WasteApp: recolha seletiva de lixo (Parte 1)

Turma 1 - Grupo 1

up201806551@fe.up.pt	Beatriz Costa Silva Mendes
up201806524@fe.up.pt	Daniel Garcia Lima Sarmento da Silva
up201806538@fe.up.pt	Henrique Manuel Ruivo Pereira

24 de abril de 2020

Projeto CAL - 2019/20 - MIEIC

Professora das Aulas Laboratoriais: Liliana da Silva Ferreira

Índice

1 Descrição

Neste trabalho pretende-se criar uma aplicação para gestão e localização dos pontos de recolha seletiva de resíduos, denominada *WasteApp*. Esta *app* deverá permitir aos seus utilizadores localizar os pontos de recolha seletiva mais próximos e os tipos de resíduos que lá se podem depositar, bem como realizar a gestão da sua capacidade.

Para além disso, a *app* tem ainda por objetivo dar origem a um modelo de negócio que tem por base a recolha ao domicílio de determinados tipos de resíduos para exploração financeira (otimizando o itinerário percorrido).

A aplicação deve ser capaz de determinar a acessibilidade aos pontos de depósito/recolha.[1]

1.1 1ª Abordagem: Ponto de Recolha Mais Próximo

Numa fase inicial, despreza-se a capacidade do ponto de recolha e os tipos de resíduos que poderão ser depositados neste. Deste modo, o único objetivo acaba por ser apenas determinar qual o ponto de recolha mais próximo do utilizador, calculando a rota mais curta até um qualquer ponto de recolha (partindo-se do princípio de que este se encontra acessível).

1.2 2ª Abordagem: Ponto de Recolha de um Determinado Resíduo Mais Próximo

Neste segundo passo, acrescenta-se ao problema o facto de que determinados pontos de recolha se limitam a aceitar certos tipos de resíduos. Assim, cada utilizador terá de ter pelo menos 5 pontos de recolha mais próximos, um para cada tipo de resíduo (papel, vidro, plástico, pilhas e lixo indiferenciado).

1.3 3ª Abordagem: Ponto de Recolha de um Determinado Resíduo Mais Próximo com Capacidade Suficiente

Posteriormente, é necessário considerar que os pontos de recolha têm uma capacidade máxima, ou seja, depois de atingir essa capacidade, não poderá ser depositado nele mais nenhum resíduo. Assim sendo, se o ponto de recolha mais próximo do utilizador de um determinado resíduo ultrapassar a sua capacidade máxima com o depósito do utilizador, terá de ser atribuído a este um novo ponto de recolha desse mesmo resíduo, que será o mais próximo ainda com capacidade.

1.4 4ª Abordagem: Implementação Paralela do Modelo de Negócio

Por fim, neste última abordagem, terá de ser implementado um serviço de recolha de resíduos ao domicílio para exploração financeira, que depois serão levados para uma central de reciclagem. Assim sendo, terá de ser determinado o menor itinerário possível que passe pelas casas dos utilizadores que fornecem os resíduos a ser recolhidos.

2 Identificação e formalização do problema

2.1 Ponto de Recolha Mais Próximo

Identificação do Problema

Nesta vertente de utilização da *app*, pretende-se disponibilizar ao utilizador o ponto de recolha mais próximo em que pode depositar o tipo e a quantidade de lixo que pretende, bem como o itinerário para lá chegar.

Dados de Entrada

- *type* – tipo de lixo que se pretende depositar (plástico, vidro, papel, etc.);
- *quantity* – quantidade de lixo que se pretende depositar;
- $G = (V, E)$ – grafo dirigido pesado, composto por:
  - V – vértices, que representam interseções, compostos por:
    - $Dist$ – distância a L;
    - $Adj \subseteq E$ – conjunto de arestas que partem do vértice.
  - E – arestas, que representam vias de comunicação, compostas por:
    - W – peso da aresta (distância entre os dois vértices);
    - ID – identificador único da aresta;
    - $V_{\mathrm{i}} \in V$ – vértice de partida;
    - $V_{\mathrm{f}} \in V$ – vértice de chegada.
- P – conjunto de pontos de recolha, cada um caracterizado por:
  - *type* – tipo de lixo que recolhem;
  - Q – quantidade do lixo que lá está;
  - $Q_{\max}$ - quantidade máxima do lixo que pode lá estar;
  - $R \in E$ – aresta onde está;
  - D – distância ao vértice de chegada da aresta.
- L - Localização do utilizador, caracterizada por:
  - $R \in E$ - aresta onde está;
  - D - distância ao vértice de chegada da aresta.

Dados de Saída

- $P_{\mathrm{f}} \in P$ - ponto de recolha do tipo correspondente mais próximo do utilizador com capacidade para a quantidade de lixo que se pretende depositar;
- $C = e \in E : 1 \leq i \leq |E|$ - sequência de arestas, correspondente ao caminho a percorrer ($\mathrm{e_i}$ = i-ésima aresta).

Restrições

- Dados de Entrada:
  - $quantity > 0$, visto que representa peso ou volume;
  - $\forall p \in P, Q \geq 0 \wedge Q_{\max} > 0 \wedge Q \leq Q_{\max}$, pelo mesmo motivo;
  - $\forall e \in E, W > 0$, visto que o peso das arestas representa uma distância;
  - $\forall L \wedge \forall p \in P, D \geq 0$, pelo mesmo motivo;
  - $\forall v \in V, Dist \geq 0$, pelo mesmo motivo.
- Dados de Saída:
  - $R(P_{\mathrm{f}}) = e_{\mathrm{final}}$.

Funções Objetivo

A solução ótima do problema passa por minimizar a distância que o utilizador tem de percorrer para depositar o lixo, de modo que pretende-se minimizar a seguinte função:

$$f = \sum_{c \in C} W(c)$$

2.2 Novo Modelo de Negócio

Identificação do Problema

Nesta vertente da *app*, pretende-se disponibilizar ao utilizador um itinerário desde a sua localização até à localização da central de reciclagem, passando por domicílios que tenham o tipo de resíduos que pretendem recolher, sem nunca ultrapassar a quantidade possibilitada pelo utilizador.

Dados de Entrada

- *type* – tipo de lixo que se pretende depositar (plástico, vidro, papel, etc.);
- *quantity* – quantidade de lixo que se pretende recolher;
- $G = (V, E)$ – grafo dirigido pesado, composto por:
  - V – vértices, que representam interseções, compostos por:
    - $Dist$ – distância a $\mathrm{L_i}$;
    - $Adj \subseteq E$ – conjunto de arestas que partem do vértice.
  - E – arestas, que representam vias de comunicação, compostas por:
    - W – peso da aresta (distância entre os dois vértices);
    - ID – identificador único da aresta;
    - $V_\mathrm{i} \in V$ – vértice de partida;
    - $V_\mathrm{f} \in V$ – vértice de chegada.
- P_i – conjunto de domicílios que pretendem recolha, cada um caracterizado por:
  - *type* – tipo de lixo que recolhem;
  - Q – quantidade do lixo que lá está;
  - $R \in E$ – aresta onde está;
  - D – distância ao vértice de chegada da aresta.
- L_i – localização do utilizador, caracterizada por:
  - $R \in E$ – aresta onde está;
  - D – distância ao vértice de chegada da aresta.
- L_f – localização da central de reciclagem:
  - $R \in E$ – aresta onde está;
  - D – distância ao vértice de chegada da aresta.

Dados de Saída

- $C = e \in E : 1 \leq i \leq |E|$ - sequência de arestas, correspondente ao caminho a percorrer ($\mathrm{e_i}$ = i-ésima aresta);
- $P_\mathrm{f} = p \in P : 1 \leq i \leq |P|$ - sequência de domicílios de onde o utilizador deve recolher resíduos.

Restrições

- Dados de Entrada:
  - $quantity > 0$, visto que representa peso ou volume;
  - $\forall p \in P, Q \geq 0$, pelo mesmo motivo;
  - $\forall e \in E, W > 0$, visto que o peso das arestas representa uma distância;
  - $\forall L \wedge \forall p \in P, D \geq 0$, pelo mesmo motivo;
  - $\forall v \in V, Dist \geq 0$, pelo mesmo motivo.
- Dados de Saída:
  - $\forall p \in P_{\mathrm{f}}, R(p) \in C$.
  - $\sum_{p \in P_{\mathrm{f}}} Q(p) < quantity$.

Funções Objetivo

A solução do problema passa por maximizar a quantidade recolhida, sem ultrapassar a quantidade possível de recolher, e por minimizar a distância percorrida, dando prevalência ao primeiro critério. Assim, queremos minimizar as seguintes funções:

$$f = quantity - \sum_{p \in P_{\mathrm{f}}} Q(p)$$

, sendo que nunca pode ser < 0

$$g = \sum_{c \in C} W(c)$$

3 Perspetiva de Solução

A perspetiva de solução adotada tem por base a aplicação de várias fases prévias ao processamento do problema.

Inicialmente, e para reduzir a complexidade temporal do processamento, os passos iniciais baseiam-se na eliminação de arestas que não proporcionem nenhum tipo de vantagem (tais como as correspondentes a vias inutilizáveis - por obras, por exemplo - e as que tenham arestas/conjuntos de arestas com origem e destino comuns de peso total menor).

O passo seguinte corresponde à ordenação dos pontos de recolha por proximidade a cada utilizador. Tendo em conta que a morada de um utilizador e os pontos de recolha podem corresponder a um nó ou a uma parte de uma aresta, se se tratar do primeiro caso apenas se tem de ter em conta as arestas adjacentes; quanto ao segundo caso, terá de se localizar o ponto de recolha ou a morada dentro da aresta, isto é, saber a distância deste(a) a cada uma das extremidades da aresta.

3.1 Ponto de Recolha Mais Próximo

Para o primeiro caso, usa-se o algoritmo de *Dijsktra* abordado nas aulas, cujo pseudocódigo é apresentado na figura seguinte, uma vez que o grafo não contem arestas de peso negativo.

```
DIJKSTRA(G, s):  // G=(V,E), s ∈ V
1.    for each v ∈ V do                          Tempo de execução:
2.       dist(v) ← ∞                             O((|V|+|E|) * log |V|)
3.       path(v) ← nil
4.    dist(s) ← 0
5.    Q ← ∅   // min-priority queue
6.    INSERT(Q, (s, 0)) // inserts s with key 0
7.    while Q ≠ ∅ do
8.       v ← EXTRACT-MIN(Q) // greedy
9.       for each w ∈ Adj(v) do
10.         if dist(w) > dist(v) + weight(v,w) then
11.            dist(w) ← dist(v)+ weight(v,w)
12.            path(w) ← v
13.            if w ∉ Q then  // old dist(w) was ∞
14.               INSERT(Q, (w, dist(w)))
15.            else
16.               DECREASE-KEY(Q, (w, dist(w)))
```

Figura 1: Pseudocódigo do Algoritmo de *Dijsktra*.

A utilização deste algoritmo de cálculo do caminho mais curto desde o vértice inicial (morada do utilizador) até aos outros nós (pontos de recolha) resulta numa árvore de caminhos ordenada.

A preparação do algoritmo tem complexidade temporal O($|V|$) e consiste na inicialização do campo **dist** a ∞ e **path** a "nil"(correspondente a *nullptr*) em todos os vértices (à exceção da origem).

Posteriormente, a partir do vértice inicial, percorre-se todas as arestas adjacentes e adiciona-se os vértices de destino a uma fila de prioridade, atualizando a dist (distância do vértice anterior somada com o peso da aresta) e o path (vértice anterior), se este vértice ainda não tiver sido visitado, ou se a distância obtida for menor do que o valor prévio de dist.

Este processo tem complexidade temporal O($(|V| + |E|) * log(|V|)$), uma vez que a cada passo - O($|V| + |E|$) - pode ser necessário adicionar, remover ou mover elementos na fila de prioridade - O($log(|V|)$).

Para o segundo caso, ter-se-ia de aplicar uma "translação"ao mesmo algoritmo (uma vez que não se teria de calcular a distância de um nó a todos os outros, mas de um potencial ponto interno da aresta até um potencial ponto interno de outra aresta), tendo que, para todas as distâncias calculadas a partir de cada uma das extremidades dessa aresta (pelo algoritmo referido), somar-se a distância do ponto interno a essa mesma extremidade.

Posteriormente, para implementar a especificidade dos resíduos, seria apenas necessário, em cada ponto de recolha, guardar o(s) tipo(s) de resíduos nele recolhidos.

Após essa implementação, resta ter em consideração o fator capacidade, a ser guardado também em cada ponto de recolha e para cada um dos resíduos recolhidos.

3.2 Novo Modelo de Negócio

O processamento do novo modelo de negócio proposto baseia-se no cálculo do itinerário mais curto partindo da morada do utilizador e destinado a uma central de reciclagem, passando por todas as casas que desejam ver esse tipo de resíduo recolhido.

Este problema assemelha-se ao ***Travelling Salesman Problem*** **(TSP)**[2], pelo que os métodos a aplicar serão semelhantes às resoluções propostas. As três opções ponderadas consistem em:

- **Força bruta:** Esta alternativa consiste no cálculo de todas as opções possíveis e seleção da mais rentável. Obviamente, este é o método mais temporalmente complexo (O($|V|!$)) e, portanto, apesar de garantir um resultado ótimo, é pouco recomendável.
- **Algoritmo do Vizinho mais Próximo:** Este algoritmo baseia-se na ordenação dos pontos, encontrando o ponto não visitado mais próximo do anterior processado. Este algoritmo não garante um resultado ótimo, mas tem complexidade bastante inferior ao anterior (complexidade temporal O($|V|^2$) no pior caso).

- **Algoritmo de *Held-Karp*:** Este é um algoritmo de programação dinâmica baseado na recursividade da distância mínima, no qual para cada vértice, se calcula a distância do vértice inicial até esse passando por todos os vértices de um subconjunto. A sua complexidade temporal é de $O(2^n n^2)$ e o algoritmo garante um resultado ótimo. O pseudocódigo do algoritmo de *Held-Karp* é apresentado na figura seguinte.[3]

```
function algorithm TSP (G, n) is
    for k := 2 to n do
        C({k}, k) := d_1,k
    end for

    for s := 2 to n-1 do
        for all S ⊆ {2, . . . , n}, |S| = s do
            for all k ∈ S do
                C(S, k) := min_m≠k,m∈S [C(S\{k}, m) + d_m,k]
            end for
        end for
    end for

    opt := min_k≠1 [C({2, 3, . . . , n}, k) + d_k, 1]
    return (opt)
end function
```

Figura 2: Pseudocódigo do Algoritmo de *Held-Karp*.

4 Casos de Utilização

A aplicação *Wasteapp* que vamos implementar terá como base uma interface simples de texto com a qual o utilizador poderá interagir. Deste modo, para facilitar a interação, esta terá um conjunto de menus com várias opções a serem disponibilizadas, dividindo-se em 2 menus gerais: 1 menu para o utilizador comum que procura os pontos de recolha mais próximos e 1 menu para o utilizador que irá explorar financeiramente o novo modelo de negócio. Para além disso, será possível guardar a informação de um mapa sob a forma de um grafo através da leitura de ficheiros de texto, no qual incidirão as diferentes funcionalidades da *app*, dentro das quais se destacam:

- Visualização dos dados do grafo e da informação representada por este (vias de trânsito, pontos de recolha, etc.) no *GraphViewer*;
- Ordenação dos pontos de recolha por proximidade ao utilizador e do itinerário ótimo para o modelo de negócio;
- Consulta da informação sobre os pontos de recolha disponíveis(capacidade e tipo de resíduo);
- Verificação da acessibilidade dos pontos de recolha (verificação se as ruas se encontram desimpedidas, sem estarem em obras, por exemplo);
- Ordenação e atribuição das casas dos utilizadores que solicitaram a recolha de resíduos ao domicílio.

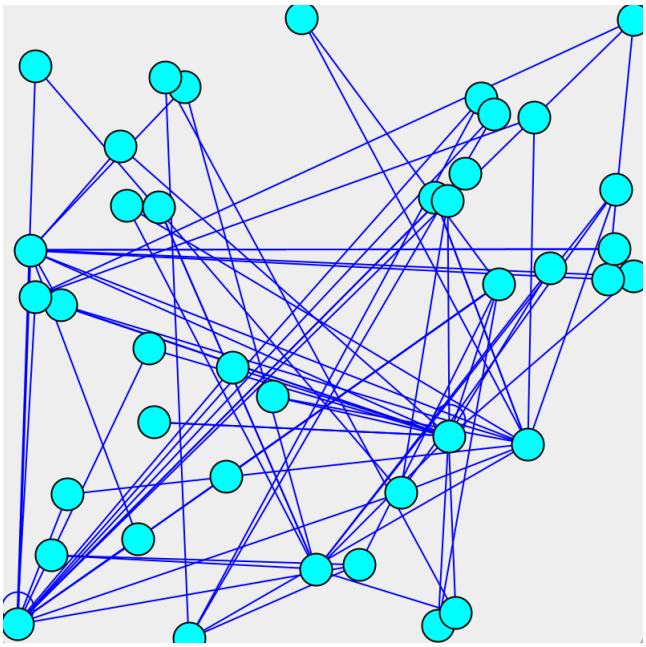

Figura 3: Exemplo de um grafo visualizado no *GraphViewer*.

5 Conclusão

Após um estudo rigoroso do problema que nos foi atribuído, percebemos que existem diversas maneiras para resolver problemas deste género. Depois do aconselhamento por parte da professora das aulas práticas e dos monitores, acreditamos que obtivemos uma abordagem correta para a solução do problema de identificação dos itinerários ótimos.

A nosso ver, a parte que requereu mais tempo foi o ponto a "Perspetiva de Solução"uma vez que necessitou de bastante pesquisa em diversas plataformas de forma a encontrarmos o algoritmo mais apropriado para a solução dos problemas que surgiram, pois serão utilizados diferentes algoritmos tanto na utilização "normal"da aplicação como na perspetiva de novo modelo de negócio que surge com esta.

Por fim, podemos concluir que, após a realização deste relatório, dominamos agora a matéria lecionada em aula que incide na manipulação de grafos com algoritmos, encontrando-nos, assim, preparados para a implementação do que sugerimos na parte do trabalho que se segue.

6 Contribuição

Beatriz Costa Silva Mendes - up201806551@fe.up.pt

- Contribuição: 1/3
- Tarefas:
  - Descrição do Tema;
  - Casos de Utilização;
  - Colaboração na Perspetiva de Solução.

Daniel Garcia Lima Sarmento da Silva - up201806524@fe.up.pt

- Contribuição: 1/3
- Tarefas:
  - Formalização do Problema;
  - Conclusão.

Henrique Manuel Ruivo Pereira - up201806538@fe.up.pt

- Contribuição: 1/3
- Tarefas:
  - Colaboração na Descrição do Tema;
  - Colaboração nos Casos de Utilização;
  - Perspetiva de Solução

7 Referências Bibliográficas

Grupo B, Turma 6, 2016. *Ecoponto: Recolha De Lixo Seletiva (Tema 4) - Parte 1.* Projeto de Concepção e Análise de Algoritmos.

[1]: Docs.google.com. 2020. 2019-20 2S CAL Trabalhos Publicados. Disponível no Moodle.

[2]: Wikipedia Contributors (2019). *Travelling salesman problem.* [online] Wikipedia. Disponível em: `https://en.wikipedia.org/wiki/Travelling_salesman_problem`.

[3]: Wikipedia Contributors (2019). *Held–Karp algorithm.* [online] Wikipedia. Disponível em: `https://en.wikipedia.org/wiki/Held%E2%80%93Karp_algorithm`.

Figura 1: Rossetti, R., Ferreira, L., Cardoso, H. L. e Andrade, F., 2020. *Algoritmos Em Grafos: Caminho Mais Curto (Parte I).*

Figura 2: Wikipedia Contributors (2019). *Held–Karp algorithm.* [online] Wikipedia. Disponível em: `https://en.wikipedia.org/wiki/Held%E2%80%93Karp_algorithm`.

Figura 3: GitHub. (2020). *STEMS-group/GraphViewer.* [online] Disponível em: `https://github.com/STEMS-group/GraphViewer`.