UNIVERSIDADE FEDERAL DE LAVRAS
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO
GCC218 - ALGORITMOS EM GRAFOS
Prof. Mayron César de O. Moreira

Descrição do Problema

Durante a pandemia, a demanda por serviços logísticos cresceu em virtude do aumento de compras online (Fonte: "Logística cresce na pandemia com aumento de compras pela internet", por Paula Monteiro, em Pequenas Empresas & Grandes Negócios). Grandes empresas que realizam suas operações de entregas de produtos buscam sempre uma redução de custos logísticos, a fim de utilizar tal economia de recursos em outros investimentos de interesse corporativo. Um dos problemas mais comuns nesses contextos é descrito formalmente abaixo.

Considere que uma empresa possua uma frota de veículos *M* e um conjunto de clientes *C* a serem atendidos. O deslocamento dos veículos pode ser modelado através de um grafo direcionado *G = (V, A)*. O conjunto de vértices do grafo pode ser representado por *V = P ∪ D ∪ {0}*, em que *P* representa o conjunto de pontos de coleta, *D* o conjunto de pontos de entrega e *0* representa o depósito. O conjunto de arcos é denotado por *A*, e representa as conexões entre os vértices, seja depósito ou clientes. A cada arco *(i, j)*, *i ≠ j*, associa-se um custo c_{ij} e um tempo t_{ij}. Assume-se que os tempos de deslocamento respeitam a desigualdade triangular ($\forall i, j, k \in V,\ t_{ij} \leq t_{ik} + t_{kj}$). Considera-se que o custo de deslocar-se entre *i* e *j* é diretamente proporcional ao tempo de deslocamento entre *i* e *j*.

Cada veículo *m* tem uma capacidade q_m, enquanto cada ponto *i* possui uma demanda d_i e um tempo de serviço, para ser atendido, denotado por s_i. Em um pedido $r = (i, j),\ i \in P, j \in D$, assume-se que $d_i \geq 0$ e que $d_i + d_j = 0$. Cada cliente *i* tem uma janela de tempo $[a_i, b_i]$. Um veículo deve chegar ao cliente antes do tempo b_i. Pode ser que um veículo chegue ao cliente *i* antes do tempo a_i. No entanto, o cliente não será atendido antes desse instante. O depósito também possui uma janela de tempo $[a_0, b_0]$. Os veículos não podem sair do depósito antes de a_0 e devem estar de volta antes ou no horário b_0. Adota-se os seguintes valores para a janela de tempo de depósito: $a_0 = 0$ e $b_0 = H$, em que H é o horizonte de tempo da roteirização.

Uma solução factível (ou viável) para tal problema deve respeitar as restrições apresentadas a seguir.

1. **precedência de coleta e entrega:** para um pedido *r = (i, j)*, em que *i* e *j* são seus respectivos pontos de coleta e entrega, o ponto de entrega $j \in D$ não pode ser visitado antes de seu correspondente ponto de coleta $i \in P$.
2. **origem e horário de serviço:** cada veículo $f \in \{1,\ldots, m\}$ deve partir e retornar do ponto de origem *0* no intervalo da janela de tempo $[a_0, b_0]$.
3. **janelas de tempo:** O tempo de chegada do veículo $f \in \{1,\ldots, m\}$ ao ponto $i \in V$ não pode exceder b_i. Caso o motorista do veículo $f \in \{1,\ldots, m\}$ chegue antes de a_i, o mesmo deve esperar até a_i para realizar o atendimento do ponto.
4. **obrigatoriedade e exclusividade de visita:** para cada pedido *r=(i, j)*, seus pontos de coleta e entrega devem ser visitados exatamente uma vez.
5. **atendimento de pedido:** em um pedido *r=(i, j)*, a visita ao ponto $i \in P$ por um veículo $f \in \{1,\ldots, m\}$ torna obrigatório que o atendimento ao ponto $j \in D$ seja feito pelo mesmo veículo.
6. **capacidade do veículo:** o somatório das demandas referentes aos pontos atendidos por um veículo $f \in \{1,\ldots, m\}$ não pode ultrapassar a capacidade do veículo, denotada por q_m.

A função objetivo a ser otimizada é hierarquicamente definida por:

1. Minimização da quantidade de veículos utilizada para atender todos os pedidos, respeitando as restrições do problema.
2. Minimização do custo total gasto por todas as rotas.

Objetivos

Desenvolva um algoritmo eficiente, que retorne uma solução viável para o problema descrito acima. Para tanto, as linguagens de programação C++ e Python podem ser utilizadas.

Metas

O trabalho será desenvolvido em quatro macroentregas. As tarefas de cada uma das macros são descritas abaixo.

Macroentrega 1: Criação de funções para leitura dos dados / Proposição de um algoritmo para a resolução do problema

1. Implementação das estruturas de dados.

2. Implementação de função(ões) para a leitura de dados.
3. Implementação de função de verificação de solução.
    - Gere uma solução aleatória. Passando tal solução como parâmetro, crie uma função que verifica se as restrições descritas acima (janelas de tempo, precedência e capacidade do veículo) são respeitadas nessa solução.
4. Crie um documento, com no máximo 2 páginas, que explique em detalhes o algoritmo delineado. Apresente um pseudocódigo.

- Data de entrega: **30/01/2022**, até às **14h**.
- Local de entrega: **Campus Virtual.**
- Valor: até 15%.
- Conteúdo a ser inserido no Campus Virtual: ***link*** de um repositório ***GitHub*** de um dos membros do grupo. No repositório, as tarefas da macroentrega serão disponibilizadas. Lembre-se que o repositório deve ser **público**.

Macroentrega 2: Implementação do algoritmo

1. Implemente um algoritmo que encontre uma solução viável para o problema descrito acima.
    - Critério de qualidade das soluções será um dos itens a serem avaliados. Para saber o quão boa é uma solução, define-se:
        - I uma instância a ser resolvida;
        - X_I a solução que o seu algoritmo retorna, em uma dada execução;
        - X_I^* a melhor solução conhecida (referente a ambas as funções de qualidade), referente a instância I;
        - $M(Y)$: número de veículos gastos pela solução Y;
        - $C(Y)$: custo gasto pelos veículos da solução Y;
        - Desvio da solução X_I em relação à solução X_I^*: $\Delta_I(X_I, X_I^*) = (\frac{M(X_I)-M(X_I^*)}{M(X_I^*)} \times 100;\ \frac{C(X_I)-C(X_I^*)}{C(X_I^*)} \times 100)$.
        - Prioriza-se o menor desvio em relação ao número de veículos como prioritário. Posteriormente, como critério de desempate, prioriza-se o menor desvio com relação ao custo da solução.
        - O grupo deverá disponibilizar um arquivo com todos os valores de solução encontrados pelo algoritmo, para que o professor possa calcular o desvio em relação às melhores soluções. O padrão de arquivo de saída será disponibilizado.
2. Seja criativo na implementação do seu algoritmo! Não pare o seu desenvolvimento em um algoritmo construtivo. Proponha algoritmos de busca local para aprimorar o valor da solução encontrada.

- Data de entrega: **20/02/2022**, até às **14h**.
- Local de entrega: **Campus Virtual.**
- Valor: até 50%.
- Conteúdo a ser inserido no Campus Virtual: ***link*** de um repositório ***GitHub*** de um dos membros do grupo. No repositório, as tarefas da macroentrega serão disponibilizadas. Lembre-se que o repositório deve ser **público** e que um **README** deve ser disponibilizado, para instruir o professor sobre como o programa pode ser executado.

Macroentrega 3: Implementação de interface via Google Colab para interação e visualização das soluções

A interface de seu programa deve ser disponibilizada por meio do *Google Colab*. No *notebook*, o usuário deve ter a opção de fazer o upload de um arquivo. O processamento do algoritmo deve ocorrer chamando o seu projeto dentro do *Colab*. Ao imprimir sua resposta, utilize uma ferramenta para visualização da solução, tal como Google My Maps. As estatísticas relativas à solução podem ser exibidas por meio de *dashboards*, utilizando as bibliotecas pandas ou ploty, por exemplo.

- Seja criativo na exibição das informações do *dashboard*. Algumas ideias:
    - Quantidade de veículos utilizados;
    - Custo total do trajeto realizado por todos os veículos utilizados;
    - Tempo ocioso de cada veículo (quando o veículo chega antes da janela de tempo, ou quando chega antes do tempo de fechamento do depósito);
    - Maior e menor distância percorrida por cada veículo considerando cada par de pontos visitados na sequência.

- Data de entrega: **06/03/2022**, até às **14h**.
- Local de entrega: **Campus Virtual.**
- Valor: até 35%.
- Conteúdo a ser inserido no Campus Virtual: ***link*** de um repositório ***GitHub*** de um dos membros do grupo. No repositório, as tarefas da macroentrega serão disponibilizadas. Lembre-se que o repositório deve ser **público** e que um **README** deve ser disponibilizado, para instruir o professor sobre como o programa pode ser executado.

Macroentrega 4: Entrevista com o professor

A entrevista será realizada com cada grupo (no máximo 3 alunos), em sala de aula. Perguntas a respeito do contexto estudado e o algoritmo implementado podem ser feitas. A entrevista é uma fase obrigatória para a atribuição da nota final. A nota será comum para todo o grupo.

Seja $E \in [0, 1]$ a nota da entrevista, e $N \in [0, 100]$ a nota final do trabalho. A nota final do trabalho, denotada por NF, é calculada por $NF = E \times N$.

Base de Dados

Trinta instâncias foram selecionadas da literatura para a seção de experimentos computacionais deste trabalho. A Tabela 1 apresenta as informações a respeito dos dados.

Tabela 1. *Instâncias e valores de referência.*

Instância (I)	$M(X_I^*)$	$C(X_I^*)$
bar-n100-1	6	733
bar-n100-2	5	554
ber-n100-3	3	713
ber-n100-4	3	494
nyc-n100-4	2	535
nyc-n100-5	2	671
poa-n100-1	12	1589
poa-n100-2	15	1539
poa-n100-6	3	562
poa-n100-7	5	779
bar-n200-1	22	1829
bar-n200-2	23	2072
bar-n200-3	8	1644
bar-n200-4	13	838
ber-n200-5	27	3944
ber-n200-6	9	3016
nyc-n200-3	7	1019
nyc-n200-4	4	1037
poa-n200-1	25	2433

poa-n200-2	13	2347
bar-n400-1	32	3085
ber-n600-1	47	7783
nyc-n800-3	26	3871
poa-n1000-1	30	8082
poa-n1500-6	141	16678
nyc-n2000-4	29	7198
ber-n2500-3	248	18483
bar-n3000-6	79	21718
poa-n4000-2	513	59115
poa-n5000-3	286	60628

As primeiras 10 linhas de cada arquivo contém informações gerais sobre a instância:

- NAME: <identificação de instância única>
- LOCATION: <cidade onde foi gerado>
- COMMENT: : <uma referência geral>
- TYPE: <o tipo da instância>
- SIZE: <número de vértices incluindo o depósito>
- DISTRIBUTION: <distribuição a qual a instância foi gerada>
- DEPOT: <tipo de localização do depósito: 'central' ou 'aleatório'>
- ROUTE-TIME: <horizonte de tempo da roteirização>
- TIME-WINDOW: <largura da janela de tempo>
- CAPACITY: <capacidade máxima do veículo>

Após a linha com a palavra “NODES”, é seguida por uma quantidade SIZE de linhas, contendo as informações completas de cada ponto (vértice) no arquivo de instância. Para cada linha, existem 9 campos separados por um único caractere de espaço em branco como abaixo: *<id> <lat> <long> <dem> <etw> <ltw> <dur> <p> <d>*.

Os campos são:

- <id>: o identificador único do ponto (o ponto 0 é o depósito único);
- <lat>: latitude deste local;

- <long>: longitude deste local;
- <dem>: a demanda deste nó (dem > 0 para coleta, dem < 0 para entrega);
- <etw>: tempo mais cedo possível para iniciar o serviço (janela de tempo);
- <ltw>: última hora possível para iniciar o serviço (janela de tempo);
- <dur>: a duração do serviço neste local;
- <p>: o par de coleta se <id> for uma entrega; e 0 caso contrário;
- <d>: o par de entrega se <id> for uma coleta; e 0 caso contrário

O <p> e <d> são apenas para fins de integridade. Em todas as instâncias, para um local de coleta <id> sua entrega é dada por (<id>+((SIZE-1)/2)). Para um local de entrega <id>, sua coleta é dada por (<id>-((SIZE-1)/2)).

Após todos os NODES, existe uma linha contendo a palavra EDGES seguida de SIZE linhas, cada uma com SIZE valores inteiros separados por um único espaço em branco. Esses números inteiros representam os tempos de viagem entre cada par de locais na instância, medidos em minutos e calculados usando o Ferramenta Open Source Routing Machine (OSRM).

Todas as instâncias terminam com uma linha contendo a palavra EOF.

Sugestão!

Dediquem-se no desenvolvimento deste trabalho! Além do aprendizado gerado, muitas empresas utilizam projetos acadêmicos disponíveis no *GitHub* como parte da avaliação dos candidatos em processos seletivos.