diferente do raiz da árvore de B&B,

vai existir um caminho fixado. Nesse caso, nenhum caminho parcial pode conter arcos que são incidentes a esse caminho previamente fixado da árvore. Por último, o algoritmo Menores Arcos é chamado novamente para encontrar mais um caminho parcial, agora com tamanho 10, e o último arco desse caminho deve ser incidente ao nó origem. A combinação dos três caminhos parciais gera um limite inferior que é utilizado pelo algoritmo B&B.



(a) Estratégia Sem Ciclos

(b) Estratégia Caminhos Parciais

Figura 4: A Figura 4a representa os arcos selecionados pela estratégia Sem Ciclos para compor o limite inferior. O limite inferior obtido através dessa estratégia foi: (7x6) + (3x5) + (11x4) + (20x3) + (34x2) + (10x1) = 239. A Figura 4b representa dois caminhos parciais gerados pela estratégia Caminhos Parciais. Cada caminho parcial possui uma cor diferente e eles são combinados de forma a gerar um limite inferior. O limite obtido através dessa estratégia com $D = 3$foi: (7x6) + (3x5) + (14x4) + (20x3) + (34x2) + (10x1) = 251.

O Algoritmo 2 apresenta a estratégia Caminhos Parciais. O conjunto V representa os vértices do grafo. O procedimento *isValid* verifica se a aresta em questão retornaria para algum vértice já fixado na árvore de B&B. O método *partial* recebe como parâmetros o vértice partida do caminho parcial, a origem (vértice inicial) e o nível da árvore B&B.

Algoritmo 2: $CaminhosParciais$

```
Data: partida, nivel
rounds = nivel/D ;
for it ← 1 to rounds do
    menorCusto = 0;
    for vertice ∈ V do
        if isValid(vertice) then
            custoAtual = menoresArestas.partial(vertice, origem, nivel);
            if custoAtual < menorCusto then
                menorCusto = custoAtual;
            end
        end
    end
    limite = limite + menorCusto;
    nivel = nivel − D;
end
return limite ;
```

5. Algoritmo Heurístico Baseado no Branch-and-Bound

Nesta Seção, será apresentado o algoritmo heurístico. Esse algoritmo funciona de maneira similar ao método Sem Ciclos, porém além de evitar ciclos, ele também evita no cálculo do limite inferior, um grau maior do que dois em todos vértices. Essa estratégia não garante otimalidade pois há casos em que a estimativa excede o valor da melhor solução possível. Essa estimativa é obtida através da seleção de arcos, assim como o método Sem Ciclos, porém sempre que um arco avaliado for gerar ciclo ou aumentar o grau de um vértice para mais que dois ele é eliminado. O valor obtido pela estimativa é então utilizado por um algoritmo B&B de forma a gerar limites superiores para o problema.

A Figura 5 mostra um exemplo dessa abordagem. Assim como no método Menores Arcos, esse seleciona primeiramente os seguintes arcos: (a, e) e (e, d). No entanto, para o terceiro arco, ao invés de selecionar (c, e) com custo 11, o algoritmo o descarta pois sua inclusão no limite aumentaria o grau de e para 3. Em seguida utilizando a mesma estratégia que Menores Arcos, selecionam-se os arcos: $(d, c) = 14$, $(b, f) = 20$. O próximo arco a ser avaliado seria $(f, d) = 34$ mas ele aumentaria o grau de d para 3. Os próximos dois arcos a serem selecionados então são: $(b, c) = 39$ e por último (voltando para origem), $(a, f) = 59$.

O limite obtido então foi: $(7\times6)+(3\times5)+(14\times4)+(20\times3)+(39\times2)+(59\times1) = 310$.

Nota-se que essa estimativa é ainda maior que o limite inferior calculado pela estratégia Caminhos Parciais.

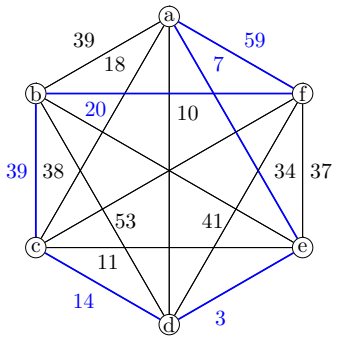


Figura 5: A Figura representa os arcos selecionados pela estratégia heurística para compor a estimativa. A estimativa através dessa estratégia foi: (7x6) + (3x5) + (14x4) + (20x3) + (39x2) + (59x1) = 310.

6. Resultados Computacionais

Os testes computacionais foram executados em um Intel Core i7 com 2.4GHz e 16Gbytes de memória RAM. O sistema operacional foi o MAC OS Yosemite. O resolvedor utilizado para comparação foi o Cplex 12.5.1. Foram utilizados dois conjuntos de instâncias. O primeiro conjunto é composto por 11 instâncias do TSPLIB e seus tamanhos variam de $N = 17$ a $N = 48$ nós. O segundo conjunto é composto por 12 instâncias que obedecem a desigualdade triangular, seus tamanhos variam de $N = 20$ a $N = 35$. Para cada instância do segundo conjunto foram gerados valores aleatórios de $[1, 100]$ e depois a matriz obtida foi triangularizada através do uso de um algoritmo de caminho mínimo. Todas instâncias utilizadas são simétricas e completas.

6.1. Instâncias TSPLIB

Nesta Seção, são apresentados os resultados obtidos para instâncias da biblioteca TSPLIB [1] que variam de tamanho $N = 17$ a $N = 48$. Os tempos computacionais das três estratégias exatas propostas são comparados com o tempo computacional necessário para resolver o problema, via CPLEX, utilizando o modelo apresentado na Seção 3. Também são apresentados as soluções encontradas pela heurística proposta, bem como o tempo computacional necessário para obtê-las.

[1] Disponíveis em: https://github.com/pdrozdowski/TSPLib.Net/tree/master/TSPLIB95/tsp

A Tabela 1 apresenta os resultados obtidos para as instâncias da TSPLIB. A coluna de cada método mostra o tempo necessário para resolver o problema. Para o método Caminhos Parciais a coluna D representa o valor usado para o parâmetro D do método. Para o método heurístico a coluna Solução Heurística mostra qual foi a solução obtida e a coluna Gap mostra o Gap da solução obtida em relação a solução ótima. Após realizar diversos experimentos, um bom valor para o parâmetro D mostrou-se ser $N/2$, no qual N é o número de nós de uma instância. Foram obtidas soluções ótimas para todas instâncias testadas.

Tabela 1: Tabela de Resultados - TSPLIB

Instância	Solução	CPLEX Tempo (s)	Menores Arcos Tempo (s)	Sem Ciclos Tempo (s)	Caminhos Parciais Tempo (s)	D	Heurística Tempo (s)	Solução Heurística	Gap (%)
gr17	12994	1.25	0.21	0.67	0.36	13	0.06	12994	0.00
gr21	24345	6.98	0.27	1.13	0.61	13	0.07	24345	0.00
gr24	13795	12.98	1.10	5.55	3.94	13	0.24	13795	0.00
fri26	10703	63.5	4.63	5.27	1.36	17	0.29	10703	0.00
bayg29	22230	284.5	22.64	89.85	67.10	18	2.57	22230	0.00
bays29	26862	238	35.64	125.6	83.23	18	2.29	26862	0.00
swiss42	22327	21294.5	1107	853.4	301.25	22	17.34	22327	0.00
dantzig42	12528	-	6389	6293	1364.48	24	131.16	12528	0.00
gr48	102378	-	-	-	56607	24	14600	102378	0.00
hk48	247926	-	-	-	33079	21	9225.72	247926	0.00
att48	209320	-	-	-	47848	24	209320	2231	0.00

6.2. Instâncias Desigualdade Triangular

Nesta Seção, são apresentados os resultados obtidos para instâncias que respeitam a desigualdade triangular que variam de tamanho $N = 20$ a $N = 35$. Da mesma forma que nos resultados das instâncias oriundas da TSPLIB, os tempos computacionais das três estratégias exatas propostas são comparados com o tempo computacional necessário para resolver o problema, via CPLEX, utilizando o modelo apresentado na Seção 3. Também são apresentados as soluções encontradas pela heurística proposta, bem como o tempo computacional necessário para obtê-las.

A Tabela 2 apresenta os resultados obtidos para as instâncias de desigualdade triangular e variam de $N = 20$ a $N = 35$. A coluna de cada método mostra o tempo necessário pelo método para resolver o problema.

Tabela 2: Tabela de Resultados - Desigualdade Triangular

Instância	Solução	CPLEX Tempo (s)	Menores Arcos Tempo (s)	Sem Ciclos Tempo (s)	Caminhos Parciais Tempo (s)	D	Heurística Tempo (s)	Solução Heurística	Gap (%)
20_1	1690	1.61	0.03	0.11	0.01	10	0.01	1690	0
20_2	2469	1.61	0.42	0.8	0.64	15	0.02	2469	0
20_3	1461	2.52	1.91	3.8	2.95	10	0.15	1461	0
25_1	1721	147.5	21.97	47	38.21	15	1.6	1721	0
25_2	1986	54	5.7	39	16.16	15	0.2	1986	0
25_3	1997	47	64.2	250	58.89	20	3.20	1997	0
30_1	2764	28	8.43	25	16.60	20	0.25	2764	0
30_2	2579	1500	262.29	929	438.81	15	11.32	2579	0
30_3	2019	168.5	597.39	762	1088	15	5.43	2019	0
35_1	2952	1103	1600.47	1737	1049	18	7.43	2952	0
35_2	3496	10158	-	-	-	-	1825	3496	0
35_3	2240	4566	-	-	-	-	85.40	2240	0

Para as instâncias da TSPLIB, os resultados obtidos pelos algoritmos propostos foram bem mais rápidos que o ILOG CPLEX utilizando a formulação apresentada na seção 3. Em vários casos, os métodos apresentados foram significativamente mais rápidos que o CPLEX*stand-alone*, *e.g.* *fri26*(46 vezes mais rápido) e *swiss42* (70 vezes mais rápido). Além disso, o CPLEX não conseguiu resolver instâncias de tamanho 48 nós no tempo limite (24 horas). Os resultados para as entradas de desigualdade triangular mostram que as estratégias propostas obtiveram resultados similares aos do CPLEX. O CPLEX foi capaz de resolver algumas instâncias de tamanho 35 que as outras estratégias não foram capazes dentro do tempo limite. Baseado nos nossos resultados é possível notar como a estratégia Menores Arcos é mais rápida que as outras para instâncias de tamanhos menores que 30. Os experimentos mostraram também que em vários casos a estratégia Caminhos Parciais obtém a solução ótima como a primeira solução por causa da abordagem de *strong branching*. O método heurístico conseguiu encontrar a solução ótima para todas as instâncias testadas da TSPLIB. Para as instâncias que obedecem desigualdade triangular, também foi possível observar que as soluções encontradas foram as ótimas.

7. Considerações Finais

Neste trabalho, foram apresentados uma formulação de programação linear mista e um novo algoritmo Branch-and-Bound (B&B) para resolver o Problema da Mínima Latência sem o uso de resolvedores comerciais. Foram apresentados resultados exatos para instâncias TSPLIB de tamanho até 48, resultados que foram atingidos em tempo muito menor que usando CPLEX sozinho. Também foi usada a ideia de que em um nó da árvore B&B, pode-se obter rapidamente um limite inferior, através da seleção dos menores arcos do grafo. Além disso foram adicionadas restrições como as apresentadas em 4.2 e 4.3 que são inerentes ao PML, essas restrições proporcionam limites inferiores melhores e portanto reduzem o custo computacional para provar otimalidade. O algoritmo heurístico, apresentado na Seção 5, obteve resultados ótimos tanto para as instâncias testadas da TSPLIB quanto para as instâncias que obedecem a desigualdade triangular, demonstrando que a estimativa apresentada funciona de maneira muito próxima a um limite inferior. Para trabalhos futuros, será explorado paralelismo na estratégia Caminhos Parciais já que o cálculo de cada caminho parcial pode ser feito individualmente. Também será feito uma decomposição da formulação apresentada (Seção 3) com objetivo de implementar uma estratégia de Branch-and-Price.

Agradecimento

Este trabalho foi parcialmente financiado pela Capes e pelo CNPq através do Programa Institucional de Bolsa de Iniciação Científica (PIBIC).

Referências

Achterberg, T., Koch, T., e Martin, A. (2004). Branching rules revisited. *Operations Research Letters*, 33:42–54.

Angel-Bello, F., Alvarez, A., e Garcia, O. (2013). Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, (37):2257–2266.

ao F. M. Sarubbi, J. (2008). *Problemas de Roteamento com Custos de Carga*. PhD thesis, Universidade Federal de Minas Gerais.

Ban, H. B., Nguyen, K., Ngo, M. C., e Nguyen, D. N. (2013). An efficient exact algorithm for the minimum latency problem. *Progress in Informatics*, (10):167–174.

Conway, R., Maxwell, W., e Miller, L. (1967). *Theory of Scheduling*. Addison-Wesley.

Dantzig, R., Fulkerson, R., e Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410.

Eijl, C. (1995). A polyhedral approch to the delivery man problem. Memorandum COSOT 95 19, Eindhoven University of Tecnology, New York University.

Fischetti, M., Laporte, G., e Martelo, S. (1993). The delivery man problem and cumulative methods. *Operations Research*, 6:1055–1064.

Godinho, M. T., Gouveia, L., e Pesneau, P. (2014). Natural and extended formulations for the time-dependent traveling salesman problem. *Discrete Appl. Math.*, 164:138–153. ISSN 0166-218X. URL `http://dx.doi.org/10.1016/j.dam.2011.11.019`.

Goemans, M. e Kleinberg, J. (1998). An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:114–124.

João F. M. Sarubbi, G. M., Henrique Pacca Luna (2008). Minimum latency problem as a shortest path problem with side constraints. In *Book of Extended Abstracts of the XIV Latin Ibero-American Congress on Operations Research (CLAIO 2008)*.

Koopmans, T. e Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica*, (25):53–76.

Lucena, A. (1990). Time-dependent traveling salesman problem - the deliveryman case. *Networks*, 20:753–763.

Méndez-Díaz, I., Zabala, P., e Lucena, A. (2008). A new formulation for the traveling deliveryman problem. *Discrete Applied Mathemathics*, 156:3223–3237.

Picard, J. C. e Queyranne, M. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26:86–110.

Sarubbi, J., Miranda, G., Luna, H., e Camargo, R. (2007). Computing sharp lower and upper bounds for the minimum latency problem. In *Hybrid Intelligent Systems-2007*, p. 71–77. IEEE Computer Society.

Simchi-Levi, D. e Berman, O. (1991). Minimizing the total flow time of n jobs on a network. *IIE Transactions*, 23:236–244.

Wu, B. Y., Huang, Z.-N., e Zhan, F.-J. (2004). Exact algorithms for the minimum latency problem. *Information Processing Letters*, 92(6):303–309.