Atividade 3 - Técnica de Relaxação Lagrangiana para o kSTSP

Lucas Guesser Targino da Silva - RA: 203534
Renan Fernando Franco da Silva - RA: 223989

1 de maio de 2022

1 Enunciado do Problema

Sejam:

1. $G = \langle V, E \rangle$: um grafo não-orientado completo:
   (a) V: conjunto de vértices;
   (b) E: conjunto das arestas;
2. $c^k : E \to \mathbb{R}_+, \ \forall k \in \{1, 2\}$, duas funções custo nos vértices;
   (a) dada uma aresta e, escrevemos $c^k(e) = c^k_e$;
3. σ: parâmetro de similaridade de ciclos;

Objetivo: encontrar dois ciclos Hamiltonianos com custo total mínimo, tal que pelo menos σ arestas do grafo sejam visitadas por ambos os ciclos.

O nome kSTPS significa *k-similar Travelling Salesman Problem*. O k é o parâmetro de similaridade σ acima, convenientemente renomeado para que não houvesse confusão com o k utilizado para indexar cada um dos ciclos (mais sobre isso na Seção 2).

2 Modelo Matemático

Nessa seção, será apresentada a formulação do problema utilizando Programação Linear Inteira. Esse não foi resolvido diretamente (já que não é o intuito da atividade), mas foi utilizado para derivar uma Relaxação Lagrangiana (Seção 3), que foi modelo de fato implementado e resolvido.

2.1 Variáveis de Decisão

- x^k_e: presença da aresta e no ciclo k;
- z_e: presença de duplicação da aresta e;

Todas as variáveis de "presença" são decisões binárias com a seguinte interpretação de valores:

0 : ausente

1 : presente

2.2 Problema de Otimização

Minimizar:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} c_e^k \; x_e^k \tag{1}$$

Sujeito a:

$$\sum_{e\in\delta(v)} x_e^k = 2 \qquad \forall v \in V, \forall k \in \{1,2\} \tag{2}$$

$$\sum_{e\in E(S)} x_e^k \leqslant |S| - 1 \qquad \forall S \subsetneq V, S \neq \emptyset, \forall k \in \{1,2\} \tag{3}$$

$$x_e^k \geqslant z_e \qquad \forall e \in E, \forall k \in \{1,2\} \tag{4}$$

$$\sum_{e\in E} z_e \geqslant \sigma \tag{5}$$

$$x_e^k, z_e \in \{0,1\} \qquad \forall e \in E, \forall k \in \{1,2\} \tag{6}$$

2.3 Explicação das Restrições

- A função objetivo (1) é soma do custo de todas as arestas selecionadas em todos os ciclos.
- A restrição (2) garante que a quantidade de arestas incidentes em todos os vértices seja 2. Essa condição faz com que todos os vértices tenham que ser visitados (duas arestas pois uma é a de "entrada" e a outra a de "saída").
- A restrição (3) garante que não existam subciclos nos ciclos. Nessa restrição, S é um subconjunto próprio e não-vazio dos vértices do problema. A expressão $E(S)$ é o conjunto das arestas cujos ambos os vértices estão em S.
- A restrição (4) garante que, se uma aresta foi escolhida para ser duplicada, então essa aresta aparecerá nos dois ciclos.
- A restrição (5) garante que pelo menos σ arestas serão escolhidas para serem duplicadas.
- A restrição (6) garante que todas as variáveis são decisões binárias, ou seja, assumem apenas um de dois possíveis valores: 0 e 1.

2.4 Tamanho das Restrições

- Restrição (2): uma para cada vértice e para cada ciclo. Total: $2 \cdot |V|$;
- Restrição (3): uma para $S \in \mathcal{P}(V), S \neq V, S \neq \emptyset$ e para cada ciclo. Total: $2 \cdot \left(2^{|V|} - 2\right)$;
- Restrições (4): uma para cada aresta e para cada ciclo. Total: $2 \cdot |E| = 2 \cdot \dfrac{|V|^2 - |V|}{2}$ (já que o grafo é completo);
- Restrições 5: apenas uma. Total: 1;

Assim, o número total de restrições é:

$$T_r = 2 \cdot |V| + 2 \cdot \left(2^{|V|} - 2\right) + 2 \cdot \frac{|V|^2 - |V|}{2} + 1 \tag{7}$$

Assim:

$$T_r \in \mathcal{O}\left(2^{|V|}\right) \tag{8}$$

Note que há um número exponencial de restrições.

3 Relaxação Lagrangiana

3.1 Escolha da Restrição a ser Relaxada

A técnica de Relaxação Lagrangiana consiste em escolher uma ou mais restrições para relaxar. Tal escolha visa remover as restrições que fazem do problem "difícil de ser resolvido" [1].

3.1.1 Análise das Restrições

- Restrição 2: não parece ser uma restrição difícil;
- Restrição 3: essa restrição é de fato difícil. Entretanto, há um número exponencial delas, de forma que é inviável relaxá-la. Além disso, há técnicas que lidam bem com ela[1];
- Restrição 4: no trabalho anterior, notou-se que o fator de similaridade causa bastante impacto no tempo computacional Figura 1. Isso significa que ele deixa o problema difícil, sendo então um bom candidato à relaxação;
- Restrição 5: essa restrição tem efeitos bem parecidos com a Restrição 4. Ela representa, entretanto, apenas uma equação de restrição, não sendo assim interessante para a relaxação;
- Restrição 6: essa restrição faz parte do modelo relaxado;

Portanto, a restrição escolhida para ser relaxada é a Restrição 4.

3.2 Modelo Matemático da Relaxação Lagrangiana

$$\max_{\lambda} \left\{ \min_{x,z} \sum_{k \in \{1,2\}} \sum_{e \in E} \mathcal{C}_e^k \; x_e^k + \lambda_e^k z_e \right\} \tag{9}$$

Em que:

$$\mathcal{C}_e^k = c_e^k - \lambda_e^k \tag{10}$$

Sujeito às restrições 2, 3, e 5 do problema original (Subseção 2.2), e a restrições de domínio:

$$0 \leqslant x_e^k, z_e \leqslant 1 \qquad \forall e \in E, \forall k \in \{1,2\} \tag{11}$$

$$\lambda_e^k \geqslant 0 \qquad \forall e \in E, \forall k \in \{1,2\} \tag{12}$$

Note que esse problema pode ser resolvido quebrando-o em 2 TSP e um problema para resolver as arestas a serem duplicadas z_e. Tais detalhes são melhor descritos nas próximas subsubseções.

[1] *Lazy constraints* por exemplo, abordado no trabalho anterior.

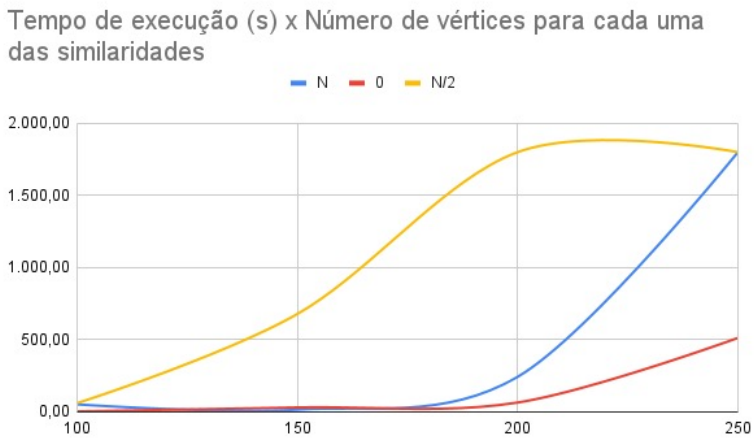


Figura 1: Tempo de execução dos experimentos do trabalho anterior.

3.2.1 Problema da escolha de ciclos derivado da Relaxação Lagrangiana

Dado $\mathcal{C}$, minimizar:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} \mathcal{C}_e^k \; x_e^k \tag{13}$$

Sujeito às restrições 2, 3, e 6. Note que esse problema é o TSP.

3.2.2 Problema da escolha das arestas similares derivado da Relaxação Lagrangiana

Minimizar:

$$\sum_{k\in\{1,2\}} \sum_{e\in E} \lambda_e^k z_e \tag{14}$$

Sujeito à restrição 6 e:

$$\sum_{e\in E} z_e \geqslant \sigma \tag{15}$$

Esse problema é facilmente resolvido escolhendo-se entre as arestas $e \in E$, as σ com menor peso $\sum_{k\in\{1,2\}} \lambda_e^k$. Para essas arestas, é dado o valor 1 e para as outras é dado o valor 0.

3.3 Método do Subgradiente

Vamos utilizar o Método do Subgradiente para resolver o dual lagrangiano mostrado na Secção 6. No começo do método, inicializamos λ e o *learning rate* π, então temos um processo iterativo. Em cada iteração recebemos uma solução ótima para (9) usando o λ atual, juntamente com o melhor *upper bound* z_{UB} e o *lower bound* atual $z_{LB}^{(i)}$, e então atualizamos o λ conforme o Algoritmo 1. Se de uma iteração para outra o *lower bound* atual piorar, atualizamos o *learning rate* fazendo $\pi = \pi \cdot \frac{\sqrt{5}-1}{2}$.

Algorithm 1 Subgradient

function INITIALIZE(λ, π)
    $\lambda_e^k = 1$, para todo $e \in E$ e $k \in \{1, 2\}$
    $\pi \leftarrow 2$
    return λ, π
function UPDATE(x, z, $z_{UB}, z_{LB}^{(i)}$, λ, π)
    $g_e^k = z_e - x_e^k$ para todo $e \in E$ e $k \in \{1, 2\}$
    $\alpha = \pi \frac{(z_{UB} - z_{LB})}{\sum_{k \in \{1,2\}} \sum_{e \in E} (g_e^k)^2}$
    $\lambda_e^k = max(0, \lambda_e^k + \alpha \cdot g_e^k)$, para todo $e \in E$ e $k \in \{1, 2\}$
    return λ

3.4 Heurística Lagrangiana

Na Heurística Lagrangiana recebemos dois ciclos hamiltonianos $tour1$ e $tour2$, além de uma constante σ, então devemos retornar dois ciclos hamiltonianos $newTour1$ e $newTour2$ tais que os mesmos compartilham no mínimo $sigma$ arestas. Nosso algoritmo está a seguir:

4 Algoritmo de Solução do Problema

5 Experimento Computacional

5.1 Configuração da Máquina

O problema foi executado num ideapad S145 81S90005BR: Lenovo IdeaPad S145 Notebook Intel Core i5-8265U (6MB Cache, 1.6GHz, 8 cores), 8GB DDR4-SDRAM, 460 GB SSD, Intel UHD Graphics 620.

O sistema operacional foi o Fedora 35 executando gcc (GCC) 11.2.1 20220127 (Red Hat 11.2.1-9), Concorde [2] e o solver QSOpt [3].

Como linguagem de programação, utilizamos C++ [4].

5.2 Dados do Problema

Os dados do problema foram fornecidos em um arquivo contendo 4 colunas e 250 linhas. A interpretação dos dados é a seguinte: cada linha representa um vértice e cada par de coluna as coordenadas desse vértice. A razão para um vértice ter duas posições diferentes é simplesmente para que as distâncias entre eles tenham valores diferentes no primeiro e no segundo ciclo.

O modelo na verdade precisa apenas de pesos. Construímos a primeira função de custo como a distância euclidiana entre os pontos das colunas 1 e 2. Da mesma forma, utilizamos distância euclidiana entre os pontos das colunas 3 e 4 para construir a segunda função de custo.

Note que, com essa interpretação, parece que temos dois conjuntos de vértices, um definido pelas colunas 1 e 2, e outro definido pelas colunas 3 e 4. Conforme explicitado no primeiro parágrafo da seção, esse não é o caso. Cada linha é um vértice e os valores fornecidos servem apenas para calcular a distância euclidiana e usá-la como peso para as arestas. Dessa forma, a aresta que liga os vértices representados pelas linhas 12 e 84, por exemplo, possui dois pesos diferentes, um para ser utilizado no primeiro ciclo e outro para ser utilizado no segundo.

```
1: function HEURISTIC(tour1, tour2, σ)
2:     equalEdges ← arestas em comum entre tour1 e tour2
3:     remainingEdges ← arestas que não estão em ao menos um dos tours
4:     Ordene as arestas dos conjuntos equalsEdges e remainingEdges por seus custos, onde
       o custo de uma aresta é a soma dos valores do seu custo em cada tour
5:     edges ← concatenação de equalsEdges e remainingEdges, nesta ordem
6:     newTour1 ← ∅
7:     newTour2 ← ∅
8:     numSimilar ← 0
9:     for cada aresta e em edges do
10:        if numSimilar ≥ σ then
11:            break
12:        if e não forma um subtour em newTour1 e newTour2 then
13:            newTour1 ← newTour1 ∪ {e}
14:            newTour2 ← newTour2 ∪ {e}
15:            numSimilar ← numSimilar + 1
16:    edgesTour1 ← arestas do tour1 ordenadas por seu custo no tour1
17:    for cada aresta e em edgesTour1 do
18:        se e não formar subtour em newTour1, adicione e no mesmo
19:    edges1 ← todas as arestas ordenadas por seu custo no tour1
20:    for cada aresta e em edges1 do
21:        se e não formar subtour em newTour1, adicione e no mesmo
22:
23:    edgesTour2 ← arestas do tour2 ordenadas por seu custo tour2
24:    for cada aresta e em edgesTour2 do
25:        se e não formar subtour em newTour2, adicione e no mesmo
26:    edges2 ← todas as arestas ordenadas por seu custo no tour2
27:    for cada aresta e em edges2 do
28:        se e não formar subtour em newTour2, adicione e no mesmo
29:    return newTour1 e newTour2
```

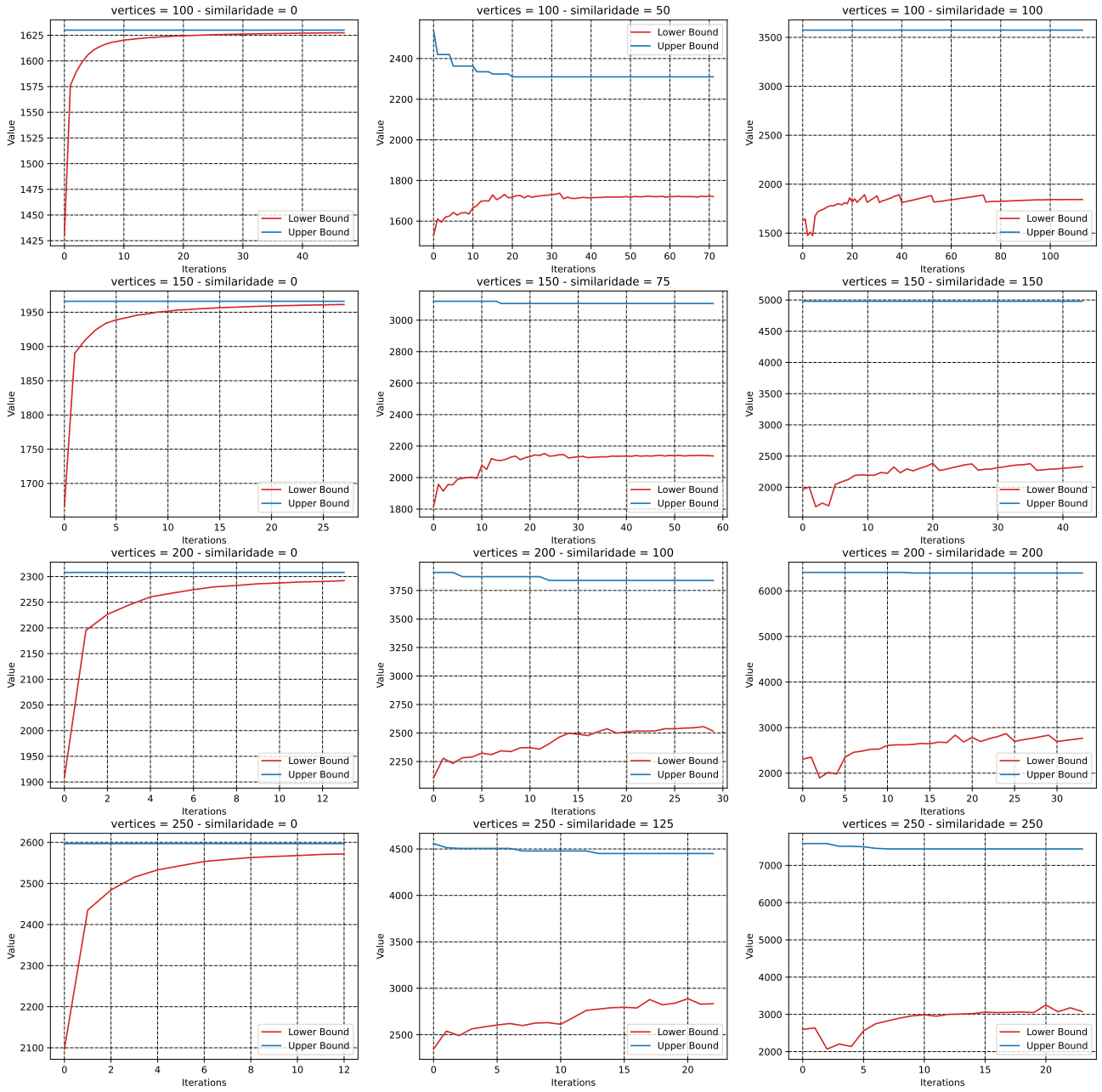
5.3 Geração das Instâncias

Para gerar instâncias de um dado tamanho N, utilizamos as primeiras N linhas dos dados fornecidos.

6 Resultados dos Experimentos

$\|V\|$	σ	z_{UB}	z_{LB}	opt gap	τ
100	0	1630	1627.41	0.002	609
100	100	3574	1842.37	0.940	329
100	50	2310	1720.67	0.343	240
150	0	1966	1961.29	0.002	605
150	150	4980	2331.49	1.136	618
150	75	3106	2137.31	0.453	507
200	0	2308	2292.01	0.007	619
200	100	3838	2516.89	0.525	617
200	200	6392	2766.68	1.310	635
250	0	2597	2571.7	0.010	733
250	125	4451	2833.34	0.571	700
250	250	7442	3076.08	1.419	616

Tabela 1: Resultados dos experimentos computacionais. τé o tempo de processamento em segundos.


vertices = 100 - similaridade = 0
vertices = 100 - similaridade = 50
vertices = 100 - similaridade = 100
vertices = 150 - similaridade = 0
vertices = 150 - similaridade = 75
vertices = 150 - similaridade = 150
vertices = 200 - similaridade = 0
vertices = 200 - similaridade = 100
vertices = 200 - similaridade = 200
vertices = 250 - similaridade = 0
vertices = 250 - similaridade = 125
vertices = 250 - similaridade = 250
Lower Bound
Upper Bound
Value
Iterations

function KSTSP($custos1, custos2, \sigma, \tau_{max}, \pi_{min}, \Delta_{min}$)
$\lambda_0 \leftarrow 1$
$\tau \leftarrow 0$
$\pi \leftarrow 2$
$\Delta z_{LB} \leftarrow +\infty$
$\Delta z_{UB} \leftarrow +\infty$
while $\tau < \tau_{max}$ or $\pi > \pi_{min}$ or ($\Delta z_{LB} > \Delta_{min}$ and $\Delta z_{UB} > \Delta_{min}$) **do**
$tsp1 \leftarrow$ Solve-TSP($custos1$)
$tsp2 \leftarrow$ Solve-TSP($custos2$)
$z \leftarrow$ Resolve-z(λ, σ
$ciclo1 \leftarrow$ Heuristic($tsp1, tsp2, \sigma$)
$ciclo2 \leftarrow$ Heuristic($tsp2, tsp1, \sigma$)
$z_{LB}, \Delta z_{LB} \leftarrow$ Compute-Lower-Bound($ciclo1, ciclo2, z, \lambda$)
$z_{UB}, \Delta z_{UB} \leftarrow$ Compute-Upper-Bound($ciclo1, ciclo2, z$)
$\lambda, \leftarrow$ Update($ciclo1, ciclo2, z, .z_{UB}, \lambda, \pi$)
if $\Delta z_{LB} < 0$ **then**
$\pi \leftarrow \frac{\sqrt{5}-1}{2}\pi$

7 Análise dos Resultados

Nota-se que um dos passos do Algoritmo 4 e encontrar duas soluções independentes de TSP, um para cada conjunto de distâncias fornecidas. Mas quando o fator de similaridade σ é zero, essa é justamente a solução ótima. Portanto, para tais casos, o algoritmo encontra a solução ótima no primeiro passo. Isso significa que o Upper Bound z_{UB}nunca será atualizado. Já o Lower Bound z_{LB}depende dos multiplicadores λescolhidos. A solução ótima do Dual Lagrangiano 9 é então atingida quando $\lambda = 0$. No caso em questão, $\sigma = 0$ e portanto a solução do problema da Subseção 3.2.2 é $z = 0$. Isso faz com que o subgradiente g no Algoritmo 3.3 seja sempre negativo, i.e. a solução de fato direciona-se para $\lambda = 0$. Tudo isso pode ser visto nos gráficos de "similaridade = 0". Neles, o Upper Bound z_{UB}é constante, enquanto que o Lower Bound z_{LB}dirige-se para a solução ótima.

Para a solução de todas as instâncias, foi dado um tempo limite de 10 minutos (600 segundos). Em muitos casos, foi tal critério que fez a execução do algoritmo parar. Em alguns casos, o tempo excedeu o limite devido a detalhes de implementação da verificação de limite de tempo. O algoritmo checa tal limite apenas no reinício do loop, de forma que sua nunca é interrompida. Isso poderia ser melhorado, mas para os objetivos da atividade não houve tal necessidade.

Nos outros casos, foi o critério de passo mínimo π que definiu a parada do algoritmo. Toda vez que o Lower Bound z_{LB} piora de uma iteração para a outra, o passo é reduzido. Assim, casos que apresentam bastante oscilação do Lower Bound têm maiores chances de parar por tal critério. Isso é justamente o que observamos nas instâncias de 100 vértices com similaridade σigual a 50 e 100, e na instância de 150 vértices com similaridade 75.

Em todos os casos, observa-se baixa variação do Upper Bound com o número de iterações. Os casos de similaridade $\sigma = 0$ já foram discutidos acima. Para os outros casos, isso pode indicar que a heurística de factibilização da solução do Dual Lagrangiano não é capaz de gerar instâncias factíveis boas. Também pode indicar que os parâmetros utilizados no algoritmo são inadequados, muito embora sejam necessárias mais investigações e experimentações para ter alguma conclusão sobre a real causa desse comportamento.

Por fim, observa-se que é obtido um gap de otimalidade, apresentado na Tabela 6. Avaliações

quando à qualidade de tais gaps de otimalidade seriam interessantes de um ponto de vista prático, muito embora tal análise tenha que levar em conta também outros fatores como tempo de execução. Portanto, apesar do valor prático, está fora do escopo dessa atividade qualificar tais resultados.

Referências

[1] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6. Athena Scientific Belmont, MA, 1997.

[2] W. Cook, "Concorde tsp solver," 2022.

[3] D. Applegate, W. Cook, S. Dash, and M. Mevenkamp, "Qsopt linear programming solver," 2022.

[4] B. Stroustrup, *The C++ programming language*. Pearson Education, 2013.