NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE MATEMÁTICA

SARA PATRÍCIA FARINHA CRUZ

Licenciada em Matemática

UMA HEURÍSTICA PARA UM PROBLEMA DE LOCALIZAÇÃO DE *HUBS* COM CAPACIDADES

MESTRADO EM MATEMÁTICA E APLICAÇÕES

Universidade NOVA de Lisboa
setembro, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO
DE MATEMÁTICA

UMA HEURÍSTICA PARA UM PROBLEMA DE LOCALIZAÇÃO DE *HUBS* COM CAPACIDADES

SARA PATRÍCIA FARINHA CRUZ

Licenciada em Matemática

Orientadora: Isabel Cristina Silva Correia
Professora, Universidade NOVA de Lisboa

MESTRADO EM MATEMÁTICA E APLICAÇÕES

Universidade NOVA de Lisboa
setembro, 2023

Uma heurística para um problema de localização de *hubs* com capacidades

Aos meus pais

Agradecimentos

Gostaria de começar por agradecer à minha orientadora, Professora Isabel Correia, por toda a ajuda, disponibilidade e ensinamentos que transmitiu ao longo deste ano e que foram fundamenteis para a realização desta dissertação.

Queria também agradecer à Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa, em particular ao Departamento de Matemática, por todas as oportunidades e conhecimentos que me foram transmitidos, ao longo destes anos.

Por último, gostaria de deixar um enorme agradecimento aos meus amigos e familiares, em especial aos meus pais, por todo o apoio e suporte que me deram.

Resumo

Num problema de localização de *hubs* existe um conjunto de nós, que devem efetuar trocas de fluxo entre si. Estas trocas poder-se-iam realizar através do transporte direto entre todos os nós, no entanto este tipo de solução não é eficiente porque requer muitas ligações. Uma solução mais eficiente, consiste em selecionar um subconjunto de nós, para serem *hubs*, tendo estes como função receber fluxo proveniente de outros nós, eventualmente processá-lo e redireciona-lo para outros *hubs*, ou para o seu destino final.

A utilização deste tipo de solução permite reduzir os custos de transporte, pois envolve um menor número de ligações diretas entre pares de nós. Adicionalmente, a circulação de maiores quantidades de fluxo entre *hubs* permite a utilização de veículos de maior capacidade, podendo usufruir-se dos benefícios das economias de escala.

Assim, num problema de localização de *hubs* o objetivo consiste em determinar o subconjunto de nós, onde devem ser instalados os *hubs*, a afetação dos restantes nós aos *hubs* e o modo de enviar o fluxo na rede, de modo a otimizar uma dada função objetivo.

Estes problemas têm sido aplicados em várias áreas, entre as quais se destacam as áreas do transporte, das telecomunicações e da recolha e distribuição de correio.

Nesta dissertação estudou-se um problema de localização de *hubs* com dois tipos de capacidades tendo-se adaptado a este novo problema uma formulação existente na literatura para o problema clássico de localização de *hubs* com capacidades. Desenvolveu-se também para o problema em estudo um algoritmo genético com chaves aleatórias enviesadas. A formulação e o algoritmo foram testados computacionalmente num conjunto de instâncias de teste. Com o algoritmo foi possível determinar soluções admissíveis para todas as instâncias, enquanto que a formulação não permitiu obter soluções para algumas instâncias de maior dimensão dentro do tempo limite estabelecido.

Palavras-chave: *Hubs*, localização de *hubs* com capacidades, programação linear inteira mista, heurística, algoritmo genético com chaves aleatórias enviesado

Abstract

In a hub location problem, there is a set of nodes that must exchange flows. These exchanges could be carried out through direct transport between nodes, however this type of solution is not efficient because it requires a huge number of connections. A more efficient solution consists of selecting a subset of nodes to be hubs, with the function of receiving flow from other nodes and redirecting it to other hubs, or to its final destination.

The use of this type of solution makes it possible to reduce transportation costs, as it involves a smaller number of direct connections between nodes. Additionally, the circulation of greater amounts of flow between hubs allows the use of vehicles with larger capacity, making it possible to take advantage of the benefits of scale economies.

Hence, in a hub location problem, the objective is to determine the subset of nodes, that should become *hubs*, the assignment of the other nodes to the *hubs* and the routing of the flow in the network, in order to optimize an objective function, that typically consist on the minimization of the total costs.

These problems have been applied in several areas, such as transportation, telecommunications and mail delivery.

In this dissertation, a *hub* location problem with two types of capacities was studied, for which we adapted a formulation existing in the literature for the classic capacitated *hub* location problem. A Biased Random Key Genetic Algorithm was also developed for the problem under study. The formulation and the algorithm were computationally tested on a set of test instances. With the algorithm it was possible to determine feasible solutions for all instances, while the formulation did not allow to obtain solutions for some larger instances within the established time limit.

Keywords: *Hubs*, capacitated *hub* locating problem, mixed-integer linear programming, heuristics, biased random key genetic algorithm

Índice

Índice de Figuras

Índice de Tabelas

1

Introdução

Os problemas de localização constituem uma classe importante de problemas de Otimização Combinatória, uma vez que apresentam inúmeras aplicações em situações reais. De uma forma muito sucinta, um problema de localização pode ser descrito como sendo um problema no qual se pretende determinar onde instalar um conjunto de serviços ou equipamentos e a sua afetação aos clientes de modo a otimizar uma dada função objetivo. Este tipo de problemas tem gerado um enorme interesse entre os investigadores, sendo possível encontrar um elevado número de publicações sobre este assunto. Num livro recente, editado por Laporte, Nickel e Saldanha-da-Gama [28], podem ser encontrados estudos sobre vários problemas de localização e as suas aplicações.

Figura 1.1: Conjunto de nós que necessitam de efetuar trocas de fluxo entre si

Os problemas de localização de *hubs* constituem uma classe de problemas de localização. Num problema de localização de *hubs* existe um conjunto de nós, ou localizações, como as que se encontram na figura 1.1, que devem efetuar trocas de fluxo (pessoas, correio, mercadoria, etc.) entre si. Estas trocas de fluxo poder-se-iam realizar através do transporte direto entre todos os pares de nós, no entanto, este tipo de solução não é muito eficiente, apresentando custos financeiros e ambientais elevados devido ao elevado

número de ligações que requer. Uma solução mais eficiente, consiste em selecionar um subconjunto de nós, para serem *hubs*, tendo estes como função receber fluxo proveniente de outros nós, eventualmente processá-lo e redireciona-lo depois para outros *hubs*, ou para o seu destino final.

Na figura 1.2 apresenta-se uma rede que possibilita a circulação de fluxo entre todos os pares nós apresentados na figura 1.1. Nesta rede os nós quatro, cinco, sete e oito representam *hubs*, que neste caso estão todos ligados entre si. Os restantes nós são designados por não *hubs*. Nesta figura, o fluxo que é enviado, por exemplo, do não *hub* um para o não *hub* nove tem de passar pelos *hubs* quatro e oito. Por outro lado, o fluxo que é enviado de dez para seis tem de passar apenas pelo *hub* sete.

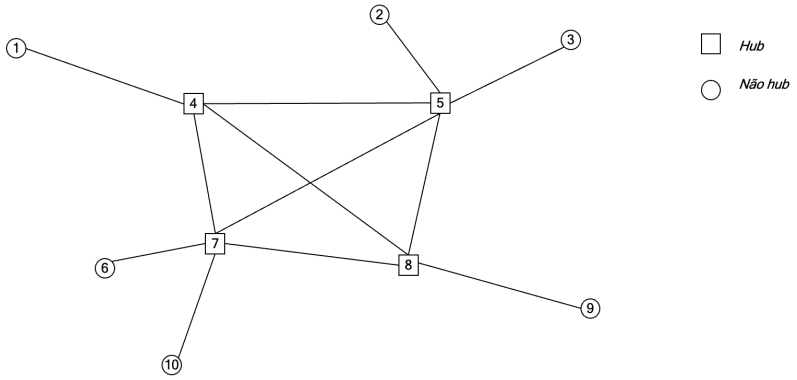


Figura 1.2: Uma rede de *hubs* e não *hubs* para o conjunto de nós da figura 1.1

A utilização de uma rede, como aquela que se encontra na figura 1.2, permite reduzir os custos de transporte, uma vez que envolve um menor número de ligações diretas entre pares de nós. Adicionalmente, a circulação de maiores quantidades de fluxo entre pares de *hubs* permite a utilização de veículos de maior capacidade, podendo usufruir-se assim dos benefícios das economias de escala nos custos de transporte.

Os problemas de localização de *hubs* têm tido um enorme impacto em aplicações na área do transporte de mercadoria e passageiros. Um exemplo clássico da utilização de *hubs*, pode ser encontrado no transporte aéreo. Como não existem ligações aéreas entre todos os pares de cidades, é comum os passageiros terem de viajar até um aeroporto, que seja *hub*, efetuarem uma escala e viajarem depois para outro *hub*, ou para o seu destino final.

Outra aplicação muito conhecida dos problemas de localização de *hubs* está relacionada com a recolha e distribuição de correio [19]. Neste caso, os *hubs* representam centros que recolhem o correio proveniente dos não *hubs*, processando-o através da sua ordenação e separação por código postal e reencaminhando-o para outros *hubs* ou para o seu destino final.

As redes de telecomunicação constituem outro exemplo da utilização de *hubs*. Estas redes são compostas por nós que transmitem mensagens ou informações através de linhas de comunicação (por exemplo, fibra ótica, ligações satélite). O desenho de uma rede

eficiente desta natureza é um problema difícil, principalmente quando há muitos nós a trocar informação, sendo o recurso a *hubs* uma prática comum.

Tal como acontece nos problemas de localização clássicos, os problemas de localização de *hubs* também podem apresentar restrições de capacidade, que normalmente, dizem respeito à quantidade máxima de fluxo que cada *hub* pode processar. No entanto, em muitas das aplicações dos problemas de localização de *hubs*, para além destas restrições de capacidade é necessário limitar o número de não *hubs*, que podem estar afetos a cada *hub*. Um exemplo desta situação ocorre quando os *hubs* representam portos marítimos. Num porto marítimo, para que seja efetuado o desembarque de mercadorias é necessário utilizar-se cais de desembarque e gruas, que operam neles, sendo que estas infraestruturas existem em número limitado. Este tipo de limitações, também se pode encontrar no transporte aéreo, visto que cada aeroporto contém um número limitado de mangas e *slots* para realizar a suas operações.

Como na literatura sobre problemas de localização de *hubs* não se encontrou nenhum estudo envolvendo os dois tipos de restrições de capacidade referidas optou-se, neste dissertação, por estudar este novo problema.

Assim, neste trabalho começou-se por adaptar, a este novo problema, uma formulação em Programação Linear Mista existente na literatura para o problema clássico de localização de *hubs* com capacidades. Esta formulação foi testada computacionalmente num conjunto de instâncias de teste retiradas da literatura. Como a utilização da formulação não permitiu a obtenção de soluções ótimas para todas as instâncias de teste, foi desenvolvida uma heurística baseada em chaves aleatórias com o objetivo de determinar boas soluções admissíveis para este problema.

Esta dissertação está organizada em cinco capítulos, sendo que neste primeiro capítulo é apresentada uma pequena introdução do problema em estudo. No capítulo 2 é apresentada uma revisão da literatura. O problema em estudo é apresentado no capítulo 3. No capítulo 4 descreve-se a heurística para o problema. E finalmente, no capítulo 5 são apresentadas as conclusões sobre o trabalho desenvolvido e são lançadas algumas direções para o trabalho futuro.

2

Revisão da literatura

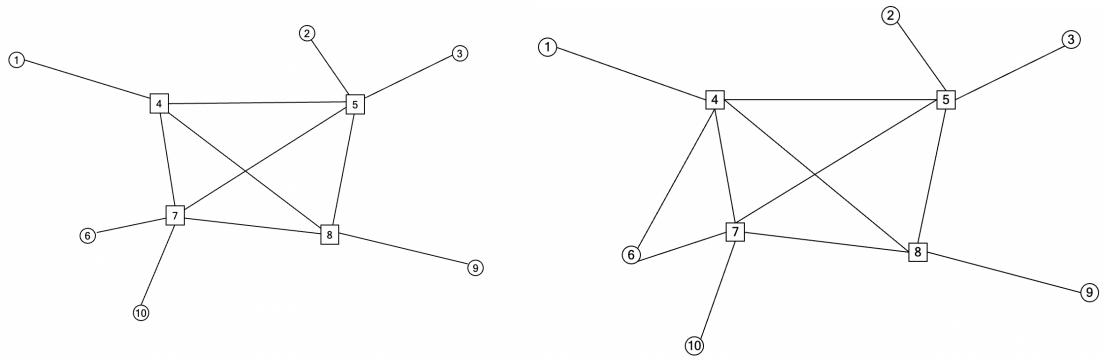
Num trabalho publicado em 1987, O'Kelly [33] apresentou aquela que é considerada a primeira formulação matemática para um problema de localização de *hubs* envolvendo o estudo de uma rede de transporte aéreo. Desde então, a literatura relativa a este tipo de problemas tem vindo a crescer, tendo já sido estudadas muitas variantes e muitas aplicações destes problemas. Nos trabalhos [3, 8, 20] podem ser encontradas muitas referências nesta área.

Na secção 2.1 deste capítulo, apresentam-se algumas das características que permitem distinguir alguns dos problemas de localização de *hubs* que tem sido estudados na literatura. Na secção 2.2 são revistos alguns problemas de localização de *hubs* com capacidades, bem como as técnicas que lhes tem sido aplicadas para a sua resolução.

2.1 Características dos problemas de localização de *hubs*

2.1.1 Afetação dos não *hubs* aos *hubs*

Os problemas de localização de *hubs* decompõem-se em duas classes, quando se considera a forma como é realizada a afetação dos não *hubs* aos *hubs*.



(a) Afetação total (b) Afetação parcial

Figura 2.1: Afetação dos não *hubs* aos *hubs*

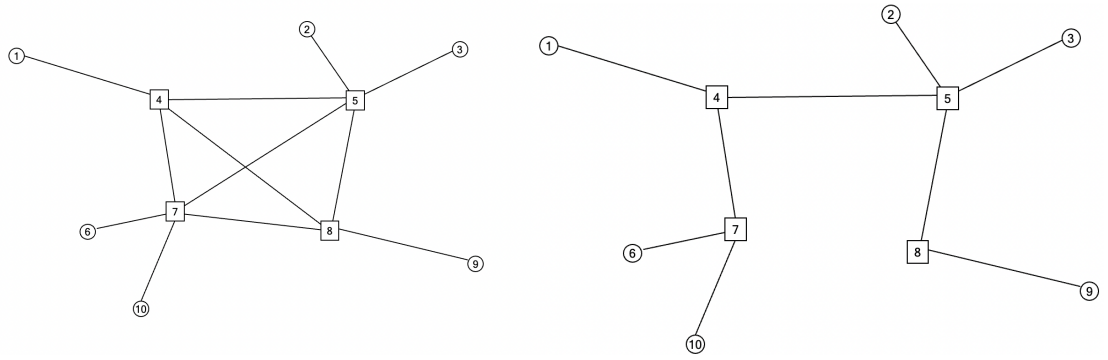
Assim, podemos encontrar na literatura problemas de localização de *hubs*, que lidam com afetações totais [1, 7, 19, 34, 42], ou seja onde cada não *hub* é afeto a um e um só *hub*, tal como acontece na rede representada pela figura 2.1(a). A possibilidade de afetar um não *hub* a mais do que um *hub* ou seja, a consideração de afetações parciais tal como se ilustra na figura 2.1(b), também tem sido adotada em muitos trabalhos [7, 16, 25, 26, 32].

Na generalidade dos problemas de localização de *hubs* admite-se que não existem ligações diretas entre pares de não *hubs*.

2.1.2 Estrutura do subgrafo induzido pelos *hubs*

Na grande maioria dos trabalhos publicados sobre estes problemas o subgrafo induzido pelos *hubs* é completo, ou seja, existe uma ligação direta entre cada par de *hubs*, tal como acontece na rede representada na figura 2.2(a).

No entanto, também têm sido estudados problemas de localização de *hubs* onde o subgrafo induzido pelos *hubs* é uma árvore ou seja, é um grafo conexo e sem ciclos, tal como acontece na rede representada pela figura 2.2(b). Como exemplos deste trabalhos temos [10, 29].



(a) Grafo completo (b) Árvore

Figura 2.2: Estrutura do subgrafo induzido pelos *hubs*

2.1.3 Capacidade dos *hubs*

Encontra-se na literatura, um elevado número de trabalhos publicados sobre problemas de localização de *hubs* sem capacidades, indicando-se como exemplos destes trabalhos as referências [40, 35]. Nestes problemas não existem quaisquer restrições em relação ao fluxo total que cada *hub* pode processar ou ao fluxo total que pode atravessar cada par de nós.

Porém, existem também na literatura problemas de localização de *hubs* com capacidades [7, 4, 31]. Nestes problemas as capacidades podem referir-se (*i*) ao fluxo total máximo que cada *hub* pode receber diretamente dos não *hubs*, (*ii*) ao fluxo máximo que cada *hub* pode receber proveniente dos restantes nós, incluindo os restantes *hubs*, (*iii*) ao fluxo

máximo que é permitido circular entre cada par de *hubs*, (iv) ao fluxo máximo que circula entre um *hub* e um não *hub*.

2.1.4 Tipo de função objetivo

Em relação ao tipo de função objetivo, os problemas de localização de *hubs* surgem habitualmente divididos nas quatro classes seguintes:

1. Problemas de mediana *p-hub*: Fazem parte desta classe os problemas cujo objetivo é minimizar os custos associados ao transporte do fluxo, sabendo que serão instalados *p hubs*. Em [7, 15] encontram-se formulações matemáticas para esta classe dos problemas de localização de *hubs*.

2. Problemas com custos fixos: Nesta classe de problemas de localização de *hubs* o objetivo é minimizar os custos totais. Estes custos são compostos pelo custo fixo da instalação dos *hubs*, mais o custo associado ao transporte de fluxo, sendo que o número de *hubs* a serem instalados é inicialmente desconhecido. Nos artigos [7, 31] podemos encontrar formulações para esta classe de problemas de localização de *hubs*.

3. Problemas de *p-hub* centro: Dado um número de *hubs p* a serem instalados, esta classe dos problemas de localização de *hubs*, tem como objetivo minimizar o custo máximo associado à circulação de fluxo entre cada par de nós. Como exemplos de referencias literárias que lidam com problemas de *p-hub* centro temos [24, 17].

4. Problemas de cobertura *p-hub*: Nesta classe de problemas, o objetivo é minimizar os custos associados ao transporte de fluxo, sendo estabelecido à priori um número *p* de *hubs* a serem instalados e tendo em consideração uma ou mais das seguintes possibilidades:

   - O custo associado ao transporte do fluxo de cada origem para cada destino não deve exceder um determinado valor previamente definido.
   - O custo associado ao transporte de fluxo, numa determinada aresta da rede, não deve ultrapassar um determinado valor previamente definido.
   - O custo de transportar fluxo entre um não *hub* e um *hub* deve não exceder um determinado valor previamente definido.

   No artigo [25] é explorada uma heurística para a obtenção de boas soluções admissíveis para um problema de cobertura *p-hub*.

2.2 Problemas de localização de *hubs* com capacidades

Como é vasta a bibliografia sobre problemas de localização de *hubs* e dado que, o problema em estudo nesta dissertação envolve capacidades, nesta secção serão apenas revistos os problemas de localização de *hubs* com capacidades

A primeira formulação em programação linear mista, para um problema de localização de *hubs* com capacidades e afetação total foi proposta em 1994 por Campbell, no artigo [7], onde também é apresentada a primeira formulação matemática para um problema de localização de *hubs* com capacidades e afetação parcial. Neste trabalho, Campbell referiu-se à capacidade de um *hub* como a quantidade máxima de fluxo que o pode atravessar, ou em alternativa como a quantidade de fluxo não processado que o *hub* pode receber.

Em 1999, Ernst e Krishnammorthy propõem em [19] uma nova formulação em programação linear mista para o problema de localização de *hubs* com capacidades e afetação total, baseando-se numa formulação que os autores já tinham desenvolvido para o problema de mediana *p-hub*. Na formulação proposta por Ernst e Krishnammorthy as restrições de capacidade estão associadas ao fluxo não processado que pode entrar em cada *hub*. Em [19], para além de ser introduzida uma nova formulação para o problema de localização de *hubs* com capacidades, são também desenvolvidas duas heurísticas bastante eficientes para este problema. Uma das heurísticas é baseada em pesquisa local e originou melhores resultados em problemas de menores dimensões. A outra heurística, baseada em *Simulated Annealing*, produziu melhores resultados nos problemas de maiores dimensões. Apesar do trabalho desenvolvido por Ernst e Krishnammorthy em [19] se ter tornado bastante conhecido, só em 2010 é que é publicado um artigo [14], onde a partir de um exemplo é demonstrado que a formulação proposta em [19] estava incompleta. Os autores de [14] propõem um conjunto de restrições para completar o modelo proposto por Ernst e Krishnammorthy.

Ebery, Krishnamoorthy, Ernst e Boland publicam em 2000 um artigo [16], onde introduzem uma nova formulação para os problemas de localização de *hubs* com afetação parcial e onde é limitado o volume de fluxo não processado que cada *hub* pode receber. Neste trabalho é também desenvolvida uma heurística eficiente, baseada no algoritmo do caminho mais curto, com o intuito de se obter limites superiores para o problema em estudo, que são posteriormente incorporados num algoritmo *Branch-and-Bound*.

Em 2005 é proposta em [27] a utilização do algoritmo *Branch-and-Cut* na resolução de problemas de localização de *hubs* com capacidades e afetação total, onde a capacidade é referente à quantidade máxima de fluxo que pode passar em cada *hub*.

Chen publica em 2008 um trabalho [9], onde é introduzida uma nova heurística, para a obtenção de soluções admissíveis para os problemas de localização de *hubs* com capacidades e afetação total, onde cada *hub* não pode receber mais do que uma dada quantidade de fluxo não processado. A heurística introduzida por Chen foi capaz de obter as soluções ótimas para quase todos os problemas de menores dimensões, usados em [19] e obteve resultados melhores do que o método *Simulated Annealing* desenvolvido

em [19], para os problemas de maiores dimensões.

Também em 2008, Randall publica um artigo [38], onde aplica uma heurística baseada em colónia de formigas ao problema de localização de *hubs* com afetação total e onde a quantidade de fluxo, que pode passar por cada *hub* é limitada. Esta heurística obteve bons resultados para o problema, num tempo computacional razoável.

Em 2010, Correia, Nickel e Saldanha-da-Gama abordam pela primeira vez em [13] um problema de localização de *hubs* com afetação total, onde a capacidade de cada *hub* é selecionada, de entre um conjunto de capacidades disponíveis, durante o processo de resolução do problema. Em [13] Correia, Nickel e Saldanha-da-Gama apresentam e comparam, a partir do limite obtido pela relaxação linear, várias formulações para o problema proposto.

Também em 2010 Stanimirović publica um estudo [41], onde é desenvolvido um algoritmo genético para a resolução de problemas de localização de *hubs* com afetação total e capacidades, sendo estas referentes à quantidade de fluxo que pode entrar em cada *hub*. Este algoritmo genético obteve bons resultados, sendo considerado, pelos autores, um algoritmo eficiente em relação à qualidade da solução que produz e ao tempo de execução.

Em 2012, Lin, Lin e Chen [30] desenvolvem um algoritmo genético para a obtenção de soluções admissíveis para o problema de mediana *p-hub*, com afetação parcial e onde existe um limite máximo de fluxo que cada *hub* pode processar, assim como um limite máximo de fluxo que pode atravessar cada aresta da rede. Neste artigo, os autores usam redes aéreas chinesas na realização dos seus testes computacionais, tendo concluído que o algoritmo genético proposto apresenta bons resultados utilizando tempos computacionais razoáveis.

Num trabalho de 2014, Correia, Nickel e Saldanha-da-Gama [12] introduzem um problema de localização de *hubs* com afetação total e capacidades, onde vários produtos devem circular entre os vários nós da rede, havendo um limite de fluxo associado a cada produto, que pode entrar em cada *hub*. Para este problema, os autores apresentam varias formulações, considerando em particular o caso em que cada *hub* apenas pode lidar com fluxo de um produto, assim como o caso em que cada *hub* pode lidar com o fluxo de todos os produtos.

Em 2018, Correia, Nickel e Saldanha-da-Gama [11] apresentam um modelo matemática para um problema estocástico de localização de *hubs* com capacidades, com afetação parcial e multi-período. Neste modelo o objetivo passa por minimizar o custo total esperado, considerando um horizonte temporal dividido em vários períodos e admitindo incerteza na quantidade de fluxo que necessita de ser transportada entre cada par de nós.

3

Problema em estudo

Neste trabalho é introduzido um problema de localização de *hubs*, com afetação total e capacidades associadas à quantidade de fluxo não processado, que pode entrar em cada *hub* e ao número de não *hubs* que podem estar ligados a cada *hub*. Neste problema, o número de *hubs* a serem instalados é desconhecido à priori e são tidos em conta os custos da instalação dos *hubs* assim como os custos envolvidos na circulação de fluxo.

Assim, na secção 3.1 deste capítulo é introduzida uma formulação para o problema em estudo. Na secção 3.2 são apresentados testes de pré-processamento com o objetivo de tentar reduzir a dimensão do problema. Na secção 3.3 são apresentados os resultados das experiências computacionais desenvolvidas para testar a formulação e os testes de pré-processamento.

3.1 Formulação

Em 1999, Ernst e Krishnammorthy [19], propõem uma formulação (EK), para o problema de localização de *hubs*, com afetação total, onde cada *hub* não pode receber mais do que uma certa quantidade de fluxo não processado proveniente dos não *hubs* e onde o objetivo é minimizar a soma dos custos associados à instalação dos *hubs* com os custos associados ao transporte do fluxo. Apresentam-se em seguida os parâmetros usado na formulação (EK):

- $N = \{1, ..., n\}$: conjunto dos nós que efetuam trocas de fluxo entre si.
- w_{ij}: quantidade de fluxo que deve ser enviada do nó i para o nó j ($i, j \in N$).
- d_{ij}: distância entre o nó i e o nó j ($i, j \in N$).
- f_i: custo fixo de instalar um *hub* no nó i ($i \in N$).
- Γ_i: quantidade máxima de fluxo que o *hub* instalado no nó i ($i \in N$) pode receber diretamente dos não *hubs*.

- O_i: fluxo total com origem no nó i ($i \in N$), sendo definido como:

$$O_i = \sum_{j \in N} w_{ij}$$

- D_i: fluxo total com destino no nó i ($i \in N$) e portanto é definido como:

$$D_i = \sum_{j \in N} w_{ji}$$

- χ: custo de transportar uma unidade de fluxo, por unidade de distância, entre um nó de origem e um *hub*.

- α: custo de transportar uma unidade de fluxo, por unidade de distância, entre dois *hubs*.

- δ: custo de transportar uma unidade de fluxo, por unidade de distância, entre um *hub* e um nó de destino.

Com o objetivo de incluir economias de escala nos custos de transporte entre *hubs* é usual considerar $0 < \alpha < 1$, $\chi > \alpha$ e $\delta > \alpha$.

Com o objetivo de se ilustrar a contabilização dos custos de transporte considere-se a rede da figura 3.1.

Figura 3.1: Rede de *hubs* e não *hubs*

O custo total de enviar o fluxo w_{ij} do nó i, para o nó j é calculado de acordo com a expressão (3.1), onde $\chi d_{ik} w_{ij}$, $\alpha d_{km} w_{ij}$ e $\delta d_{mj} w_{ij}$ representam respetivamente, os custos da deslocação do fluxo w_{ij} entre o não *hub* i e o *hub* k, entre os *hubs* k e m e entre o *hub* m e o não *hub* j.

$$\chi d_{ik} w_{ij} + \alpha d_{km} w_{ij} + \delta d_{mj} w_{ij} \tag{3.1}$$

Nestes problemas considera-se que a matriz de fluxos pode não ser simétrica e que podemos ter $w_{ii} \neq 0$ ($i \in N$). Em relação à matriz das distancias, admite-se que satisfaz a propriedade da desigualdade triangular e por isso o fluxo enviado do nó i para o nó j passa no máximo por dois *hubs*.

Na formulação (EK) são consideradas as seguintes variáveis de decisão:

y_{kl}^{i} = quantidade de fluxo com origem no nó i, que passa primeiro pelo *hub* k e de seguida pelo *hub* l ($i, k, l \in N$).

$$z_{ik} = \begin{cases} 1, & \text{se o não } \textit{hub} \text{ i é afeto ao } \textit{hub} \text{ k} \\ 0, & \text{no caso contrario} \end{cases} \qquad (i,k \in N)$$

$z_{kk} = 1$ indica que o nó k é *hub* ($k \in N$).

A formulação (EK) é dada por:

$$(EK) \qquad min \quad \sum_{i\in N}\sum_{k\in N} d_{ik}(\chi O_i + \delta D_i)z_{ik} + \sum_{i\in N}\sum_{k\in N}\sum_{l\in N} \alpha d_{kl} y^i_{kl} + \sum_{k\in N} f_k z_{kk} \tag{3.2}$$

$$s.a. \quad \sum_{k\in N} z_{ik} = 1 \qquad \forall i \in N \tag{3.3}$$

$$z_{ik} \leq z_{kk} \qquad \forall i,k \in N \tag{3.4}$$

$$\sum_{i\in N} O_i z_{ik} \leq \Gamma_k z_{kk} \qquad \forall k \in N \tag{3.5}$$

$$O_i z_{ik} + \sum_{l\in N} y^i_{lk} = \sum_{l\in N} y^i_{kl} + \sum_{j\in N} w_{ij} z_{jk} \qquad \forall i,k \in N \tag{3.6}$$

$$z_{ik} \in \{0,1\} \qquad \forall i,k \in N \tag{3.7}$$

$$y^i_{kl} \geq 0 \qquad \forall i,k,l \in N \tag{3.8}$$

A função objetivo (3.2) correspondente à minimização dos custos totais apresenta três componentes. A primeira componente contém os custos de envio de fluxo entre *hubs* e não *hubs*. A segunda componente contém os custos de circulação de fluxo entre *hubs*. Finalmente, a terceira componente contém os custos fixos da instalação dos *hubs*.

As restrições (3.3) obrigam a que cada nó ou seja um *hub*, ou esteja afeto a um *hub*. As restrições (3.4) garantem que se um nó i está afeto a um nó k, então k tem de ser um *hub*. As restrições (3.5) correspondem às restrições de capacidade dos *hubs* e garantem que a quantidade de fluxo que o *hub* recebe dos não *hubs* que lhes estão afetos juntamente com o fluxo nele originado não ultrapassa a sua capacidade. As restrições (3.6) são as equações de conservação do fluxo. No primeiro membro destas restrições encontra-se todo o fluxo com origem em i e que chega a k de forma direta, ou através de um *hub* l. No segundo membro encontra-se o fluxo com origem em i que sai de k para outros *hubs* e também o fluxo que é enviado para outros nós afetos a k. Assim, estas restrições garantem que todo o fluxo que entra no nó k proveniente de i é igual ao fluxo que sai de k e tem origem em i. Por último, as restrições (3.7) e (3.8) representam restrições de domínio, para as variáveis de (EK).

Apesar da formulação (EK) ter sido muito usada em diferentes trabalhos, apenas em 2010 surge um artigo [14], onde através de um exemplo é demonstrado que esta

formulação estava incompleta, permitindo em certos casos que, para $k,l,i \in N$, a variável y_{kl}^{i} seja diferente de zero, enquanto z_{ik} é igual a zero. Em [14] é ainda demonstrado que as restrições em falta na formulação proposta por Ernst e Krishnamoorthy são dadas pelas desigualdades (3.9)

$$\sum_{l \in N, l \neq k} y_{kl}^{i} \leq O_i z_{ik} \qquad \forall i,k \in N. \tag{3.9}$$

No problema em estudo nesta dissertação é também necessário limitar o número de não *hubs* que estão afetos a cada *hub* o que é possível juntando à formulação (EK) completada pelas restrições (3.9) o seguinte conjunto de restrições

$$\sum_{i \in N \setminus \{k\}} z_{ik} \leq L_k z_{kk} \qquad \forall k \in N, \tag{3.10}$$

onde

L_k: número máximo de não *hubs* que podem estar afetos a cada *hub* k ($k \in N$).

Assim, a formulação para o problema em estudo neste trabalho é dada por:

$$(P) \qquad min \quad \sum_{i \in N} \sum_{k \in N} d_{ik}(\chi O_i + \delta D_i) z_{ik} + \sum_{i \in N} \sum_{k \in N} \sum_{l \in N} \alpha d_{kl} y_{kl}^{i} + \sum_{k \in N} f_k z_{kk}$$

$$s.a. \quad \sum_{k \in N} z_{ik} = 1 \qquad \forall i \in N$$

$$z_{ik} \leq z_{kk} \qquad \forall i,k \in N$$

$$\sum_{i \in N} O_i z_{ik} \leq \Gamma_k z_{kk} \qquad \forall k \in N$$

$$O_i z_{ik} + \sum_{l \in N} y_{lk}^{i} = \sum_{l \in N} y_{kl}^{i} + \sum_{j \in N} w_{ij} z_{jk} \qquad \forall i,k \in N|$$

$$\sum_{l \in N, l \neq k} y_{kl}^{i} \leq O_i z_{ik} \qquad \forall i,k \in N$$

$$\sum_{i \in N \setminus \{k\}} z_{ik} \leq L_k z_{kk} \qquad \forall k \in N$$

$$z_{ik} \in \{0,1\} \qquad \forall i,k \in N$$

$$y_{kl}^{i} \geq 0 \qquad \forall i,k,l \in N$$

3.2 Pré-processamento

Nesta secção são apresentados alguns testes de pré-processamento ao modelo desenvolvido na secção anterior com o objetivo de tentar fixar algumas das variáveis e assim reduzir a dimensão do modelo.

Teste 1: Se $\Gamma_k < O_k$ para um nó $k \in N$, então o nó k, não pode ser *hub*, dado que não consegue processar o fluxo nele originado. Assim devemos fixar as variáveis seguintes:

$$z_{ik} = 0 \qquad i \in N$$

$$y^i_{kl} = 0 \qquad i,l \in N$$

$$y^i_{lk} = 0 \qquad i,l \in N$$

Teste 2: Se $\Gamma_k < O_i$ e $\Gamma_i \geq O_i$ para um nó $i \in N$ e para todo o nó $k \in N\backslash\{i\}$, então o nó i têm de ser *hub*, visto que não existe outro nó com capacidade para processar o seu fluxo. Deste modo, devemos fixar as seguintes variáveis:

$$z_{ii} = 1$$

$$z_{ik} = 0$$

$$y^i_{kl} = 0 \qquad l \in N$$

Teste 3: Se $\Gamma_k - O_k < O_i$, para $i \in N$ e $k \in N\backslash\{i\}$, então o nó k não tem capacidade de processar o fluxo originado em i. Assim, devemos fixar as seguintes variáveis:

$$z_{ik} = 0$$
$$y^i_{kl} = 0 \qquad l \in N$$

3.3 Experiência computacional

Nesta secção são apresentados os resultados computacionais obtidos com o intuito de testar o modelo matemático desenvolvido na secção 3.1, assim como o pré-processamento desenvolvido em 3.2. Nas experiências realizadas foi utilizada a versão 22.1 e o ambiente OPL do *solver* IBM ILOG CPLEX Optimization Studio num computador com o processador M1 da Apple, que apresenta 3.2GHz e 16GB de memoria RAM. Consideraram-se para o CPLEX todos os parâmetros pré-definidos excetuando o tempo máximo de execução.

Esta secção está dividida em duas subsecções. Na subsecção 3.3.1 é descrito o conjunto de dados utilizados nas experiências computacionais e na subsecção 3.3.2 são apresentados os resultados obtidos nestas experiências.

3.3.1 Descrição das instâncias de teste

Para a obtenção dos resultados computacionais foi utilizado um conjunto de dados conhecido na literatura como *Australian Post* (*AP*), que teve a sua primeira aparição em 1996 num trabalho de Ernst e Krishnamoorthy [18] e que pode ser obtido em [36].

O conjunto de dados *AP* é composto por duzentas coordenadas, que representam duzentos distritos postais australianos e pelo volume de correspondência, que é enviado entre pares de distritos postais, incluindo o volume de correspondência enviado de um distrito postal para ele próprio. No conjunto de dados *AP* são ainda definidos $\chi = 3$, $\alpha = 0.75$ e $\delta = 2$, o que quer dizer que, no conjunto de dados *AP* o custo unitário da recolha da correspondência dos distritos postais, para os centros de triagem (*hubs*) é superior ao custo unitário da distribuição do correio dos centros de triagem para os distritos postais de destino, que por sua vez é superior ao custo da circulação de fluxo entre centros de triagem.

Como forma de serem gerados instâncias de menores dimensões a partir do conjunto de dados completo, são combinados os nós e os seus fluxos, tal como descrito em [18], conseguindo-se assim gerar instâncias com $n \in \{10, 20, 25, 40, 50, 60, 70, 75, 90, 100, 125, 150, 175, 200\}$ nós.

Para cada instância o conjunto de dados *AP* define também, os custos associados à instalação de *hubs*, havendo dois tipos de custos fixos para cada dimensão da instância. Nas instâncias onde os custos fixos são do tipo T, os nós que recebem e enviam maiores quantidades de fluxo apresentam custos de instalação de *hubs* mais elevados, tornando este tipo de instâncias difíceis de resolver, uma vez que torna difícil para o modelo selecionar os nós com maior afluência de fluxo, para se tornarem *hubs*. Nas instâncias onde os custos fixos são do tipo L, esta tendência não se verifica.

Em cada instância do conjunto de dados *AP* são ainda definidas as capacidades $\Gamma_k (k \in N)$ máximas de fluxo não processado que cada *hub* pode receber. Para cada dimensão existem instâncias com capacidades do tipo T, onde cada potencial *hub* apresenta uma capacidade mais apertada e instâncias do tipo L, onde a capacidade dos *hubs* não é tão apertada.

Assim, para cada uma das dimensões $n \in \{10, 20, 25, 40, 50, 60, 70, 75, 90, 100, 125, 150, 175, 200\}$, o conjunto de dados *AP* apresenta quatro instâncias, sendo elas as instâncias nLL, nLT, nTL e nTT, onde a primeira maiúscula indica o tipo de custo fixo e a segunda o tipo de capacidade da instância.

É de notar que, em nenhuma das instâncias do conjunto de dados *AP* são definidos os valores L_k $(k \in N)$, sendo por isso necessário gerar estes valores.

Deste modo, para cada nó $(k \in N)$, pertencente a uma instância do conjunto de dados *AP*, com n nós, definiu-se L_k tal como:

$$L_k = \left\lceil \frac{\Gamma_k * n}{\sum_{i=1}^{n} \Gamma_i} \right\rceil + R \tag{3.11}$$

onde R é um número inteiro gerado aleatoriamente entre um e $\lceil 0.3 * n \rceil$ e $\lceil \Delta \rceil$ representa o menor inteiro não inferior a Δ. Com esta expressão pretende-se que nós com maior valor de Γ_k possam ter um maior número de não *hubs* afetos do que nós com menor valor de Γ_k.

3.3.2 Resultados computacionais

Antes de se apresentarem os resultados computacionais optou-se por analisar o número de variáveis e restrições que o modelo (P) apresenta em função do número de nós das instâncias de teste. Estes resultados encontram-se na tabela 3.1.

Tabela 3.1: Número de variáveis e de restrições do problema em estudo

n	nº variáveis binarias	nº variáveis contínuas	nº restrições
10	100	1000	1230
20	400	8000	8860
25	625	15625	16950
40	1600	64000	67320
50	2500	125000	130150
60	3600	216000	223380
70	4900	343000	353010
75	5625	421875	433350
90	8100	729000	745470
100	10000	1000000	1020300
125	15625	1953125	1984750
150	22500	3375000	3420450
175	30625	5359375	5421150
200	40000	8000000	8080600

A tabela 3.2 contém os resultados computacionais obtidos pela formulação (P). Foi dado ao *solver* um tempo máximo de execução de uma hora. Esta tabela apresenta na primeira coluna os nomes das instâncias de teste, que quando seguidas de um asterisco indicam que o *solver* atingiu o tempo limite sem fornecer uma solução com a garantia de ser ótima. Neste caso indica-se na tabela a melhor solução admissível conhecida. Quando o nome da instância não vem seguido de um asterisco, então, para essa instância foi possível a determinação de uma solução ótima dentro do tempo limite. Note-se que não são apresentados resultados para as instâncias com $n \in \{150, 175, 200\}$, porque não foi possível determinar uma solução admissível para estas instâncias dentro do tempo máximo permitido.

Nas segunda e terceira colunas da tabela 3.2 apresentam-se, para o modelo (P), os *hubs* instalados e os valores da função objetivo para as soluções obtidas indicando-se na quarta coluna o tempo, em segundos, necessário para a obtenção de cada uma das soluções.

Na quinta coluna da tabela 3.2 é indicado o *gap* da relaxação linear (em percentagem), sendo este calculado através de 3.12. Nesta expressão o limite superior representa o valor ótimo, ou quando este não é conhecido representa o custo da melhor solução admissível conhecida. Na última coluna apresentam-se os tempos, em segundos, necessários para a obtenção da solução da relaxação linear.

$$gap = \frac{limite\,superior - valor\,ótimo\,da\,relaxação\,linear}{limite\,superior} * 100 \tag{3.12}$$

Na tabela 3.1 podemos observar que o problema em estudo apresenta uma elevada complexidade, contendo um elevado número de variáveis binárias e restrições, para instâncias de grandes dimensões. Este facto pode ser responsável por não ter sido possível a obtenção de soluções ótimas para as instâncias assinaladas com um asterisco e também para a obtenção de qualquer solução admissível para instâncias com cento e cinquenta ou mais nós.

Importa ainda referir que se tentou resolver as instâncias com duzentos nós numa workstation (Texto2*CPU: AMD EPYC™ 7702 (64 Cores 256MB Cache, 2.0GHz to 3.35GHz GHz), 512GB RAM @ 3,200GHz) existente no Centro de Matemática e Aplicações, com uma capacidade superior à do computador utilizado neste trabalho, tendo o *solver* indicado problemas de falta de memória.

Com o intuito de se avaliar de que forma é que a dimensão das instâncias influencia o tempo necessário para a obtenção de soluções ótimas e de que forma influencia o *gap* da relaxação linear foi criada a tabela 3.3. Esta tabela apresenta na primeira coluna o número de nós das instâncias, para as quais foi possível a obtenção de de soluções ótimas. Assim, cada linha da tabela contém os resultados agregados para as instâncias LL, LT, TL e TT para as quais se conhece uma solução ótima. Assim, a linha associada a $n = 10$ contém os resultados agregados para quatro instâncias, enquanto a linha correspondente a $n = 100$ contém resultados agregados para apenas duas instâncias. A segunda coluna da tabela 3.3 contém o número de instâncias para as quais foi possível obter-se a solução ótima. A terceira, quarta e quinta colunas indicam, respetivamente, os tempos médios, mínimos e máximos, em segundos, necessários para a obtenção das soluções ótimas. As colunas seis, sete e oito apresentam, respetivamente, as percentagens do *gap* médio, mínimo e máximo da relaxação linear, em função da dimensão da instância em estudo.

Para se compreender de que forma o tipo de instância (LL, LT, TL, TT) influencia o tempo necessário para a obtenção das soluções ótimas e de que forma influencia o *gap* da relaxação linear criou-se a tabela 3.4. Esta tabela apresenta na primeira coluna os vários tipos de instâncias (LL, LT, TL, TT). Na segunda coluna o número de instâncias para as quais foi possível obter-se a solução ótima e que foram usadas para calcular os valores nas colunas seguintes. Na terceira, quarta e quinta colunas são apresentados, respetivamente, os tempos médios, mínimos e máximos, em segundos, necessários para a obtenção das soluções ótimas. Nas colunas seis, sete e oito encontram-se, respetivamente, as percentagens média, mínima e máxima do *gap* da relaxação linear, em função do tipo de instância em estudo.

Da tabela 3.3 verifica-se, que para todos as instâncias com até setenta e cinco nós, foi possível a determinação de todas as soluções ótimas, sendo que em média o tempo necessário para a obtenção das soluções foi aumentando, com o aumento do número de nós. Deste modo, conclui-se que o aumento do tamanho das instâncias faz com que o

Tabela 3.2: Resultados computacionais para a formulação (P)

Instância	Modelo (P)			Relaxação Linear	
	hubs	valor ótimo	tempo (s)	*gap* (%)	tempo (s)
10LL	3-4-7	224 913,04	0,24	0,56	0,07
10LT	1-4-5-10	250 992,26	0,25	1,22	0,07
10TL	4-5-10	263 399,94	0,24	0,88	0,08
10TT	4-5-10	263 399,94	0,25	0,88	0,07
20LL	6-8-9-14	260 651,71	0,93	3,78	0,42
20LT	6-8-9-14	260 678,86	0,62	2,65	0,14
20TL	6-8-19	296 359,16	0,80	2,04	0,26
20TT	6-8-10-19	319 296,00	0,57	3,18	0,14
25LL	8-14-23	249 883,98	0,83	0,44	1,32
25LT	8-12-14-19	285 766,17	3,02	4,50	0,97
25TL	8-14-23	329 421,83	0,85	0,07	0,83
25TT	9-14-16-25	372 456,57	5,28	4,20	0,44
40LL	12-16-26-29	271 404,35	30,61	2,25	1,45
40LT	12-16-26-30	287 863,92	19,41	3,28	0,97
40TL	14-16-19-36	377 875,71	36,04	2,82	1,60
40TT	6-14-19-40	399 275,39	26,19	3,53	1,76
50LL	15-27-32-35	263 040,55	16,20	0,32	2,82
50LT	13-19-32-46	292 551,51	50,35	2,62	12,40
50TL	3-20-27-41-45	410 211,11	53,08	2,09	11,39
50TT	13-19-25-26	481 912,77	169,78	3,26	14,87
60LL	6-27-33-55	251 303,92	99,81	1,58	5,80
60LT	6-19-33-39-42	272 791,67	71,64	2,35	6,00
60TL	15-19-46-55	304 114,42	24,95	2,66	4,93
60TT	9-26-33-40	375 203,66	113,35	1,76	9,80
70LL	20-30-34-64	267 246,25	219,88	1,52	19,60
70LT	21-36-47-64	267 891,43	113,03	1,99	14,29
70TL	20-22-58-64	335 968,29	64,01	4,00	7,98
70TT	22-24-41-48-65	471 459,81	1 144,55	2,78	33,58
75LL	8-47-55-58	255 429,12	106,52	1,43	23,16
75LT	24-48-52-58	274 606,72	172,00	1,74	24,52
75TL	22-32-48-67-75	383 508,14	223,48	1,33	20,39
75TT	8-12-33-53-74	404 052,99	2 505,06	3,63	22,93
90LL	4-50-58-65	251 845,13	402,63	1,23	72,47
90LT*	25-52-62-65	262 233,77	3 604,76	2,01	137,39
90TL	28-50-77-86	301 478,51	154,15	1,24	54,49
90TT	9-51-59-61-86	375 089,17	234,98	1,91	75,04
100LL	29-52-64-73	257 090,32	942,14	1,46	142,99
100LT	29-52-68-96	267 904,86	1 218,32	2,42	141,26
100TL*	5-19-52-58-61	484 425,47	3 605,97	3,64	313,08
100TT*	5-52-58-61-95	528 139,68	3 751,30	3,21	567,52
125LL*	7-47-59-81-86	255 953,11	3 613,04	2,71	562,63
125LT*	6-42-78-85-87	263 047,78	3 610,82	1,83	522,45
125TL*	3-29-53-80-112	323 407,82	3 731,38	4,00	913,73
125TT*	1-16-47-78-81-87-111	974 124,48	3 859,34	66,28	833,71

Tabela 3.3: Resultados computacionais para a formulação (P) em função de n

n	nº de soluções ótimas	tempo (s)			*gap (%)*		
		média	mínimo	máximo	média	mínimo	máximo
10	4	0,24	0,24	0,25	0,88	0,56	1,22
20	4	0,73	0,57	0,93	2,91	2,04	3,78
25	4	2,50	0,83	5,28	2,30	0,07	4,50
40	4	28,06	19,41	36,04	2,97	2,25	3,53
50	4	72,35	16,20	169,78	2,07	0,32	3,26
60	4	77,44	24,95	113,35	2,09	1,58	2,66
70	4	385,37	64,01	1 144,55	2,57	1,52	4,00
75	4	751,77	106,52	2 505,06	2,03	1,33	3,63
90	3	263,92	154,15	402,63	1,46	1,23	1,91
100	2	1 080,23	942,14	1 218,32	1,94	1,46	2,42

Tabela 3.4: Resultados computacionais para a formulação (P) em função do tipo de instância

tipo	nº de soluções ótimas	tempo (s)			gap (%)		
		média	mínimo	máximo	média	mínimo	máximo
LL	10	181,98	0,24	942,14	1,46	0,32	3,78
LT	9	183,18	0,25	1 218,32	2,53	1,22	4,50
TL	9	61,96	0,24	223,48	1,90	0,07	4,00
TT	9	466,67	0,25	2 505,06	2,79	0,88	4,20

tempo necessário à sua resolução também aumente. Estes resultados não surpreendem dado que, de acordo com a tabela 3.1 o aumento do número de nós de um problema leva a um aumento do número de variáveis e restrições do problema, tornando-o mais difícil de resolver.

Ainda, da tabela 3.3 pode-se inferir que a dimensão da instância em estudo não interfere com o *gap* da relaxação linear. As instâncias com quarenta nós apresentam o *gap* médio mais elevado, enquanto as instâncias com dez nós apresentam o menor *gap* médio.

A partir da tabela 3.4 verifica-se que as instâncias do tipo LL são os mais fáceis de resolver, tendo sido possível a obtenção da solução ótima para dez das onze instâncias do tipo LL apresentados na tabela 3.2, enquanto para cada um dos restantes tipos de instâncias foram apenas obtidas soluções ótimas para nove instâncias. A partir da tabela 3.4 verifica-se também que a seguir às instâncias do tipo LL, as instâncias do tipo TL são os mais fáceis de resolver, visto que apresentam um menor tempo médio para a obtenção das soluções ótimas que os restantes tipos de instâncias. Para além disso, as instâncias do tipo TT são os que apresentam um maior tempo médio para a obtenção das soluções ótimas sendo deste modo consideradas as instâncias mais difíceis de resolver.

Ainda, a partir da tabela 3.4 verifica-se que em média as instâncias do tipo TT são as que apresentam o *gap* médio da relaxação linear mais elevado, enquanto as instâncias do tipo LL são as que apresentam o *gap* médio da relaxação linear mais baixo.

Com o intuito de se testar o pré-processamento desenvolvido em 3.2 realizaram-se várias experiências computacionais a partir das quais se criou a tabela 3.5 com a mesma constituição que a tabela 3.2 apresenta.

Tabela 3.5: Resultados computacionais com pré-processamento

Instâncias	Modelo (P)			Relaxação linear	
	hubs	valor ótimo	tempo (s)	*gap* (%)	tempo (s)
10LL	3-4-7	224 913,04	0,30	0,56	0,33
10LT	1-4-5-10	250 992,26	0,30	1,17	0,39
10TL	4-5-10	263 399,94	0,29	0,88	0,10
10TT	4-5-10	263 399,94	0,28	0,88	0,12
20LL	6-8-9-14	260 651,71	1,07	3,78	0,19
20LT	6-8-9-14	260 678,86	0,63	2,65	0,15
20TL	6-8-19	296 359,16	0,78	2,04	0,42
20TT	6-8-10-19	319 296,00	0,61	3,18	0,18
25LL	8-14-23	249 883,98	0,87	0,44	0,27
25LT	8-12-14-19	285 766,17	3,31	4,12	0,56
25TL	8-14-23	329 421,83	1,19	0,07	0,27
25TT	9-14-16-25	372 456,57	4,56	4,05	0,50
40LL	12-16-26-29	271 404,35	40,35	2,25	5,84
40LT	12-16-26-30	287 863,92	38,98	3,28	1,57
40TL	14-16-19-36	377 875,71	28,80	2,82	1,79
40TT	6-14-19-40	399 275,39	24,92	3,53	1,38
50LL	15-27-32-35	263 040,55	18,93	0,32	2,20
50LT	13-19-32-46	292 551,51	51,47	2,62	3,99
50TL	3-20-27-41-45	410 211,11	76,81	2,09	13,66
50TT	13-19-25-26	481 912,77	167,70	3,26	4,47
60LL	6-27-33-55	251 303,92	95,10	1,58	6,00
60LT	6-19-33-39-42	272 791,67	68,80	2,35	5,44
60TL	15-19-46-55	304 114,42	24,95	2,66	3,79
60TT	9-26-33-40	375 203,66	85,18	1,76	10,09
70LL	20-30-34-64	267 246,25	190,54	1,52	20,20
70LT	21-36-47-64	267 891,43	114,24	1,99	9,60
70TL	20-22-58-64	335 968,29	65,75	4,00	8,24
70TT	22-24-41-48-65	471 459,81	2 663,88	2,78	30,68
75LL	8-47-55-58	255 429,12	105,77	1,43	16,07
75LT	24-48-52-58	274 606,72	167,88	1,74	16,05
75TL	22-32-48-67-75	383 508,14	214,02	1,33	36,71
75TT	8-12-33-53-74	404 052,99	2 491,41	3,63	28,21
90LL	4-50-58-65	251 845,13	359,38	1,23	69,17
90LT*	5-22-28-62-65	263 480,48	3 749,82	2,48	104,64
90TL	28-50-77-86	301 478,51	145,45	1,24	52,93
90TT	9-51-59-61-86	375 089,17	225,01	1,91	49,00
100LL	29-52-64-73	257 090,32	1 029,72	1,46	161,19
100LT	29-52-68-96	267 904,86	1 261,19	2,42	73,07
100TL*	5-19-52-58-61	484 425,47	3 606,56	3,64	366,39
100TT*	5-52-58-61-95	527 649,77	3 606,56	3,12	450,93
125LL*	7-47-59-81-86	255 953,11	3 638,58	2,71	564,83
125LT*	6-42-78-85-87	263 047,78	3 612,30	1,83	448,74
125TL*	3-29-53-80-112	323 407,82	3 610,23	4,00	1 036,97
125TT*	11-71-77-78-88	339 235,33	3 611,02	3,18	1 171,54
150LL*	1-26-62-81-100-113-121-124	583 307,02	3 724,55	57,49	3 265,89
150LT*	1-27-43-75-94-100-109	510 670,75	3 765,10	50,16	4 408,68
150TL*	1-26-62-82-100-121-124	1 374 131,04	3 619,07	78,21	3 512,79
150TT*	1-27-55-86-94-105-111-125	1 177 655,58	3 624,60	72,03	1 242,09

A partir da tabela 3.5 verifica-se que o pré-processamento realizado permitiu (i) aumentar o número de soluções admissíveis obtidas para as instâncias em estudo, com a obtenção de soluções para as instâncias $150LL$, $150LT$, $150TL$ e $150TT$, (ii) melhorar o custo da solução admissível da instância $125TT$. No entanto, o pré-processamento realizado não possibilitou o aumento do número de soluções ótimas obtidas.

Com o objetivo de se realizar uma melhor análise aos resultados obtidos para o pré-processamento, criaram-se as tabelas 3.6 e 3.7, com informação análoga à das tabelas 3.3 e 3.4.

Tabela 3.6: Resultados computacionais com pré-processamento em função de n

n	nº de soluções ótimas	tempo (s)			gap (%)		
		média	mínimo	máximo	média	mínimo	máximo
10	4	0,29	0,28	0,30	0,87	0,56	1,17
20	4	0,77	0,61	1,07	2,91	2,04	3,78
25	4	2,48	0,87	4,56	2,17	0,07	4,12
40	4	33,26	24,92	40,35	2,97	2,25	3,53
50	4	78,73	18,93	167,70	2,07	0,32	3,26
60	4	68,51	24,95	95,10	2,09	1,58	2,66
70	4	758,60	65,75	2 663,88	2,57	1,52	4,00
75	4	744,77	105,77	2 491,41	2,03	1,33	3,63
90	3	243,28	145,45	243,28	1,46	1,23	1,91
100	2	1 145,45	1 029,72	1 261,19	1,94	1,46	2,42

Tabela 3.7: Resultados computacionais com pré-processamento em função do tipo de instância

tipo	nº de soluções ótimas	tempo (s)			gap		
		média	mínimo	máximo	média	mínimo	máximo
LL	10	184,20	0,30	1 029,72	1,46	0,32	3,78
LT	9	189,64	0,30	1 261,19	2,48	1,17	4,12
TL	9	62,00	0,29	214,02	1,90	0,07	4,00
TT	9	1 375,48	0,28	3 624,60	8,61	0,88	72,03

Analisando essas tabelas verifica-se que, tal como ocorreu quando se resolveu o problema sem recorrer ao pré-processamento, ao recorrer-se ao pré-processamento as instâncias de maiores dimensões e do tipo LT e TT continuam a ser os mais difíceis de resolver.

Comparando as tabelas 3.6 e 3.7 com as tabelas 3.3 e 3.4 conclui-se, que apesar do pré-processamento ter permitido a obtenção de soluções admissíveis para um maior número de instâncias de teste, este não foi assim tão vantajoso, visto que em média o tempo necessário à obtenção das soluções ótimas manteve-se praticamente o mesmo, ou até, no caso das instâncias do tipo TT aumentou consideravelmente. Para além disso, o pré-processamento também não possibilitou uma significante diminuição do *gap* médio da relaxação linear, para nenhuma instância com determinado número de nós ou tipo de instância.

Deste modo, com objetivo de se tentar perceber a razão pela qual o pré-processamento desenvolvido não melhorou os resultados obtidos pelo modelo (P), analisou-se o número

de variáveis binárias e contínuas que o pré-processamento permitiu fixar em cada instância. A partir desta análise criaram-se as tabelas 3.8 e 3.9.

A tabela 3.8 apresenta na primeira coluna o número de nós das instâncias de teste. Na segunda, terceira e quarta colunas apresenta, respetivamente, a percentagem média, mínima e máxima de variáveis binárias fixas em função da dimensão da instância. Na quinta, sexta e sétima colunas são indicadas, respetivamente, as percentagens médias, mínimas e máximas de variáveis contínuas fixas em função da dimensão da instância. A tabela 3.9 é idêntica a tabela 3.8, diferindo no facto de as percentagens médias, mínimas e máximas das variáveis fixas virem em função do tipo de instância (LL, LT, TL e TT).

Tabela 3.8: Percentagem de variáveis binárias e contínuas fixas em função de n

n	Variáveis Binárias			Variáveis Contínuas		
	média	mínimo	máximo	média	mínimo	máximo
10	13,50	0,00	27,00	18,50	0,00	37,00
20	0,18	0,01	0,36	25,63	0,75	50,50
25	19,60	1,12	38,08	27,60	1,12	54,08
40	16,13	5,44	26,81	22,38	7,94	36,81
50	9,26	0,28	18,24	12,26	0,28	24,24
60	3,70	0,14	9,86	4,53	0,14	13,19
70	1,12	0,10	2,94	1,12	0,10	2,94
75	3,93	0,11	7,95	5,27	0,11	10,61
90	0,84	0,10	1,86	0,84	0,10	1,86
100	4,22	2,16	6,28	4,45	3,16	7,28
125	0,88	0,04	2,68	1,08	0,04	3,48
150	0,74	0,04	2,22	0,91	0,04	2,89
175	0,60	0,04	1,62	0,75	0,04	2,19
200	1,78	0,06	3,51	2,53	0,06	5,01

Tabela 3.9: Percentagem de variáveis binárias e contínuas fixas em função do tipo de instância

tipo	Variáveis Binárias			Variáveis Contínuas		
	média	mínimo	máximo	média	mínimo	máximo
LL	0,71	0,00	7,94	1,01	0,00	7,94
LT	10,51	0,36	38,08	17,80	0,66	54,08
TL	0,70	0,00	5,44	1,01	0,00	7,94
TT	9,93	0,36	38,08	16,93	0,71	54,08

Com base nas tabelas 3.8 e 3.9 conclui-se que em média a percentagem de variáveis binárias e contínuas fixas nas instâncias em estudo, a partir do pré-processamento, é muito baixa. Isto justifica porque não se obtiveram resultados tão satisfatórios com a implementação do pré-processamento. Note-se, no entanto, que foi nas instâncias do tipo LT e TT que a percentagem média de variáveis fixas foi mais elevada. Uma possível justificação para este tipo de resultados poderá estar relacionada com o facto desta instâncias apresentarem capacidades apertadas e os testes de pré-processamento recorrerem ao valor das capacidades dos nós para fixarem variáveis.

Dado que, nem o modelo (P), nem modelo (P) com o pré-processamento permitiram a obtenção de boas soluções para todas as instâncias em tempos computacionais razoáveis, torna-se fundamental o desenvolvimento de uma heurística que obtenha boas soluções

admissíveis para todas as instâncias em estudo, em tempos computacionais aceitáveis. Assim, no capítulo seguinte é desenvolvida uma heurística para o problema em estudo.

4

Heurística

Neste trabalho é desenvolvido um algoritmo genético com chaves aleatórias enviesado, com o intuito de se obterem boas soluções admissíveis para o problema em estudo.

Deste modo, na secção 4.1 deste capítulo é apresentada uma breve evolução dos algoritmos do tipo genético, começando-se por introduzir os algoritmos genéticos, passando-se depois a apresentar os algoritmos genéticos com chaves aleatórias e os algoritmos genéticos com chaves aleatórias enviesados. Na secção 4.2 é explicada a adaptação dos algoritmos genéticos com chaves aleatórias enviesados ao problema em estudo. Os resultados computacionais obtidos para o algoritmo trabalhado são apresentados na secção 4.3.

4.1 Heurísticas do tipo genético

Algoritmos genéticos

O termo algoritmo genético foi introduzido em 1975 por John Holland em [23]. Este termo refere-se a uma metaheurística utilizada na resolução de problemas de otimização baseada em mecanismos de seleção natural e genética.

Num algoritmo genético as populações de indivíduos evoluem ao longo de várias iterações, denominadas de gerações, até que o(s) critério(s) de paragem seja(m) atingido(s). Cada indivíduo de uma população, também conhecido por cromossoma é representado por um vetor com um número de genes (tamanho) que depende do problema em estudo. Os valores de cada gene são chamados de alelos.

A conversão de um cromossoma numa solução para o problema é feita através de um algoritmo determinístico conhecido por descodificador, que depende do problema em estudo.

Em cada geração os cromossomas mais aptos, isto é aqueles que dão origem a soluções com melhor valor na função objetivo, apresentam uma maior probabilidade de gerarem descendentes. Os descendentes são obtidos através de operadores de cruzamento e de mutação.

Em [39] pode encontrar-se informação mais detalhada sobre os algoritmos genéticos, em particular sobre o funcionamento da seleção dos cromossomas utilizados nos cruzamentos, nas mutações e sobre as diferentes formas de se efetuarem os cruzamentos e as mutações.

Algoritmos genéticos com chaves aleatórias

Apesar do grande sucesso que os algoritmos genéticos têm tido, para determinados problemas podem ser obtidas soluções não admissíveis durante o processo de cruzamento e mutação, o que provoca um aumento no tempo computacional necessário à recuperação da admissibilidade das soluções.

Com o objetivo de colmatar este comportamento existente nos algoritmos genéticos, Bean em 1994 introduziu em [5] os algoritmos genéticos com chaves aleatórias. Nestes, os cromossomas são representados por vetores de números reais gerados aleatoriamente entre zero e um e os descodificadores são utilizados na conversão dos cromossomas em soluções admissíveis.

Nos algoritmos genéticos com chaves aleatórias o primeiro passo consiste na criação da população inicial, composta por p cromossomas, cujos alelos são gerados de forma aleatória e independente. De seguida o descodificador converte os cromossomas em soluções para o problema, para depois, avaliar a qualidade dos cromossomas e dividi-los em dois grupos. Um grupo, denominado de elite, contém os $\lceil p * p_e \rceil$ $(0 < p_e < 1)$ cromossomas mais aptos, onde p_e representa a proporção de elementos elite. O outro grupo, denominado de não elite, contém os restantes cromossomas.

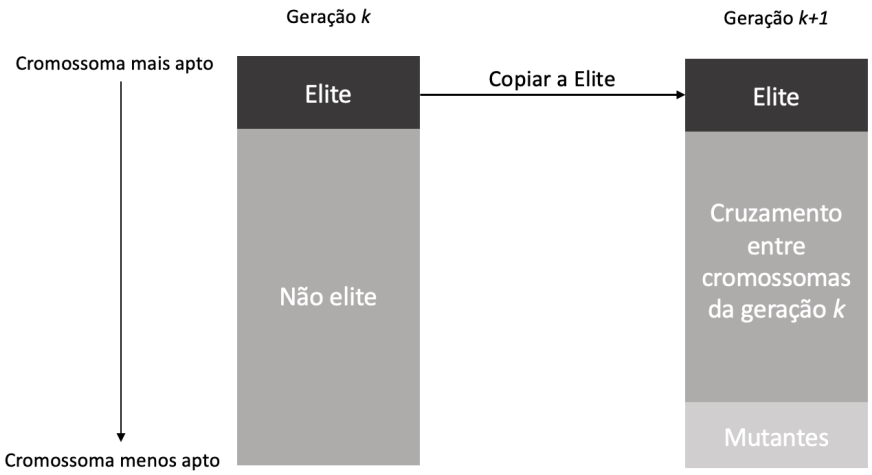


Figura 4.1: Evolução dos indivíduos entre gerações nos algoritmos genéticos com chaves aleatórias

Tal como a figura 4.1 (inspirada numa figura encontrada em [22]) sugere, na criação

das novas gerações os algoritmos genéticos com chaves aleatórias copiam todos os indivíduos elite de uma geração para a geração seguinte, garantindo assim, que as melhores soluções encontradas vão sendo conservadas ao longo das gerações.

Sempre que é criada uma nova geração são introduzidos $\lceil p * p_m \rceil$ $(0 < p_m < 1 - p_e)$ indivíduos mutantes, obtidos da mesma forma que os indivíduos da população inicial e onde p_m representa a proporção de mutantes com que se está a trabalhar. A introdução destes indivíduos tem como objetivo garantir a variabilidade populacional, pretendendo-se evitar que o algoritmo fique preso em ótimos locais.

Os restantes $p - \lceil p * p_e \rceil - \lceil p * p_m \rceil$ indivíduos introduzidos numa geração obtém-se do cruzamento entre dois indivíduos escolhidos aleatoriamente da geração anterior.

Após serem selecionados os dois indivíduos usados no cruzamento é gerado, para cada gene, um valor aleatório entre zero e um, que indicará qual dos indivíduos passará o seu alelo ao descendente. Na figura 4.2 (inspirada numa figura encontrada em [22]) é possível observar-se o cruzamento entre dois indivíduos com quatro genes. Neste exemplo, a probabilidade do indivíduo um passar o seu alelo ao descendente é $\rho = 0,7$, o que quer dizer que se, para um dado gene o número aleatoriamente gerado for inferior a 0,7, então o descendente receberá, para esse gene, o alelo do indivíduo um, no caso contrário herdará o alelo do indivíduo dois.

Indivíduo 1	0,32	0,77	0,53	0,85
Indivíduo 2	0,26	0,15	0,91	0,44
Números aleatórios	0,58	0,89	0,68	0,25
Relação com ρ	<	>	<	<
Descendente	0,32	0,15	0,53	0,85

Figura 4.2: Exemplo de cruzamento entre dois indivíduos

Algoritmos genéticos com chaves aleatórias enviesados

Os algoritmos genéticos com chaves aleatórias enviesados foram introduzidos em 2011, por Gonçalves e Resende no artigo [22]. Estes algoritmos apenas diferem dos algoritmos genéticos com chaves aleatórias na forma como são selecionados os indivíduos utilizados nos cruzamentos. Nos algoritmos genéticos com chaves aleatórias os dois indivíduos utilizados no cruzamento são selecionados aleatoriamente, de entre todo o conjunto populacional. Nos algoritmos genéticos com chaves aleatórias enviesados, tal como a figura 4.3 (inspirada numa figura encontrada em [22]) sugere, um dos indivíduos usados no cruzamento é selecionado aleatoriamente de entre o conjunto de indivíduos elite e o outro é selecionado aleatoriamente de entre o conjunto de indivíduos que não pertencem

à elite.

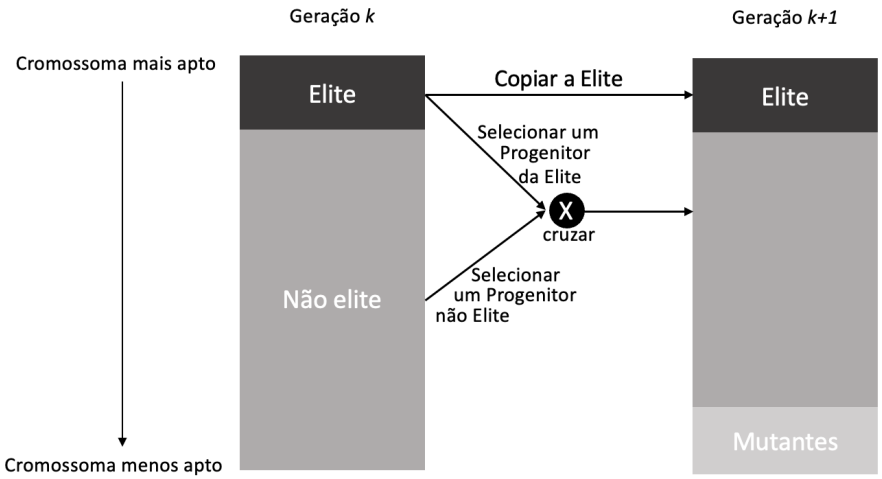


Figura 4.3: Evolução dos indivíduos entre gerações nos algoritmos genéticos com chaves aleatórias enviesados

Nos algoritmos genéticos com chaves aleatórias enviesados ρ é conhecido por ρ_e e representa a probabilidade de um descendente receber o alelo do progenitor que pertence à elite, considerando-se que ρ_e é sempre superior ou igual a 0,5.

Na tabela 4.1 é possível observarem-se os valores recomendados por Gonçalves e Resende [22], para os parâmetros p, p_e, p_m e ρ_e dos algoritmos genéticos com chaves aleatórias enviesados.

Tabela 4.1: Recomendações de valores para os parâmetros

Parâmetro	Descrição dos parâmetros	Valores recomendados
p	tamanho da população	$p = an$, onde $1 \leq a \in \mathbb{N}$ é uma constante e n é o número de genes de cada indivíduo
p_e	proporção da população elite	$0.10 \leq p_e \leq 0.25$
p_m	proporção da população mutante	$0.10 \leq p_m \leq 0.30$
ρ_e	probabilidade de herança do alelo elite	$0.5 \leq \rho_e \leq 0.8$

O algoritmo 1 descreve detalhadamente todas as etapas de um algoritmo genético com chaves aleatórias enviesado genérico. Neste algoritmo g é o contador para o número de gerações, m representa o número de genes de cada indivíduo e c^* e f^* representam, respetivamente, o melhor indivíduo encontrado e o valor da função objetivo associado a esse indivíduo.

Este tipo de algoritmo tem sido aplicado com sucesso a muitos problemas de otimização combinatória. Em 2013, Gonçalves e Resende publicam um artigo [21], onde aplicam um algoritmo genético com chaves aleatórias enviesado a um problema de empacotamento tridimensional. Este problema consiste em empacotar um conjunto de caixas retangulares no menor número possível contentores, sem haver sobreposição de caixas.

Estes autores conseguiram com este algoritmo obter resultados muito positivos, confirmando que a abordagem utilizada era significativamente melhor que as abordagens utilizadas até ao momento.

Algoritmo 1: Algoritmo genético com chaves aleatórias enviesado

Dados: p, p_e, p_m, m, ρ_e
Resultado: c^*, f^*

início
  Gerar uma população inicial P_1 com p indivíduos, em que cada alelo é gerado a partir de uma distribuição $U[0,1]$;
  Calcular a qualidade de cada indivíduo através do descodificador;
  Inicializar f^* e c^*;
  $g \longleftarrow 1$;
  enquanto *critério de paragem não é satisfeito* **faça**
    Guardar em P_g^e os $\lceil p_e \times p \rceil$ melhores indivíduos da geração P_g ;
    Copiar P_g^e para P_{g+1};
    Gerar $\lceil p_m \times p \rceil$ mutantes, avaliar a qualidade de cada mutante, usando o descodificador e copiar os mutantes para P_{g+1};
    para $i = 1$ *até* $(p - \lceil p_e \times p \rceil - \lceil p_m \times p \rceil)$ **faça**
      Selecionar aleatoriamente um indivíduo p_1 de P_g^e;
      Selecionar aleatoriamente um indivíduo p_2 de $P_g \setminus P_g^e$;
      para $j = 1$ *até* m **faça**
        $u = U[0,1]$;
        se $u \leq \rho_e$ **então**
          $p_3[j] \longleftarrow p_1[j]$;
        senão
          $p_3[j] \longleftarrow p_2[j]$;
        fim
      fim
      Avaliar a qualidade de p_3, a partir do descodificador, e copiar p_3 para P_{g+1};
    fim
    $g \longleftarrow g + 1$;
    se *Foi encotrado um indivduo melhor* **então**
      Atualizar f^* e c^*;
    fim
  fim
fim

Num trabalho de 2017, Pessoa, Santos e Resende [37] aplicam um algoritmo genético com chaves aleatórias enviesado a um problema de mediana *p-hub*, com afetação total, sem capacidades e no qual o sub-grafo induzido pelos *hubs* é uma árvore. Neste artigo os autores concluem que o algoritmo proposto apresenta bons resultados, tendo sido capaz de melhorar algumas das melhores soluções conhecidas para as instâncias da literatura.

Em 2018, Almeida, Correia e Saldanha-da-Gama publicam um artigo [2], onde recorrem a um algoritmo genético com chaves aleatórias enviesado para obterem soluções admissíveis para um problema de calendarização de atividades de um projeto com restrições de recurso e onde o objetivo é minimizar a duração do projeto. As atividades deste projeto requerem recursos específicos estando disponível para as realizar um conjunto

de recursos com varias competências. Os autores concluíram o artigo afirmando que a heurística implementada produzia soluções viáveis de elevada qualidade.

Em 2019, Biajali, Chaves e Lorena [6] aplicaram um algoritmo genético com chaves aleatórias enviesado a um problema de localização de serviços com capacidades em dois níveis. O problema consiste em determinar os locais onde devem ser instalados armazéns e fabricas, de modo a minimizar os custos envolvidos no transporte de produtos das fábricas até aos clientes, passando pelos armazéns. Biajali, Chaves e Lorena concluíram que o algoritmo proposto apresentava bons resultados computacionais.

Na adaptação do algoritmo genético com chaves aleatórias enviesado ao problema em estudo, considerou-se que cada indivíduo continha um número de genes igual ao número de nós da instância em resolução. O descodificador desenhado para o problema em estudo vai permitir converter os indivíduos (vetores de chaves aleatórias) em soluções admissíveis para o problema.

4.2 Algoritmos genéticos com chaves aleatórias enviesados para o problema em estudo

Nesta secção são inicialmente descritos, com o auxílio de um exemplo, dois descodificadores para o problema em estudo, apresentando-se em seguida o pseudocódigo para cada um dos descodificadores. Posteriormente, ainda com o auxílio do exemplo, é descrita a forma como é calculado o custo da solução obtida pelo descodificador, sendo também apresentado o pseudocódigo referente ao cálculo deste custo.

Descodificador 1

O descodificador 1 começa por receber os indivíduos (vetores de chaves aleatórias) e associar-lhes um outro vetor, denominado de índices, que é um vetor numerado de um a n (onde n representa a dimensão da instância de teste). Este processo está ilustrado na figura 4.4 .

Indivíduo	0,23	0,60	0,55	0,48	0,92	0,19	0,33	0,88	0,74	0,14
Índices	1	2	3	4	5	6	7	8	9	10

Figura 4.4: Representação de um indivíduo de uma população e o vetor de índices que lhe está associado

Posteriormente o vetor de chaves aleatórias é ordenado por ordem decrescente. Neste processo os movimentos realizados no vetor de chaves aleatórias são também realizados no vetor dos índices pelo que os vetores representados na figura 4.4 passam a ter a configuração dos vetores representados na figura 4.5.

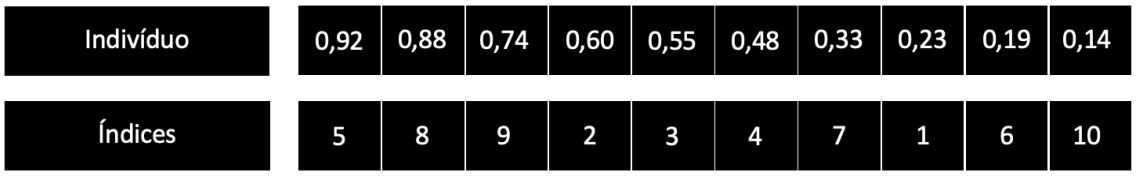


Figura 4.5: Representação de um indivíduo de uma população e o vetor de índices que lhe está associado após a ordenação do vetor que representa os indivíduos

Após a ordenação, o descodificador 1 seleciona para serem *hubs* os primeiros nós, que aparecem no vetor índices e que tenham capacidade para processar todo o fluxo neles originado. Assim, para o indivíduo representado na figura 4.4, o primeiro nó a ser selecionado para ser *hub* seria o nó 5, caso este tivesse capacidade para processar todo o seu fluxo. Este descodificador seleciona nós para serem *hubs*, até que se verifiquem em simultâneo as duas condições seguintes:

1. A soma das quantidades máximas de fluxo que os nós já selecionados para serem *hubs* podem receber diretamente dos não *hubs*, seja maior ou igual à quantidade total de fluxo que necessita de circular entre os vários nós.

2. A soma do número máximo de não *hubs* que podem estar afetos aos *hubs* já selecionados seja maior ou igual ao número de não *hubs*.

Deste modo, considerando a informação presente na tabela 4.2, facilmente se concluí que o primeiro nó a ser selecionado pelo descodificador 1 para ser *hub* é o nó 5, visto que este é o primeiro nó que se encontra representado no vetor índices após a ordenação do vetor indivíduo e que $\Gamma_5 > O_5$.

Tabela 4.2: Parâmetros utilizados no exemplo

i	1	2	3	4	5	6	7	8	9	10
L_i	3	3	2	2	2	3	2	2	2	3
Γ_i	25	30	35	10	50	40	20	20	45	50
O_i	20	5	5	10	15	20	25	30	5	10
D_i	15	10	15	15	21	15	10	13	15	16

Uma vez que $\Gamma_5 < 145 = \sum_{i=1}^{10} O_i$, o descodificador 1 terá de selecionar outros nós para serem *hubs*. Como o nó 8 é o nó que se segue ao 5 no vetor índices, este seria o próximo nó onde o descodificador abriria um *hub*. No entanto, como $\Gamma_8 < O_8$, tal não ocorrerá.

Assim, para além de abrir um *hub* no nó 5 este descodificador abrirá *hubs* nos nós 9, 2 e 3, garantindo assim, que a soma da quantidade máxima de fluxo que os *hubs* podem receber diretamente dos não *hubs* que lhes estão afetos é maior ou igual a 145 e que a soma do número máximo de não *hubs* que podem estar afetos aos *hubs* é maior ou igual ao número de não *hubs*.

Após serem selecionados todos os *hubs*, o descodificador 1 passa a fazer a afetação dos não *hubs* aos *hubs*. Para tal, o descodificador 1 seleciona por ordem os nós que aparecem

no vetor índices ordenado e que não tenham sido escolhidos para serem *hubs* e afeta-os aos *hubs* que lhes ficam mais próximos e têm capacidades para que esta afetação ocorra. Caso nenhum dos *hubs* já selecionados tenha capacidade para lhe ter afeto um determinado nó e se esse nó tiver capacidade para processar o seu fluxo, então esse nó passa a ser *hub*. Se o nó também não tiver capacidade para processar o seu fluxo, então é atribuído um custo elevado ao indivíduo, para que este se perca ao longo das gerações dado que corresponde a uma solução não admissível.

Considerando que os nós se distribuem no espaço de acordo com a figura 4.6 vejamos de que forma o descodificador 1 fará a afetação dos nãos *hubs* aos *hubs*.

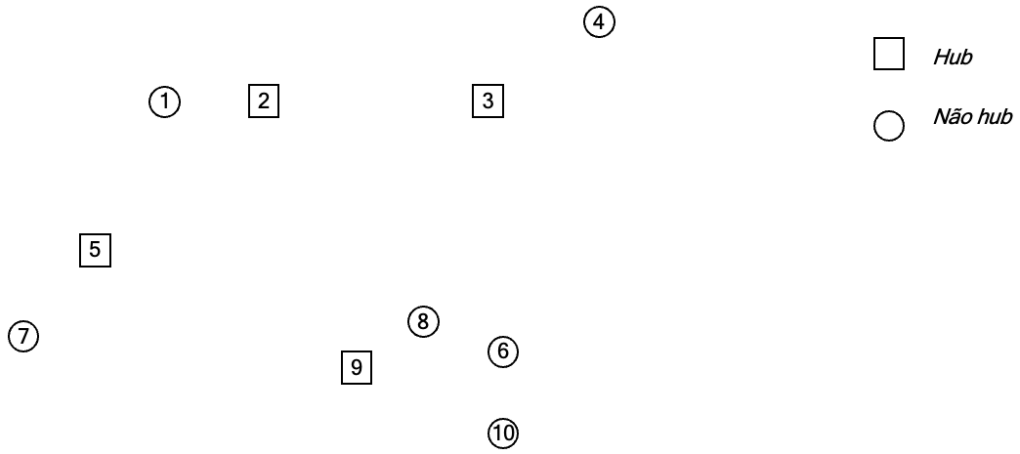


Figura 4.6: Conjunto de nós

Sendo o nó 8 o primeiro não *hub* que se encontra representado no vetor índices ordenado, este será o primeiro não *hub* que o descodificador afetará. Dado que o *hub* 9 é o *hub* que se encontra mais próximo do nó 8, que $\Gamma_9 - O_9 \geq O_8$ e que o *hub* 9 pode ainda receber 2 não *hubs*, então o não *hub* 8 será afeto ao *hub* 9.

Seguidamente, o descodificador afetará o não *hub* 4 ao *hub* 3, o não *hub* 7 ao *hub* 5 e o não *hub* 1 ao *hub* 2, ficando a faltar afetar os nós 6 e 10.

Como o não *hub* 6 aparece representado no vetor índices antes do não *hub* 10, o não *hub* 6 será o próximo não *hub* a ser afeto pelo descodificador. Apesar do *hub* que se encontra mais próximo do nó 6 ser o 9, o nó 6 não poderá ficar afeto a 9, uma vez que $\Gamma_9 - O_9 - O_8 < O_6$. Deste modo, o nó 6 vai ter de ficar afeto ao segundo *hub* que lhe está mais próximo, ou seja ao *hub* 3.

Por fim, sendo o *hub* 9 o mais próximo do nó 10 e dado que $\Gamma_9 - O_9 - O_8 \geq O_{10}$ e que o *hub* 9 pode receber dois não *hubs* e apenas lhe está afeto o não *hub* 8, então o não *hub* 10 ficará afeto ao *hub* 9.

Assim, a rede construída pelo descodificador terá o aspeto da rede representada na figura 4.7, onde já se incluíram as ligações entre *hubs* e será apresentada pelo descodificador na forma de um vetor idêntico ao vetor representado na figura 4.8. Este vetor apresenta na $i-ésima$ posição o *hub* ao qual o nó i foi afeto. Se na $i-ésima$ posição do

vetor estiver o valor i então quer dizer que o nó i foi considerado um hub.

Algoritmo 2: Descodificador 1

Dados: $Indivíduo$ (vetor de chaves aleatórias), conjunto de dados que caracterizam a instância em estudo
Resultado: *solução*, *custo*

início
  Inicializar o vetor *solução* a zero;
  $M \longleftarrow 10^{10}$;
  $Total \longleftarrow \sum_{i \in N} O_i$;
  $contar \longleftarrow 1$;
  $aberto \longleftarrow 0$;
  $capGerada \longleftarrow 0$;
  $capPontosGerada \longleftarrow 0$;
  Construir o vetor Índices (vetor numerado de um a n);
  Ordenar o vetor Indivíduo, por ordem decrescente, efetuando os mesmos movimentos no vetor Índices;
  enquanto $capGerada < Total$ *ou* $capPontosGerada < n - aberto$ **faça**
    $ind \longleftarrow Índices_{contar}$;
    se $\Gamma_{ind} \geq O_{ind}$ **então**
      $solução_{ind} \longleftarrow ind$;
      $capGerada \longleftarrow capGerada + \Gamma_{ind}$;
      $capPontosGerada \longleftarrow capPontosGerada + L_{ind}$;
      $nosDisp_{ind} \longleftarrow L_{ind}$;
      $capDisp_{ind} \longleftarrow \Gamma_{ind} - O_{ind}$;
      $aberto \longleftarrow aberto + 1$;
    fim
    $contar \longleftarrow contar + 1$;
  fim
  para $i = 1$ *até* n **faça**
    $ind \longleftarrow Índices_i$;
    $k \longleftarrow 1$;
    enquanto $\text{solução}_{ind} = 0$ *e* $k \leq n$ **faça**
      $hub \longleftarrow k - ésimo \quad hub \quad mais \quad próximo \quad de \quad ind$;
      se $capDisp_{hub} \geq O_{ind}$ *e* $nosDisp_{hub} > 0$ **então**
        $solução_{ind} \longleftarrow hub$;
        $capDisp_{hub} \longleftarrow capDisp_{hub} - O_{ind}$;
        $nosDisp_{hub} \longleftarrow nosDisp_{hub} - 1$;
      fim
      $k \longleftarrow k + 1$
    fim
    se $\text{solução}_{ind} = 0$ **então**
      se $\Gamma_{ind} \geq O_{ind}$ **então**
        $solução_{ind} \longleftarrow ind$;
        $capDisp_{ind} \longleftarrow \Gamma_{ind} - O_{ind}$;
        $nosDisp_{ind} \longleftarrow Lind$;
      senão
        $custo \longleftarrow M$;
      fim
    fim
  fim
  se $custo < M$ **então**
    $custo \longleftarrow CalcularCusto(solução)$;
  fim
fim

No final o descodificador calcula o custo associado à solução obtida.

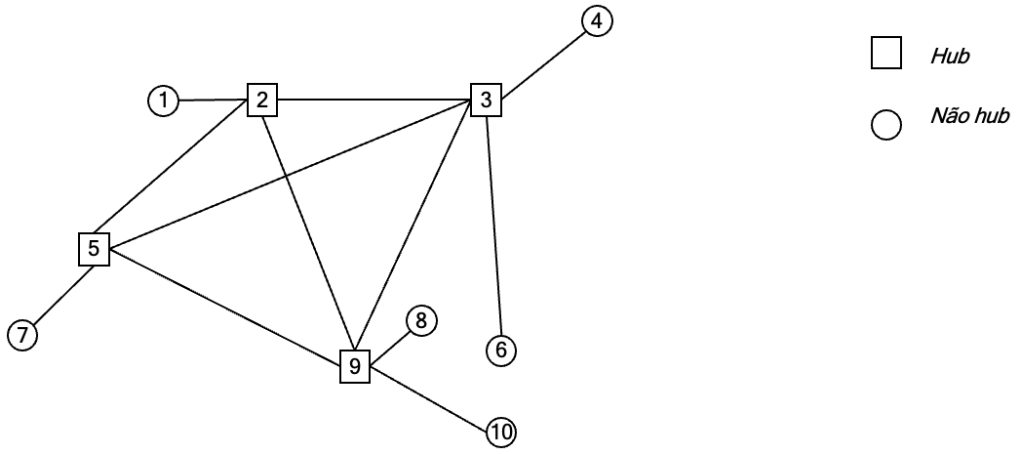


Figura 4.7: Uma solução obtida pelo descodificador 1, apresentada na forma de uma rede de *hubs* e não *hubs*

solução	2	2	3	3	5	3	5	9	9	9

Figura 4.8: Uma solução obtida pelo descodificador 1, apresentada na forma de vetor

O algoritmo 2 descreve detalhadamente todas as etapas do descodificador 1. Neste algoritmo são inicializadas algumas variáveis, entre as linhas dois e oito. Na linha nove ocorre a construção do vetor índices e na linha dez a ordenação do vetor indivíduos. Da linha onze à linha vinte e dois são definidos os *hubs*. Entre as linhas vinte e três e trinta e quatro ocorre a afetação dos não *hubs* aos *hubs*. Se um não *hub* não for afeto a um *hub*, neste segmento do código, então ele passa a ser *hub*, se tiver capacidade para processar todo o fluxo nele originado (linhas trinta e cinco a trinta e nove), ou caso não tenha capacidade para processar o seu fluxo é atribuído um custo elevado à solução (linha quarenta a quarenta e dois). Finalmente da linha quarenta e cinco à linha quarenta e sete é calculado o custo da solução, recorrendo-se a uma outra função denominada *CalcularCusto* cujo pseudocódigo se apresenta no algoritmo 4.

Descodificador 2

O processo realizado pelo descodificador 2 na conversão de um vetor de chaves aleatórias numa solução admissível para o problema em estudo é muito semelhante ao processo realizado pelo descodificador 1. Estes dois descodificadores apenas diferem na forma como é feita a afetação dos não *hubs* aos *hubs*.

Na afetação dos não *hubs* aos *hubs*, o descodificador 2 começa por selecionar por ordem os nós que aparecem no vetor índices e que não tenham sido escolhidos para serem *hubs*, procurando para cada nó o *hub* que lhe está mais próximo.

Após ser encontrado o *hub* mais próximo de um dado nó, será verificado um dos três casos seguintes:

1. O *hub* tem capacidades para que o nó lhe seja afeto e tem pelo menos uma das suas capacidades maior ou igual à respetiva capacidade que teria um *hub* instalado nesse nó. Neste caso o descodificador 2 afeta o nó ao *hub*.

2. O *hub* tem as duas capacidades inferiores às do nó e o nó tem capacidades para ser *hub* e ter-lhe afeto o *hub* e todos os nós que lhe estão afetos. Neste caso o *hub* deixa de ser *hub* e o não *hub* passa a ser *hub* e a ter-lhe afeto o anterior *hub* e os nós que lhe estavam afetos.

3. O *hub* não tem capacidades para que o nó lhe fique afeto e o nó não tem capacidades para ser *hub* e ter-lhe afeto o *hub* e todos os nós que lhe estavam afetos. Neste caso o descodificador 2 vai à procura de um *hub* que verifique as condições do caso um ou do caso dois, dando sempre preferência aos *hubs* mais próximos do nó.

Se para um dado nó não for possível encontrar um *hub* que verifique o caso um, ou o caso dois, então o descodificador 2 simplesmente abrirá um *hub* nesse nó, caso ele tenha capacidade para processar todo o fluxo nele originado, se não é atribuído um custo muito elevado à solução para que esta se perca ao longo das gerações dado que não é admissível.

Vejamos então a rede que o descodificador 2 construiria para o exemplo anterior.

Como as etapas iniciais do descodificador 2 são iguais às etapas iniciais do descodificador 1, o descodificador 2 começa por abrir *hubs* nos nós 2, 3, 5 e 9.

De seguida, como neste exemplo $\Gamma_8 \leq \Gamma_9$, $\Gamma_4 \leq \Gamma_3$, $\Gamma_7 \leq \Gamma_5$ e $\Gamma_1 \leq \Gamma_2$, o descodificador 2, tal como o descodificador 1, afetará o nó 8 ao *hub* 9, o nó 4 ao *hub* 3, o nó 7 ao *hub* 5 e o nó 1 ao *hub* 2.

Posteriormente, o descodificador 2 procurará o *hub* mais próximo do nó 6, que neste caso é o *hub* 9. Como $\Gamma_9 - O_9 - O_8 < O_6$ e $\Gamma_6 - O_6 - O_9 - O_8 < 0$, o descodificador 2 terá de procurar o segundo *hub* mais próximo do nó 6, que é o *hub* 3. Uma vez que $\Gamma_6 > \Gamma_3$, $L_6 > L_3$, $\Gamma_6 - O_6 - O_3 - O_4 \leq 0$ e que um *hub* aberto no nó 6 pode ter-lhe afeto 3 não *hubs*, então descodificador 2 converte o *hub* 3 num não *hub* e converte o não *hub* 6 num *hub*, afetando-lhe os nós 3 e 4.

Finalmente, o descodificador 2 procurará o *hub* mais próximo do nó 10, que para este exemplo, é o *hub* 9. Dado que, $\Gamma_{10} > \Gamma_9$, $L_{10} > L_9$, $\Gamma_{10} - O_{10} - O_9 - O_8 \geq 0$ e que um *hub* aberto no nó 10 pode ter-lhe afeto 3 não *hubs* então o descodificador 2 converte o *hub* 9 num não *hub* e converte o não *hub* 10 num *hub*, afetando-lhe os nós 9 e 8.

Portanto, a rede construída pelo descodificador 2 terá o aspeto da rede representada na figura 4.9 e será representada por um vetor idêntico ao vetor representado em 4.10.

No final o descodificador 2 calcula o custo associado à solução obtida.

O algoritmo 3 descreve detalhadamente todas as etapas do descodificador 2. Neste algoritmo temos a inicialização de algumas variáveis nas linhas dois e três. Na linha quatro o algoritmo constrói o vetor índices e na linha cinco ordena o vetor indivíduos. Da linha seis à linha dezassete o algoritmo define alguns *hubs*. Entre as linhas dezoito e quarenta e um o algoritmo afeta os não *hubs* aos *hubs*, quando as capacidades dos não

Algoritmo 3: Descodificador 2

Dados: $Indivíduo$ (vetor de chaves aleatórias), conjunto de dados que caracterizam a instância em estudo
Resultado: *solução*, *custo*

início
  Inicializar o vetor *solução* a zero;
  Inicializar as variáveis M, $Total$, $contar$, $aberto$, $capGerada$ e $capPontosGerada$, tal como são inicializadas no algoritmo2;
  Construir o vetor Índices (vetor numerado de um a n);
  Ordenar o vetor Indivíduo, por ordem decrescente, efetuando os mesmo movimentos no vetor Índices;
  enquanto $capGerada < Total$ *ou* $capPontosGerada < n - aberto$ **faça**
    $ind \longleftarrow Índices_{contar}$;
    se $\Gamma_{ind} \geq O_{ind}$ **então**
      $solução_{ind} \longleftarrow ind$;
      $capGerada \longleftarrow capGerada + \Gamma_{ind}$;
      $capPontosGerada \longleftarrow capPontosGerada + L_{ind}$;
      $nosDisp_{ind} \longleftarrow L_{ind}$;
      $capDisp_{ind} \longleftarrow \Gamma_{ind} - O_{ind}$;
      $aberto \longleftarrow aberto + 1$;
    fim
    $contar \longleftarrow contar + 1$;
  fim
  para $i = 1$ *até* n **faça**
    $ind \longleftarrow Índices_i$;
    $k \longleftarrow 1$;
    enquanto $\text{solução}_{ind} = 0$ *e* $k \leq n$ **faça**
      $hub \longleftarrow k - ésimo \quad hub \quad mais \quad próximo \quad de \quad ind$;
      se $(\Gamma_{ind} \leq \Gamma_{hub}$ *ou* $L_{ind} \leq L_{hub})$ *e* $(CapDisp_{hub} \geq O_{ind}$ *e* $nosDisp_{hub} > 0)$ **então**
        $solução_{ind} \longleftarrow hub$;
        $CapDisp_{hub} \longleftarrow CapDisp_{hub} - O_{ind}$;
        $nosDisp_{hub} \longleftarrow nosDisp_{hub} - 1$;
      fim
      se $(\Gamma_{ind} > \Gamma_{hub}$ *e* $L_{ind} > L_{hub})$ *e* $(\Gamma_{ind} > \Gamma_{hub} - CapDisp_{hub} + O_{ind}$ *e* $L_{ind} \geq L_{hub} - nosDisp_{hub} + 1)$ **então**
        $solução_{ind} \longleftarrow ind$;
        $capDisp_{ind} \longleftarrow \Gamma_{ind} - \Gamma_{hub} + capDisp_{hub} - O_{ind}$;
        $nosDisp_{ind} \longleftarrow L_{ind} - L_{hub} + nosDisp_{hub} - 1$;
        $capDisp_{hub} \longleftarrow \Gamma_{hub}$;
        $nosDisp_{hub} \longleftarrow L_{hub}$;
        para $t = 1$ *até* n **faça**
          se $\text{solução}_t = hub$ **então**
            $solução_t \longleftarrow ind$;
          fim
        fim
      fim
      $k \longleftarrow k + 1$;
    fim
    se $\text{solução}_{ind} = 0$ *e* $\Gamma_{ind} \geq O_{ind}$ **então**
      $solução_{ind} \longleftarrow ind$;
      $capDisp_{ind} \longleftarrow \Gamma_{ind} - O_{ind}$;
      $nosDisp_{ind} \longleftarrow Lind$;
    senão
      $custo \longleftarrow M$;
    fim
  fim
  se $custo < M$ **então**
    $custo \longleftarrow CalcularCusto(solução)$;
  fim
fim

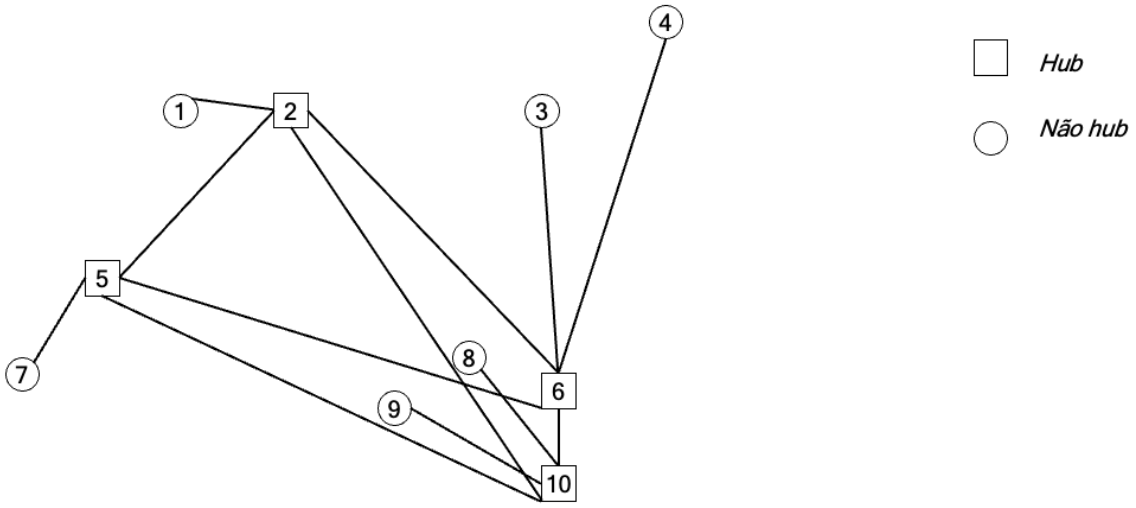


Figura 4.9: Uma solução obtida pelo descodificador 2, apresentada na forma de uma rede de *hubs* e não *hubs*

solução	2	2	6	6	5	6	5	10	10	10

Figura 4.10: Uma solução obtida pelo descodificador 2, apresentada na forma de vetor

hubs são inferiores às dos *hubs* (linhas vinte e três a vinte e sete), ou redefine os *hubs* e afeta os não *hub* aos *hub*, quando as capacidades dos não *hub* são superiores às dos *hub* (linhas vinte e oito a trinta e nove).

Se entre as linhas dezoito e quarenta do algoritmo 3 houver um nó que fique por afetar, então ele tornar-se-á *hub* se tiver capacidade para processar todo o fluxo nele originado (linhas quarenta e dois a quarenta e cinco), se não é atribuído um custo elevado à solução (linhas quarenta e seis a quarenta e oito). Finalmente da linha cinquenta e um à linha cinquenta e três é calculado o custo da solução, recorrendo-se à função *CalcularCusto*.

Cálculo do custo

A função *CalcularCusto* começa por construir uma matriz Z_{nxn} definida pela equação 4.1.

$$Z_{ik} = \begin{cases} 1, & \text{se na posição } i \text{ do vetor } \textit{solução} \text{ está o valor } k \\ 0, & \text{no caso contrario} \end{cases} \tag{4.1}$$

No caso da solução representada em 4.8, a matriz Z, que a função *CalcularCusto* deverá construir é a matriz 4.2.

$$Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.2}$$

Com a matriz Z já construída, a função $CalcularCusto$ aplica a formula 4.3, para obter o custo associado à solução dada como argumento. Nesta formula, a primeira componente calcula o custo associado à abertura dos $hubs$, a segunda calcula o custo da circulação do fluxos entre $hubs$ e não $hubs$ e a terceira calcula o custo da circulação de fluxo entre $hubs$.

$$custo = \sum_{i=1}^{n} f_i Z_{ii} + \sum_{i=1}^{n}\sum_{j=1}^{n} (O_i\chi + D_i\delta)d_{ij}Z_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \alpha w_{ij} d_{kl} Z_{ik} Z_{jl} \tag{4.3}$$

A implementação desta função teve como objetivo simplificar a implementação computacional do cálculo do custo de uma solução.

O algoritmo 4 descreve, em detalhe, todas as etapas da função $CalcularCusto$.

Algoritmo 4: CalcularCusto

Dados: vetor *solução* com n componentes, χ, δ, α, f_i, w_{ij}, d_{ij}, $i, j \in N$
Resultado: $custo$

```
1  início
2      Construir uma matriz nula Z_nxn;
3      custo ⟵ 0;
4      para i = 1 até n faça
5          Z_{i solução_i} ⟵ 1;
6      fim
7      para i = 1 até n faça
8          custo ⟵ custo + f_i Z_ii;
9          para j = 1 até n faça
10             custo ⟵ custo + (O_i χ + D_i δ) d_ij Z_ij;
11             para k = 1 até n faça
12                 para l = 1 até n faça
13                     custo ⟵ custo + α w_ij Z_ik Z_jl d_kl;
14                 fim
15             fim
16         fim
17     fim
18 fim
```

4.3 Resultados Computacionais

Nesta secção são apresentados os resultados computacionais obtidos com o objetivo de testar os descodificadores desenvolvidos na secção 4.2. Nas experiências computacionais realizadas foi utilizada a versão 1.77.0 do Visual Studio Code num computador com o processador M1 da Apple, que apresenta 3.2GHz e 16GB de memoria de RAM.

Nas experiências computacionais efetuadas seguiram-se as recomendações de Gonçalves e Resende [22] e atribui-se ao parâmetro p_e o valor 0,2, ao parâmetro p_m o valor 0,3 e ao parâmetro ρ_e o valor 0,7. Como critério de paragem considerou-se que o algoritmo pararia ao fim de executar um determinado número de gerações ou assim que no inicio de uma geração fosse detetado um tempo computacional superior a uma hora.

Para se determinar qual dos descodificadores desenvolvidos obtém melhores resultados computacionais e para se calibrar tanto o número de gerações utilizadas no critério de paragem, como o parâmetro p (número de indivíduos de cada geração), considerou-se como critério de paragem vinte e cinco, cinquenta e setenta e cinco gerações e considerou-se $p = 2n$ e $p = 5n$. Para cada um dos descodificadores e para cada par de parâmetros considerados, correu-se as cinquenta primeiras instâncias do problema cinco vezes guardando-se para cada instância a melhor solução encontrada e o tempo necessário à execução das cinco corridas. Com estas informações criaram-se as tabelas 4.3 e 4.4 referentes ao descodificador 1 e ao descodificador 2, respetivamente.

As tabelas 4.3 e 4.4 apresentam, em função dos valores dos parâmetros considerados e do tamanho da instância de teste, a percentagem média de *gap* da melhor solução encontrada nas cinco corridas e o tempo total médio em segundos necessário à realização das cinco corridas. Sendo a percentagem média de *gap* de uma solução calculada através de 4.4, onde o limite inferior representa o valor ótimo, ou quando este é desconhecido representa o valor ótimo da relaxação linear.

$$gap = \frac{valor\,obtido\,pelo\,algoritmo - limite\,inferior}{limite\,inferior} * 100 \tag{4.4}$$

As tabelas 4.3 e 4.4 são compostas por oito colunas. A primeira coluna identifica os tamanhos das instâncias de teste. A segunda o valor p considerado, ou seja, o tamanho da população. A terceira e quarta, contém respetivamente, as percentagens médias dos *gaps* e os tempos totais médios em segundos, quando consideradas vinte e cinco gerações. A quinta e sexta apresentam as percentagens médias dos *gaps* e os tempos totais médios em segundos, quando consideradas cinquenta gerações. A sétima e oitava colunas contém as percentagens médias dos *gaps* e os tempos totais médios em segundos, quando consideradas setenta e cinco gerações.

A partir das tabelas 4.3 e 4.4 verifica-se que o tempo computacional necessário à obtenção de uma solução admissível, pelo algoritmo genético com chaves aleatórias enviesado, é praticamente o mesmo para os dois descodificadores desenvolvidos. No entanto as soluções encontradas pelo algoritmo, quando considerado o descodificador 1, apresentam

Tabela 4.3: Resultados obtidos pelo descodificador 1 para instâncias de menores dimensão

		25 gerações		50 gerações		75 gerações	
	p	*gap* (%)	tempo (s)	*gap* (%)	tempo (s)	*gap* (%)	tempo (s)
10	2n	0,41	0,08	0,41	0,15	0,41	0,22
20		0,41	2,31	0,30	4,52	0,30	6,74
25		0,87	6,82	0,87	13,37	0,87	19,93
40		1,33	72,59	1,33	142,57	1,33	212,75
50		1,68	227,73	1,54	445,66	1,54	663,37
média		**0,94**	**61,90**	**0,89**	**121,25**	**0,89**	**180,60**
10	5n	0,41	0,19	0,41	0,37	0,41	0,55
20		0,01	5,75	0,01	11,26	0,01	16,75
25		0,16	16,94	0,16	26,61	0,16	49,59
40		0,66	181,81	0,66	356,78	0,66	532,15
50		0,48	565,78	0,45	1 106,86	0,45	1 647,64
média		**0,34**	**154,09**	**0,34**	**300,38**	**0,34**	**449,34**

Tabela 4.4: Resultados obtidos pelo descodificador 2 para instâncias de menores dimensão

		25 gerações		50 gerações		75 gerações	
	p	gap (%)	tempo (s)	gap (%)	tempo (s)	gap (%)	tempo (s)
10	2n	2,75	0,08	2,75	0,15	2,75	0,22
20		1,77	2,34	1,77	4,57	1,77	6,80
25		6,07	7,04	6,07	13,75	6,07	20,47
40		6,48	75,01	6,48	146,51	6,48	218,12
50		6,33	229,76	6,33	448,98	6,33	668,40
média		**4,68**	**62,85**	**4,68**	**122,79**	**4,68**	**182,80**
10	5n	2,75	0,19	2,75	0,37	2,75	0,55
20		1,77	5,85	1,77	11,41	1,77	16,97
25		6,14	17,63	6,01	34,42	6,01	51,21
40		6,36	188,53	6,36	368,45	6,36	548,58
50		5,99	575,99	5,14	1 147,99	5,14	1 695,87
média		**4,60**	**157,64**	**4,41**	**312,53**	**4,41**	**462,64**

gaps muito menores. Uma possível razão para estas diferenças de *gaps* poderá dever-se ao facto da variabilidade de soluções que o descodificador 1 permite obter ser superior aquela que o descodificador 2 permite obter, já que na grande maioria dos casos o descodificador 2 obriga a que os *hubs* sejam os nós de maiores capacidades, não havendo uma garantia de que se obtenham sempre as melhores soluções ao serem escolhidos estes nós. Assim consideramos que o descodificador 1 é mais adequado para o problema em estudo do que o descodificador 2.

Na tabela 4.3 verifica-se que o aumento do número de gerações não apresenta uma grande influencia na qualidade das soluções encontradas. Assim, e visto que com o aumento do número de gerações os tempos computacionais também aumentam consideraram-se como critérios de paragem a obtenção de vinte e cinco gerações ou a obtenção de um tempo computacional superior a uma hora no início de uma geração.

Ainda a partir da tabela 4.3 verifica-se que para $p = 5n$ a qualidade das soluções encontradas é superior à qualidade das soluções encontradas para $p = 2n$, deste modo e apesar dos tempos computacionais necessários à obtenção das soluções ser bastante superior para $p = 5n$, decidiu-se que valeria a pena atribuir a p o valor $5n$.

Uma vez que da observação das tabelas 4.3 e 4.4, se concluiu que os melhores resultados computacionais obtidos pelo algoritmo genético com chaves aleatórias enviesado ocorriam quando era utilizado o descodificador 1, $p = 5n$ e quando o algoritmo parava ao fim de executar vinte e cinco gerações, ou assim que no início de uma geração fosse detetado um tempo computacional superior a uma hora, todas as instâncias foram corridas cinco vezes tendo em conta este cenário. A tabela 4.5 apresenta, para todas as instâncias, a melhor solução encontrada nas cinco corridas.

Na primeira coluna da tabela 4.5 são identificadas as instâncias de teste. Na segunda são apresentados os *hubs* da melhor solução obtida pelo algoritmo. Na terceira encontram-se os custos associados a essa solução. Na quarta coluna são apresentados os tempos totais, em segundos, das cinco corridas. Note-se que a partir da instância $100LL$ os tempos apresentados são superiores a dezoito mil segundos, indicando que em pelo menos uma das cinco corridas o algoritmo esteve a correr durante mais de uma hora. Isto acontece devido ao facto de o tempo de execução do algoritmo só ser avaliado no início de cada geração, podendo-se dar o caso de termos um tempo de execução inferior a uma hora no início de uma geração e no final dessa geração o tempo já ser bastante superior a uma hora.

A partir da tabela 4.5 verifica-se que o algoritmo genético com chaves aleatórias enviesado apresenta vantagens relativamente ao modelo (P), dado que o algoritmo permitiu obter soluções admissíveis para todas as instâncias de teste e o modelo apenas havia obtido soluções para instâncias com um máximo de cento e cinquenta nós. Para uma melhor comparação entre os resultados obtidos pela heurística e pelo modelo criaram-se as tabelas 4.6 e 4.7.

A tabela 4.6 apresenta na primeira coluna o número de nós das instâncias de teste, para as quais o modelo (P) com o pré-processamento obteve solução. Na segunda coluna da tabela 4.6 são apresentados os *gaps* médios, em função do tamanho das instâncias, das melhores soluções encontradas pela heurística nas cinco corridas. A terceira coluna apresenta os desvios padrões médios dos *gaps* obtidos nas cinco corridas, em função do tamanho da instância. A quarta coluna exibe os tempos computacionais, em segundos e em função de n, necessários à realização das cinco corridas. Na quinta coluna são indicados os números de soluções ótimas que o modelo obteve para cada n. A sexta coluna indica os *gaps* médios das soluções obtidas pelo modelo (P) com o pré-processamento, em que não se tem a certeza de serem as soluções ótimas. Por último, a sétima coluna apresenta os tempos médios, em segundos necessários à obtenção das soluções pelo modelo (P) com o pré-processamento.

A tabela 4.7 contém os resultados agrupados pelo tipo de instância. Excetuando a primeira coluna, as restantes colunas têm o mesmo tipo de informação da tabela 4.6.

A partir da tabela 4.6 é possível observar-se que com o aumento do tamanho das instâncias de teste, os tempos necessários à obtenção de soluções também aumenta, aumentando muito mais quando é utilizada a heurística. Assim, conclui-se que é preferível utilizar o modelo (P) na obtenção de soluções com um número de nós não superior a cento

Tabela 4.5: Resultados computacionais para o descodificador 1

	melhor solução admissível		
Instâncias	*hubs*	custo	tempo (s)
10LL	3-4-7	224 913,04	0,19
10LT	1-4-10	252 973,55	0,19
10TL	4-10	264 543,97	0,20
10TT	4-10	264 543,97	0,18
20LL	6-8-9-14	260 678,86	5,84
20LT	6-8-9-14	260 678,86	5,66
20TL	6-8-19	296 403,64	5,84
20TT	6-8-10-19	319 296,00	5,65
25LL	8-14-23	249 883,98	17,78
25LT	8-14-16-19-24	287 584,50	16,07
25TL	8-14-23	329 421,83	17,77
25TT	9-14-16-25	372 456,57	16,12
40LL	6-14-26-29	274 156,19	185,72
40LT	12-16-26-30	290 513,40	178,24
40TL	6-19-21-36	378 737,09	186,68
40TT	6-19-25-40	401 131,24	176,60
50LL	15-27-32-35	263 040,55	575,47
50LT	13-19-32-46	292 598,89	555,84
50TL	3-17-27-31-45	417 466,92	574,78
50TT	12-13-19-25-26	482 469,45	557,02
60LL	6-27-33-55	251 617,12	1 427,33
60LT	6-19-39-42-59	276 866,58	1 374,48
60TL	15-19-46-55	304 136,32	1 429,02
60TT	9-26-33-40	379 953,33	1 432,30
70LL	8-21-30-64	270 984,68	3 079,48
70LT	21-36-47-64	268 053,85	3 088,49
70TL	18-20-22-58-64	341 953,65	3 085,14
70TT	24-30-41-48-65	482 944,20	3 091,90
75LL	8-47-55-58	255 429,12	4 409,82
75LT	24-48-52-58	277 171,44	4 291,54
75TL	3-24-32-57-67	404 658,56	4 374,64
75TT	12-27-31-53-74	414 469,33	3 952,50
90LL	4-50-58-65	252 476,33	10 935,36
90LT	25-50-62-65	267 027,44	10 858,11
90TL	3-50-77-86	308 895,12	10 895,18
90TT	9-51-59-61-63	388 752,57	10 924,16
100LL	29-52-64-73	257 522,79	18 322,06
100LT	29-35-68-96	269 656,98	18 477,22
100TL	5-19-35-52-61	501 501,90	18 208,02
100TT	5-15-52-61-95	576 913,00	18 363,77
125LL	38-55-62-86	267 614,59	19 835,95
125LT	6-66-78-85-89	281 560,88	18 935,11
125TL	3-80-84-112	337 480,80	19 919,33
125TT	11-59-69-77-88	360 917,97	19 860,00
150LL	12-44-99-134	265 060,16	22 751,20
150LT	47-64-104-149	294 587,67	22 778,25
150TL	58-59-63-110-134	374 956,59	22 791,52
150TT	18-65-74-135	442 470,07	20 296,91
175LL	47-49-113-155	270 171,97	19 498,44
175LT	56 113 123 135 152	320 735,41	22 808,99
175TL	52-62-129-148	448 725,03	25 790,89
175TT	29-33-85-119-121-146-151	579 939,50	25 866,69
200LL	56-126-139-199	285 173,44	50 405,49
200LT	52-73-139-170	295 661,81	43 465,44
200TL	26-29-88-123	469 532,44	50 189,02
200TT	52-68-91-168-173	623 954,50	43 437,90

Tabela 4.6: Comparação dos resultados em função de n

	Heurística			Modelo (P)		
n	*gap* (%)	σ_{gap}	tempo (s)	nº de soluções ótimas	*gap* (%)	tempo (s)
10	0,41	0,00	0,19	4	-	0,29
20	0,01	0,92	5,75	4	-	0,77
25	0,16	0,57	16,94	4	-	2,48
40	0,66	0,61	181,81	4	-	33,26
50	0,48	1,07	565,78	4	-	78,73
60	0,72	0,59	1 415,78	4	-	68,51
70	1,42	0,77	3 086,25	4	-	758,60
75	2,26	1,21	4 257,12	4	-	744,77
90	1,92	1,26	10 903,20	3	2,54	1 119,91
100	3,42	0,92	18 342,77	2	3,50	2 376,00
125	5,58	1,96	19 637,60	0	3,03	3 618,03
150	20,55	6,37	22 154,47	0	213,10	3 683,33

Tabela 4.7: Comparação dos resultados em função do tipo de instância

	Heurística			Modelo (P)		
tipo	*gap* (%)	σ_{gap}	tempo (s)	nº de soluções ótimas	*gap* (%)	tempo (s)
LL	1,20	0,87	6 795,52	10	69,03	767,10
LT	2,82	1,12	7 653,41	9	35,01	1 069,50
TL	3,78	1,18	6 790,68	9	122,31	949,49
TT	5,08	2,24	6 556,43	9	88,01	1 375,48

e vinte e cinco. No entanto, é de realçar o facto da heurística ter obtido bons *gaps* para estas instâncias e do desvio padrão não ser muito elevado, levando-nos a acreditar que esta heurística é robusta e adequa-se ao problema em estudo. Dado que o modelo (P) não permitiu a obtenção de soluções para as instâncias com cento e setenta e cinto ou mais nós e que as soluções obtidas pelo modelo (P), para as instâncias com cento e cinquenta nós apresentam *gaps* muito superiores aos *gaps* das soluções obtidas pela heurística, conclui-se que a heurística desenvolvida obtém melhor resultados para as instâncias de maiores dimensões.

A tabela 4.7 comprova, mais uma vez, a elevada complexidade das instâncias do tipo TT, instâncias para as quais os *gaps* médios das soluções obtidas pela heurística foram maiores. Por outro lado, a tabela 4.7 também comprova que as instâncias do tipo LL são as mais fáceis de resolver, tendo a heurística conseguido obter soluções com *gaps* médios de $1,2\%$ para estas instâncias.

Como forma de tentar melhorar os tempos computacionais da heurística desenvolvida, poderia-se ter utilizado uma função mais eficiente para calcular o custo das soluções. Assim, tal como será sugerido na conclusão deste trabalho, poderia-se ter utilizado a função $CalcularCusto2$, descrita no algoritmo 5.

Algoritmo 5: CalcularCusto2

Dados: vetor *solução* com n componentes, χ, δ, α, f_i, w_{ij}, d_{ij}, $i, j \in N$
Resultado: $custo$

1 **início**
2 $custo \longleftarrow f_{solução_1} + d_{1solução_1}(\chi O_1 + \delta D_1)$;
3 **para** $i = 2$ *até* n **faça**
4 $contador \longleftarrow 0$;
5 $custo \longleftarrow custo + d_{isolução_i}(\chi O_i + \delta D_i)$;
6 **para** $j = 1$ *até* $i - 1$ **faça**
7 **se** solução$_i \neq$ solução$_j$ **então**
8 $custo \longleftarrow custo + \alpha d_{solução_i solução_j}(w_{ij} + w_{ji})$;
9 $contador \longleftarrow contador + 1$;
10 **fim**
11 **se** $contador = i - 1$ **então**
12 $custo \longleftarrow custo + f_{solução_i}$;
13 **fim**
14 **fim**
15 **fim**
16 **fim**

O algoritmo 5 começa por, na linha dois, atribuir ao custo o valor correspondente á abertura do *hub* ao qual o nó um está afeto mais o custo associado à circulação de fluxo entre o nó um e o *hub*. Na linha cinco é somado o custo da circulação de fluxo entre os restantes nós e os *hubs* aos quais foram afetos. Na linha oito é adicionado o custo da circulação de fluxo entre *hubs*. E na linha doze é somado o custo fixo de abertura dos restantes *hubs*.

5

Conclusões

Neste trabalho estudou-se um problema de localização de *hubs*, com afetação total e capacidades associadas à quantidade de fluxo não processado que cada *hub* pode receber e ao número máximo de não *hubs* que podem ser afetos a cada *hub*. Neste problema, o número de *hubs* a serem instalados é desconhecido à priori e são tidos em conta os custos associados à instalação dos *hubs* e à circulação de fluxo.

Tanto quanto se sabe, este foi o primeiro trabalho sobre problemas de localização de *hubs* a considerar capacidades referentes ao número máximo de não *hubs* que podem estar afetos a cada *hub*. Assim, começou-se por adaptar um modelo de Programação Linear Misto existente na literatura para o problema clássico de localização de *hubs* com capacidades ao novo problema. Para este modelo foram ainda desenvolvidos alguns testes de pré-processamento, com o objetivo de tentar fixar variáveis e assim reduzir a dimensão do modelo. Dado que, nem o modelo, nem o modelo após os testes de pré-processamento permitiram a obtenção de soluções para todas as instâncias de teste utilizadas nas experiências computacionais, desenvolveu-se um algoritmo genético com chaves aleatórias enviesado. Para este algoritmo foram desenvolvidos dois descodificadores.

Com o objetivo de afinar os parâmetros do algoritmo começou-se por testar os descodificadores nas instâncias de menor dimensão. Posteriormente, aplicou-se o algoritmo, com o descodificador e os valores dos parâmetros para os quais se obteve melhores resultados nas instâncias de menor dimensão, às restantes instâncias de teste.

Verificou-se então que o algoritmo obtinha soluções admissíveis para todas as instâncias com *gaps* relativamente reduzidos para a maior parte das instâncias. No entanto, também se notou que para as instâncias de maiores dimensões os tempos computacionais necessários à obtenção das soluções eram muito elevados.

Assim, deixa-se para trabalho futuro a utilização da função *CalcularCusto*2, descrita no algoritmo 5, durante cálculo dos custos. Esta função permitirá aumentar a eficiência do cálculo dos custos, já que requer menos operações do que a função *CalcularCusto*.

Outra direção de trabalho futuro consiste no desenvolvimento de descodificadores mais sofisticados, com o objetivo de se tentar melhorar a qualidade das soluções admissíveis.

Referências

[1]S. Abdinnour-Helm. "A hybrid heuristic for the uncapacitated hub location problem". Em: *European Journal of Operational Research* 106 (1998), pp. 489–499.

[2]B.F. Almeida, I. Correia e F. Saldanha-da-Gama. "A biased random-key genetic algorithm for the project scheduling problem with flexible resources". Em: *Top* 26 (2018), pp. 283–308.

[3]S. Alumur e B.Y. Kara. "Network hub location problems: The state of the art". Em: *European Journal of Operational Research* 190 (2008), pp. 1–21.

[4]T. Aykin. "Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem". Em: *European Journal of Operational Research* 79 (1994), pp. 501–523.

[5]J.C. Bean. "Genetic algorithms and random keys for sequencing and optimization". Em: *ORSA Journal on Computing* 6 (1994), pp. 154–160.

[6]F.L. Biajoli, A.A. Chaves e L.A.N. Lorena. "A biased random-key genetic algorithm for the two-stage capacitated facility location problem". Em: *Expert Systems with Applications* 115 (2019), pp. 418–426.

[7]J.F. Campbell. "Integer programming formulations of discrete hub location problems". Em: *European Journal of Operational Research* 72 (1994), pp. 387–405.

[8]J.F. Campbell e M.E. O'Kelly. "Twenty-Five Years of Hub Location Research". Em: *Transportation Science* 46 (2012), pp. 153–169.

[9]J. Chen. "A Heuristic for the Capacitated Single Allocation Hub Location Problem". Em: *Advances in Industrial Engineering and Operations Research*. Ed. por Alan H. S. Chan e Sio-Iong Ao. Vol. 5. 2008, pp. 185–196.

[10]I. Contreras, E. Fernández e A. Marín. "The Tree of Hubs Location Problem". Em: *European Journal of Operational Research* 202 (2010), pp. 390–400.

[11]I. Correia, S. Nickel e F. Saldanha-da-Gama. "A stochastic multi-period capacitated multiple allocation hub location problem: Formulation and inequalities". Em: *Omega* 74 (2018), pp. 122–134.

[12]I. Correia, S. Nickel e F. Saldanha-da-Gama. "Multi-product capacitated single-allocation hub location problems: Formulations and Inequalities". Em: *Networks and Spatial Economics* 14 (2014), pp. 1–25.

[13]I. Correia, S. Nickel e F. Saldanha-da-Gama. "Single-assignment hub location problems with multiple capacity levels". Em: *Transportation Research Part B: Methodological* 44 (2010), pp. 1047–1066.

[14]I. Correia, S. Nickel e F. Saldanha-da-Gama. "The capacitated single-allocation hub location problem revisited: A note on a classical formulation". Em: *European Journal of Operational Research* 207 (2010), pp. 92–96.

[15]J. Ebery. "Solving large single allocation p-hub problems with two or three hubs". Em: *European Journal of Operational Research* 128 (2001), pp. 447–458.

[16]J. Ebery, M. Krishnamoorthy, A. Ernst e N. Boland. "The capacitated multiple allocation hub location problem: Formulations and algorithms". Em: *European Journal of Operational Research* 120 (2000), pp. 614–631.

[17]A.T. Ernst, H.W. Hamacher, H. Jiang, M. Krishnamoorthy e G.J. Woeginger. "Uncapacitated single and multiple allocation p-hub center problems". Em: *Computers and Operations Research* 36 (2009), pp. 2230–2241.

[18]A.T. Ernst e M. Krishnamoorthy. "Efficient algorithms for the uncapacitated single allocation p-hub median problem". Em: *Location Science* 4 (1996), pp. 139–154.

[19]A.T. Ernst e M. Krishnamoorthy. "Solution algorithms for the capacitated single allocation hub location problem". Em: *Annals of Operations Research* 86 (1999), pp. 293–306.

[20]R.Z. Farahani, M.Hekmatfar, A.B. Arabani e E. Nikbakhsh. "Hub location problems: A review of models, classification, solution techniques, and applications". Em: *Computers and Industrial Engineering* 64 (2013), pp. 1096–1109.

[21]J.F. Gonçalves e M.G.C. Resende. "A biased random key genetic algorithm for 2D and 3D bin packing problems". Em: *International Journal of Production Economics* 145 (2013), pp. 500–510.

[22]J.F. Gonçalves e M.G.C. Resende. "Biased random-key genetic algorithms for combinatorial optimization". Em: *Journal of Heuristics* 17 (2011), pp. 487–525.

[23]J.H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[24]B.Y. Kara e B.Ç. Tansel. "On the single-assignment p-hub center problem". Em: *European Journal of Operational Research* 125 (2000), pp. 648–655.

[25]H. Karimi e M. Bashiri. "Hub covering location problems with different coverage types". Em: *Scientia Iranica* 18 (2011), pp. 1571–1578.

[26]J.G. Klincewicz. "A dual algorithm for the uncapacitated hub location problem". Em: *Location Science* 4 (1996), pp. 173–184.

[27]M. Labbé, H. Yaman e E. Gourdin. "A branch and cut algorithm for hub location problems with single assignment". Em: *Mathematical Programming* 102 (2005), pp. 371–405.

[28]G. Laporte, S. Nickel e F. Saldanha-da-Gama. *Location Science*. Springer, Cham, 2019.

[29]Y. Lee, B.H. Lim e J.S.Park. "A hub location problem in designing digital data service networks: Lagrangian relaxation approach". Em: *Computers and Operations Research* 4 (1996), pp. 185–194.

[30]C.C. Lin, J.Y. Lin e Y.C. Chen. "The capacitated p-hub median problem with integral constraints: An application to a Chinese air cargo network". Em: *Applied Mathematical Modelling* 36 (2012), pp. 2777–2787.

[31]A. Marín. "Formulating and solving splittable capacitated multiple allocation hub location problems". Em: *Computers and Operations Research* 32 (2005), pp. 3093–3109.

[32]A. Marın, L. Cánovas e M. Landete. "New formulations for the uncapacitated multiple allocation hub location problem". Em: *European Journal of Operational Research* 172 (2006), pp. 274–292.

[33]M.E. O'Kelly. "A quadratic integer program for the location of interacting hub facilities". Em: *European Journal of Operational Research* 32 (1987), pp. 393–404.

[34]M.E. O'Kelly. "Hub facility location with fixed costs". Em: *Papers in Regional Science* 71 (1992), pp. 293–306.

[35]M.E. O'Kelly. "Rectilinear minimax hub location problems". Em: *Journal of Geographical Systems* 11 (2009), pp. 227–241.

[36]*OR Library*. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/phub1.txt. visto pela ultima vez em 5/01/2023.

[37]L.S. Pessoa, A.C. Santos e M.G.C Resende. "A biased random-key genetic algorithm for the tree of hubs location problem". Em: *Optimization Letters* 11 (2017), pp. 1371–1384.

[38]M. Randall. "Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation". Em: *Computational Optimization and Applications* 39 (2008), pp. 239–261.

[39]C.R. Reeves. "Genetic Algorithms". Em: *Handbook of Metaheuristics*. Springer US, 2010, pp. 109–139.

[40]D. Skorin-Kapov, J. Skorin-Kapov e M. O'Kelly. "Tight linear programming relaxations of uncapacitated p-hub median problems". Em: *European Journal of Operational Research* 94 (1996), pp. 582–593.

[41]Z. Stanimirović. "A genetic algorithm approach for the capacitated single allocation p-hub median problem". Em: *Computing and Informatics* 29 (2012), pp. 117–132.

[42]H. Topcuoglu, F. Corut, M. Ermis e G. Yilmaz. "Solving the uncapacitated hub location problem using genetic algorithms". Em: *Computers and Operations Research* 32 (2005), pp. 967–984.