UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Otimização Multi-Objetivo de Caminhos

Duarte da Cruz Melão

Mestrado em Engenharia Informática

Dissertação orientada por:
Professora Doutora Ana Luísa do Carmo Correia Respício

2022

Agradecimentos

Os meus agradecimentos vão para a minha família, especialmente os meus pais e o meu irmão, que me dão apoio todos os dias. Também queria agradecer à minha namorada e à sua família, pelo apoio constante e preocupação para finalização desta dissertação.

Um grande obrigado à aluna Ana Sofia Pereira Nunes, por trabalhar lado a lado com a sua dissertação, a fim da continuação do projeto, como também pela ajuda prestada.

Finalmente queria agradecer à minha orientadora, professora Ana Respício pela sua orientação e correção na realização deste trabalho, uma vez que sem ela, este trabalho não tinha sido realizado.

Resumo

O problema do caminho mais curto é descrito e tratado em inúmeras situações, criando-se algoritmos e aplicativos para resolvê-lo. Esse problema geralmente é abordado quando temos um único objetivo, ou seja, quando queremos encontrar o caminho mais curto entre dois pontos. Para isso, normalmente é utilizado o algoritmo de Dijkstra, um algoritmo genérico para a solução desse tipo de problema. Porém, quando a possibilidade de vários objetivos é adicionada, a complexidade aumenta. Neste caso, não basta ter um único critério para calcular o caminho mais curto, com a ajuda de um algoritmo típico, mas mais critérios, para que possamos então classificar cada caminho de uma forma mais detalhada e obter uma resposta mais ponderada. Esses critérios podem ser causas naturais, altitude, sombra, exposição ao sol, bem como outros tipos de fenómenos, tráfego, tipo de piso, etc.

Assim, com a presença de diversos critérios, é necessário fazer mais comparações entre os caminhos, de forma a poder eliminar os que não interessam. Estas comparações são feitas entre os valores do mesmo critério dos diferentes caminhos, caminhos estes que tenham o mesmo vértice origem e destino.

Desta forma, foi feita uma implementação de um algoritmo multi-objetivo exato, para obtenção do conjunto de soluções ótimas para o caso de uso escolhido. É executado num ponto inicial, fazendo a sua pesquisa de forma local (constrói o caminho iterativamente), guardando todos os caminhos que são potencialmente ótimos, de acordo com os critérios especificados pelo agente de decisão (AD). Considerou-se como caso de uso a aplicação do algoritmo a uma secção do mapa real de Lisboa, tendo como objetivos a minimização da distância, a maximização de proximidade a áreas verdes e a minimização de exposição a poluição (maximização da qualidade do ar), nos caminhos a encontrar.

Palavras-chave: Caminho mais curto; Recomendação de Rotas Pedestres; Agente de Decisão; Problema do Caminho mais Curto Multi-Objetivo; Frente de Pareto

Abstract

The shortest path problem is described and addressed in numerous situations, creating algorithms and applications to solve it. This problem is usually addressed when we have a single objective, that is, when we want to find the shortest path between two points. For this, the Dijkstra algorithm is normally used, which is a generic algorithm for solving this type of problem. However, when the possibility of several objectives is added, the complexity increases. In this case, it is not enough to have a single criterion to calculate the shortest path, with the help of a typical algorithm, but more criteria, so that we can then classify each path in a more detail way and obtain a more weighted answer. These criteria can be natural causes, altitude, shade, sun exposure, as well as other types of phenomena, traffic, floor type, etc.

Thus, with the presence of several criteria, it is necessary to make more comparisons between the paths, in order to eliminate the ones that do not matter. These comparisons are made between the values of the same criterion of different paths, paths that that have the same source and destination vertex.

In this way, an implementation of an exact multi-objective algorithm was made, to obtain the set of optimal solutions for the chosen use case. It is executed at an initial point, doing its search locally (iteratively builds the path), saving all the paths that are potentially optimal, according to the criteria specified by the decision agent (AD). It was considered as a use case the application of the algorithm to a section of the real map of Lisbon, with the objectives of minimizing distance, maximizing proximity to green areas and minimizing exposure to pollution (maximization of air quality), on the paths to find.

Keywords: Shortest Path; Recommendation of Pedestrian Routes; Decision Agent; Multi-Objective Shortest Path Problem; Pareto Front

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AD - Agente de Decisão

POM - Problema de Otimização Multi-objetivo

CP - Conjunto de Pareto

FP - Frente de Pareto

GA - Algoritmos Genéticos

ACO - Otimização baseada em Colónias de Formigas

SA - Arrefecimento Simulado

PS - Enxame de Partículas

TS - Pesquisa Tabu

SS - Pesquisa por Dispersão

NSGA-II - Fast and Elitist Non-Dominated Sorting Genetic Algorithm II

SPEA-2 - Strength Pareto Evolutionary Algorithm 2

MOACO - Multi-Objective Network Optimization based on an Ant Colony Optimization

VNS – Pesquisa de Vizinhança Variável

Capítulo 1 Introdução

1.1 Motivação

Nos dias que correm, torna-se importante a deslocação da população, quer seja para o emprego, para a escola, para lazer, ou para outra atividade. Com os avanços da tecnologia, cada vez mais surgem sistemas para recomendação de rotas, no entanto, esta é uma área ainda pouco desenvolvida, a nível dos critérios a considerar para a recomendação. Para isso, é necessário continuar a melhorar e a criar novas ferramentas, para que se consiga optar pelas melhores escolhas, isto é, escolher dentro dos diferentes percursos disponíveis, o mais adequado para a situação em questão. Quando falamos em mais adequado, podemos ter uma grande diferenciação em quais os melhores/piores fatores para cada um de nós. Rapidez e menor custo monetário poderão ser fatores decisivos para uma pessoa, como uma menor poluição e um trajeto numa zona florestal poderão ser decisivos para outra. Uma vez que estes fatores podem ser conflituosos entre si, temos de ter um compromisso entre os mesmos para chegarmos a uma solução ideal, que seja de agrado ao agente de decisão (AD).

Os investimentos têm vindo a aumentar nesta área, uma vez que, não só beneficia diretamente a fluidez e rapidez da circulação da população, ao se conseguirem movimentar de uma forma mais eficiente, como também indiretamente, de forma a haver uma maior dispersão automóvel, um maior aproveitamento do espaço verde, entre outros.

Atualmente, existem ferramentas que tentam preencher essa lacuna, como é o caso do Moovit[1], Waze[2], Google Maps[3], entre outras. Apesar de já existirem ferramentas capazes de produzir recomendações de percursos em mapas, muitas vezes não dão poder de especificação das preferências do lado do AD, de forma a decidir quais os pontos positivos e negativos num trajeto. Consequentemente, acaba por não se conseguir encontrar as melhores alternativas para cada pessoa, sendo isto um ponto crucial a melhorar.

A motivação para este trabalho é contribuir para uma melhor solução no âmbito do problema do caminho mais curto com diversos objetivos, isto é, dar uma melhor

[1] https://moovitapp.com/
[2] https://www.waze.com/pt-BR/live-map/
[3] https://www.google.pt/maps/

proposta para que seja usada no quotidiano das pessoas, e que estas consigam escolher rotas de acordo com as suas preferências.

1.2 Problema

Habitualmente, é comum existir problemas na escolha de rotas pedestres, onde não se tem apenas um critério a ponderar como objetivo para resolução do problema, mas vários, normalmente conflituosos. A este tipo de problemas dá-se o nome de problema de otimização multi-objetivo (POM) com caminhos. Estes problemas são de difícil resolução devido à existência de soluções incomparáveis, por exemplo, tendo presente um problema de otimização com critérios de distância e custo, com uma solução com melhor distância que outra mas com pior custo, acontece não existir uma solução superior para apresentar ao AD. Deste modo, é necessário apresentar ao mesmo não uma, mas várias soluções ótimas, de forma a ter do seu lado um maior poder de escolha. Em muitos casos, acaba-se por fazer uma média ponderada dos valores dos critérios, a fim de se conseguir resolver o problema de uma forma mais simplificada. Para além disso, existe a possibilidade de se escolher os pesos de cada objetivo nessa média, de forma não adequada, o que pode levar a erros. A resolução do problema é transformada, tendo em conta única e exclusivamente um objetivo, obtendo-se uma só solução. Consequentemente, é levado para o resultado, uma decisão final que pode conter informação errada.

Múltiplos trabalhos foram feitos na área dos problemas multi-objetivo, no entanto várias lacunas continuam presentes. Uma das principais dificuldades deve-se ao facto de termos presente inúmeros caminhos ótimos, o que faz com que tenha de existir um outro fator de decisão, e não apenas o "critério de otimalidade de Pareto", que será exposto posteriormente. Temos exemplos em sistemas de recomendação de transporte, aquando presentes diversos caminhos ótimos, é escolhido um destes, não dando ao AD qualquer poder decisivo de determinar o/os objetivo/os a que dará mais ênfase e os que dará menos.

Outra das dificuldades caracteriza-se por encontrar a melhor ou as melhores soluções possíveis em tempo computacional razoável, o que por sua vez faz com que se tente encontrar soluções perto do ótimo, em vez das soluções ótimas. Desta forma, limita-se o espaço de pesquisa e procura-se as melhores soluções neste espaço, isto é, tem-se o compromisso de não se obter as soluções ótimas, mas sim aproximadamente ótimas.

Este tipo de problemas são classificados como NP-difícil, uma vez que a eficiência da resolução dos mesmos depende do número de caminhos ótimos obtidos,

que cresce de forma exponencial no número de vértices presentes no grafo (Martins, 1984).

1.3 Objetivos

Para resolver problemas uni-objetivo, quer-se encontrar uma solução ótima, isto é, uma solução admissível que otimize a função objetivo, podendo existir várias soluções alternativas com esse valor para a mesma função. Na resolução de problemas multi-objetivo, o mesmo princípio não se aplica, uma vez que, tendo uma solução admissível que otimize um dos objetivos, existe a possibilidade de deterioração dos restantes.

Por isso, existe não uma solução ótima, mas sim múltiplas soluções igualmente ótimas. A estas soluções damos o nome de não-dominadas (também chamadas de eficientes ou ótimas de Pareto). No contexto do trabalho, estas soluções serão designadas de caminhos não-dominados, caminhos estes com uma origem e um destino, previamente escolhidos pelo AD.

Desta forma, este trabalho integra-se num projeto em que se pretende desenvolver uma aplicação móvel para apoiar a decisão na escolha de rotas pedestres multi-objetivo.

Foi escolhido como caso de uso a otimização de caminhos num mapa da região de Lisboa, uma vez que, o trabalho em (Nunes, 2022) que resolve o problema uni-objetivo e o presente estudo, em conjunto, irão contribuir num trabalho futuro de forma a desenvolver um sistema de recomendação móvel.

As contribuições deste trabalho são:

- Realização de um estudo sobre metodologias do caminho mais curto, nomeadamente algoritmos multi-objetivo.

- Processamento dos dados recolhidos para o caso de uso, a fim da construção de um grafo direcionado conexo.

- Adaptação e desenvolvimento do algoritmo multi-objetivo proposto em (Martins, 1984), capaz de encontrar todas as soluções de Pareto no grafo acima mencionado.

- Avaliação da execução do algoritmo, aplicando-o num caso real usando parte de uma secção do mapa de Lisboa, e comparação das diferenças nos resultados obtidos por (Nunes, 2022), face a múltiplos objetivos.

1.4 Organização do Documento

Este documento está organizado da seguinte forma:

Capítulo 1 (Introdução) - Apresenta o problema e os objetivos que se planeiam atingir com a realização da dissertação.

Capítulo 2 (Conceitos) - Apresenta conceitos fundamentais para a perceção do problema.

Capítulo 3 (Trabalhos Relacionados) - Apresenta a informação necessária para a compreensão de problemas uni-objetivo e multi-objetivo, mostrando trabalhos anteriores relacionados com o tema.

Capítulo 4 (Caso de Uso e Resultados) - Apresenta uma abordagem ao caso de uso e o seu tratamento com soluções propostas.

Capítulo 5 (Conclusão) - Apresenta as conclusões retiradas após estudadas diversas aplicações de soluções para o mesmo problema.

Capítulo 2 Conceitos

Neste capítulo serão introduzidos na secção 2.1 conceitos básicos para a perceção do problema multi-objetivo, explicado na secção 2.2.

2.1 Conceitos Básicos

Para contextualização do trabalho, primeiro há que mencionar algumas definições. Começamos por falar em grafos, vértices e arestas. Um grafo é uma estrutura finita que contém dois conjuntos, vértices e arestas, representado na forma G = (V,A), onde V é o conjunto de vértices e A o conjunto de arestas.

Uma aresta liga então dois vértices, a=(v,w), v e w ∈ V. O vértice v diz-se adjacente a outro, quando os dois formam uma aresta da forma (v,w). Uma aresta (v,w) é adjacente a outra, quando o vértice da entrada da primeira aresta é o mesmo vértice da saída da segunda aresta, representado na forma (v,w) e (w,z). Esta aresta poderá ser dirigida, ou seja, composta com uma orientação, ou não dirigida. Se for dirigida, a sua representação será feita como um par ordenado de vértices, traduzindo-se pela saída do vértice v e na entrada do vértice w, enquanto que não dirigida (não orientada), será representada por um par não ordenado, sendo neste caso (v,w) ou (w,v) a mesma aresta (Kleinberg & Tardos, 2006).

Com esta informação podemos ter dois tipos de grafos; grafos dirigidos, constituídos apenas por arestas dirigidas, e grafos não dirigidos, com apenas arestas não dirigidas. Um caminho p num grafo dirigido G = (V,A), com origem em u e destino em w é uma sequência de arestas (u_0,u_1), (u_1,u_2), …, (u_{n-1},u_n), sendo $u_0 = u$ e $u_n = w$, onde $u_0,u_1,\ldots$ pertencem a V e (u_0,u_1), $(u_1,u_2),\ldots$ pertencem a A. Um grafo também pode ser conexo ou desconexo, de acordo com a interligação dos vértices e arestas. Se existir sempre um caminho entre qualquer par de vértices do grafo, o mesmo designa-se por conexo, caso contrário é desconexo (Sedgewick & Wayne, 2011).

A prática do uso de grafos é importante, na medida em que, pode-se representar inúmeros parâmetros, como custos, distâncias, tempos relativos a trânsito, fluxo, ou também capacidades de carga, de memória, entre outros, de uma forma apelativa e de fácil compreensão.

2.2 Problema Multi-objetivo

Para a resolução de problemas de caminho mais curto uni-objetivo, pretende-se descobrir o caminho de valor mínimo. Para descobrir este caminho é necessário que cada uma das arestas constituintes do grafo tenha um valor, a que se dá o nome de comprimento, ou peso. O caminho de valor mínimo consiste na sequência de arestas onde o custo total das mesmas é o menor, contendo o vértice inicial na origem da primeira aresta e o vértice final no destino da última aresta, como descrito por (Kleinberg & Tardos, 2006), da forma:

$$\sum c_{ij} \text{ , com } i,j \in p, \quad (1)$$

sendo i e j uma aresta na forma (i,j), com um custo associado c_{ij}, pertencentes a um caminho p.

A função objetivo deste problema passará por minimizar a soma dos custos associados ao caminho formado, caminho este começando no vértice inicial e acabando no vértice final.

Para problemas em que se pretende otimizar mais que um objetivo, temos uma função objetivo como um vetor de várias funções objetivo, tantas quanto o número de objetivos presentes. Deste modo, associado às arestas, podemos ter um ou vários k atributos que serão utilizados para calcular os valores dos diferentes m objetivos. Então, para cada aresta (i,j) é associado um vetor custo,

$$c(i,j) = c_{ij} = (c^1_{i,j}, \dots, c^k_{i,j}), \text{ de tal modo que } c^{\ell}_{i,j} \geq 0, \forall\, \ell \in \{1, \dots, k\}. \quad (2)$$

Assim, a minha função objetivo vetorial f será da forma:

$$f(p) = (f_1(p), \dots, f_m(p)), \text{ onde } f_{\ell}(p) = f(c_{(i,j)}), \text{ com } (i,j) \in p, \forall\, \ell \in \{1, \dots, m\}. \quad (3)$$

2.2.1 Dominância

Um caminho admissível x diz-se dominado por outro caminho admissível y, se e só se o caminho y for melhor em pelo menos um dos objetivos, comparativamente ao caminho x, sem que se deteriore qualquer um dos restantes, isto é,

$$\exists\, \ell,\ 1 \leqslant \ell \leqslant m : f_{\ell}(y) < f_{\ell}(x) \text{ e } f_i(y) \leqslant f_i(x)\ \forall\, i,\ 1 \leqslant i \leqslant m \wedge i \neq \ell. \quad (4)$$

(Demeyer et al., 2013) menciona no seu artigo, “A path is called Pareto optimal if no objective can be ameliorated without deteriorating another objective”. Mais detalhadamente, é explicado este conceito em uma definição, por (Paixão & Santos, 2013), sobre caminhos não-dominados:

Seja p um caminho em $P_{i,j}$, i, j $\in \mathbb{N}$, onde $P_{i,j}$ é o conjunto de todos os caminhos do vértice i ao vértice j. Se não existir um caminho q em $P_{i,j}$ de tal modo que $q <_D p$, isto é, se não existir um caminho q que domine p, ou que p é dominado por um caminho existente q, então p é designado de não-dominado, eficiente ou caminho ótimo de Pareto.

Na Figura 2.1 é apresentado o espaço dos objetivos e as respetivas zonas de dominância, a que domina e a dominada, a partir de uma solução x. Também é representada a zona a tracejado, contendo soluções incomparáveis com a solução x.

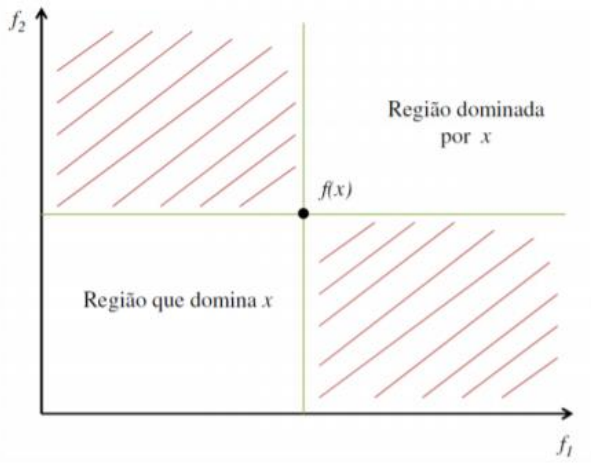

Figura 2.1 Representação do conceito de dominância com 2 objetivos no espaço dos objetivos (Raupp, 2012).

O nosso objetivo será dar a conhecer ao AD um conjunto variado de soluções, de forma a que não exista nenhuma solução que domine qualquer uma pertencente ao conjunto.

Um POM de minimização pode ser descrito da forma (Raupp, 2012):

$$\text{(POM) Min } f(x) = \{f_1(x), \dots, f_m(x)\} \quad (5)$$

$$\text{s.a } x \in X$$

tendo um vetor $x = \{x_1,...,x_n\}$ de variáveis de decisão de tamanho n, no espaço de pesquisa X, queremos descobrir um conjunto de vetores $x^* \in X$ que não sejam dominados por nenhum outro vetor, tendo presente as m funções objetivo $f(x^*) = \{f_1(x^*), \dots, f_m(x^*)\}$, onde $f_\ell : \mathbb{R}^n \rightarrow \mathbb{R}$, $1 \leqslant \ell \leqslant m$, e $m \geq 2$. Desta forma, os vetores x^*

irão produzir valores para todas as m funções objetivo, de forma a que não se consiga arranjar melhores vetores, fabricando assim todas as soluções ótimas que formarão um conjunto no espaço de soluções. No contexto do problema, este conjunto também é designado de conjunto de Pareto (CP).

2.2.2 Solução Ótima de Pareto e sua Fronteira

Temos presente uma solução ótima de Pareto, quando não é dominada por nenhuma solução no nosso espaço de pesquisa X. Não poderá ser melhorada em nenhum dos objetivos sem piorar em pelo menos um dos outros. O problema passa por descobrir o conjunto das soluções ótimas de Pareto em X, de forma a criar no espaço dos objetivos, a fronteira de Pareto.

Considerando de novo o nosso espaço dos objetivos de duas dimensões (Figura 2.1), após calculado o nosso conjunto ótimo de Pareto, é possível verificar uma fronteira, fronteira esta que nos demonstra as melhores soluções que o AD poderá escolher. Como observamos no exemplo da Figura 2.2, todos os seis pontos incluídos na fronteira (CP), dominam as restantes 4 soluções (C, D, E e F). Como resultado, nota-se que não existe relação de dominância entre o nosso conjunto ótimo de Pareto, uma vez que, trocando de uma solução dentro deste conjunto para outra, implica o melhoramento num objetivo mas o pioramento no outro. Logicamente, haverá sempre um "trade-off" entre os objetivos. Por consequência, sabemos que todas as soluções dentro desta frente de Pareto (FP), são as melhores.

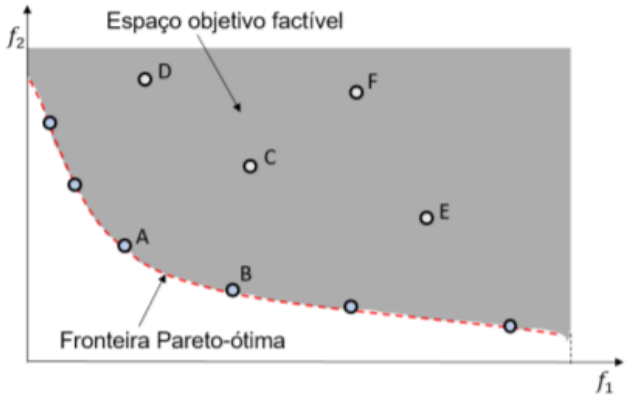

Figura 2.2 Exemplo de fronteira ótima de Pareto (Abreu & Pereira, 2019).

Esta fronteira, contendo soluções igualmente ótimas incomparáveis, é então apresentada ao AD, sendo que o mesmo terá a responsabilidade de escolher qual implementar.

Capítulo 3 Trabalhos Relacionados

O tratamento de problemas que envolvem caminhos é um tema amplamente estudado, uma vez que a resolução destes problemas faz parte do dia a dia das pessoas. Um dos tipos de problemas mais conhecidos é designado de problema do caminho mínimo, ou mais curto. A resolução deste passa por, num leque variado de caminhos, descobrir qual a travessia mais curta entre dois vértices do grafo, que compreende o caminho de menor distância, menor custo, entre outros. O resultado será o somatório dos pesos de cada uma das arestas, presente nesse caminho.

Existem várias variantes nestes problemas, entre elas:

- Problema com uma só origem, que passa por descobrir o caminho mínimo, a partir de um vértice origem, para todos os outros pertencentes ao mesmo grafo.

- Problema com um só destino, que passa por descobrir o caminho mínimo de todos os vértices do grafo, para um vértice final.

- Problema de origem-destino, que se baseia na descoberta do menor caminho de um vértice inicial para um vértice final, do mesmo grafo.

- Problema de todos os pares do grafo, que se baseia em determinar o menor caminho entre todos os pares de vértices do grafo.

Para este estudo, o foco será somente no problema de origem-destino, sendo o mais indicado para o objetivo proposto.

O estudo deste tipo de problemas começou a ser feito na primeira metade do século XX, e estando em constante evolução, continua a ser um tópico de alta importância. É intensamente estudado em diversas áreas e aplicado em muitos problemas ocorrentes em grafos de computadores, telecomunicações, transporte, entre muitos outros (Kleinberg & Tardos, 2006) Barrico, 1998).

Irá-se destacar em seguida, alguns algoritmos conhecidos e especializados para a resolução do problema do caminho mais curto.

3.1 Algoritmos Uni-Objetivo

Temos o algoritmo de Dijkstra, um algoritmo que soluciona o problema do caminho mais curto num grafo, com arestas de peso não negativo, entre 2 pontos (Figura 3.1). Este algoritmo está presente em (Nunes, 2022), de forma a se poder obter as soluções para cada objetivo individualmente, presentes na região de Lisboa. Para o

seu funcionamento, é necessário o uso de etiquetas, para saber a ordem de pesquisa dos vértices e posteriormente alcançar o caminho mais curto. Uma vez que este algoritmo apenas possui um objetivo, cada etiqueta terá apenas um valor, recorrente à distância mais curta encontrada até ao momento de pesquisa, do vértice inicial s, para um outro vértice a verificar i, ou seja, ao caminho mais curto entre s e i.

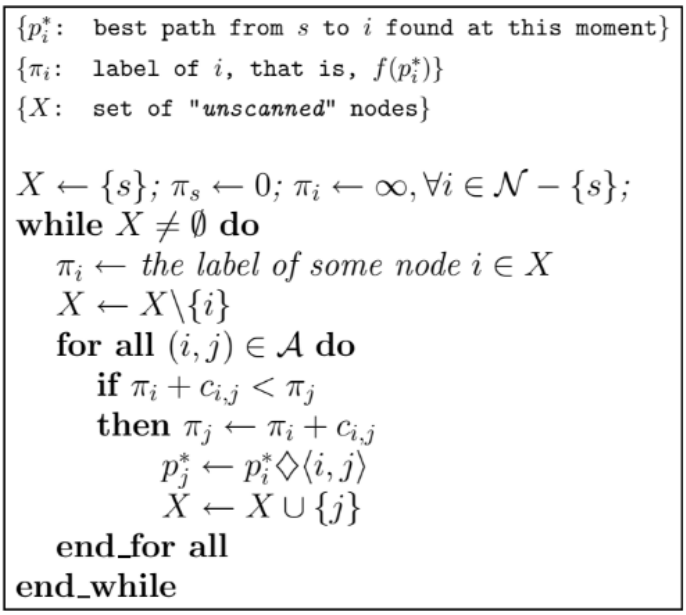

Figura 3.1 Algoritmo de Rotulação de Dijkstra (Paixão & Santos, 2013).

De seguida, é explicado este algoritmo de uma forma mais detalhada. Parte-se de um estado inicial, onde temos num conjunto X apenas o vértice inicial. Em cada um dos vértices i do grafo iremos ter uma etiqueta π_i, etiqueta esta que tem valor infinito, que irá sendo ajustado à medida que se descobre valores menores. No entanto, a etiqueta inicial π_s, terá sempre valor nulo, logicamente por não existir um distanciamento entre o vértice inicial e o mesmo.

Sempre que um vértice é inspecionado, é retirado de X. No entanto, se um outro vértice adjacente ao inspecionado validar a condição dentro do ciclo, explicada de seguida, então é adicionado a X. Dá-se a condição de paragem deste algoritmo quando todos os vértices são verificados, isto é, quando X fica vazio. Uma vez que estamos a aplicar a técnica de label setting (escolhe-se as etiquetas por ordem crescente de valor), é escolhida a etiqueta de menor valor (menor distância), entre todas as etiquetas dos restantes vértices pertencentes a X. Para cada aresta (i,j), adjacente ao vértice i, é feita a soma da etiqueta com a distância da mesma ($\pi_i+c_{i,j}$). Caso esta soma seja menor que o valor da etiqueta π_j, então o valor deste é atualizado com a soma anterior, como também

é adicionado o vértice j ao conjunto X. Desta forma, o melhor caminho encontrado até ao momento, desde o vértice inicial até j (p_j^*), será formado pela concatenação do melhor caminho encontrado até ao momento, do vértice inicial até i (p_i^*) mais a aresta (i,j), ou seja, ($p_i^* \Diamond (i,j)$).

De acordo com (de Carvalho, 2008), podemos verificar um exemplo da execução deste algoritmo, isto é, tendo um grafo composto com cidades e a ligação das mesmas, calcula-se dentro dos caminhos possíveis, o caminho mais curto entre a cidade origem (A), e o resto das cidades. A Figura 3.2 representa a evolução do grafo com a computação do Dijkstra.

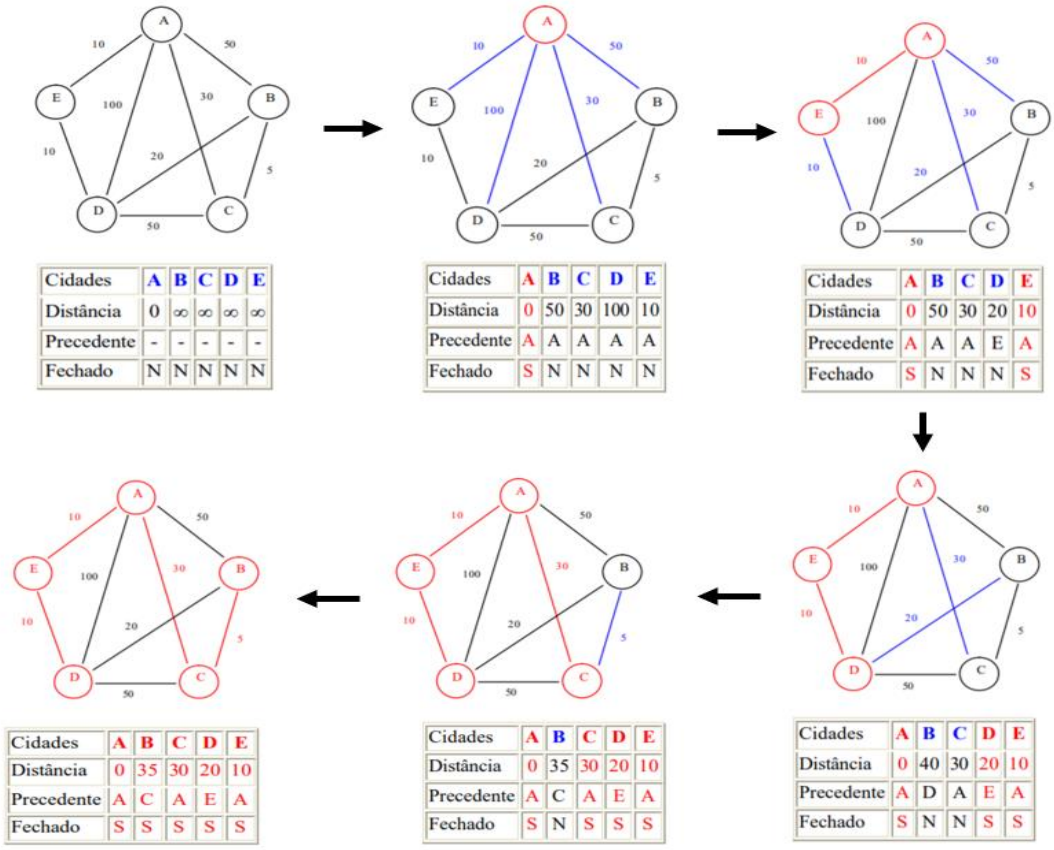

Figura 3.2 Evolução do grafo resultante da aplicação do algoritmo de Dijkstra (de Carvalho, 2008).

Como é possível observar, em cada iteração do algoritmo, os valores das arestas vão atualizando para distâncias mais baixas, até se obter o caminho mais curto, entre a cidade A e as restantes. A condição de paragem é executada quando todas as cidades forem "fechadas", isto é, quando todos os vértices forem visitados/inspecionados. Sempre que é encontrada uma distância inferior à armazenada na tabela, procede-se à substituição da etiqueta e é alterada a cidade precedente. Este algoritmo consegue ser

eficiente para o problema em questão, com complexidade O(|m|+|n| log|n|), sendo n o número de vértices ou vértices e m o número de arestas presentes no grafo.

Outro algoritmo muito conhecido é o algoritmo de Bellman-Ford (Richard, 1958), que consegue não só resolver o problema do caminho mais curto com arestas de peso positivo, como também com peso negativo. Enquanto que o Dijkstra é um algoritmo greedy (faz uma procura local de forma a atingir um ótimo local), este tem uma abordagem de programação dinâmica, sendo que a sua pesquisa é feita de forma ascendente, calculando primeiro as distâncias mais curtas dos vértices com uma aresta de distância, seguindo-se dos vértices com duas arestas, e por aí adiante. Este algoritmo rotula os vértices da mesma maneira que o de Dijkstra, diferenciando-se por possuir apenas etiquetas permanentes quando a condição de paragem é atingida, terminando a execução do algoritmo, enquanto que no de Dijkstra existem etiquetas permanentes, mesmo durante a execução do ciclo, de forma a atingir o ótimo local.

Comparando estes dois algoritmos, o de Dijkstra continua a ser executado em menor tempo, quando se tem presente apenas arestas positivas, sendo apenas utilizado o de Bellman-Ford para problemas que contenham arestas de peso negativo (AbuSalim et al., 2020). Isto acontece porque, não só o algoritmo de Bellman-Ford pode ter de visitar um vértice mais do que uma vez, o que não sucede no de Dijkstra, como também realiza sucessivas aproximações das distâncias até chegar a uma solução final, ao contrário de obter a solução de uma só vez.

Temos também o algoritmo de Floyd-Warshall (Floyd, 1962), focado na variante da descoberta do caminho mínimo entre todos os pares de vértices do grafo. Criado em 1962, é um algoritmo que trabalha tanto com arestas de valor positivo como negativo, usando duas matrizes, uma matriz M de tamanho n.n, sendo n o número de vértices do grafo (guarda o caminho mínimo entre cada par de vértices descoberto ao momento), e outra matriz T para descobrir os vértices intermédios de cada par de vértices. Assim, com o uso de T, é possível ir atualizando a matriz M, de forma a melhorar o caminho de um par e consequentemente, no final do algoritmo, se obter o caminho mais curto do par. Este algoritmo é muito eficaz, uma vez que poupa no uso de recursos e tempo, quando comparado com outros algoritmos, como é o caso do Dijkstra, que neste caso, teria de ser executado tantas vezes quantas o número de pares de vértices presentes.

3.2 Algoritmos Multi-Objetivo

Os algoritmos mencionados na secção 3.1 são eficientes, no entanto, com a introdução de múltiplos critérios a otimizar, tornou-se necessário desenvolver novos algoritmos que permitam lidar com o aumento da complexidade. Tornou-se

imprescindível a criação de técnicas que considerassem as preferências do AD. Os métodos clássicos exatos foram sendo cada vez menos usados, devido à maior complexidade dos problemas, não garantindo a resolução dos mesmos em tempo útil.

3.2.1 Meta-Heurísticas

Nas últimas décadas, foram desenvolvidos diversos algoritmos meta-heurísticos para resolver problemas deste tipo. A necessidade da criação destes algoritmos foi pelo simples facto de não se conseguir atingir o ótimo para instâncias de grandes dimensões, com os algoritmos exatos existentes. Desta forma, faz-se uma aproximação do ótimo, com estratégias de alto nível baseadas em conceitos e princípios já conhecidos (heurísticas), que são capazes de limitar a forma como procuram o espaço de decisões, de modo a considerar certas características das soluções encontradas para explorar novas regiões promissoras (meta).

Estes algoritmos caem em diferentes categorias, consoante o tipo de iterações que fazem, a transformação da informação, a forma como atingem o resultado final, etc. Temos algoritmos genéticos (GA), baseados em colónias de formigas (ACO), arrefecimento simulado (SA), enxame de partículas (PS), pesquisa tabu (TS), pesquisa por dispersão (SS), meméticos, entre muitos outros.

A criação dos GA baseiam-se nos princípios da seleção natural, criado por Charles Darwin (Darwin, 1859), durante o século XX. A seleção natural processa-se em todos os organismos reprodutores, onde as características favoráveis à sobrevivência são passadas na descendência, de geração em geração, tornando-se características comuns, enquanto que características que desfavorecem os mesmos organismos tornam-se obsoletas e incomuns, na passagem para a descendência.

John Henry Holland (Holland, 1992) foi o primeiro a ser bem sucedido na criação de algoritmos evolutivos genéticos, baseados nestes conceitos. Foram incorporados nos mesmos, fenómenos como, **avaliação**, **seleção**, **recombinação** e **mutação**. Para isso, é necessário ter presente o conceito de cromossoma e população. Um cromossoma é nada mais que uma representação de uma solução do problema. Uma população é um conjunto de cromossomas, que irão ser testados e comparados, de forma a que os genes presentes nos melhores cromossomas sejam passados para as gerações futuras. Assim, inicia-se o algoritmo com uma população de cromossomas:

- É feita a **avaliação**, entre todos os cromossomas, para que cada um destes tenha um valor, valor este designado de fitness (função avaliadora de soluções), capaz de dizer-nos a priori, se uma solução é melhor que outra, ou não.

- De seguida, é feita uma **seleção**, de modo a que os melhores cromossomas possam sobreviver, enquanto que os de pior fitness desapareçam da população.

- Posteriormente, é iniciada a etapa de **recombinação**, feita a partir dos cromossomas da etapa anterior. Extrai-se dois ou mais cromossomas, de forma a parti-los em secções, e uni-los com estas mesmas, para que se obtenha, possivelmente, melhores soluções. A ideia deste passo é conseguir criar cromossomas, que não sejam idênticos a nenhum pai. Esta etapa é executada a partir de uma probabilidade, conhecida antes do início do algoritmo.

- Finalizando, procede-se para a **mutação**, que serve como passo para manter a diversidade genética entre as populações. Em cada cromossoma pós-recombinado, é feita uma alteração em parte da sua constituição, de forma aleatória, para que se consiga explorar uma maior parte do espaço de pesquisa. Acabando este passo, a nova população elitista de cromossomas é passada para a próxima geração para que seja avaliada de novo. Para se proceder à mutação, é usada uma probabilidade para evitar acabar com uma população uniforme, sem características para evoluir.

Estes quatro passos formam um ciclo, que terminará quando a condição de paragem for satisfeita (Figura 3.3).

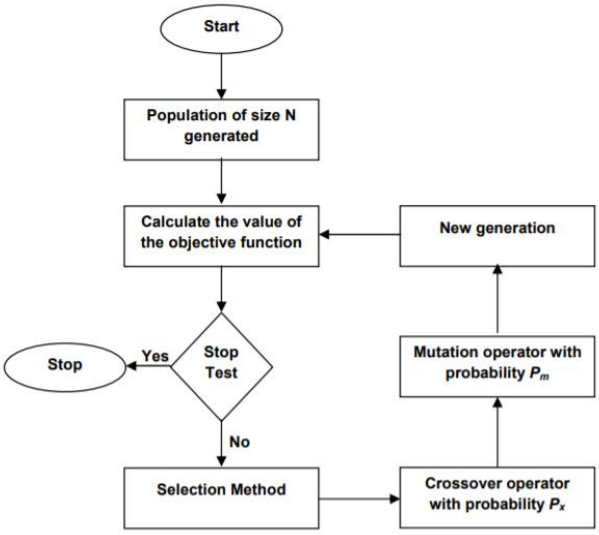

Figura 3.3 Esquema representante do funcionamento de um GA (Abdoun et al., 2012).

Atualmente, existem diversos algoritmos genéticos para problemas de otimização multi-objetivo, no entanto, dois destacam-se dos demais, por serem mais eficientes em diversos casos. Estes são; o Fast and Elitist Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) e Strength Pareto Evolutionary Algorithm 2 (SPEA-2) (Zitzler et al., 2001).

Os mesmos atuam de forma distinta na etapa de avaliação. O algoritmo SPEA-2 calcula o fitness de cada um dos cromossomas, baseando-se na distância euclidiana de cada um destes para os demais. O algoritmo NSGA-II calcula o rank de cada um dos cromossomas com um método designado de profundidade de dominância, criando várias frentes, de forma a que todos os cromossomas da frente 1 são melhores que os da frente 2, os da frente 2 melhores que os da 3, e assim sucessivamente. Ambos os algoritmos têm em atenção a preservação da diversidade das populações, de forma a que se consigam soluções que preencham o espectro do espaço de decisões.

Na seleção, ambos utilizam a Binary Tournament Selection, uma vez que comparam as soluções, aos pares, de forma a conseguir uma pool de cromossomas, chamada de mating pool. Esta mating pool irá ser constituída pelos melhores cromossomas, que iniciarão a etapa de recombinação para que possam passar para a descendência as suas características.

Ao nível da recombinação, ambos usam o SBX Crossover, que produz pesquisa local combinada com pesquisa aleatória. No SPEA-2 esta última pesquisa é feita perto dos cromossomas pai, enquanto que no NSGA-II é feita na frente pertencente ao cromossoma. Na Figura 3.4 é possível verificar a atuação deste tipo de recombinação.

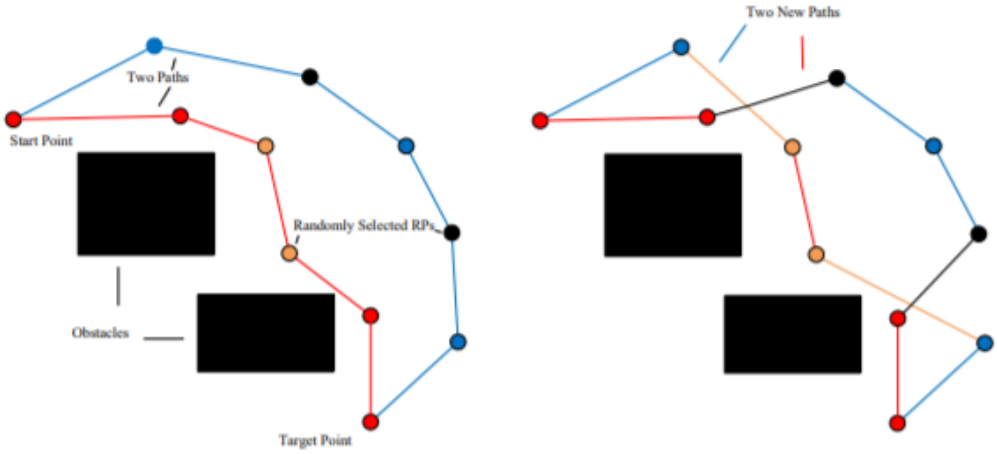

Figura 3.4 Etapa de crossover. Os dois pontos amarelos são os nós escolhidos aleatoriamente do cromossoma da esquerda e os dois pretos são os escolhidos aleatoriamente da direita. Os segmentos são então trocados entre estes pontos, de forma a formar dois novos cromossomas (Xue, 2018).

A mutação é igual nos dois algoritmos, designada de mutação polinomial. Como se pode observar pela Figura 3.5, este tipo de mutação consegue obter soluções mais vastas, quando comparadas com uma mutação linear. É também fácil de verificar que com o uso de mutação polinomial, consegue-se atingir soluções mais próximas do nosso CP.

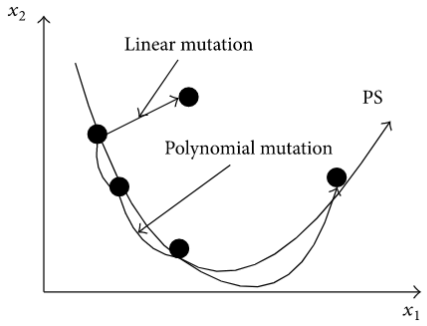

Figura 3.5 Comparação de mutação polinomial com mutação linear (Zhao et al., 2016).

Na passagem da população para a geração seguinte, os algoritmos diferem. O SPEA-2 considera primeiro passar os cromossomas não dominados e depois os cromossomas dominados com melhor fitness. O NSGA-II pondera passar os cromossomas pela ordem das frentes, primeiro os da frente 1, seguindo os da frente 2, por aí adiante, até preencher a população.

Em termos computacionais, o SPEA-2 acaba por ser um pouco mais complexo, no entanto, em alguns casos, acaba por superar o NSGA-II em termos de performance (King et al., 2010).

Outra categoria de algoritmos muito conhecida, criada por Maresta Dorigo, é designado de Ant Colony Optimization (ACO), que tem como propósito simular o comportamento de uma colónia de formigas. Para encontrar alimento da forma mais rápida e eficaz, compartilham a informação sobre os caminhos efetuados num grafo, de maneira a revelar quais os melhores caminhos. Cada formiga utiliza uma componente, designada de feromona, complementada com o uso de informação heurística, em cada aresta do grafo, para se movimentar pelos vértices pertencentes ao mesmo. Assim, toda a formiga que passar por arestas semelhantes, irá juntar à sua informação processada, informação de outras formigas. Nos locais do grafo onde existir mais feromona, maior probabilidade de transição de formigas haverá (Dorigo, 1992), (Maniezzo et al., 1996).

Este último, pertence aos métodos heurísticos probabilísticos de pesquisa local, uma vez que a sua execução é tomada partindo da vizinhança, com apoio de meta-heurísticas, de forma a que se chegue a soluções eficientes, com alguma rapidez, a partir de estruturas mais ou menos genéricas (Hashimoto, 2004). Em muitos casos, este tipo de algoritmos pode ser mais eficiente, uma vez que, ao construir o grafo com o uso de um algoritmo ACO, o espaço de pesquisa pode-se tornar menos redundante.

Com a presença de vários objetivos, foram feitas implementações de algoritmos com a meta-heurística ACO, com acrescento de diversas otimizações, de forma a aumentar a robustez e diminuir o tempo computacional dos mesmos.

Em (Ning et al., 2018) conseguimos ver um variado leque de algoritmos Multi-Objective Network Optimization based on an Ant Colony Optimization (MOACO). Todos eles baseiam-se numa estrutura básica, vista na Figura 3.6.

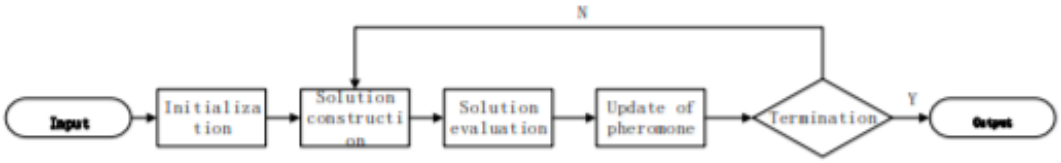

Figura 3.6 Esquema básico de um algoritmo ACO (Ning et al., 2018).

Segundo a mesma, é possível verificar a **inicialização** do algoritmo com a introdução dos parâmetros de input e a informação respetiva das heurísticas e da feromona. Parte-se a seguir para a **construção da solução** onde, para cada formiga f, constrói-se uma nova solução, com o uso de uma probabilidade. Esta probabilidade apenas servirá para resolver o subproblema da solução da formiga f, com o uso das suas heurísticas e informação da feromona. Assim teremos tantos subproblemas quanto o número de formigas.

Depois, passa-se por fazer uma **avaliação da solução** obtida no passo anterior. Após terem sido avaliadas todas as soluções de todas as formigas, guarda-se as soluções não-dominadas e apaga-se as dominadas. De seguida, é feita uma **atualização da feromona** numa matriz, relativamente à informação retida nas soluções previamente construídas. A feromona presente nos arestas relativos a soluções não-dominadas aumentará. Por fim, para a **terminação** do algoritmo, é preciso verificar se a condição de paragem do problema em específico foi atingida. Se sim, é dado como output o conjunto de soluções não-dominadas, se não, retorna-se ao passo de construção da solução.

Estes passos variam consoante as características do algoritmo. Podemos ver as variantes que este pode tomar em (Ning et al., 2018), desde a forma como as colónias são formadas, a quantas matrizes são usadas para armazenamento de informação. As duas maiores diferenças nestes algoritmos são: determinar o conjunto de soluções baseados nas relações de dominância entre cada solução, construídas por cada uma das formigas; ou decompor o problema em diversos subproblemas, tantos quanto o número de objetivos presentes no problema original. Todos estes são resolvidos simultaneamente, onde cada um portará uma solução ótima para o objetivo correspondente.

Algoritmos de arrefecimento simulado (SA) são muito comuns em problemas de caminho mais curto, isto porque são muito fáceis de implementar e geralmente retornam uma boa solução. A utilização de arrefecimento como método de otimização começou a ser usado nas décadas de 70 e 80 (Kirkpatrick et al., 1983). Os mesmos baseiam-se no processo térmico de arrefecimento, usado na metalurgia, para obter estados de baixa energia quando manuseando um sólido, de forma a moldá-lo da maneira pretendida. Aumenta-se a temperatura do mesmo, para que depois se faça um controlo na descida da temperatura, controlo este feito de forma a que os átomos do material se organizem numa estrutura uniforme, isto é, se consigam movimentar livremente, para se unirem de forma a minimizar os defeitos no resultado final. Da mesma forma, um algoritmo SA substitui uma solução corrente por uma solução próxima, ou seja, na vizinhança do espaço de soluções, baseia-se numa variável correspondente à temperatura. Quanto maior a temperatura, maior aleatoriedade haverá na seleção da próxima solução. O algoritmo progride na redução desta variável, de modo a que haja uma convergência para a obtenção de uma solução ótima. De acordo com a variância da energia em cada iteração, diferentes casos são aplicados para a obtenção da próxima solução (Figura 3.7).

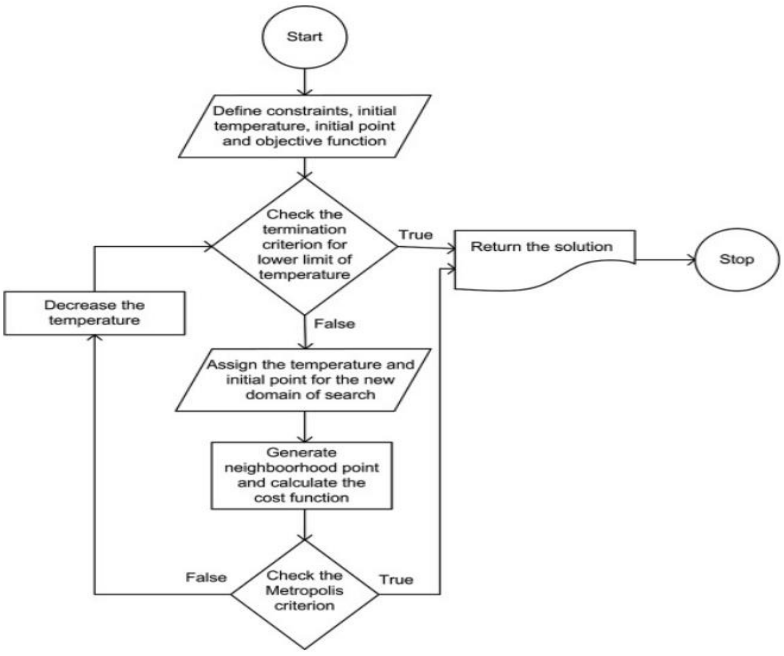

Figura 3.7 Processo na criação de um algoritmo SA (Eren et al., 2017).

Outra categoria de algoritmos que dá uso a meta-heurísticas são os de enxame de partículas (PS), que visam imitar o comportamento de um bando de pássaros, sendo cada um destes uma partícula da população. Este método de otimização existe há pouco mais de duas décadas, sendo um tema conhecido, mas recente, criado por Kennedy e

Eberhart (Kennedy & Eberhart, 1995). Cada partícula terá a sua própria velocidade e posição. Perante a cooperação com outras partículas e pela aprendizagem própria, esta partícula levará o espaço de soluções cada vez mais próximo da fronteira de Pareto. Assim, são usadas duas variáveis, que correspondem à experiência pessoal e geral de cada partícula, de forma a explorarem o espaço de pesquisa. Com uso de fórmulas relativas à sua velocidade e posição de cada uma destas, consegue-se mover a população de partículas à melhor solução.

A meta-heurística da pesquisa tabu (TS) também é muito conhecida, sendo proposta por (Glover, 1986), de onde é elaborado um mecanismo de adaptação de soluções, de forma a guardar movimentos que deteriorem as mesmas, isto é, é utilizada uma estratégia de memorização de movimentos proibidos (utilizando-se o nome tabu), que de alguma forma piorem as soluções da proximidade do ótimo.

Os métodos de pesquisa de vizinhança variável (VNS) têm sido cada vez mais estudados, usando uma pesquisa local e testando os vizinhos até se encontrar um vizinho local que seja ótimo, a partir do qual se expande a nossa solução para essa vizinhança, reinicializando este processo até se obter uma ou mais soluções finais.

3.2.2 Algoritmo exato de Martins

O algoritmo proposto por (Martins, 1984), explicado de seguida com mais ênfase, realiza a sua pesquisa e recorre aos vizinhos localmente, no entanto não é considerado uma meta-heurística, uma vez que obtém soluções ótimas, e não aproximações. Podemos dizer que é uma extensão do algoritmo Simplex (Gass, 2011), dado que este último procura o valor mínimo ou máximo, respeitando as restrições do problema. No entanto, Martins alargou o seu algoritmo para tratar problemas com diversos objetivos. Na figura seguinte é mostrado o algoritmo multi-objetivo de Martins (Figura 3.8).

Como já mencionado anteriormente, nestes problemas temos vários caminhos ótimos de Pareto, mais propriamente, de um vértice inicial s para qualquer outro vértice do grafo. Então, será preciso uma estrutura que guarde todos os caminhos não-dominados. Logo teremos presente um conjunto π_i, contendo caminhos não dominados, de s a i, que irão sendo atualizados, isto é, apagados se forem dominados por um novo caminho encontrado, ou caso contrário, mantidos no conjunto.

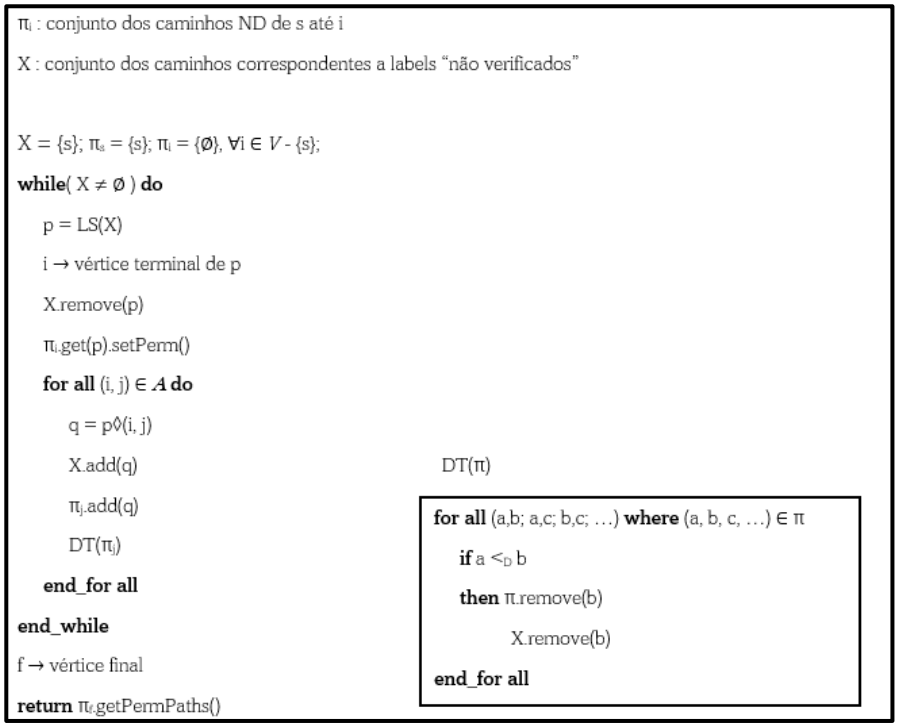

Figura 3.8 Algoritmo de etiquetagem para problemas de otimização multi-objetivo de caminhos (Martins, 1984).

Contrariamente dos problemas com um único objetivo, temos várias etiquetas associadas ao mesmo vértice. Consequentemente, existe um conjunto X com etiquetas não verificadas, onde, em cada iteração do algoritmo, é seguida uma política de seleção de etiqueta, isto é, uma única etiqueta não verificada é retirada de X, com mudança da propriedade da etiqueta de temporária para permanente, dentro do conjunto de etiquetas do seu vértice terminal i. Ao ser permanente, sabemos que nenhum outro caminho irá dominar o mesmo, entre s e i. Seguidamente, o caminho p correspondente de s a i é expandido, adicionando as arestas (i,j). Esta política não é aleatória, mas sim, feita de acordo com a etiqueta lexicograficamente mais pequena (LS). Quando se diz lexicograficamente mais pequena, procura-se a etiqueta mais favorável, a partir dos seus valores de cada objetivo, começando pelo primeiro, seguindo-se do segundo, por aí adiante, comparando com as restantes etiquetas do conjunto. Empate entre etiquetas não é um acontecimento possível, visto que para duas etiquetas empatarem teriam de ser a mesma. Como exemplo podemos ter uma instância com minimização de três objetivos. Neste caso, a escolha lexicográfica mais pequena entre duas etiquetas é feita, se para o primeiro objetivo, uma das etiquetas tiver menor valor que a outra. Se ambas as etiquetas tiverem o mesmo valor para o primeiro objetivo, compara-se o segundo, e assim sucessivamente, até se encontrar um valor mais baixo que o outro. Se algum objetivo for de maximização, procura-se a etiqueta com o maior valor para o mesmo.

Este novo caminho p com o incremento da expansão (p◊(i,j)), agora q, é adicionado a X e ao conjunto π_j pertencente ao seu vértice terminal j. A condição de

paragem também tem de se ter em conta. Como o número de caminhos não dominados de s a qualquer outro vértice i não é sabido a priori, o algoritmo apenas para quando o conjunto X fica vazio, mais propriamente, só para quando todas as etiquetas forem verificadas. Isto significa que tem de se computar (expandir) todos os caminhos não dominados de s para cada um dos vértices i. É feito então um teste de dominância DT(π_j) para cada uma das arestas (i,j) presentes no caminho p de cada iteração. Neste teste de dominância é verificado, entre todos os pares de caminhos do conjunto π_j, se existem caminhos dominados. Caso um caminho a domine outro b, isto é a $<_D$ b, então b é removido do conjunto π_j e de X.

Quando a condição de paragem for atingida, havendo inexistência de etiquetas não verificadas, o algoritmo termina, retornando as soluções não-dominadas que obteve, a partir dos caminhos permanentes do conjunto π_f, conjunto este pertencente ao vértice final f. Os caminhos temporários deste conjunto simplesmente são dominados por algum/ns dos permanentes.

No estudo de Paixão e Santos (Paixão & Santos, 2013) são apresentados diversos métodos para aprofundar o conhecimento deste problema, mais propriamente, diversas formas de aplicar os algoritmos de etiquetagem, de forma a se descobrir, em termos computacionais, as melhores técnicas e estruturas a utilizar.

Cada vez mais procura-se otimizar este algoritmo, uma vez que se tem de examinar todo o grafo para encontrar os caminhos de Pareto. Isto faz com que seja pouco eficiente a sua execução, principalmente em grafos de grande dimensão. Outra forma de abordar este problema, proposta por (Demeyer et al., 2013), passa por não só colocar uma segunda condição de paragem para interromper o processo de pesquisa no algoritmo, entre outras palavras, etiquetas que são dominadas por uma etiqueta que contém o vértice destino, nunca levarão a uma solução de Pareto e por isso torna-se irrelevante as investigar, como também procurar no grafo bidirecionalmente, isto é, simultaneamente pela origem e pelo destino.

Capítulo 4 Caso de Uso e Resultados

Para podermos circular em Lisboa de forma agradável, muitos fatores são considerados, tais como, o local com menos poluição para dar um passeio de bicicleta ao fim de semana, o trajeto mais económico de se fazer de carro para o trabalho, ou ainda qual o trajeto menos íngreme para dar uma corrida, de forma a não se fazer tanto esforço físico. Todos estas circunstâncias são, muitas vezes, não respondidas, ou na melhor das hipóteses, respondidas de forma não clara ou errada.

Este trabalho estuda a resolução do problema de caminho mais curto multi-objetivo, a partir de uma secção do mapa de Lisboa. Para isto, é descoberto o caminho de custo mínimo, de um ponto inicial deste mapa até outro ponto de destino. Desta forma, processou-se esta informação num grafo, a fim de transformar as ruas e estradas em arestas, e as suas intersecções em vértices (Figura 4.1). O grafo obtido é dirigido, visto que a informação usada a priori, a partir do trabalho de (Nunes, 2022), contém estradas com um ponto origem e destino, ou seja, toda a informação relativa às arestas possui orientação. Com a aplicação do algoritmo de Martins, explicado na secção 3.2, é possível conjugar diversos parâmetros, pontos de origem/destino e múltiplos objetivos.

Existiram diversos problemas com o uso da transformação do mapa num grafo. Um deles foi o facto de haver arestas repetidas, ou seja, contendo os mesmos vértices de origem e destino mas com diferentes valores para os diferentes objetivos. Isto levou a que se tivessem de escolher as melhores arestas, com o critério de maior número de objetivos favoráveis. Neste caso, tendo presente três objetivos a minimizar, explicado mais adiante, escolheu-se as arestas com o maior número de valores mais baixo. Outro problema surgiu quando se percebeu que certas arestas levavam a becos, isto é, havia um rompimento na conetividade do grafo. Uma vez que cada entrada das estradas já possuía um vértice origem e destino, optou-se por criar uma bidirecionalidade em cada uma das arestas, de forma a que, para cada aresta com vértice origem a e vértice destino b, fossem criadas duas arestas, (a,b) e (b,a). Assim, os valores dos critérios presentes na aresta (a,b) são passados para (b,a). Após este passo, foram tratados dos conflitos, caso existissem arestas repetidas. Deste modo, o grafo torna-se conexo, visto que é possível ir de um vértice a qualquer outro, por um caminho pertencente ao grafo.

Figura 4.1 Representação de uma pequena secção do mapa de Lisboa, com indicação dos vértices (vermelho) e arestas (azul).

4.1 Função Objetivo

Para caracterizar a nossa função objetivo, é necessário recorrer aos parâmetros que foram possíveis de identificar nos ficheiros extraídos da base de dados. Esta função pode ser caracterizada, combinando as diferentes funções objetivo apresentadas seguidamente, tendo em conta as suas maximizações/minimizações.

Distância

Esta função objetivo passará por maximizar ou minimizar a distância, em metros, atravessada por um peão na cidade de Lisboa, de uma localização inicial para uma final. Será calculada tendo em conta as distâncias individuais de cada estrada constituinte de cada par de vértices do caminho final, isto é, tendo um caminho [a,b,c,d], a sua distância será calculada pela soma das distâncias das arestas (a,b), (b,c) e (c,d).

Custo Verde

Para esta função é necessário representar a maior proximidade a áreas verdes, de forma a maximizá-la, caso queiramos ter a menor exposição à mesma no nosso caminho, caso contrário, minimizá-la para ter a maior exposição no nosso caminho. Desta forma, para se obter o custo verde é necessário multiplicar a distância da estrada, pelo valor da coluna greenmeter, mostrado na Tabela 4.1, como proposto em (Nunes, 2022), que pode ser:

1 - Se a estrada interseta uma zona verde.

2 - Se a estrada tem pelo menos uma zona verde num raio menor que 25 metros de distância.

3 - Se não existe nenhuma zona verde a pelo menos 25 metros de distância da estrada.

Custo Limpo

De forma a calcular o valor desta função para cada aresta, será essencial obter a exposição à poluição presente nas estradas, de forma a ter uma melhor qualidade de ar, para consequente cálculo do custo no caminho. A minimização implica uma área menos poluída, isto é, uma melhor qualidade de ar, enquanto que a maximização uma área mais poluída. O custo limpo é calculado tendo em conta a multiplicação da distância da estrada pelo valor da coluna airquality_pos, presente na Tabela 4.1. Este segundo valor é recolhido, tendo em conta a média dos valores da qualidade do ar medidos num raio de 50 metros de distância da estrada, tal como proposto em (Nunes, 2022).

A função objetivo será então um vetor de diversas funções objetivo, tantas quanto o número de objetivos escolhidos. Para a resolução do problema multi-objetivo presente, será usada a função objetivo abaixo, combinando entre dois e três dos objetivos, com uso de minimização em todos estes. Minimizando os três objetivos, tendo como vértice origem **x** e vértice destino **y**, a função objetivo será:

$$[\min \text{dist}(x,y), \min \text{custoVerde}(x,y), \min \text{custoLimpo}(x,y)], \text{ onde:}$$

$$\min \text{dist}(x,y) = \min \sum_{(i,j)\in A} \text{dist}(i,j)$$

$$\min \text{custoVerde}(x,y) = \min \sum_{(i,j)\in A} \text{custoVerde}(i,j)$$

$$\min \text{custoLimpo}(x,y) = \min \sum_{(i,j)\in A} \text{custoLimpo}(i,j),$$

(i,j) representando cada aresta pertencente ao conjunto de arestas A do caminho, com vértice origem i e vértice destino j.

4.2 Resolução do Problema

Nesta secção será mostrado os passos necessários para a resolução do problema com vários objetivos, bem como, a explicação de cada um destes.

4.2.1 Extração da Informação

Foi primeiramente feito por (Nunes, 2022) um estudo relativamente ao desenvolvimento de caminhos uni-objetivo. Como fundação foi usado uma secção do mapa de Lisboa, extraída da base de dados da plataforma OpenStreetMaps em ficheiros .csv (valores separados por vírgulas), e identificadas localizações e estradas pertencentes à mesma, que posteriormente processou-se em vértices e arestas, de forma a ser possível integrar um grafo. Finalmente, foram resolvidas instâncias nesta grafo, de forma a comparar resultados com aplicações conhecidas.

Desta forma, dá-se a continuação deste estudo, passando para um panorama multi-objetivo. Portanto, são aplicados diversos critérios, de forma a se conseguir promover uma melhoria nas escolhas do AD e na abrangência de soluções. Foram usados os mesmos dados, que serviram como input para o algoritmo implementado, a partir de tabelas em Excel, uma relativa às estradas (Tabela 4.1), e outra às localizações.

ID Aresta	Source	Target	Length	Custo Verde	Custo Limpo	greenmeter	airquality_pos
1	1	2	74.26	148.52	6699.73	2	90.22
2	2	3	110.6	221.2	9812.74	2	88.72
3	4	5	58.21	116.43	5080.21	2	87.27
4	5	6	92.55	185.1	7885.14	2	85.20
5	6	7	7.93	15.86	662.26	2	83.31
6	8	9	87.24	174.47	7088.37	2	81.25
7	9	10	44.7	89.4	3733.95	2	83.54
8	11	12	218.32	654.97	20709.47	3	94.86
9	12	13	138.17	414.51	13242.35	3	95.84
10	13	14	38.87	116.6	3737.281	3	96.16
11	3	15	131.05	393.14	11765.72	3	89.78
12	15	16	23.3	69.91	2133.13	3	91.54
13	17	18	12.52	37.57	1118.42	3	89.31

Tabela 4.1 Ficheiro CSV exemplo de arestas.

4.2.2 Conversão da Informação

Após ser feita a extração dos ficheiros, foi necessário transformar esta informação em dados utilizáveis. Para isso, foram processadas as arestas e vértices, a partir dos ficheiros, contendo a informação das estradas e das localizações, respetivamente e feito a bidirecionalidade em todas as arestas, como indicado na introdução deste capítulo. Estes ficheiros contêm a informação disposta em forma de matriz, onde cada coluna representa uma característica específica, e cada linha representa cada localização ou estrada, dependendo do ficheiro.

Do ficheiro das localizações, apenas foi retirada a informação relativa a uma coluna, pertencente ao identificador de cada uma destas. Do ficheiro das estradas, usou-se uma coluna como identificador, como também duas colunas para informar os identificadores das localizações inicial e final da estrada (source e target). Usaram-se três colunas para representar cada um dos objetivos que, poderão ou não, aparecer no problema, de acordo com as escolhas do AD. As restantes colunas destes ficheiros foram ignoradas, uma vez que não traziam informação útil para o problema.

Desta forma, consegue-se extrair toda a informação destes ficheiros, de uma forma automática, caso se queira no futuro trabalhar noutras zonas geográficas, ou com diferentes objetivos.

4.2.3 Criação do Grafo Multi-Objetivo

Com a conversão de toda a informação, é possível formar vértices e arestas, para compor um grafo, representante da área geográfica escolhida. Cada vértice é formado a partir de uma localização, tendo presente os valores respetivos à linha extraída do ficheiro das localizações. Da mesma forma, cada aresta é formado a partir de uma estrada, tendo presente os valores respetivos à linha extraída do ficheiro das estradas.

Após a criação do grafo, é possível a execução de qualquer algoritmo multi-objetivo sobre este. Assim, temos um problema com uma localização de origem e uma localização de destino, pré-fabricadas, de forma a obter uma ou mais soluções, que contenham estas duas localizações, sendo todas as soluções do conjunto final de Pareto, igualmente ótimas. Estas soluções não são mais que caminhos, constituídos por localizações e estradas, alternadamente, isto é, tendo localizações L e estradas E, um caminho terá a composição L_i -> E -> L -> E -> … -> L_f , sendo L_i a localização de origem e L_f a de destino.

4.2.4 Aplicação do Algoritmo

Para este trabalho, foi aplicado o algoritmo proposto por (Martins, 1984), por ser capaz de computar e encontrar o caminho mais curto entre dois vértices num grafo, aplicando os diversos critérios a cada uma das arestas, de forma muito eficaz, fugindo do uso da média ponderada dos vários objetivos num só, o que provoca muita perda de informação.

Foi feita uma modificação ao mesmo, para se poder guardar caminhos eliminados que durante a execução do algoritmo foram qualificados como sendo de Pareto, mas que

no entanto acabaram por ser dominados por outros caminhos. Para isto criou-se um outro conjunto, de forma a acolher todas as etiquetas eliminadas. Da mesma forma que se obtém as soluções de Pareto a partir do conjunto do vértice final, obtém-se as soluções dominadas, a partir deste conjunto. Não se adquire todas as soluções dominadas, uma vez que, com a dimensão do mapa escolhido, o número das mesmas seria muito grande.

4.2.5 Obtenção das Soluções

Após a terminação do algoritmo, são obtidas todas as soluções na FP. Como resultado, o AD tem um leque variado de soluções, não ficando dependente dos algoritmos uni-objetivo existentes, que são executados dentro de softwares que não dão o poder necessário de escolha.

Uma instância será então nada mais que um caso particular do problema, tendo presente um grafo com certos vértices e arestas, diversos objetivos com otimização em maximização e/ou minimização, um vértice origem e um vértice destino.

Na conclusão da execução do algoritmo, teremos um conjunto de soluções, para a construção da FP, sendo algumas destas, caminhos de Pareto, igualmente ótimos, e caminhos dominados, que durante o processamento dos dados, eram predeterminados como caminhos de Pareto temporários, que acabaram por ser dominados por outros.

4.3 Testes e Resultados

Neste capítulo serão mostrados os testes computacionais executados, aplicando diferentes instâncias no algoritmo escolhido. Os testes foram executados numa máquina com um processador AMD Ryzen 5 3500U, 2.1GHz e memória RAM de 12GB, com o sistema operativo Windows 10.

Os vértices origem e destino das instâncias I1 a I4 foram selecionados de acordo com o estudo em (Nunes, 2022), de forma a se poder comparar as vantagens do uso de diversos objetivos em simultâneo, relativamente a apenas um. A diferença das instâncias relativamente ao segundo algarismo parte do uso de dois ou três objetivos. As restantes instâncias foram criadas escolhendo arbitrariamente para cada uma delas o vértice origem e o vértice destino.

A Tabela 4.2 apresenta os detalhes de cada instância resolvida, bem como tempos de execução e complexidade do grafo das mesmas.

ID - identificador da instância, que será único.

VértO - id do respetivo vértice origem da instância.

VértD - id do respetivo vértice destino da instância.

Length - indicador de minimização, maximização, ou não utilização do objetivo relativo à distância.

Custo Verde - indicador de minimização, maximização, ou não utilização do objetivo relativo à quantidade de área verde presente.

Custo Limpo - indicador de minimização, maximização, ou não utilização do objetivo relativo à quantidade de área limpa presente.

ID	VértO	VértD	Length	Custo Verde	Custo Limpo
1-2	8	461	MIN	MIN	--
1-3	8	461	MIN	MIN	MIN
2-2	2100	460	MIN	MIN	--
2-3	2100	460	MIN	MIN	MIN
3-2	2164	1391	MIN	MIN	--
3-3	2164	1391	MIN	MIN	MIN
4-2	200	319	MIN	MIN	--
4-3	200	319	MIN	MIN	MIN
5-3	2520	64	MIN	MIN	MIN
6-3	21	1997	MIN	MIN	MIN

Tabela 4.2 Instâncias criadas.

Na Tabela 4.3 apresentam-se os resultados do algoritmo para as instâncias, bem como os respetivos tempos de execução.

ID - identificador da instância (tabela 4.2).

NumObj - número de objetivos presentes na respetiva instância.

NumSolP - número de soluções não-dominadas obtidas.

NumSolD - número de soluções dominadas.

%Ótima - percentagem do número de soluções de Pareto, em relação ao número de todas as soluções obtidas.

ValÓtimoD – melhor valor encontrado para o objetivo distância.

ValÓtimoV – melhor valor encontrado para o objetivo custo verde.

ValÓtimoL – melhor valor encontrado para o objetivo custo limpo.

T(ms) - tempo de execução do algoritmo em milissegundos.

Dim- - dimensão da menor solução de Pareto, em número de vértices.

Dim+ - dimensão da maior solução de Pareto, em número de vértices.

ID	Num Obj	Num SolP	Num SolD	%Ótima	ValÓtimo D	ValÓtimo V	ValÓtimo L	T(ms)	Dim-	Dim+
1-2	2	4	2	66,7%	351,41	559,07	--	1856	8	14
1-3	3	5	1	83,3%	351,41	559,07	28134,57	2548	8	14
2-2	2	4	4	50%	1275,33	2810,26	--	2798	20	26
2-3	3	4	3	57,1%	1275,33	2810,26	106475,96	3999	20	26
3-2	2	2	2	50%	844,89	2145,37	--	2456	12	14
3-3	3	2	2	50%	844,89	2145,37	71547,91	3425	12	14
4-2	2	1	1	50%	409,65	888,54	--	4556	11	11
4-3	3	2	2	50%	409,65	888,54	33058,96	21839	8	11
5-3	3	3	3	50%	1933,14	4822,15	147102,19	8417	52	53
6-3	3	8	14	36,4%	2162,26	5941,36	152905,63	2786	44	57

Tabela 4.3 Resultados do algoritmo.

Para a apresentação dos resultados, serão exibidas imagens retratando cada solução no mapa de Lisboa, sendo as soluções a azul de Pareto, e as a vermelho dominadas, como também a cruz azul o vértice de origem e a cruz verde o vértice de destino. As soluções a vermelho são obtidas a partir do algoritmo, uma vez que são processadas quando este é executado, tendo no entanto sido encontradas uma ou mais soluções que a dominam.

Figura 4.2 Soluções com minimização de 1 objetivo (distância à esquerda, custo verde no meio e custo limpo à direita) para a instância 1-2 (Nunes, 2022).

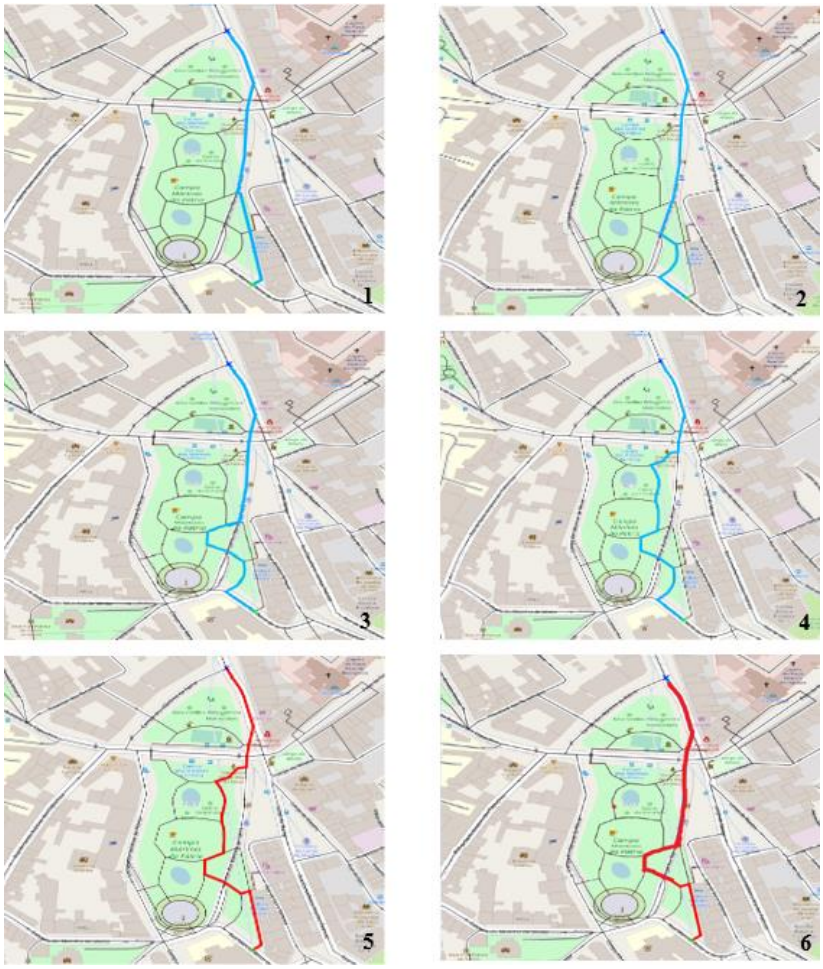

Figura 4.3 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 1-2.

Na Figura 4.2 são apresentadas as rotas obtidas pela aplicação HealthyTrack de (Nunes, 2022), de forma a otimizar cada um dos objetivos, individualmente. Na tentativa de otimizar, tanto a distância como o custo verde, foram obtidas quatro soluções de Pareto (Figura 4.3), observando-se que duas destas são semelhantes às rotas de otimização de distância e de custo verde (solução 1 de distância, solução 4 do custo verde), obtidas por Nunes, com acrescento de duas novas soluções de Pareto e duas dominadas (solução 2 e 3 de Pareto, solução 5 e 6 dominada). É possível formar a FP com a presença de todas as soluções, como observado na Figura 4.4, verificando-se que, com o uso de algoritmos uni-objetivo apenas se obteria as soluções de Pareto (a azul) das extremidades, ou seja, a solução com menor distância e com menor custo verde. Aplicando o algoritmo multi-objetivo, conseguiu-se proporcionar ao AD um leque maior de escolhas, mais precisamente duas novas soluções alternativas que apresentam um compromisso entre os dois objetivos.

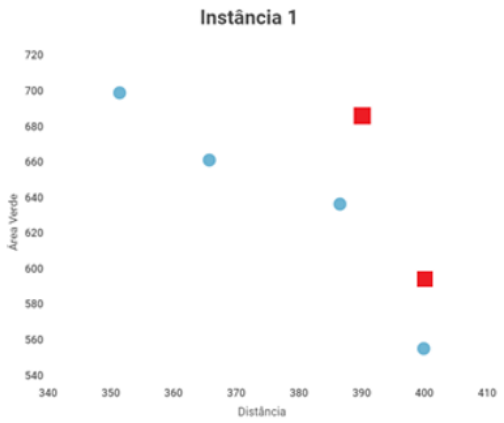

Figura 4.4 Frente de Pareto da instância 1-2.

De seguida passamos para a análise das soluções da instância que se obtém da 1-2, mas com três objetivos, acrescentando à função objetivo a minimização do custo limpo, dando origem à instância 1-3.

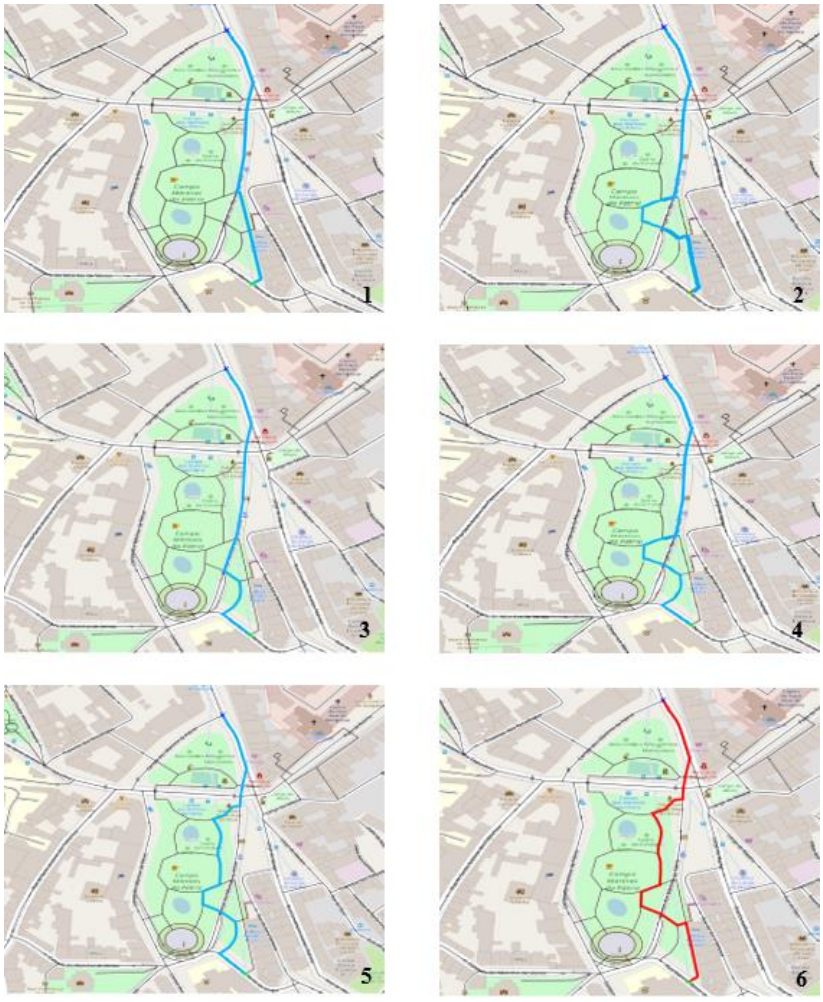

Figura 4.5 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 1-3.

É percetível o aumento do número de soluções de Pareto, quando transformamos a instância 1-2 num problema de minimização dos três objetivos. As quatro soluções de Pareto mantiveram-se, com o acrescento de uma nova solução de Pareto (solução 2), que era precisamente uma solução dominada da instância 1-2 (solução 6). Isto deve-se ao cancelamento da dominância desta solução no terceiro objetivo, relativamente ao custo limpo. Deste modo, o AD formulando um problema com três objetivos consegue ter mais opções de escolha. A solução dominada 6 veio da solução dominada 5 da instância 1-2.

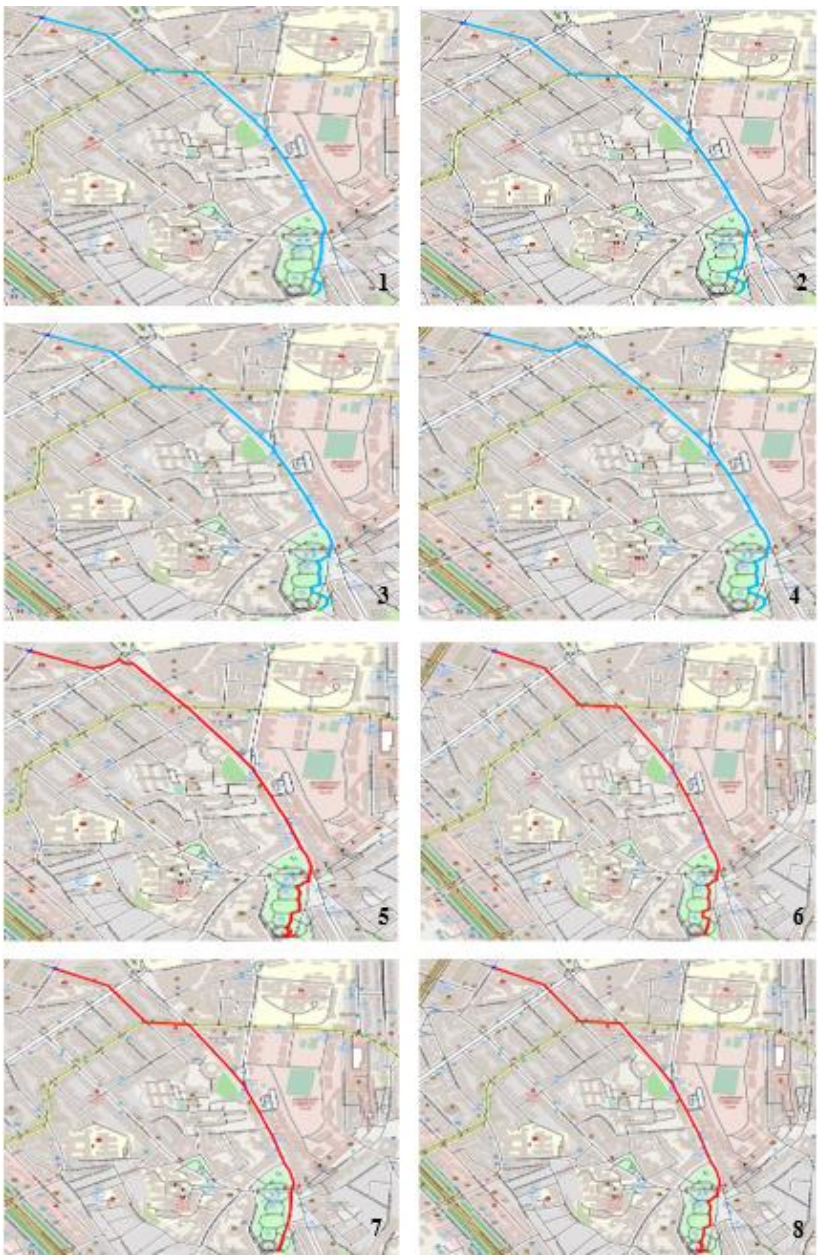

Figura 4.6 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 2-2.

Para a instância 2-2, observa-se um trajeto mais longo (Figura 4.6), tendo presente parte rodoviária, como também parte florestal, sendo esta última coincidente com as instâncias 1-2 e 1-3, o que criou um aumento no número de soluções relativamente às mesmas. Uma vez que o objetivo proposto nesta instância é minimizar tanto a distância como o custo verde, conseguimos perceber pelas rotas obtidas que tendem a dirigir-se à parte florestal (mais perto do vértice destino), o mais rapidamente possível. Deste modo, a primeira parte de todas as soluções está disposta apenas de duas formas, havendo maior diversidade das soluções na segunda parte da rota. Apresenta-se ao AD um leque de soluções, o que com a minimização de apenas um objetivo não aconteceria (Figura 4.7).

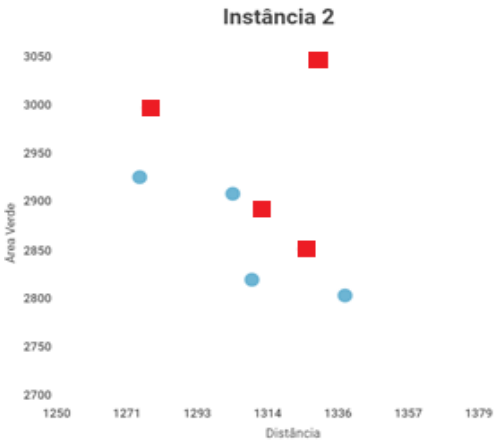

Figura 4.7 Frente de Pareto da instância 2-2.

É mostrado na Figura 4.8 a passagem desta instância para a minimização dos três objetivos.

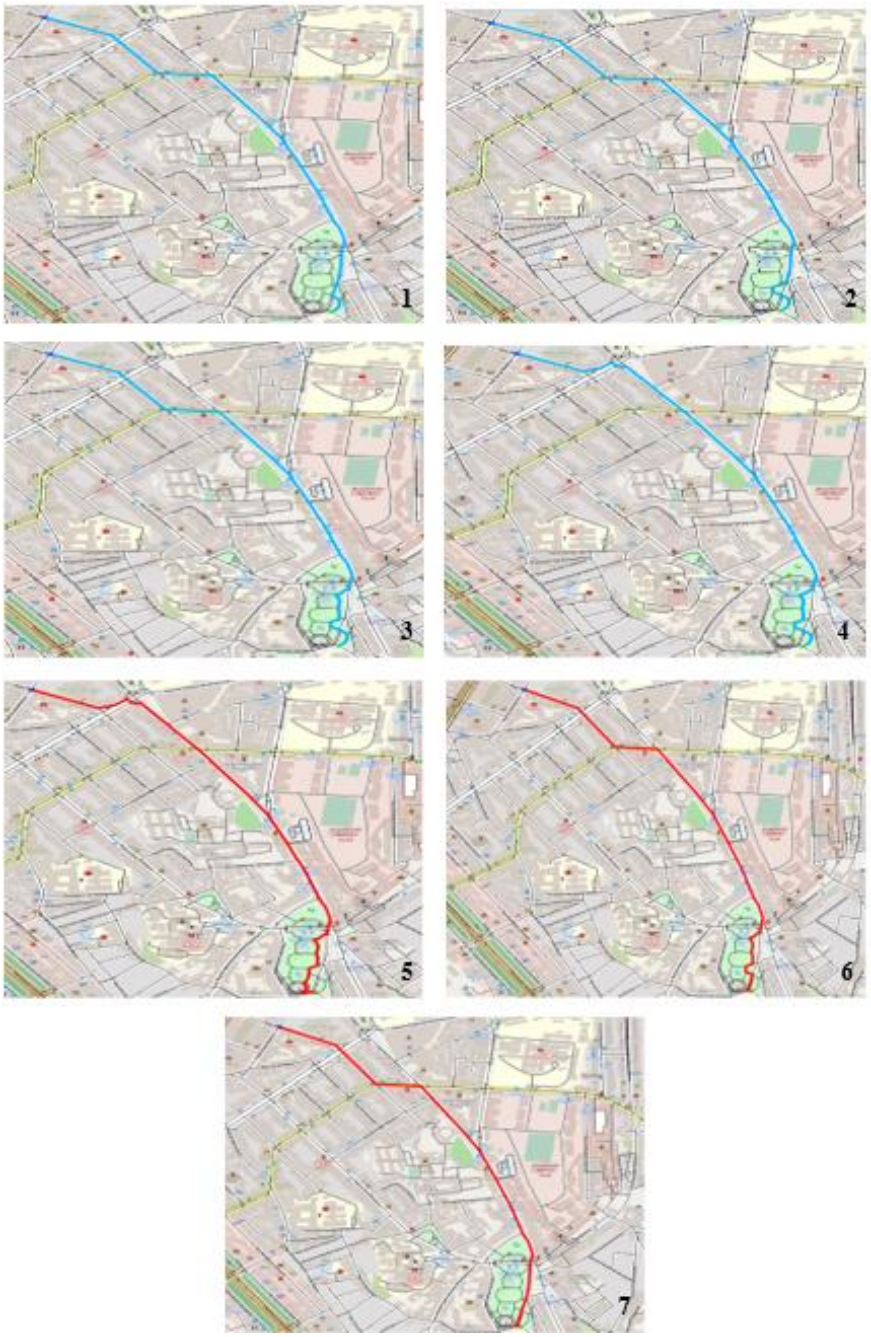

Figura 4.8 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 2-3.

Relativamente à resolução da instância 2-3, pouco mudou nas rotas obtidas (Figura 4.8). Obteve-se sete das oito soluções da instância prévia com dois objetivos, mantendo-se as mesmas quatro soluções de Pareto. Isto deve-se à solução 1 de Pareto da instância 2-2 minimizar simultaneamente o objetivo da distância e o objetivo do custo limpo. Consequentemente não se obtiveram melhores soluções para o objetivo.

Figura 4.9 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 3-2.

Para a instância 3-2, foi criado um trajeto relativamente linear, tendo-se obtido duas soluções de Pareto, sendo a solução 1 designada à minimização da distância e a solução 2 à minimização do custo verde (Figura 4.9).

É possível verificar claramente que a solução 3 foi dominada pela solução 1, uma vez que, sendo a primeira metade do caminho equivalente, a segunda parte do caminho da solução 3 é tanto mais longa, como também menos vantajosa em termos de proximidade à área verde. O mesmo sucede com as soluções 2 e 4. Conseguimos perceber melhor estas relações de dominância, a partir da FP da Figura 4.10.

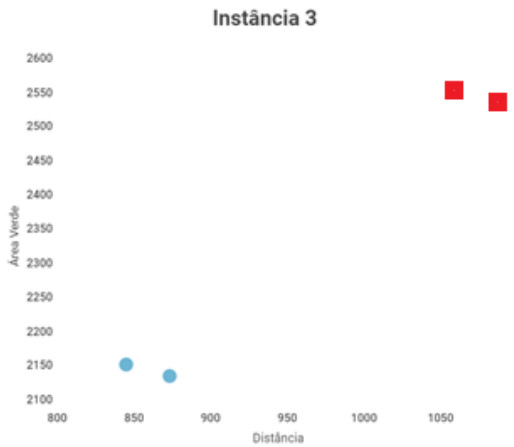

Figura 4.10 Frente de Pareto da instância 3-2.

Com a introdução de mais um objetivo, as mesmas rotas foram obtidas (Figura 4.11), relativamente à instância 3-2. Isto deve-se à solução de Pareto 1 inclusive minimizar o custo limpo.

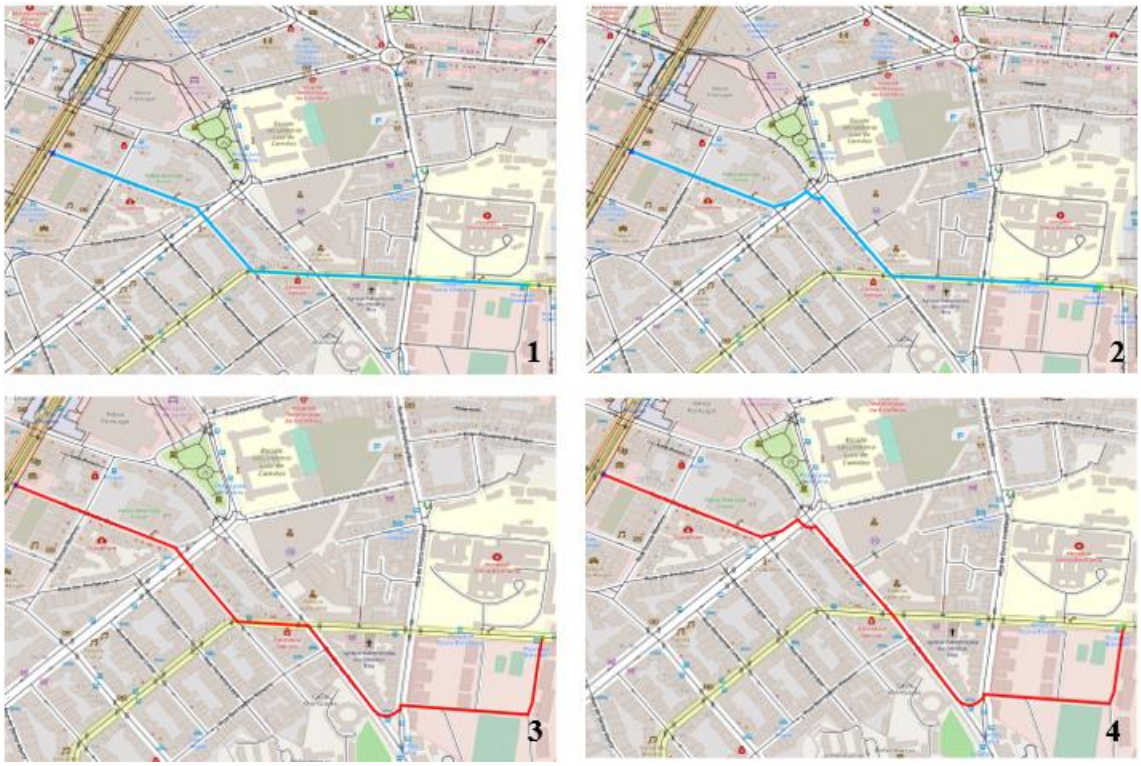

Figura 4.11 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 3-3.

A instância 4-2 é testada a seguir, comparando com os resultados de Nunes.

Figura 4.12 Soluções com minimização de 1 objetivo (distância à esquerda, custo verde no meio e custo limpo à direita) para a instância 4-2 (Nunes, 2022).

Figura 4.13 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 4-2.

Na Figura 4.12 são apresentadas as rotas obtidas pela aplicação HealthyTrack de (Nunes, 2022), de forma a otimizar cada um dos objetivos, individualmente. No experimento de minimizar ambos os objetivos da distância e do custo verde, foi obtido na Figura 4.13 uma solução de Pareto (solução 1), observando-se que é semelhante à solução de otimização da distância, e à do custo verde, obtidas por Nunes. Neste caso, o AD tem apenas ao seu dispor uma rota, uma vez que só existe uma solução ótima. O algoritmo obtém ainda uma outra solução, semelhante no espaço dos objetivos, mas que é dominada. É possível observar este cenário na FP da Figura 4.14.

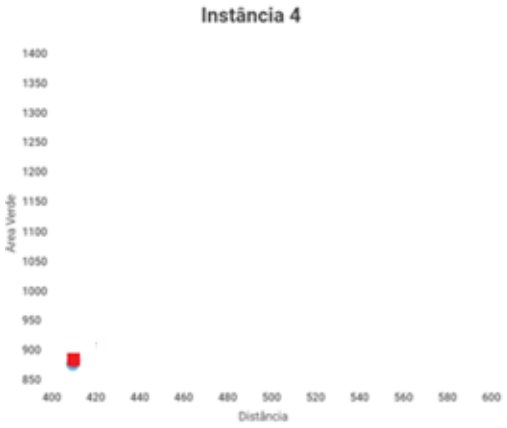

Figura 4.14 Frente de Pareto da instância 4-2.

Figura 4.15 Representação gráfica das rotas obtidas pelo algoritmo, para a instância 4-3.

Para a instância 4-3 foi feito o acréscimo do terceiro objetivo, relativo ao custo limpo. Deste modo, obteve-se uma segunda solução de Pareto (solução 2), que visa minimizar este novo objetivo, como também é possível observar na solução da direita obtida por Nunes (Figura 4.12). Obteve-se a mesma solução dominada (solução 3), com a presença de uma nova (solução 4 dominada), ambas dominadas pela primeira solução.

Relativamente aos resultados obtidos, é possível observar pela Tabela 4.3 que, passando de dois objetivos para três, o número de soluções de Pareto mantém-se ou aumenta, o que era de esperar visto que, minimizando três objetivos, onde dois destes estão repetidamente contidos, as soluções de Pareto têm obrigatoriamente de se manter. Consequentemente a percentagem de soluções ótimas (%Ótima) também se mantém ou acresce.

A modificação feita no algoritmo para guardar soluções dominadas, onde previamente tinham sido potencialmente de Pareto, expõe-nos que, maioritariamente, estas também se mantêm ou aumentam. Pode suceder uma diminuição do valor, caso alguma solução dominada passe a ser de Pareto, levando o novo objetivo adicionado a cancelar a dominância de uma solução de Pareto que previamente a dominava.

O tempo de execução das instâncias mostra a existência de um aumento na passagem dos dois para três objetivos. Isto deve-se à necessidade do algoritmo ter de fazer cálculos extra entre duas etiquetas, ou seja, comparando duas soluções, são feitas três comparações em relação às duas, quando presentes apenas dois objetivos. Isto reflete-se de duas formas:

- A primeira, em relação à maior dificuldade na eliminação de soluções provenientes da propriedade da dominância das mesmas. Uma solução mais rapidamente é dominada e portanto eliminada, quando tem apenas presente dois valores agregados.

- A segunda deve-se à escolha da etiqueta lexicograficamente mais pequena. Em vez de, no mínimo ser feita uma comparação e no máximo duas, quando presentes dois objetivos, passa-se a no mínimo ser feita uma comparação e no máximo três.

Pelas instâncias 1 a 3, consegue-se verificar que o aumento do tempo computacional, passando de dois para três objetivos, aumenta sensivelmente entre 35% a 45%, uma vez que as soluções obtidas não variam muito. Acontece haver "outliers", isto é, tempos computacionais muito acima do esperado, como é o caso da instância 4, que aumentou mais de quatro vezes devido às novas possibilidades de caminhos que se formaram a partir do vértice origem, para tentar minimizar o objetivo do custo limpo que foi adicionado ao problema. Desta forma, o tempo computacional não se reflete pela dimensão dos caminhos, ou seja, pelo número de vértices constituintes em cada uma das soluções, mas sim pelo número de subcaminhos que são formados, isto é, pela densidade de vértices e arestas na região, refletindo-se numa maior ramificação de soluções. A dimensão das soluções apenas consegue-nos dizer se o caminho é relativamente linear, no sentido de não ter muitas curvas, ou se possui muitas curvas, caso a sua dimensão seja alta.

O verdadeiro poder do algoritmo é verificado na instância 6-3, com vértices origem e destino muito mais distanciados, uma vez que os seus três valores ótimos, indicados na Tabela 4.3, são muito mais elevados, e por isso, resultando num grande aumento no número de soluções, quer de Pareto, quer dominadas. No entanto, o seu tempo computacional não foi além de pouco mais de dois segundos, mostrando-se capaz de responder às necessidades impostas.

Capítulo 5 Conclusão

Com o aumento da mobilidade e trânsito, cada vez mais é necessário suporte para o apoio direto nas recomendações de rotas para toda a população. Este tema tem sido estudado ao longo dos tempos, no entanto existem inúmeras lacunas no que toca ao uso de mais do que um objetivo. As aplicações usadas no quotidiano são rápidas e eficazes, no entanto, muito limitadoras na abrangência dos objetivos, na conciliação dos mesmos, bem como no poder de decisão fornecido ao AD.

Este trabalho teve como objetivo proporcionar um apoio na investigação de rotas multi-objetivo, dentro de uma secção do mapa de Lisboa, para que, em conjunto com a investigação feita em (Nunes, 2022), seja possível integrar-se uma aplicação móvel para o uso da população geral. Desta forma, todo este tema pode ser expandido futuramente, em diferentes territórios, ou em áreas específicas, como rotas para bicicletas, onde mais objetivos poderão ser incluídos, como a altitude, o tipo de piso, etc.

Deste modo, as principais contribuições deste trabalho são:

1. Estudo relativamente a algoritmos uni e multi-objetivo.
2. Extração e processamento de dados, referente a uma secção do mapa real de Lisboa, a partir de ficheiros .csv da base de dados da plataforma OpenStreetMaps.
3. Utilização da informação mencionada em 2., a fim de construção de um grafo direcionado conexo, permitindo assim o deslocamento de um qualquer vértice origem para outro vértice destino.
4. Implementação do algoritmo para otimização de caminhos multi-objetivo de (Martins, 1984) em casos concretos, modificado para fornecer caminhos que durante a sua execução eram de Pareto, mas que se tornaram obsoletos devido à dominância por outros.
5. Apresentação de uma comparação relativa aos resultados que podem ser apresentados ao AD face a um objetivo, em relação a múltiplos objetivos.

No futuro, é possível melhorar em diversos aspetos, tais como:

- Tratar de otimizar o algoritmo proposto, quer através de melhores escolhas de estruturas, quer através de melhorias na forma como é feita a pesquisa, com o intuito de tratar dos “outliers” e estabilizar os seus resultados.

- Aplicação de mais algoritmos, uma vez que uns algoritmos podem ser mais benéficos em certas situações, mas mais prejudiciais noutras.

- Uso de diferentes tipos de regiões, e aplicação em outros ramos, como por exemplo passeios de bicicleta.

- Introduzir mais e diferentes objetivos, de forma a diversificar os problemas e consequentes soluções.

- Uso desta informação em aplicações móveis, de forma a poder testar em tempo real a resolução deste tipo de problemas e, posteriormente, tê-las disponíveis ao público.

Referências

Abdoun, O., Abouchabaka, J., & Tajani, C. (2012). Analyzing the performance of mutation operators to solve the travelling salesman problem. *arXiv preprint arXiv:1203.3099*.

Abreu, J. C., & de Souza Pereira, A. A. (2019). Meta-heurística multiobjetivo para sequenciamento de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência. *Revista Científica UNIFAGOC-Multidisciplinar*, *3*(1).

AbuSalim, S. W. G., Ibrahim, R., Zainuri Saringat, M., Jamel, S., & Abdul Wahab, J. (2020). Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization. *IOP Conference Series: Materials Science and Engineering*, *917*(1), 012077. https://doi.org/10.1088/1757-899X/917/1/012077

Barrico, C. M. C. S. (1998). *Uma abordagem ao problema de caminho mais curto multiobjectivo–Aplicação ao problema de encaminhamento em redes integradas de comunicações* (Doctoral dissertation, Dissertação de Mestrado, Departamento de Engenharia Electrotécnica, Faculdade de Ciências e Tecnologia, Universidade de Coimbra, Portugal).

Darwin, C. (1859). *The origin of species by means of natural selection* (Vol. 247, p. 1859). EA Weeks.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.

de Carvalho, B. M. P. S. (2008). Algoritmo de Dijkstra. *Universidade de Coimbra, Coimbra, Portugal*.

Demeyer, S., Goedgebeur, J., Audenaert, P., Pickavet, M., & Demeester, P. (2013). Speeding up Martins' algorithm for multiple objective shortest path problems. *4or*, *11*(4), 323-348.

Dorigo, M. (1992) Optimization, Learning and Natural Algorithms. Ph.D. Thesis, Politecnico di Milano, Italian.

Eren, Y., Küçükdemiral, İ. B., & Üstoğlu, İ. (2017). Introduction to optimization. In *Optimization in renewable energy systems* (pp. 27-74). Butterworth-Heinemann.

Floyd, R. W. (1962). On ambiguity in phrase structure languages. *Communications of the ACM*, *5*(10), 526.

Gass, S. I. (2011). George b. dantzig. In *Profiles in Operations Research* (pp. 217-240). Springer, Boston, MA.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, *13*(5), 533-549.

Hashimoto, K. (2004). *Técnicas de otimização combinatória multiobjetivo aplicadas na estimação do desempenho elétrico de redes de distribuição* (Doctoral dissertation, Universidade de São Paulo).

Holland, J. H. (1992). Genetic Algorithms. *Scientific American*, *267*(1), 66–73. http://www.jstor.org/stable/24939139

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

King, R. A., Deb, K., & Rughooputh, H. C. S. (2010). Comparison of NSGA-II and SPEA2 on the multiobjective environmental/economic dispatch problem. *University of Mauritius Research Journal*, *16*(1), 485-511.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson Education India.

Maniezzo, V., Dorigo, M., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, *26*(1), 29-41.

Martins, E. Q. V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, *16*(2), 236-245.

Ning, J., Zhang, C., Sun, P., & Feng, Y. (2018). Comparative study of ant colony algorithms for multi-objective optimization. *Information*, *10*(1), 11.

Nunes, A. S. P. (2022). *Healthy Track: healthy route recommendation* (Doctoral dissertation).

Paixão, J. M., & Santos, J. L. (2013). Labeling methods for the general case of the multi-objective shortest path problem–a computational study. In *Computational Intelligence and Decision Making* (pp. 489-502). Springer, Dordrecht.

Raupp, F. M. P. (2012). *Um método heurístico para o problema de escalonamento multiobjetivo em vários ambientes de máquinas* (Doctoral dissertation, PUC-Rio).

Richard, B. (1958). On a routing problem. *Quarterly of Applied Mathematics*, *16*(1), 87-90.

Sedgewick, R., & Wayne, K. (2011). *Algorithms, 4th Edition*. Addison-Wesley.

Xue, Y. (2018). Mobile robot path planning with a non-dominated sorting genetic algorithm. *Applied Sciences*, *8*(11), 2253.

Zhao, F., Lei, W., Ma, W., Liu, Y., & Zhang, C. (2016). An improved SPEA2 algorithm with adaptive selection of evolutionary operators scheme for multiobjective optimization problems. *Mathematical Problems in Engineering*, *2016*.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, *103*.