Universidade do Minho
Escola de Ciências

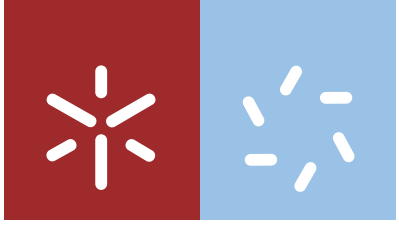
Ana Carolina Oliveira de Sousa

Modelação com técnicas de *clustering* de participações geoespaciais de cidadãos

outubro de 2017

Ana Carolina Oliveira de Sousa **Modelação com técnicas de *clustering* de participações geoespaciais de cidadãos**

UMinho | 2017



Universidade do Minho
Escola de Ciências

Ana Carolina Oliveira de Sousa

Modelação com técnicas de *clustering* de participações geoespaciais de cidadãos

Relatório de Estágio
Mestrado em Matemática e Computação

Trabalho realizado sob orientação do
Doutor Stéphane Clain
e do
Doutor Pedro Patrício

outubro de 2017

Nome: Ana Carolina Oliveira de Sousa

Endereço eletrónico: acsousa1994@gmail.com

Número do bilhete de identidade: 14317171

Título de Relatório de Estágio:
"Modelação com técnicas de *clustering* de participações geoespaciais de cidadãos"

Orientadores:
Doutor Stéphane Clain
Doutor Pedro Patrício

Ano de conclusão: 2017

Designação do Mestrado:
Matemática e Computação

Universidade do Minho, 31 de outubro de 2017

_____

Ana Carolina Oliveira de Sousa

Modelação com técnicas de *clustering* de participações geoespaciais de cidadãos

Resumo

O desenvolvimento de aplicações móveis para a participação ativa dos cidadãos nas suas cidades tem vindo a aumentar com o avanço do conceito de *smart cities*. A aplicação móvel *JuntarAJunta* permite à população reportar problemas e sugestões da sua cidade, com o objetivo de melhorar as infraestruturas urbanas e a qualidade de vida. Com a integração deste tipo de aplicações no dia-a-dia da população, a gestão destes dados recolhidos torna-se cada vez mais importante. Este relatório apresenta um método de deteção de participações semelhantes a partir da sua localização geográfica, orientação e distância ao problema reportado. Os métodos desenvolvidos baseiam-se no método de *compass clustering*, para o processamento dos dados, e no método de *clustering* hierárquico para a identificação de grupos de participações semelhantes. Os resultados obtidos com este método evidenciam a importância de informação complementar, para além da posição geográfica, de forma a identificar grupos com características semelhantes. A partir da análise de cenários sintéticos, concluiu-se que este método produz resultados mais precisos comparativamente aos resultados obtidos utilizando o método de *clustering* hierárquico com apenas a posição geográfica das participações.

Palavras-Chave: *Machine Learning*, *Clustering* Hierárquico, *Compass Clustering*, geolocalização

Modeling with clustering techniques of geospatial participations of citizens

Abstract

The development of mobile applications for the active participation of citizens in their cities has been increasing with the concept of *smart cities*. The mobile application *JuntarAJunta* allows the population to report problems and suggestions from their cities, with the aim of improving urban infrastructure and quality of life. With the integration of this type of technology in the day to day life of the population, the management of collected data is becoming increasingly important. This report presents a method for detecting similar participations based on their geographical location, compass orientation and distance to the reported problem. The developed methods are based on the compass clustering method, for data processing, and the hierarchical clustering method, for the identification of groups with similar participations. The results obtained with this method show the importance of complementary information, besides the geographical position, to identify groups with similar characteristics. From the analysis of synthetic scenarios, it was concluded that this method produces more precise results compared to the results obtained using only the geographical position of the participations.

Keywords: Machine Learning, Hierarchical Clustering, Compass Clustering, geolocation

Conteúdo

Lista de Figuras

Lista de Tabelas

1 | Introdução

Este relatório é o resultado do estágio realizado na empresa Codeangel, entre 5 de dezembro de 2016 e 31 de agosto de 2017, no âmbito da unidade curricular "Estágio/Dissertação" do Mestrado em Matemática e Computação.

O objetivo inicial deste estágio era utilizar técnicas de *machine learning* para determinar a *performance* dos vários intervenientes no processo de venda eletrónica na plataforma da empresa Codeangel, designadamente ao nível das redes sociais.

Inicialmente, foi proposto pela empresa a aprendizagem de ferramentas de desenvolvimento *web* para, numa fase final do estágio, fazer uso destes conhecimentos para apresentar, numa *dashboard*, os resultados finais do trabalho realizado durante o estágio (que seria apresentar indicadores de performance da página de comércio eletrónico).

Em fevereiro de 2017, iniciei um novo projeto de desenvolvimento *web* relacionado com a aplicação móvel *JuntarAJunta*[1], na qual a empresa apoia o seu desenvolvimento.

A aplicação *JuntarAJunta* propõe a participação ativa da comunidade em decisões das juntas de freguesias. Através dela, os cidadãos têm a oportunidade de reportar problemas e sugerir soluções e alterações das suas cidades. Além de uma fotografia e de uma descrição, cada participação tem associada uma informação geoespacial, que permite as juntas de freguesia encontrar os problemas e sugestões reportadas.

Em março de 2017, o objetivo inicial do estágio foi alterado e, tendo em conta o trabalho desenvolvido anteriormente, ficou decidido o desenvolvimento de técnicas de *machine learning* na aplicação móvel *JuntarAJunta*.

Com a divulgação da aplicação, prevê-se que o número de participações irá aumentar. Um dos problemas que poderá surgir é um número elevado de participações que correspondem a um mesmo problema. Nesse caso, haverá informação redundante na base de dados, o que poderá provocar problemas na gestão da informação. Perante este problema, foi proposto implementar mecanismos de *machine learning* para a identificação e diminuição de participações

[1] juntarajunta.pt

repetidas ou muito semelhantes.

Durante o período de estágio foram analisadas, por mim e pela aluna Maria Soares, duas abordagens diferentes para resolver o problema proposto. Primeiramente, uma abordagem usando apenas os dados fornecidos pela aplicação móvel e, de seguida, uma abordagem complementar introduzindo novas informações com o intuito de ultrapassar alguns dos problemas encontrados na primeira abordagem. A primeira abordagem está descrita com mais detalhe em [10]. A segunda abordagem será descrita neste relatório.

Este relatório está organizado em cinco capítulos. Neste primeiro capítulo definiu-se o novo tema a ser abordado, assim como o seu enquadramento tendo em conta o objetivo que se pretende atingir.

O capítulo 2 descreve a área de *Business Intelligence*, a sua importância nos dias de hoje e como o objetivo inicial deste estágio se enquadra nessa área. Além disso, descreve a aprendizagem de ferramentas de desenvolvimento *web* que foi feita na empresa, durante os primeiros meses de estágio e são apresentados alguns dos trabalhos realizados.

O capítulo 3 apresenta a aplicação *JuntarAJunta*, as suas funcionalidades e uma pequena análise aos dados e à base de dados da aplicação.

O capítulo 4 descreve a metodologia utilizada para o estudo e resolução do problema proposto. Também é feita uma introdução ao tema de *clustering*, apresentando alguns conceitos fundamentais, o método de *clustering* hierárquico e métodos de validação de resultados de *clustering*.

O capítulo 5 contém a parte principal do trabalho realizado. Primeiramente, são identificados os parâmetros e as variáveis a serem utilizadas no método de *clustering* hierárquico, a partir dos resultados descritos em [10]. Com base nesses resultados, são introduzidas novas variáveis com o intuito de melhorar a qualidade dos resultados, adaptando o algoritmo às novas variáveis. É descrito uma forma de processamento dos dados para que estes sejam usados no método de *clustering* hierárquico. Por fim, todos os métodos desenvolvidos anteriormente são aplicados em dois cenários sintéticos.

O capítulo 6 apresenta as conclusões retiradas deste estágio, que justificam a necessidade de introdução de novas variáveis e do desenvolvimento de novos métodos para a resolução do problema proposto.

2 | *Business Intelligence* e desenvolvimento *web*

O objetivo inicial deste estágio passava pela modelação matemática de indicadores de performance para a plataforma de *eCommerce* da empresa Codeangel, através de técnicas de *machine learning*. Ou seja, desenvolver uma solução de *Business Intelligence* sobre as vendas, estratégias e meios de *marketing* da plataforma de *eCommerce* e desenvolver fórmulas preditivas de apoio à decisão.

A área de *Business Intelligence* tem como objetivo converter dados em informações relevantes ao negócio, através da organização e análise de dados. Nas últimas duas décadas, esta área e a da análise de grande volume de dados (*big data*) tornaram-se cada vez mais importantes na comunidade académica e empresarial, refletindo o impacto que os problemas relacionados com dados têm nas organizações empresariais dos dias de hoje. Assim, a gestão dos dados e seu o armazenamento podem ser consideradas como a base destas duas áreas. Em relação ao armazenamento, o desenvolvimento de *data marts*[1] para extração, transformação e introdução de dados são essenciais para a conversão e integração de dados específicos da empresa. Em relação à gestão dos dados, a consulta da base de dados, o processamento analítico e relatórios baseados em gráficos simples e intuitivos são usados para explorar e identificar características importantes nos dados. Além disso, técnicas de análise estatística e de *data mining* podem ser utilizadas, por exemplo, para classificar e agrupar dados, assim como para detetar anomalias e fazer previsões em várias aplicações de negócio. Através de *scorecards* e *dashboards* para a gestão do desempenho empresarial é possível analisar e visualizar as métricas de desempenho [5].

No contexto do tema inicialmente proposto, os dados para análise provinham de origens distintas: dados relacionados com os produtos e com as vendas e dados provenientes de serviços de análise *web*. Um desses serviços

[1]Parte da *data warehouse* que contém dados relativos a uma área específica ou equipa da empresa

seria o *Google Analytics*[2], de onde seria possível obter informações acerca das visualizações por página *web*, localização geográfica do visitante e a forma pelo qual chegou à página, são alguns exemplos. Outro serviço que seria utilizado era o *Google AdWords*[3], um serviço de publicidade que consiste em anúncios encontrados, principalmente, nos mecanismos de pesquisa e relacionados com as palavras-chave que o visitante utiliza. Assim, com estes dados seria possível, por exemplo, analisar o impacto que os anúncios criados com o *Google AdWords* tinham na venda de produtos da página de *eCommerce*.

Tendo em mente o objetivo inicial do estágio, começou-se pela aprendizagem da linguagem **HTML** (*HyperText Markup Language*) e **CSS** (*Cascading Style Sheets*), para a construção e personalização de páginas *web*, seguindo-se de **JavaScript** e **jQuery**, para a dinâmica e interatividade da página. O passo seguinte foi aprender a utilizar a *framework* **Bootstrap** para facilitar a programação de páginas *web* responsivas, ou seja, páginas que se adequam ao tamanho do ecrã e ao *browser*. Além disso, esta possui *templates* para *dashboards* muito úteis e de fácil utilização.

Como o objetivo de uma *dashboard* é a apresentação de dados, foi utilizado a ferramenta **MySQL** e uma base de dados já existente, fornecida no *site* da ferramenta. Para aceder e fazer uso da informação, foi utilizada a linguagem **PHP** (*Hypertext Preprocessor*). O objetivo desta tarefa era criar uma *dashboard* para os colaboradores de uma empresa fictícia, cujo resultado está representado na Figura 2.1.

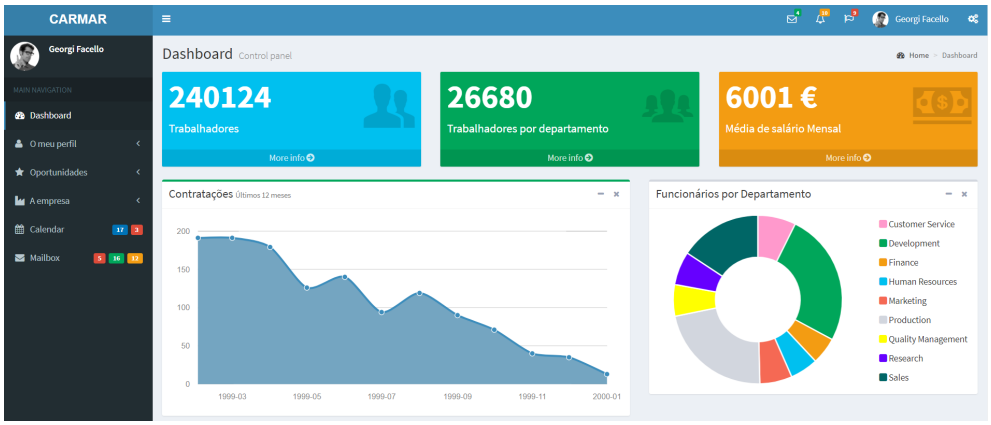


Figura 2.1: Tarefa de desenvolvimento de uma *dashboard*

Relembrando que um dos objetivos era a implementação de arquiteturas de *machine learning* para o tratamento dos dados e que a linguagem utilizada era **Python**, é necessário uma forma de utilizar a informação obtida anteriormente

[2]https://www.google.com/analytics/
[3]https://adwords.google.com/home/

e apresentá-la na *dashboard*. Para tal, foi utilizada a *framework* **Django**, que é uma *framework* escrita em **Python** para desenvolvimento rápido para *web*. Esta permite a criação de formulários, criar e manipular base de dados e utilizar resultados obtidos a partir de código em **Python** na página *web*. Nesta fase, a empresa forneceu um dos seus projetos, relacionado com uma rede social, para continuar a ser desenvolvida, utilizando esta ferramenta e as aprendidas anteriormente.

De seguida foi proposto iniciar um novo projeto no âmbito da aplicação *JuntarAJunta*, da qual a empresa é parceira. O objetivo era desenvolver um portal para os presidentes das juntas de freguesia, onde estes pudessem aceder às informações conseguidas a partir da aplicação.

Numa primeira fase, foi analisada a base de dados da aplicação e foram corrigidos alguns problemas relacionados com a inconsistência desta.

Na fase seguinte, prosseguiu-se com o desenvolvido de um projeto em **Django** para a construção do portal, utilizando um *template* da *framework* **Bootstrap**. Primeiramente, fez-se a ligação do projeto à base de dados da aplicação e o desenvolvimento e integração de um mapa *Google* com a apresentação dos limites da freguesia do utilizador. A partir do acesso à base de dados, desenvolveu-se os marcadores das participações no mapa, como se pode ver na Figura 2.2. Esta parte da tarefa foi desenvolvida a partir de *Google Maps API*[4][5][6]. Foram também desenvolvidas duas páginas *web* adicionais para a apresentação da informação em tabelas. Por fim, foram desenvolvidos mecanismos de autenticação do utilizador.

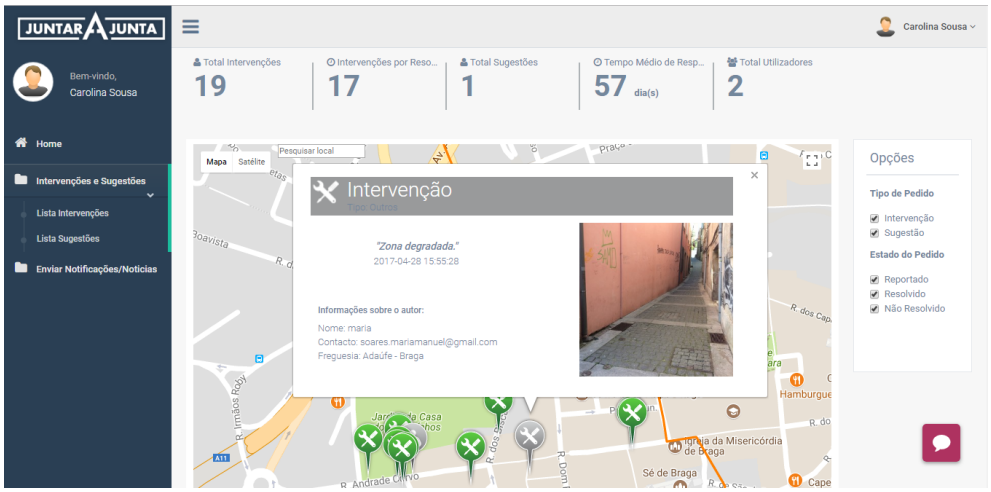


Figura 2.2: Tarefa de desenvolvimento do portal *JuntarAJunta*

[4]https://developers.google.com/maps/documentation/geocoding/start
[5]https://developers.google.com/maps/documentation/javascript/adding-a-google-map
[6]https://developers.google.com/maps/documentation/embed/guide

3 | A aplicação *JuntarAJunta*

A aplicação *JuntarAJunta* tem como objetivo a participação ativa da comunidade em decisões das juntas de freguesias de Portugal. Uma das características da aplicação é a possibilidade de reportar problemas e sugerir soluções e alterações. As zonas problemáticas são reportadas como *intervenções*. Já as *sugestões* possibilitam ao utilizador liberdade para sugerir alterações para melhorar a qualidade de vida dos cidadãos.

Cada participação (sugestão ou intervenção) é enviada à freguesia correspondente à localização da participação para esta tomar conhecimento do problema/sugestão e resolvê-lo ou reportá-lo a uma entidade competente.

Para o cidadão reportar uma sugestão ou intervenção, é necessário preencher um formulário, que permite a junta de freguesia obter informações importantes acerca da participação. No caso da sugestão, apenas é necessário o utilizador tirar uma foto e descrever a sugestão. A partir do local da submissão, a aplicação consegue obter as coordenadas GPS do local e, consequentemente, a freguesia onde se encontra. Já no caso de uma intervenção é necessário ainda selecionar uma categoria, na qual o problema está inserido.

De seguida, encontra-se a lista de categorias disponíveis.

1. Acessibilidade;
2. Árvores (Perigo);
3. Estacionamento;
4. Estrutura Perigosa;
5. Grafitti;
6. Iluminação Pública;
7. Jardins Públicos;
8. Limpeza de ruas;
9. Mobiliário urbano;
10. Obstrução da estrada ou passeio;
11. Plantação de árvores;
12. Recolha Carros abandonados;
13. Recolha de lixo;
14. Recolha Reciclagem;
15. Reparação estrada ou pavimento;
16. Saneamento;

17. Sinalética;
18. W.C. público;
19. Outros;
20. Transportes Públicos;
21. Animais abandonados;
22. Maus tratos de animais.

A Tabela 3.1 apresenta todos os dados associados a uma participação e sugestão, assim como uma pequena descrição de cada um deles.

Atributo	**Descrição**	**Intervenção**	**Sugestão**
ID	Identificação do autor da participação	x	x
Latitude	Obtida ao submeter a participação	x	x
Longitude	Obtida ao submeter a participação	x	x
Freguesia	Obtida a partir da geolocalização	x	x
Foto	Tirada através da aplicação	x	x
Descrição	Descrição do problema/sugestão	x	x
Categoria	Categoria a qual se insere o problema	x	
Estado	Resolvido, Não resolvido ou Reportado	x	x
Data	Data de submissão da participação	x	x
Alteração	Data de alteração do estado da participação	x	x

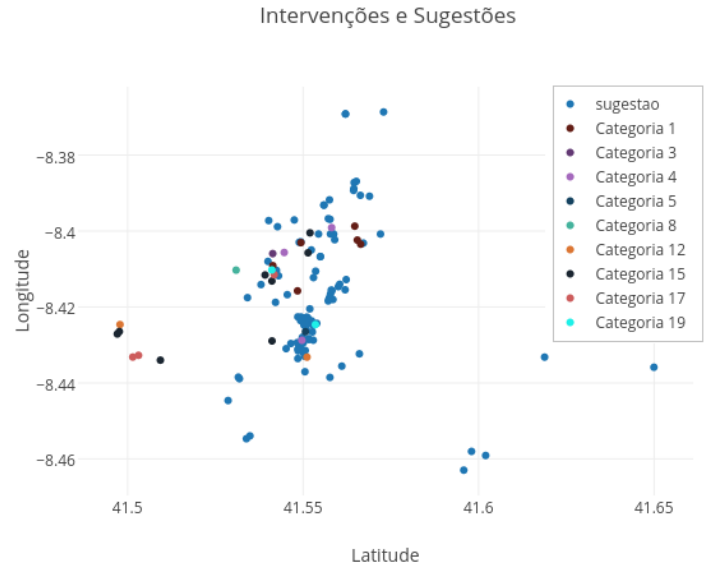
Tabela 3.1: Descrição de atributos associados a cada participação submetida na aplicação *JuntarAJunta*

No primeiro acesso à base de dados da aplicação, a 17 de março de 2017, esta continha 142 entradas. Ao longo do tempo, a informação tem vindo a aumentar e a 3 de agosto de 2017, a base de dados continha 421 entradas.

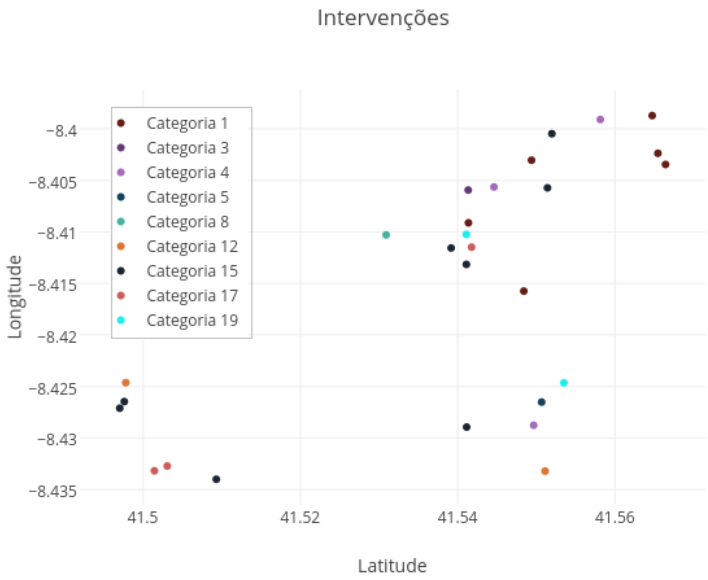
Durante a análise da informação contida na base de dados verificou-se que existiam alguns problemas de inconsistência, que estão listados de seguida.

- Há entradas repetidas na base de dados, ou seja, entradas que são do mesmo utilizador e que possuem a mesma descrição, foto, data e geolocalização, constituindo informação redundante;
- Há entradas de teste e, portanto, não constituem informação relevante;
- Há entradas que estão associadas à freguesia errada, ou seja, a freguesia obtida a partir da geolocalização não é a mesma a que foi associada a participação;
- Há entradas onde a latitude e a longitude são zero.

Verifica-se que, no momento em que foi feito este estudo, a aplicação móvel não recolhia e transformava a informação corretamente. Portanto, a informação recolhida não era muito fiável.



(a) Intervenções e Sugestões



(b) Intervenções

Figura 3.1: Estado inicial da base de dados da aplicação *JuntarAJunta*

4 | Metodologia

O problema de identificação de participações semelhantes pode ser visto como um problema de agrupamento, ou seja, como agrupar as participações em grupos representativos do problema a que estes se referem.

Assim, a metodologia que será utilizada terá como base os seguintes passos, sugeridos em [6] e [9].

1. **Identificar as variáveis a serem utilizadas**, que devem conter informação suficiente para o agrupamento correto das participações.
2. **Escolher a medida de dissemelhança**, ou seja, uma medida que reflete o grau de separação das participações a serem agrupadas.
3. **Escolher o método de *clustering*** que seja adequado ao tipo de agrupamento que se espera que esteja presente nos dados. Essa decisão é importante porque diferentes métodos de *clustering* tendem a encontrar diferentes estruturas de *clusters*.
4. **Escolher o número de *clusters*** através de critérios de paragem.
5. **Testar** vários cenários para validar o método escolhido.
6. **Interpretação dos resultados** através de representações gráficas e estatística descritiva, por exemplo.

Durante a análise da base de dados da aplicação, verificou-se que, em geral, não existia mais do que uma participação referente a um problema. Por outro lado, ao filtrar os dados de forma a não considerar aqueles que estão nas condições referidas no capítulo 3, o número de dados diminuiu consideravelmente. Por esses motivos, todas as participações que serão consideradas foram criadas sinteticamente, a fim de analisar a capacidade de funcionamento do algoritmo com cenários representativos.

Todas as implementações realizadas neste trabalho foram escritas na linguagem **Python**, fazendo uso de bibliotecas com funcionalidades de *machine*

learning como **SciPy** [1] e **scikit-learn** [2], da biblioteca **NumPy** para a geração aleatória de dados e estruturas de dados, da biblioteca **gpxpy.geo** [3] para a manipulação de dados geográficos, da biblioteca **Matplotlib** para a construção de gráficos e de funções para o cálculo do índice *Dunn* obtidas em [8].

Tendo como base [1], [2], [4] e [11], nas secções seguintes serão apresentados alguns conceitos fundamentais, onde será definida a notação utilizada neste relatório, será introduzido o método de *clustering* hierárquico e métodos de validação de *clusters*.

4.1 Conceitos Fundamentais

Seja $X = \{x_1, x_2, \ldots, x_N\}$ um conjunto de N objetos, aos quais se quer aplicar um método de *clustering*, e K o número de *clusters*. Considere-se ainda que X está contido num espaço vetorial.

Define-se um *clustering* de X em K *clusters* como sendo a partição de X em K conjuntos, $C_1, \ldots, C_K$, tal que as seguintes propriedades são satisfeitas:

1. $C_i \neq \emptyset, \quad i = 1, \ldots, K;$
2. $X = \bigcup_{i=1}^{K} C_i;$
3. $C_i \cap C_j = \emptyset, \quad i \neq j, \quad i, j = 1, \ldots, K.$

Além disso, os objetos de um *cluster* são mais "similares" entre si do que entre objetos de qualquer outro *cluster*. Dualmente, os objetos de quaisquer dois *clusters* são mais "dissimilares" entre si do que os objetos de cada *cluster*.

Seja U o conjunto constituído por K subconjuntos de X, $U = \{C_1, \ldots, C_K\}$. A *medida de dissimilaridade*, d, em U é uma função

$$d : U \times U \to \mathbb{R}$$

tal que:

1. $\exists d_0 \in \mathbb{R} : \quad -\infty < d_0 \leq d(C_i, C_j) < +\infty, \quad \forall C_i, C_j \in U$

   onde d_0 é o grau mínimo de dissimilaridade
2. $\forall C_i \in U, \quad d(C_i, C_i) = d_0$
3. $\forall C_i, C_j \in U, \quad d(C_i, C_j) = d(C_j, C_i)$

[1] https://www.scipy.org/
[2] http://scikit-learn.org/stable/
[3] https://pypi.python.org/pypi/gpxpy

d é uma métrica se também verificar as seguintes propriedades:

4. $\forall C_i, C_j \in U, \quad d(C_i, C_j) = d_0 \quad$ se e só se $\quad C_i = C_j$
5. $\forall C_i, C_z \in U, \quad d(C_i, C_z) \leq d(C_i, C_j) + d(C_j, C_z)$

Seja $d : X \times X \to \mathbb{R}$ uma métrica. Alguns exemplos de medidas de dissimilaridade estão listadas de seguida.

- **Single linkage (nearest neighbor)**: a distância entre dois *clusters* é determinada pela distância dos dois objetos mais próximos nos dois *clusters*.
$$d_{min}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$
- **Complete linkage (furthest neighbor)**: a distância entre dois *clusters* é determinada pela distância dos dois objetos mais distantes nos dois *clusters*.
$$d_{max}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$
- **Group average linkage**: a distância entre dois *clusters* é determinada pela média das distâncias entre todos os pares de objetos dos dois clusters.
$$d_{avg}(C_i, C_j) = \sum_{x \in C_i, y \in C_j} \frac{d(x, y)}{|C_i| \cdot |C_j|}$$
onde $|C_i|$ representa o número de objetos de C_i e $|C_j|$ representa o número de objetos de C_j.

Note-se que, para as medidas descritas anteriormente, quando $|C_i| = |C_j| = 1$ a medida de dissimilaridade fica definida apenas pela métrica, ou seja, a medida de dissimilaridade é a distância entre objetos. Pode então dizer-se que a medida de dissimilaridade é definida por uma métrica (distâncias entre pares de objetos) e um critério de ligação que determina a distância entre *clusters*.

Sejam x e y dois pontos definidos por coordenadas geográficas, $x = (lat_x, lon_x)$ e $y = (lat_y, lon_y)$, em radianos. A *distância de Haversine* h, entre x e y, em metros, é definida por

$$h(x, y) = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{lat_y - lat_x}{2}\right) + \cos(lat_x)\cos(lat_y)\sin^2\left(\frac{long_y - long_x}{2}\right)}\right)$$

onde $r = 6371000$ e representa o "raio" da Terra.

Seja $1 \leq i, j \leq N$. A *matriz de dissimilaridade* de X, $D(X)$, é uma matriz $N \times N$, simétrica, tal que as linhas e colunas representam os objetos de X e D_{ij} é o valor da dissemelhança entre os objetos x_i e x_j.

O *centróide* $\overline{C}$ é o ponto médio do conjunto C e é definido por

$$\overline{C} = \frac{1}{|C|} \sum_{x \in C} x$$

onde $|C|$ representa o número de objetos do conjunto C.

Neste contexto, os objetos a serem agrupados são as participações. Os métodos de *clustering* permitem agrupar objetos similares e podem ser vistos como um método conveniente para a organização de um grande conjunto de dados [6]. Consequentemente, se os dados forem resumidos de forma válida por um pequeno número de grupos de participações, os rótulos dos grupos fornecem uma descrição muito concisa do tipo de evento reportado, ou seja, uma descrição dos padrões de semelhança e diferença nos dados.

4.2 *Clustering* Hierárquico

O método de *clustering* Hierárquico tem como objetivo encontrar grupos tal que os objetos de um grupo são mais semelhantes entre si do que entre objetos de outros grupos, através de uma medida de dissimilaridade [3].

De uma forma geral, o método cria uma hierarquia de *clusters* tal que, a cada etapa, um novo *clustering* de X é obtido a partir do *clustering* obtido na etapa anterior. Existem duas estratégias: aglomerativa e divisiva.
A estratégia aglomerativa é uma abordagem *bottom up* sendo que, no início, cada objeto representa um *cluster* e vai-se subindo na hierarquia à medida que se vão unindo pares de *clusters*, até que termina com um único *cluster*, contendo todos os objetos. Além disso, se dois *clusters* se unem num único *cluster* na etapa t, estes continuarão no mesmo *cluster* em todas as etapas seguintes.
Por outro lado, a estratégia divisiva é uma abordagem *top down* tal que, no início todos objetos estão num único *cluster* e à medida que se vai descendo na hierarquia os *clusters* vão se dividindo até que, no fim, cada objeto representa um *cluster*. Cada um dos k *clusters* formados na etapa t é um subconjunto de um dos $k-1$ *clusters* formados na etapa $t-1$.

Assim, ambas as estratégias podem ser vistas como tentativas de encontrar o passo ideal em cada etapa da divisão ou união [6].

Utilizando uma estratégia aglomerativa e uma medida de dissemelhança, o método comporta-se da seguinte forma:

1. A etapa 1, inicializa-se com N *clusters*, tal que cada *cluster* contém um objeto, $U_1 = \{\{x_1\}, \ldots, \{x_N\}\}$;

2. Na etapa t, calcula-se a dissemelhança entre todos os possíveis pares de *clusters* definidos na etapa anterior, $t-1$, e identifica-se o par que produz a dissimilaridade mínima. Seja (C_i, C_j) esse par;

3. Define-se $C_q = C_i \cup C_j$ e produz-se um novo *clustering* de X, U_t, tal que $U_t = (U_{t-1} \setminus \{C_i, C_j\}) \cup \{C_q\}$;

4. Os passos 2 e 3 repetem-se até todos os objetos de X pertencerem ao mesmo *cluster*, na etapa $t = N$.

Assim, pode-se dizer que a medida de dissimilaridade é a medida que decide a união ou não de *clusters* e esta medida é escolhida em função dos dados e do contexto onde será aplicado.

Todo este processo pode ser representado na forma de um *dendrograma*, que é uma estrutura de duas dimensões que sumariza medidas de dissimilaridade e ilustra as divisões ou uniões que ocorreram em cada passo na análise. A altura da união ou divisão no dendrograma é a dissimilaridade entre os dois *clusters*.

Na maior parte das aplicações do método de identificação de *cluster*, é necessário estimar o número de *clusters* adequado, sendo que o investigador que procura o número "ideal" de *clusters* terá de decidir quando o algoritmo pára. Existe uma grande variedade de métodos que ajudam nesta decisão, sendo que os mais utilizados são métodos informais que involvem, essencialmente, representações gráficas a partir do valor do critério de *clustering* e o número de grupos. Na análise da representação gráfica, as grandes alterações sugerem o valor de *clusters* a ser considerado [6].

Supondo que o limite para a dissimilaridade é z, o algoritmo devolve os *clusters* formados tal que a dissimilaridade de todos esses *clusters* seja inferior a z. Por outras palavras, "corta-se" o dendrograma horizontalmente em z e os *clusters* devolvidos pelo algoritmo são os *clusters* identificados abaixo do corte.

4.3 Validação de *clusters*

A validação dos resultados de *clustering* é um passo importante pois permite determinar a qualidade dos *clusters*, o grau de adaptação do método de agrupamento ao conjunto de dados e a identificar o número "ideal" de *clusters*. Além disso, também permite a comparação entre dois métodos de *clustering*.

As estatísticas de validação de *cluster* podem ser agrupadas em três grupos:

- **Validação interna de *clusters***, que consiste em utilizar a informação interna do processo de *clustering* para avaliar o quão bom é a estrutura do *cluster*, sem referência a informações externas;
- **Validação externa de *clusters***,que consiste em comparar os resultados de uma análise de *cluster* com um resultado conhecido externamente, como rótulos fornecidos externamente;
- **Validação relativa de *clusters***, que consiste em avaliar a estrutura do agrupamento através da variação dos valores dos parâmetros, para o mesmo algoritmo.

Dado o objetivo dos algoritmos de *clustering*, quer-se que a distância média entre os objetos do mesmo *cluster* seja tão pequena quanto possível e a distância média entre *clusters* seja a maior possível. Assim, as medidas de validação interna passam por avaliar a coesão, a conexão e a separação do *cluster*.

- A **coesão do *cluster*** mede o quão perto estão os objetos dentro de um mesmo *cluster*. Assim, uma pequena variação dos objetos do *cluster* é um indicador de uma boa coesão, ou seja, um bom agrupamento. Esta medida baseia-se em distâncias, por exemplo, média das distâncias entre objetos.
- A **separação** mede o quão bem separados os *clusters* estão. Assim, os indicadores usados como medida de separação passa por analisar as distâncias entre os centros dos *clusters* e as distâncias mínimas entre diferentes *clusters*.
- A **conexão** mede até que medida os objetos devem estar no mesmo *cluster* que os objetos vizinhos.

Em geral, a maior parte dos indicadores usados para a validação interna de *cluster* combinam as medidas de coesão e separação no mesmo indicador.

Três dos indicadores mais usados para avaliar a qualidade dos *clusters* são o índice *Silhouette*, o índice *Dunn* e o índice *Calinski-Harabasz*.

Resumidamente, o índice *Silhouette* mede o quão bem os objetos foram agrupados e estima a distância média entre *clusters*, sendo que a coesão é medida com base na distância entre objetos de um mesmo *cluster* e a separação é medida com base na distância ao *cluster* vizinho mais próximo.

Assim, o valor de *silhouette* S_k, para k *clusters*, é dado por:

$$S_k = \frac{1}{N} \sum_{i=1}^{k} \sum_{x \in C_i} \frac{\mathrm{b}(x, C_i) - \mathrm{a}(x, C_i)}{\max(\mathrm{b}(x, C_i) - \mathrm{a}(x, C_i))}$$

onde,

$$\begin{aligned} \mathrm{a}(x, C_i) &= \frac{1}{|C_i|} \sum_{y \in C_i} d(x, y) \\ \mathrm{b}(x, C_i) &= \min_{1 \leq j \leq k, i \neq j} \left(\frac{1}{|C_j|} \sum_{y \in C_j} d(x, y) \right) = \min_{1 \leq j \leq k, i \neq j} (\mathrm{a}(x, C_j)) \end{aligned}$$

O valor deste índice varia entre -1 e 1, sendo que um valor próximo de 1 significa que os objetos estão bem agrupados, um valor próximo de 0 significa há *clusters* que se sobrepõem e um valor negativo significa que os objetos não foram bem agrupados.

O índice *Dunn* é calculado a partir do cálculo das distâncias entre os objetos de diferentes *clusters* e do cálculo das distâncias entre os objetos de um mesmo *cluster*. Assim, o índice de *Dunn* D_k, para k *clusters*, é definido por :

$$D_k = \frac{\min\limits_{1 \leq i < j \leq k} \delta(C_i, C_j)}{\max\limits_{1 \leq z \leq k} \Delta(C_z)}$$

onde,

$$\begin{aligned} \delta(C_i, C_j) &= \min_{x \in C_i, y \in C_j} d(x, y) \\ \Delta(C_i) &= \max_{x, y \in C_i} d(x, y) \end{aligned}$$

Quanto maior o valor obtido para o índice *Dunn*, melhor estão definidos os *clusters*.

O índice *Calinski-Harabasz* também é um índice baseado num quociente. A coesão é estimada com base na distância entre os objetos de um mesmo *cluster* e o seu centróide, ou seja, o ponto médio do *cluster*. A separação é

baseada na distância entre os centróides dos *clusters* e o centróide global, ou seja, o ponto médio de todos os objetos.

Assim, o índice *Calinski-Harabasz* CH_k, para k *clusters*, é definido por:

$$CH_k = \frac{N-k}{k-1} \times \frac{\sum_{i=1}^{k} |C_i| d(\overline{C_i}, \overline{X})}{\sum_{i=1}^{k} \sum_{x \in C_i} d(x, \overline{C_i})}$$

onde N é o número total de objetos.

Quanto maior o valor obtido para o índice *Calinski-Harabasz*, melhor estão definidos os *clusters*.

O índice *Fowlkes-Mallows* é um índice de validação externa que permite calcular a precisão de um resultado de *clustering* quando se sabe o verdadeiro agrupamento dos dados.

Define-se um *Verdadeiro Positivo* quando dois objetos pertencem ao mesmo *cluster* no agrupamento verdadeiro e no resultado do método de *clustering*.

Define-se um *Falso Positivo* quando dois objetos pertencem ao mesmo *cluster* no agrupamento verdadeiro e a *clusters* diferentes no resultado do método de *clustering*.

Define-se um *Falso Negativo* quando dois objetos pertencem ao mesmo *cluster* no resultado do método de *clustering* e a *clusters* diferentes no agrupamento verdadeiro.

Assim, o índice de *Fowlkes-Mallows* FM é definido por:

$$FM = \frac{VP}{\sqrt{(VP+FP)(VP+FN)}}$$

onde VP é o número de *verdadeiros positivos*, FP é o número de *falsos positivos* e FN é o número de *falsos negativos* de U. O valor deste índice varia entre 0 e 1, sendo que um valor próximo de 1 significa que existe uma grande similaridade entre os *clusters* obtidos e os *clusters* corretos.

5 | Identificação de participações semelhantes

O desafio proposto foi a identificação de participações semelhantes. Mais especificamente, estudar e analisar o problema a partir de técnicas de *machine learning*, de forma a produzir indicadores que ajudem na identificação do número de *clusters* e, consequentemente, identificar grupos de participações semelhantes. Nesse estudo serão primeiro definidas e justificadas as variáveis a serem utilizadas, seguindo-se de um método de processamento dos dados. De seguida será descrito o método de *clustering* hierárquico utilizado neste relatório e a sua capacidade em identificar participações semelhantes será testada em cenários sintéticos.

5.1 Seleção de variáveis

Uma participação representa um dado obtido através da aplicação móvel *JuntarAJunta*, relativo a um problema ou sugestão. A localização geográfica do problema ou sugestão será designada por *evento*, sendo que cada participação está associada a um evento.

Seja m o número de eventos e $1 \leq i \leq m$. O *evento* i é denotado por $E_i \in \mathbb{R}^2$, e é definido por:

$$E_i = (latitude, longitude)$$

Seja n o número de participações e $1 \leq j \leq n$. A *participação* j é denotada por P_j.

Assim, as participações associadas a um mesmo evento representam participações semelhantes.

Uma das fases principais na aplicação de métodos de *clustering* é a identificação de variáveis que serão importantes no processo. A latitude e a longitude são as variáveis mais importantes porque é a partir destas que se consegue

determinar a proximidade entre duas participações. Por um lado, sabe-se que, se existem duas participações que identificam o mesmo problema/sugestão, é provável que estas estejam próximas. Por outro lado, é possível haver duas participações muito próximas mas que, na realidade, são relativas a assuntos diferentes. Assim, o conceito de proximidade e o valor que define são cruciais pois é a partir destes que se decide se duas participações representam o mesmo problema/sugestão ou não.

Ora, a noção de proximidade difere consoante o objeto fotografado. Por exemplo, um objeto de maiores dimensões é fotografado a uma distância maior do que um de menores dimensões. Consequentemente, as participações referentes a um mesmo objeto estarão a uma distância maior uma das outras. Em [10] verificou-se que a falta de informações como distância ao objeto ou tamanho do objeto, prejudicou e dificultou a identificação correta de *clusters*. Concluiu-se que apenas a posição geográfica não era suficiente para distinguir as participações no espaço e, consequentemente, não era suficiente para determinar um valor de dissemelhança máximo para a identificação de *clusters*.

Todas as participações, que sejam intervenções, têm associado uma categoria e esta pode ajudar na identificação de participações semelhantes. Por exemplo, se houver duas participações muito próximas geograficamente mas com categorias muito distintas, então é muito provável que estas duas participações não se estejam a referir ao mesmo problema. Por outro lado, duas participações semelhantes podem ter associadas categorias diferentes mas provavelmente essas categorias estão próximas, no sentido em que ambas se adequam ao problema. Portanto, no caso de se utilizar dados reais, seria necessário integrar essa semelhança de categorias na análise porque pode ser uma característica diferenciadora na procura de participações semelhantes.

Uma das formas de determinar essa distância entre categorias seria aplicando métodos de aprendizagem supervisionada. Ora, esses métodos necessitam de um grande volume de dados, que não se tem neste momento. Em alternativa, poderia-se agrupar as 22 categorias em grupos representativos mais gerais. Um exemplo de um possível agrupamento está listado de seguida.

- *Espaços verdes*: categorias 2, 7 e 11;
- *Estradas, passeios e caminhos*: categorias 1, 10 e 15;
- *Higiene urbana*: categorias 5, 8, 12, 13, 14, 16 e 18;
- *Iluminação*: categoria 6;
- *Sinalética*: categoria 17;

- *Estrutura perigosa*: categoria 4;
- *Outros*: categorias 3, 9 e 19;
- *Animais*: categorias 21 e 22;
- *Serviços públicos*: categoria 20;

As variáveis relativas à fotografia e à descrição da participação podem trazer novas informações pois é bastante provável que haja palavras iguais entre duas participações semelhantes e semelhanças entre as fotografias. Como os dados que serão utilizados não são os dados reais, não será possível fazer uso destas variáveis.

Assim, será considerado que todos os dados criados artificialmente estão associados à mesma categoria representativa e à mesma freguesia.

5.2 Introdução de novas variáveis

A solução encontrada para contornar os problemas encontrados e descritos em [10] foi a introdução de novas variáveis.

Ao contrário da maior parte da literatura existente acerca de *clustering* espacial para a identificação de pontos de interesse (POI) que utiliza apenas as coordenadas geográficas, em [7] é utilizado a orientação geográfica, ou seja, o ângulo do dispositivo móvel relativamente ao eixo Sul-Norte, como um critério para o *clustering* de POI's. Uma das vantagens do algoritmo proposto é a obtenção de resultados mais precisos em certos cenários.

(a) Usando geolocalização

(b) Usando geolocalização e orientação

Figura 5.1: Diferença na metodologia de identificação de *clusters*

Na Figura 5.1 está representado um exemplo onde esta vantagem é visível. Sejam E_1 e E_2 dois eventos e P_1 uma participação. Utilizando um algoritmo de *clustering* com apenas a posição geográfica, P_1 seria relacionado ao evento E_1 pois este é o evento que se encontra mais próximo (Figura 5.1a). Supondo

que a orientação é a demonstrada na Figura 5.1b, esse agrupamento não faria sentido pois a câmara está direcionada a E_2.

De modo a melhorar a precisão dos resultados e obter mais informação para a identificação do valor máximo de dissemelhança, serão consideradas duas novas variáveis: orientação geográfica da fotografia e a distância do local onde o utilizador se encontra e o objeto fotografado.

Assim, a *participação* $P_j \in \mathbb{R}^4$ será definida por:

$$P_j = (\ \textit{latitude, longitude, orientação, distância}).$$

onde *latitude* e *longitude* representam as coordenadas geográficas da participação, *orientação* representa a orientação obtida pelo GPS ao tirar a fotografia e *distância* representa um intervalo de distância referente a uma aproximação da distância entre o utilizador e o evento.

Neste momento serão consideradas as seguintes opções de intervalos de distância: de 0 a 10 metros, de 10 a 30 metros, de 30 a 50 metros e mais de 50 metros.

5.3 Processamento dos dados

O método apresentado em [7] para processar os dados com o objetivo de os usar num método de *clustering* começa por traçar segmentos de reta a partir da localização das participações, de tamanho d e seguindo a sua orientação geográfica. Neste contexto, o segmento de reta terá início a uma distância correspondente ao valor mínimo do intervalo e acabará a um distância correspondente ao valor máximo do intervalo. No caso da distância ser mais de 50 metros, será considerada uma semirreta com início a uma distância de 50 metros da localização da participação. Portanto, define-se que uma participação P_j *tem associado uma reta* r_j, definida a partir da orientação e do intervalo de distância de P_j.

Como as fotografias estão, normalmente, direcionadas ao centro do objeto, as retas acabam por se intersetar, criando novos POI's (os pontos de interseção) que serão usados no método de *clustering*.

No caso da reta não gerar cruzamento, será considerada como critério para a identificação de *clusters* uma posição geográfica. Se for um segmento de reta, a posição geográfica a considerar será a posicionada a meio do segmento de reta pois, desta forma, minimiza-se a distância entre o ponto e o evento reportado. No caso de uma semirreta, será considerada a posição geográfica posicionada no início da semirreta. Todos estes pontos serão considerados como pontos de interseção.

Assim, defini-se que um ponto de interseção, POI, está associado a uma ou duas participações consoante houve ou não interseção das retas associadas.

A Figura 5.2 representa a ideia geral do algoritmo de processamento dos dados que está a ser proposto. Pela Figura verifica-se que as participações P_1 e P_3 estão a uma distância maior do que 10 metros dos eventos E_1 e E_2, respetivamente, pois os seus segmentos de reta não têm início nas suas posições e existe limite para a proximidade entre as participações e os eventos. A participação P_2 está a uma distância de 0 a 10 metros de E_1 pois o seu segmento de reta tem início na sua localização. Já a participação P_4 está a uma distância de E_3 superior a 50 metros, produzindo uma semirreta. Assim, as linhas representadas a cinzento são os segmentos de reta e a semirreta produzidos pelo método. Os segmentos de retas correspondentes às participações P_1 e P_2 cruzam-se próximo do evento E_1 (em POI_1) criando um ponto de interesse que será utilizado no método de *clustering* hierárquico. O segmento de reta correspondente à participação P_3 e a semirreta correspondente à participação P_4 não intersetam com nenhuma outra reta. Nestes casos, é necessário verificar que tipo de reta se trata de modo a que se considere os pontos POI_2 e POI_3 como os pontos de interesse que serão utilizados no método de *clustering* hierárquico.

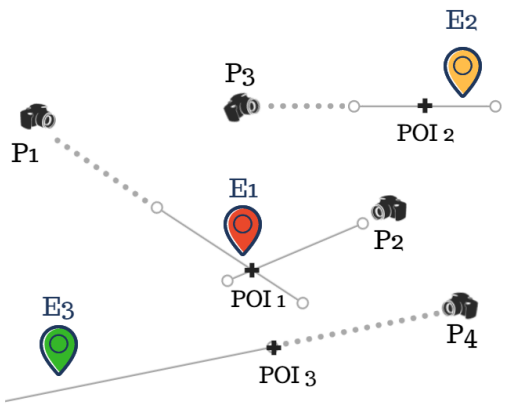


Figura 5.2: Exemplo do comportamento do algoritmo de processamento dos dados

Mais especificamente, o algoritmo de processamento dos dados comporta-se como está representado na Figura 5.3, sendo que n representa o número de participações, $1 \leq i \leq n$, $i+1 \leq j \leq n$ e $M \in \mathbb{R}^n$ uma matriz, inicialmente de zeros, que irá guardar a informação das retas que se intersetam. Sabendo que $m_{ij} \in M$, se $m_{ij} = 0$ então pode dizer-se que as retas i e j não se intersetam ou são segmentos de reta coincidentes. Por outro lado, se $m_{ij} = 1$ então

pode dizer-se que as retas i e j se intersetam. O intervalo de distância será representado por d.

Assim, o algoritmo devolverá a matriz M e os pontos de interseção, que serão utilizados no método de *clustering* para identificar grupos de participações semelhantes.

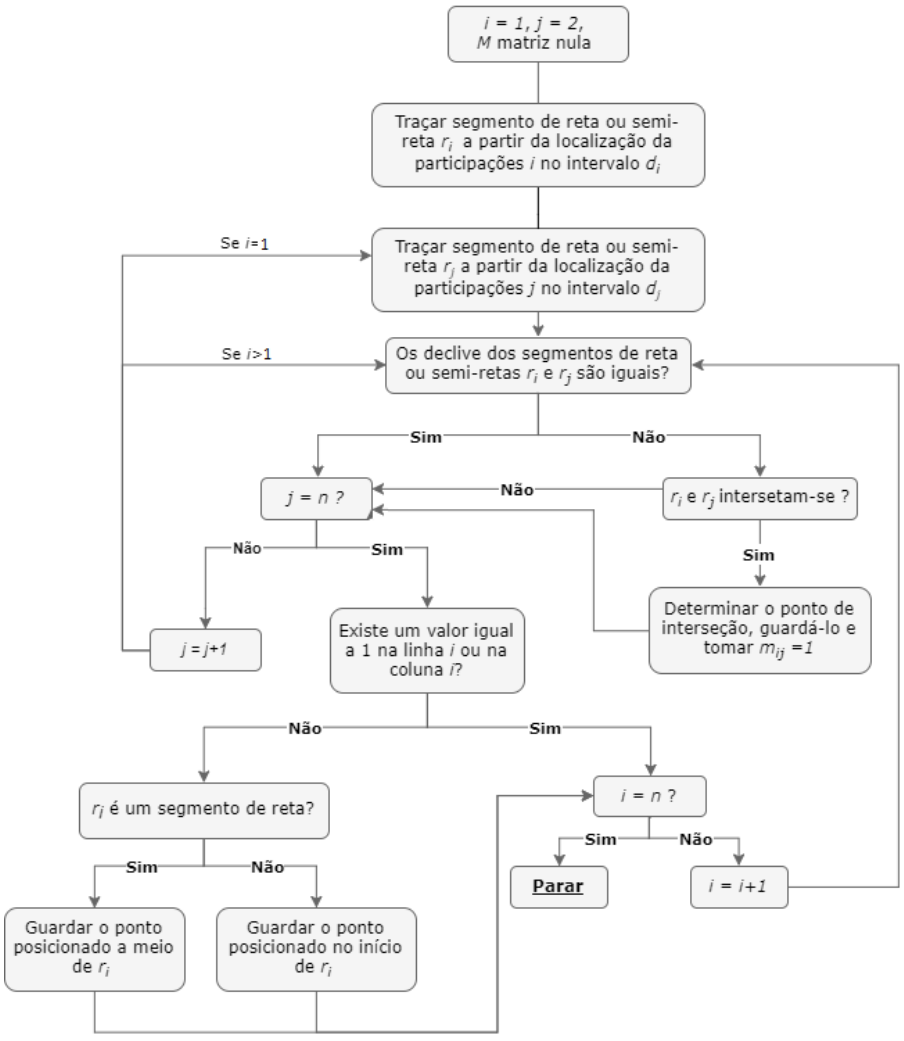


Figura 5.3: Diagrama do algoritmo de processamento dos dados

5.4 Escolha da medida de dissimilaridade

Como foi visto anteriormente, a medida de dissimilaridade é definida por uma métrica e um critério de ligação. Como cada ponto de interseção é definido pela sua latitude e longitude, faz sentido escolher a métrica de *Haversine* como sendo a distância entre pontos de interseção. Por outro lado, a escolha do critério de ligação não é tão simples.

Em [10] foram criados cenários prováveis a partir da geração aleatória de

coordenadas geográficas, para avaliar qual dos critérios de ligação, mencionados na secção 4.2, era o mais adequado para agrupar participações. Concluiu-se que o critério *group average* era o critério que produzia melhores resultados.

Esta nova versão também utiliza coordenadas geográficas e os pontos de interseção, em geral, formam cenários isotrópicos, ou seja cenários caracterizados pela uniformidade da distribuição das participações em todas as direções. Assim, os resultados obtidos continuam válidos e, portanto, será utilizado como critério de ligação o critério *group average*.

5.5 Método de *clustering* hierárquico

Depois de processados os dados, como foi explicado na secção 5.3, pode-se aplicar o método de *clustering* hierárquico.

A aplicação do método a pontos de interseção resultará em *clusters* de pontos de interseção. Ora, cada ponto de interseção está associado a uma ou duas retas. Consequentemente, os pontos de interseção estão associados a participações. Assim, os *clusters* de pontos de interseção também são *clusters* de participações.

Por exemplo, suponhamos que os segmentos de reta r_1 e r_2 se intersetam no ponto POI_1 e que as retas r_2 e r_3 se intersetam no ponto POI_2. Suponhamos também que o método de *clustering* identifica o *cluster* formado pelos pontos POI_1 e POI_2. Pode-se então dizer que o *cluster* é formado pelas participações que dão origem às retas r_1, r_2 e r_3.

Por outro lado, estes tipo de dados permitem a existência de mais do que um ponto de interseção relativo à mesma participação. Consequentemente, possibilita a participação de estar associada a mais do que um *cluster*, contrariando os princípios do método de *clustering* hierárquico.

Outra desvantagem é a possibilidade de existirem pontos de interseção de retas que não representam participações relativas ao mesmo evento. Basta uma fotografia ser tirada mais longe do local do evento, que é muito provável que a reta que lhe está associada intersetar outras retas relativas a outros eventos. Pode-se considerar tais pontos como sendo pontos enganosos, que podem interferir negativamente no processo de identificação de *clusters*.

Um ponto é um *ponto enganoso* se as retas que lhe dão origem estão associadas a *clusters* distintos.

Observa-se ainda que, se existirem várias participações relativas ao mesmo evento, os pontos de interseção estão muito próximos uns dos outros porque, em geral, as fotografias estão direcionadas ao centro do evento. Consequentemente,

é muito provável que, durante o processo de *clustering*, estes pontos sejam dos primeiros a serem unidos. Portanto, o mais provável é, no momento da união do ponto enganoso com outro ponto ou *cluster*, as retas que o formam já estarem associadas a *cluster* distintos, tornando esta união impossível de ocorrer e definindo este ponto como sendo um ponto enganoso.

Na Figura 5.4 está representada esta ideia. Suponhamos que o segmento de reta verde já está associado a um *cluster* e que participações de cores diferentes estão associadas a eventos diferentes. Os pontos de interseção POI_1, POI_2 e POI_3 serão os primeiros a serem associados ao mesmo *cluster* C pois são os pontos que estão mais próximos uns dos outros. Suponhamos que na próxima iteração do algoritmo de *clustering* hierárquico aglomerativo será a união do *cluster* C com o ponto de interseção POI_4. Tal união não poderá acontecer pois as duas retas associadas ao ponto de interseção já estão associadas a *clusters* diferentes.

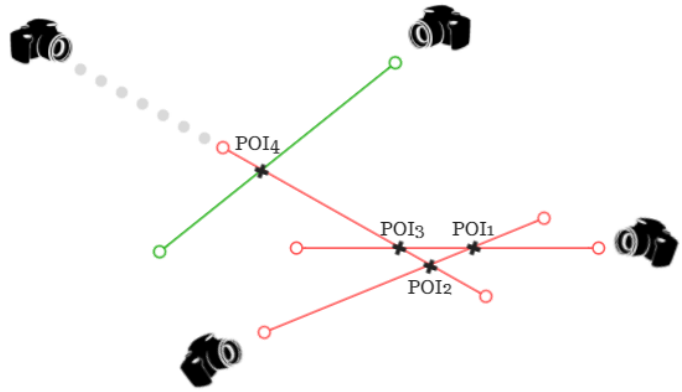


Figura 5.4: Exemplo da identificação de pontos enganosos

No caso contrário, isto é, se as retas já estiverem associadas ao mesmo *cluster*, é certo que este ponto pertence ao *cluster* já formado e, portanto, a união ocorre se o outro ponto ou *cluster* também pertencer a esse *cluster*. No caso de apenas uma reta estar associada a um *cluster*, será considerado que o ponto não é um ponto enganoso e que ambas as retas pertencem ao mesmo *cluster*. Desta forma, cada participação só estará associada a um único *cluster* e o processo de determinação de pontos enganosos terá de ser realizado durante o método de *clustering* hierárquico.

Resumidamente, o método de *clustering* hierárquico aglomerativo terá de ser reajustado de forma a filtrar aos pontos de interseção, eliminando pontos que induzem a erro e que não acrescem informação importante na identificação de *clusters*, e garantir que cada participação fica apenas associada a um único *cluster*.

Como ainda não foi definido o valor de limite para a dissemelhança, será utilizado o método *linkage*, da biblioteca *Scipy*, com o objetivo de calcular uma matriz que contém toda a informação do método de *clustering* hierárquico original, ou seja, todas as uniões e dissemelhanças calculadas. Este terá como *input* a lista de pontos de interseção, o critério de ligação *average* e a métrica de *Haversine*. A essa matriz será aplicado o método de filtragem e a restrição de que uma participação apenas pode estar associada a um único *cluster*.

Seja L a matriz obtida usando o método *linkage*, *pontosIntersecao* uma lista com as coordenadas geográficas dos pontos de interseção e *IntersecaoRetas* uma lista com a informação das retas intersetadas. Seja ainda *nParticipacoes* o número de participações e *distMaxima* o valor máximo para a dissemelhança, escolhido pelo utilizador. O algoritmo percorre L até que o valor da dissemelhança seja superior a *distMaxima*. Caso não aconteça, toda a matriz L é percorrida.

Cada linha de L é composta por uma lista definida da seguinte forma:

$$[C_i, C_j, dist, m]$$

onde C_i e C_j representam *clusters* a serem unidos, a um valor de dissemelhança *dist* e produzindo um novo *cluster* com m elementos. Se $|C_i| = 1$ (resp. $|C_j| = 1$), pode-se dizer que C_i (resp. C_j) representa um ponto de interseção.

Durante o processo, serão verificados os seguintes casos:

1. **Caso C_i e C_j representem dois pontos de interseção** então, estamos perante a possível formação de um novo cluster. Os pontos POI_i e POI_j de C_i e C_j, respetivamente, serão analisados com o objetivo de verificar se estes são pontos enganosos.

   (a) Se algum dos pontos for um ponto enganoso, então esse ponto não deve ser considerado na análise e a criação de um novo *cluster* não acontece.

   (b) Se pelo menos uma das retas que forma o ponto POI_i (resp. POI_j) já estiver associada a um *cluster* C_l então, a outra reta também pertencerá ao *cluster* C_l e a criação de um novo *cluster* não acontece.

   (c) Se nenhuma das retas que forma os pontos POI_i e POI_j tiver associado um *cluster* e a distância entre eles, $dist$, não for superior a $distMaxima$, então um novo *cluster* é formado com os pontos POI_i e POI_j.

2. **Caso apenas C_i (resp. C_j) represente um ponto de interseção** então, estamos perante a união de um ponto de interseção a um *cluster*. O ponto POI_i (resp. POI_j) será analisado com o objetivo de verificar se este é um ponto enganoso.

   (a) Se POI_i (resp. POI_j) for um ponto enganoso, então esse ponto não deve ser considerado na análise e POI_i (resp. POI_j) não é associado ao *cluster* C_j (resp. C_i).

   (b) Se pelo menos uma das retas que forma o ponto POI_i (resp. POI_j) já estiver associada a um *cluster* que não é o C_j (resp. C_i) então, POI_i (resp. POI_j) não é associado ao *cluster* C_j (resp. C_i).

   (c) Se nenhuma das retas que forma o ponto POI_i (resp. POI_j) tiver associado um *cluster* e a distância entre o ponto e o *cluster*, *dist*, não for superior a $distMaxima$, então este ponto será associado ao *cluster* C_j (resp. C_i).

3. **Caso C_i e C_j não representem pontos de interseção** então, estamos perante a união de dois *clusters*. Tal união acontece se a distância entre eles, *dist*, não for superior a $distMaxima$.

Assim, no final da execução deste algoritmo, estarão identificados os *clusters* de pontos de interseção e grupos de participações semelhantes. É de notar que, dependendo do valor máximo de dissemelhança escolhido, os pontos enganosos podem não estar todos identificados. Consequentemente, poderão existir *clusters* constituídos apenas por pontos enganosos. Apesar disso, os grupos de participações estão bem definidos.

Portanto, caso seja necessário utilizar a informação acerca dos *clusters* de pontos de interseção, é necessário verificar se nesses estão incluídos *clusters* de pontos enganosos e, em caso afirmativo, removê-los da solução dada pelo algoritmo.

Por outro lado, não é aconselhado a remoção dos pontos enganosos dos dados e, em seguida, aplicar este algoritmo porque é possível identificar pontos como sendo pontos enganosos mas que na realidade não o são. Além disso, ao eliminar pontos, é reestruturada toda a hierarquia podendo alterar os valores obtidos.

5.6 Escolha do valor máximo de dissemelhança

O método de *clustering* hierárquico baseia-se numa fusão hierárquica de objetos e, portanto, uma representação gráfica deste processo pode ser útil na

identificação de um valor máximo para a dissemelhança.

O processo descrito na secção 5.5 não permite a sua representação na forma de um dendrograma uma vez que a identificação de *clusters* a partir de pontos de interseção não é feita diretamente. O processo representado no dendrograma é o agrupamento de pontos de interseção tendo em conta a sua proximidade mas o resultado que se pretende é a identificação de grupos de participações semelhantes. Portanto, o resultado do dendrograma não é o resultado final do processo mas sim um resultado intermédio.

Contudo, o dendrograma pode dar informações importantes acerca do estrutura dos dados e, por isso, ajuda na análise e escolha de um valor máximo para a dissemelhança. Em geral, grandes mudanças nos níveis de fusão é um indicador de um limite para a dissemelhança [6]. Uma das formas que será utilizada para verificar tais mudanças é a representação gráfica do valor de fusão (ou seja, o valor da dissemelhança) em função do número de *clusters*.

No caso de se estar a analisar dados reais, não haverá informações externas acerca dos verdadeiros *clusters*. Por esse motivo, serão utilizados critérios de validação interna dos *clusters* para identificar um limite para o valor da dissemelhança e consequentemente, o número de *clusters*.

Uma vez que os dados que serão tratados neste relatório possuem informação acerca dos verdadeiros agrupamentos dos dados, os critérios de validação externa de *clusters* serão aplicados para medir a precisão dos resultados.

Por outro lado, os critérios de validação relativa foram utilizados em [10] para validar o critério de ligação escolhido para o método de *clustering*.

Assim, neste relatório serão utilizados critérios de validação interna e externa dos *clusters*.

5.7 Aplicação do método

Para ilustrar a capacidade do método em determinar *clusters* corretamente serão considerados dois cenários sintéticos. O primeiro cenário corresponde a um cenário onde estão contempladas participações distribuídas de forma anisotrópica e isotrópica e o segundo corresponde ao cenário 5 criado em [10].

Os cenários isotrópicos caracterizam-se pela uniformidade da distribuição das participações em todas as direções, enquanto que nos cenários anisotrópicos as participações estão distribuídas tendencialmente em certas direções. A aplicação do método desenvolvido em cenários isotrópicos e anisotrópicos é importante pois estes cenários representam de uma forma mais fiel situações reais.

Uma vez que se sabe como os dados estão verdadeiramente agrupados e de forma a facilitar a compreensão e leitura dos resultados obtidos, será denotado por P_{ij} a participação j relativa ao evento E_i e $[P_{ij}, P_{kl}]$ (resp. $[P_{ij}]$) o ponto de interseção associado às participações P_{ij} e P_{kl} (resp. P_{ij}).

5.7.1 Cenário 1

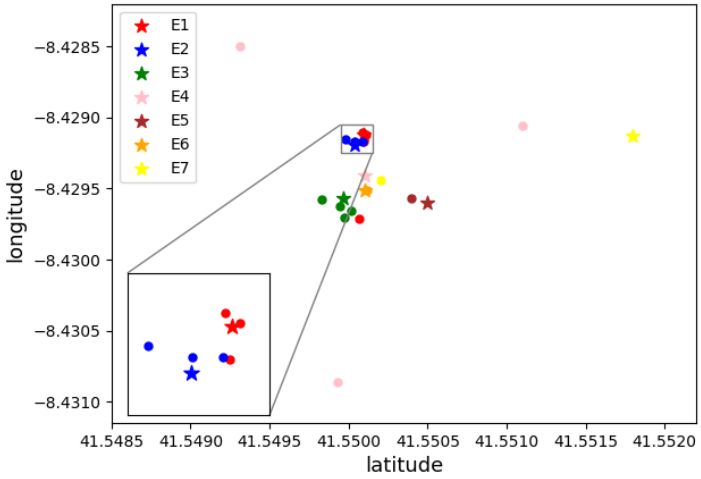


Figura 5.5: Representação gráfica do cenário 1

Este cenário, representado na Figura 5.5, consiste em sete eventos, E_1, E_2, E_3, E_4, E_5, E_6 e E_7, e dezassete participações distribuídas da seguinte forma:

O evento E_1 tem associado 4 participações: P_{11}, P_{12}, P_{13} e P_{14}.

O evento E_2 tem associado 3 participações: P_{21}, P_{22} e P_{23}.

O evento E_3 tem associado 4 participações: P_{31}, P_{32}, P_{33} e P_{34}.

O evento E_4 tem associado 3 participações: P_{41}, P_{42} e P_{43}.

O evento E_5 tem associado 1 participação: P_{51}.

O evento E_6 tem associado 1 participação: P_{61}.

O evento E_7 tem associado 1 participação: P_{71}.

Destes dados, 52.9% estão a uma distância entre 0 e 10 metros, 17.6% estão a uma distância entre 10 e 30 metros e 29.4% estão a uma distância superior a 50 metros do evento correspondente.

O objetivo deste cenário é ilustrar a capacidade do algoritmo em determinar *clusters* corretamente em dados distribuídos, maioritariamente, de forma anisotrópica. Além disso, comparar os resultados obtidos pelo método desenvolvido neste relatório com os obtidos utilizando o método de *clustering* hierárquico aplicado às coordenadas geográficas das participações.

A estas 17 participações foram aplicados o algoritmo de processamento dos dados, descrito na Figura 5.3, resultando em 23 pontos de interseção. Pela análise do resultado obtido, verificou-se que a semirreta correspondente à participação P_{71} e o segmento de reta correspondente à participação P_{51} não intersetam com nenhuma outra reta. Portanto, foram consideradas a coordenada geográfica obtida a uma distância de 50 metros de P_{71} e a coordenada geográfica do ponto médio do segmento de reta associado a P_{51}.

Na Figura 5.6 está representado graficamente o resultado do método: as linhas a cinzento representam a orientação e os pontos '+' a preto representam os pontos de interseção.

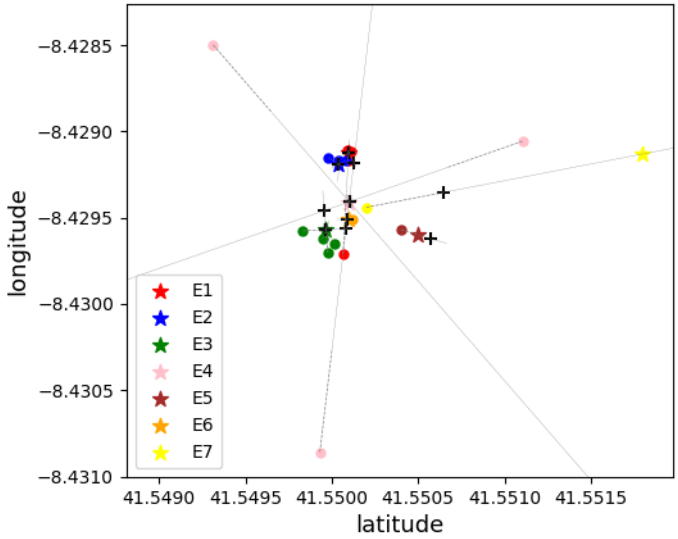
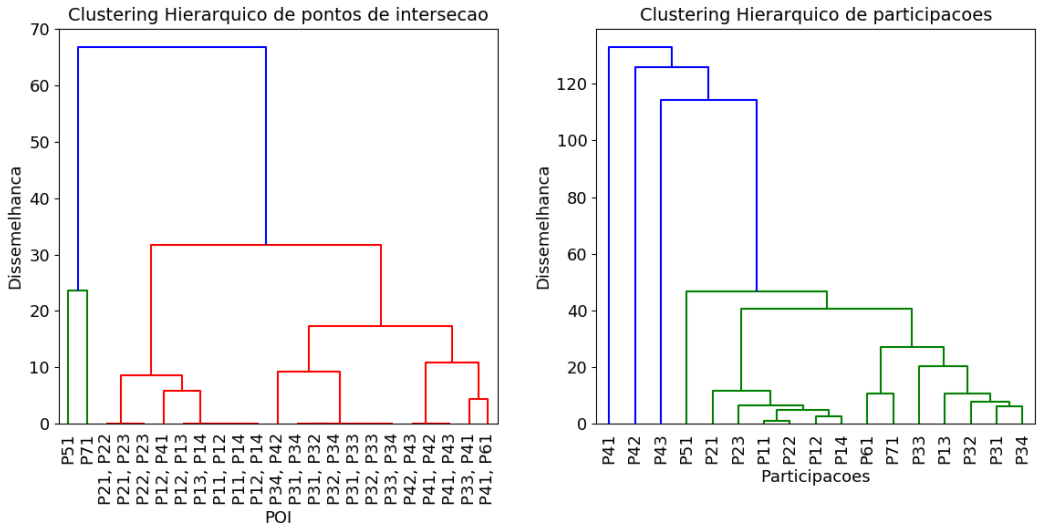


Figura 5.6: Pontos de interseção identificados no cenário 1

Análise do comportamento do algoritmo

A Figura 5.7 apresenta os resultados obtidos quando são aplicados o método de *clustering* hierárquico apenas à localização das participações e o método desenvolvido neste relatório aos pontos de interseção.

Apenas pela análise dos dendrogramas, verifica-se uma grande diferença na estrutura das hierarquias. O dendrograma da Figura 5.7a apresenta grupos bem definidos possibilitando a identificação correta dos *clusters*. Já o dendrograma da Figura 5.7b apresenta uma hierarquia que ilustra a representação real das participações, não possibilitando a identificação de *clusters* que estejam dispersos (como é o caso das participações P_{41}, P_{42}, P_{43}).



(a) De pontos de interseção

(b) De participações geoespaciais

Figura 5.7: Comparação dos métodos de *clustering* hierárquico

O dendrograma da Figura 5.7a representa o resultado obtido aplicando o método de *linkage* a todos os pontos de interseção, tomando como métrica a fórmula de *Haversine*, pois esta fornece a distância real entre duas coordenadas geográficas. Cada POI representa um ponto de interseção resultante da interseção das retas correspondentes a participações. Apenas pela análise desse dendrograma, verifica-se a existência de 3 pontos enganosos: o ponto de interseção $[P_{12}, P_{41}]$, o ponto de interseção $[P_{34}, P_{42}]$ e o ponto de interseção $[P_{33}, P_{41}]$.

Pela análise da Figura 5.8, verifica-se que a participação P_{12} foi associada a um *cluster* com um valor de dissemelhança inferior a 0.01 metros e a participação P_{41} foi associada a um outro *cluster* com um valor de dissemelhança inferior a 0.07. Portanto, a um valor de dissemelhança próximo de 10 metros, a fusão do ponto de interseção $[P_{12}, P_{41}]$ com um *cluster*, não pode acontecer pois as participações associadas ao ponto de interseção $[P_{12}, P_{41}]$ já estão associadas a *clusters* distintos, tornando este ponto um ponto enganoso. O mesmo acontece com os outros pontos enganosos identificados.

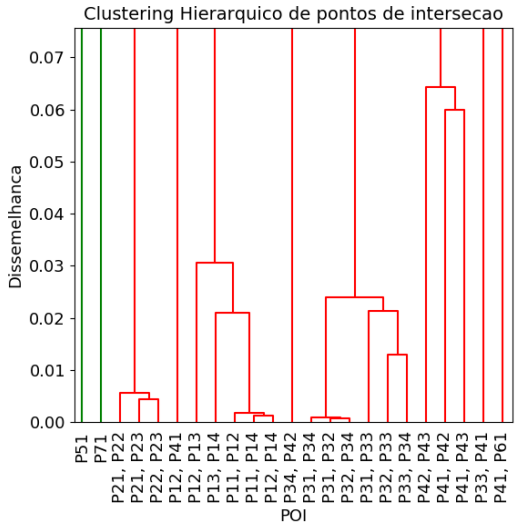


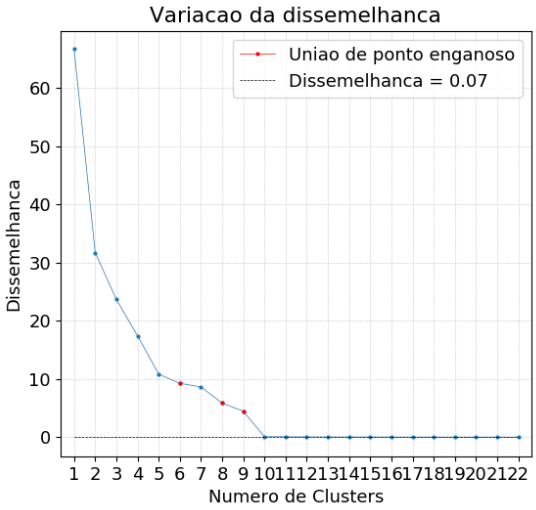
Figura 5.8: Visão detalhada do dendrograma da Figura 5.7a

Tendo em conta que os pontos de interseção de retas que representam um mesmo evento estão geralmente muito próximas, espera-se que o limite para a dissemelhança seja um valor baixo. Analisando as Figuras 5.7a e 5.8 verifica-se uma grande variação no nível de fusão a partir de 0.07 metros.

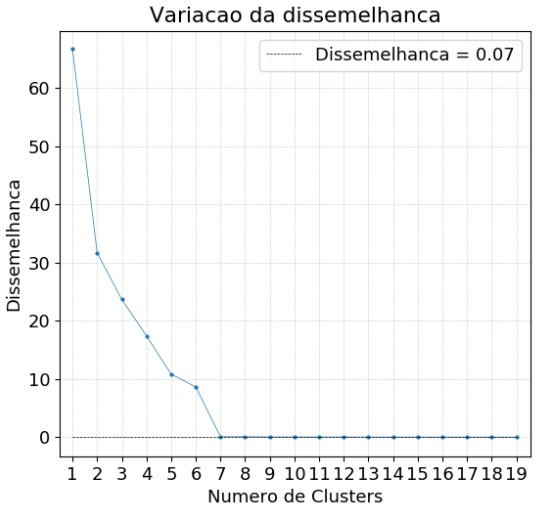
A variação do valor de dissemelhança pode ser analisada em mais pormenor a partir da Figura 5.9. Assim como se verifica no dendrograma, o valor da dissemelhança decresce rapidamente até, aproximadamente, 0.0642 metros que corresponde a um número de *clusters* de pontos de interseção igual a 10 ou 7, no caso de se eliminar os *clusters* formados apenas por pontos enganosos. Visualmente, esse é o valor onde a curva que representa a variação teria a curvatura máxima.

Como se esperava, verifica-se que a fusão de pontos enganosos ocorre num valor de dissemelhança superior a 0.07 metros. Ou seja, a fusão de pontos enganosos acontece mais tarde na hierarquia, onde se verifica uma maior variação do valor de dissemelhança conforme se varia o número de *clusters*.

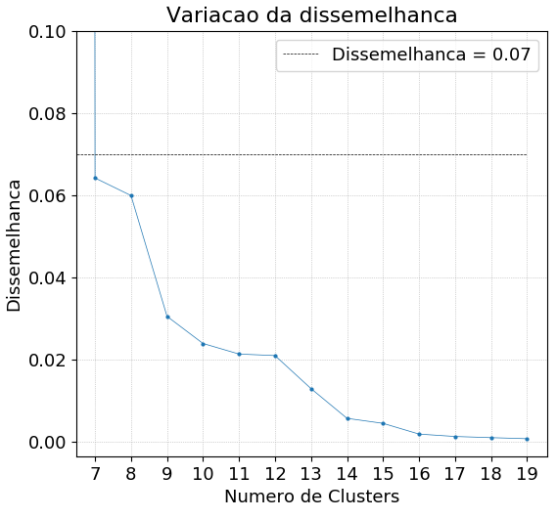
Além disso, verifica-se que ao remover as uniões de pontos enganosos (Figura 5.9b) a variação da dissemelhança entre os *clusters* é maior, para o intervalo de 1 a 7 *clusters*.



(a) Visão geral



(b) Visão geral, removendo a união de pontos enganosos



(c) Visão detalhada

Figura 5.9: Variação do valor da dissemelhança em função do número de *clusters* de pontos de interseção

Escolha do número de *clusters* e validação

Os índices de validação interna de *clusters* referidos na secção 4.3 permitem identificar o número ideal de *clusters* e validar o número de *clusters* identificados intuitivamente. Na secção anterior, o gráfico da Figura 5.9 indica, intuitivamente, o número "ideal" de *clusters* de pontos de interseção como sendo 7.

A Figura 5.10 representa a variação do valor do índice *Dunn* em função do número de *clusters*. O valor máximo obtido para este indicador corresponde a 7 *clusters* (de pontos de interseção e de participações), o que apoia a hipótese

definida inicialmente.

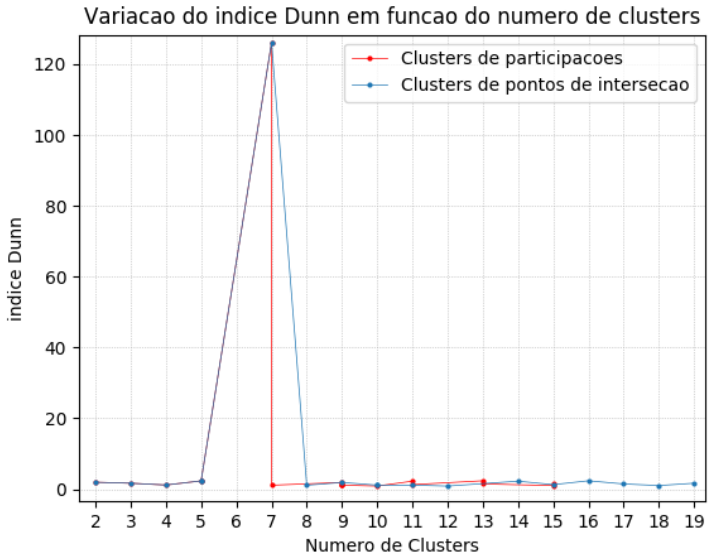


Figura 5.10: Variação do índice *Dunn* do cenário 1

Verifica-se também que o valor do índice sofre uma grande alteração nesse valor, o que mostra que uma pequena alteração no número de *clusters* provoca uma grande alteração nos níveis de coesão e separação dos *clusters*.

A Figura 5.11 representa a variação do valor do índice *Silhouette* em função do número de *clusters*.

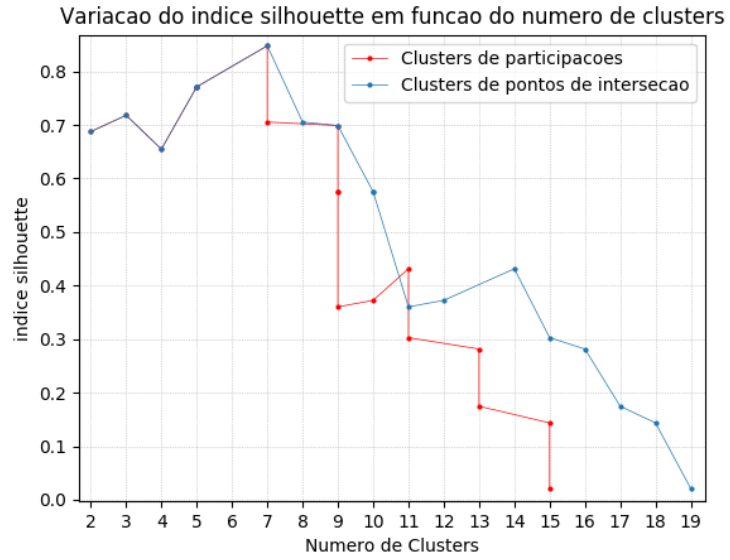


Figura 5.11: Variação do índice *Silhouette* do cenário 1

Assim como no índice *Dunn*, o valor máximo obtido para este indicador corresponde a 7 *clusters*.

A Figura 5.12 representa a variação do valor do índice *Calinski-Harabasz* em função do número de *clusters*. Verifica-se que, em geral, conforme o número de *clusters* aumenta, o valor do índice também aumenta.

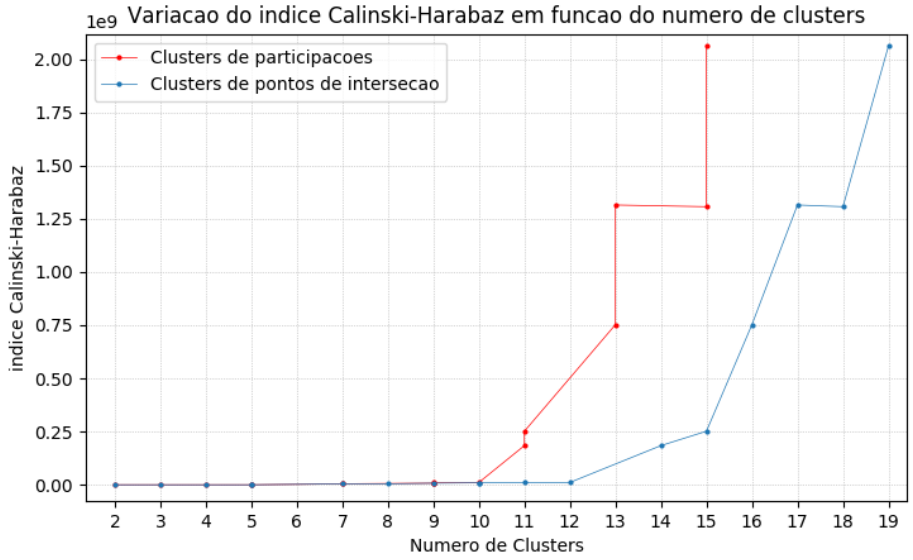


Figura 5.12: Variação do índice *Calinski-Harabasz* do cenário 1

Na Tabela 5.1 estão os valores do índice obtidos para o intervalo de 2 a 8 *clusters* de pontos de interseção. Nesse intervalo já se verifica a existência de um máximo local para o número de *clusters* igual a três.

2 *clusters*	3 *clusters*	4 *clusters*	5 *clusters*	6 *clusters*	7 *clusters*	8 *clusters*
14.6	**70.6**	62.0	227.3	480.2	4296209.2	4625742.2

Tabela 5.1: Resultados do índice de *Calinski-Harabasz* do cenário 1

Pode-se concluir que, em geral, os índices apontam para o número de *clusters* igual a sete, apoiando a hipótese identificada intuitivamente.

Utilizando o valor de dissemelhança correspondente a sete *clusters*, identificam-se os seguintes *clusters* de participações:

- O *cluster* definido pelas participações P_{21}, P_{22} e P_{23};
- O *cluster* definido pelas participações P_{11}, P_{12}, P_{13} e P_{14};
- O *cluster* definido pelas participações P_{31}, P_{32}, P_{33} e P_{34};
- O *cluster* definido pelas participações P_{41}, P_{42} e P_{43}.
- Os *clusters* definidos apenas pelas participações P_{51}, P_{61} e P_{71}.

Ou seja, o método conseguiu identificar corretamente todos os *clusters*.

O índice de Fowlkes-Mallows permite comparar as soluções obtidas pelo método de *clustering*. Como se sabe os verdadeiros *clusters* de participações, é possível calcular a precisão dos resultados obtidos.

Na Figura 5.13 está representado os valores do índice Fowlkes-Mallows para vários números de *clusters*. Além disso, é possível comparar as soluções obtidas para o método desenvolvido neste relatório e o método de *clustering* hierárquico com as coordenadas das participações.

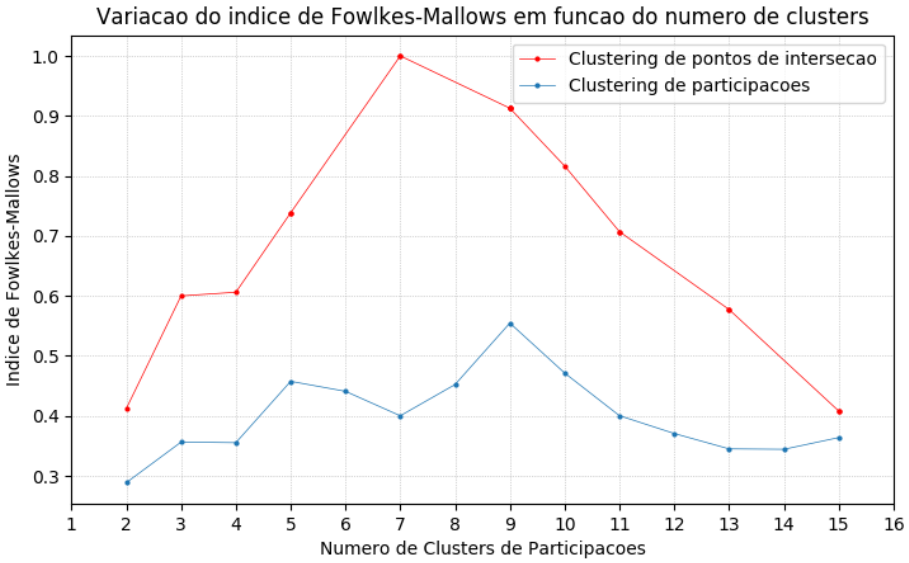


Figura 5.13: Variação do índice de Fowlkes-Mallows do cenário 1

Verifica-se que, para todos os resultados com os diferentes números de *clusters*, os resultados obtidos utilizando o método desenvolvido neste relatório são mais satisfatórios do que os obtidos com o método de *clustering* hierárquico a participações, produzindo resultados mais precisos.

Verifica-se também que os índices *Dunn* e *Silhouette* permitem retirar conclusões mais informativas do que o índice *Calinski-Harabasz*. Pode-se dizer que esses índices são mais fiáveis.

5.7.2 Cenário 5

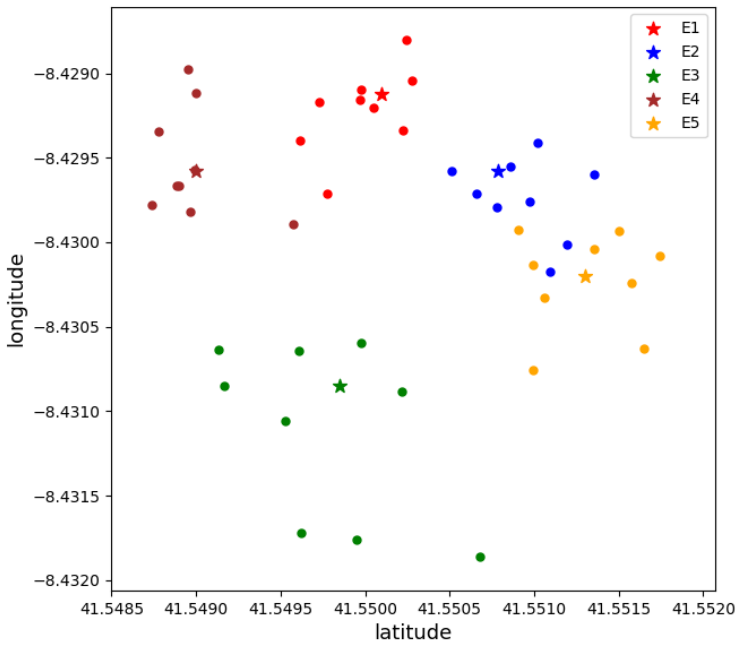


Figura 5.14: Representação gráfica do cenário 5

Este cenário, representado na Figura 5.14, consiste em cinco eventos, E_1, E_2, E_3, E_4 e E_5, onde cada um destes está associado a 10 participações. Dessas participações, 32% estão a mais de 50 metros de distância do objeto fotografado, 34% estão entre 30 a 50 metros, 28% estão entre 10 e 30 metros e, por fim, 6% estão entre 0 e 10 metros.

O objetivo é comparar os resultados obtidos através do método desenvolvido neste relatório com os resultados descritos em [10].

Ao aplicar a estes dados o algoritmo de processamento dos dados, descrito na Figura 5.3, foram identificados 276 pontos de interseção, sendo que alguns desses estão posicionados geograficamente em locais afastados da zona das participações. Isto deve-se à alta probabilidade de duas semirretas se intersectarem, formando possivelmente pontos enganosos. Uma forma de contornar este problema é identificar uma distância limite para as participações fotografadas a mais de 50 metros, de forma a reduzir o número de interseções enganosas.

Para este caso, será definido que uma participação que está a mais de 50 metros do evento, estará no intervalo de 50 a 200 metros. Portanto, para este cenário só serão considerados segmentos de reta. Utilizando este limite e aplicando o algoritmo da Figura 5.3, obteve-se 241 pontos de interseção (Figura 5.15).

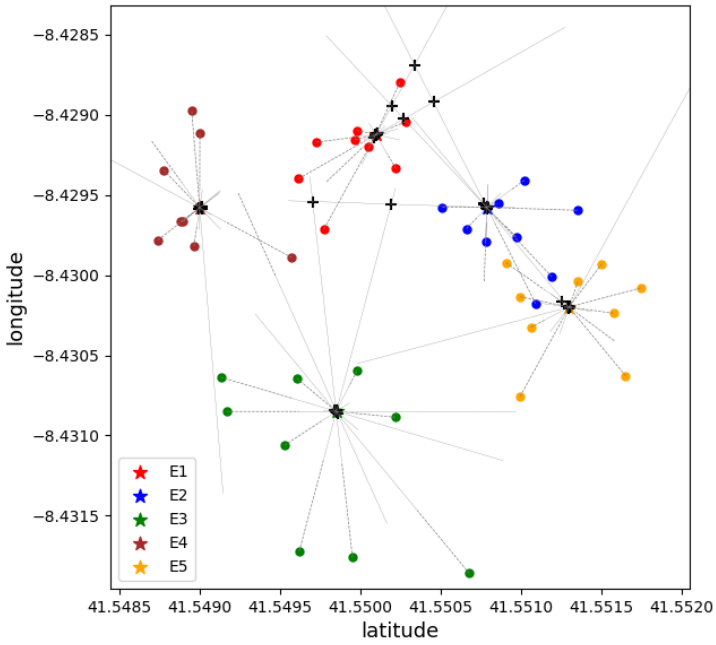
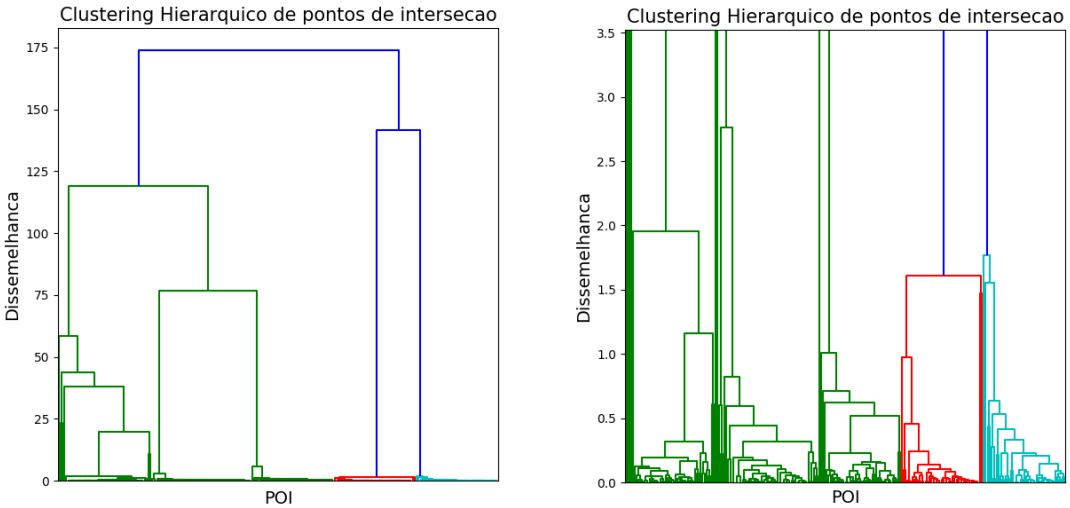


Figura 5.15: Pontos de interseção identificados no cenário 5

Análise do comportamento do algoritmo

A Figura 5.16 representa o resultado obtido aplicando o método *linkage* a todos os pontos de interseção e tomando a métrica de *Haversine*. Pela análise visual do dendrograma, verifica-se a existência de cinco grupos bem definidos.

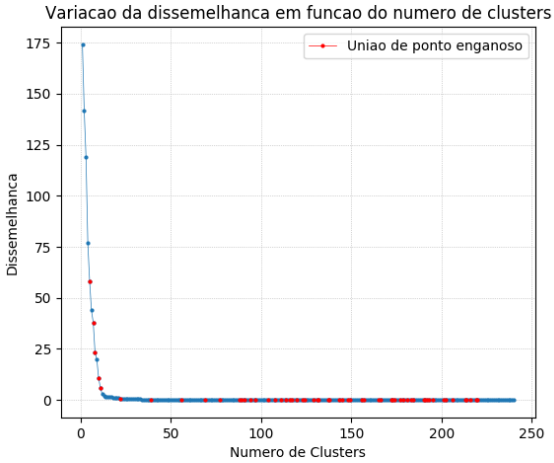


(a) Visão geral (b) Visão detalhada

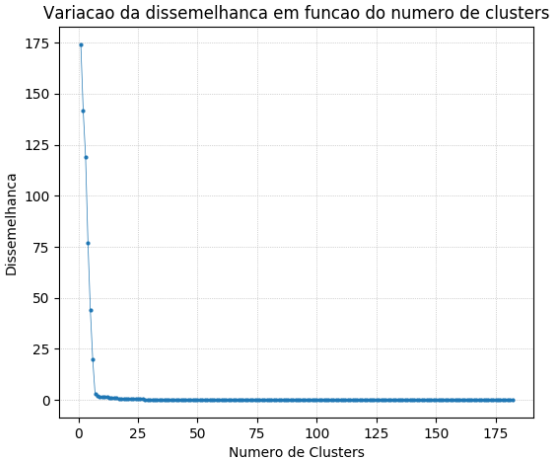
Figura 5.16: Dendrograma dos pontos de interseção do cenário 5

Ao aplicar o método de *clustering* descrito na secção 5.5 para toda a hierarquia foram identificados 73 pontos enganosos.

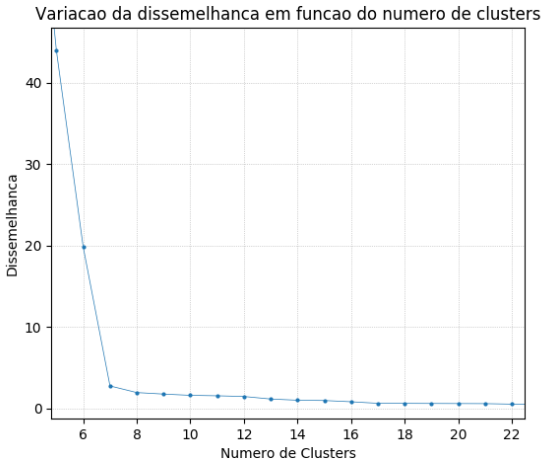
Na Figura 5.17 está representada a variação da dissemelhança e estão identificados as fusões que contêm pontos enganosos. O valor da dissemelhança decresce rapidamente até a um número de *clusters* de pontos de interseção igual a 7, considerando que se eliminou os *clusters* formados apenas por pontos enganosos.



(a) Visão geral



(b) Visão geral, sem a união de pontos enganosos



(c) Visão detalhada

Figura 5.17: Variação do valor da dissemelhança em função do número de *clusters* de pontos de interseção

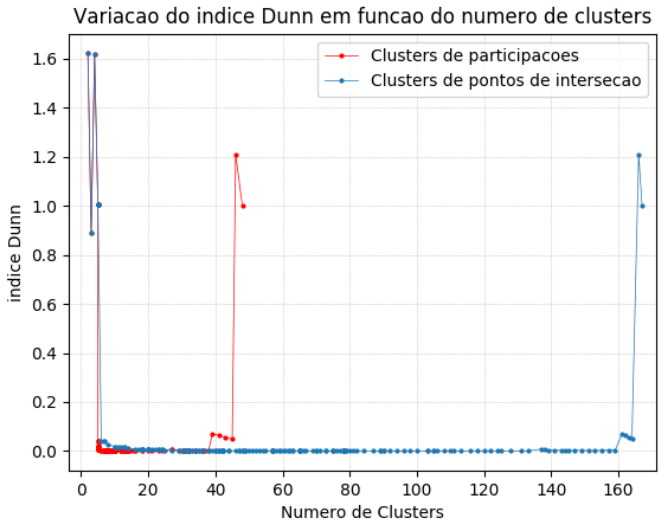
Ao contrário do que aconteceu no cenário anterior, a maior parte das fusões de pontos enganosos ocorre para um número de *clusters* elevado. Ou seja, ocorrem no início da hierarquia.

Escolha do número de *clusters* e validação

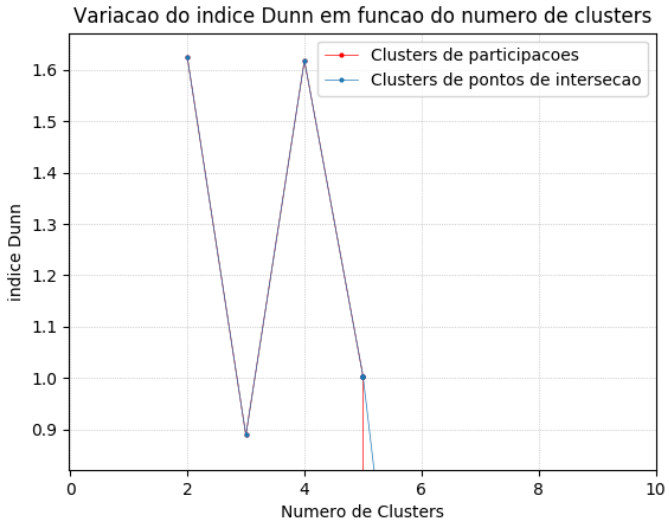
Anteriormente, identificou-se intuitivamente o número "ideal" de *clusters* de pontos de interseção como sendo 7. Assim, será analisado o intervalo entre

2 e 10 *clusters*. De seguida serão apresentados os resultados obtidos aplicando os diferentes índices de validação interna de *clusters*.

A Figura 5.18 representa a variação do valor do índice *Dunn* em função do número de *clusters*. O valor máximo obtido para este indicador corresponde a 2 *clusters* (de pontos de interseção e de participações), seguido do índice obtido para 4 *clusters* (de pontos de interseção e de participações).



(a) Visão geral



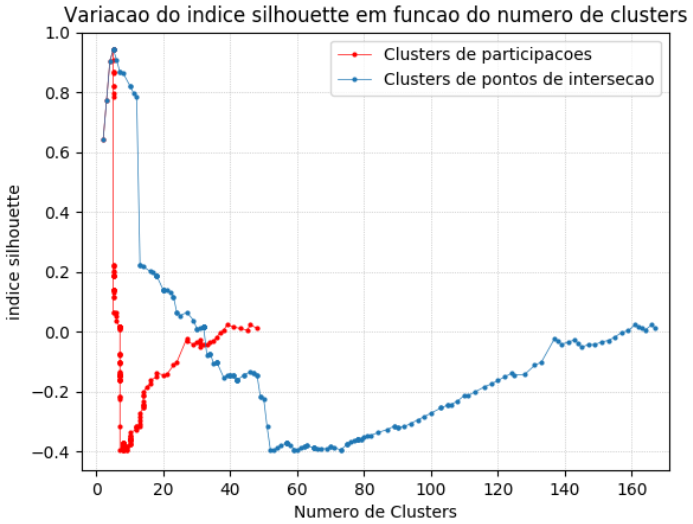
(b) Visão detalhada

Figura 5.18: Variação do índice *Dunn* do cenário 5

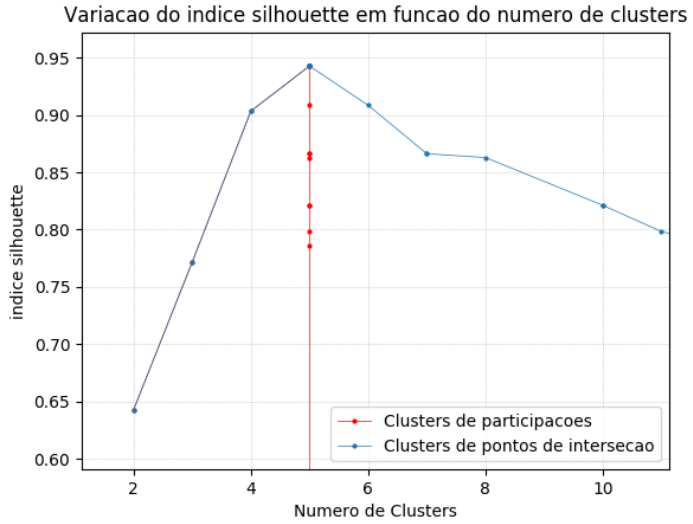
Dado o comportamento do índice entre 2 a 5 *clusters* de pontos de interseção, será mais adequado escolher o número 4 como sendo o número "ideal" de *clusters* de pontos de interseção.

A Figura 5.19 representa a variação do valor do índice *Silhouette* em função do número de *clusters*. O valor máximo obtido para este indicador corresponde

a 5 *clusters* (de pontos de interseção e de participações). Este valor corresponde, de facto, ao número correto de *clusters* presente no cenário.



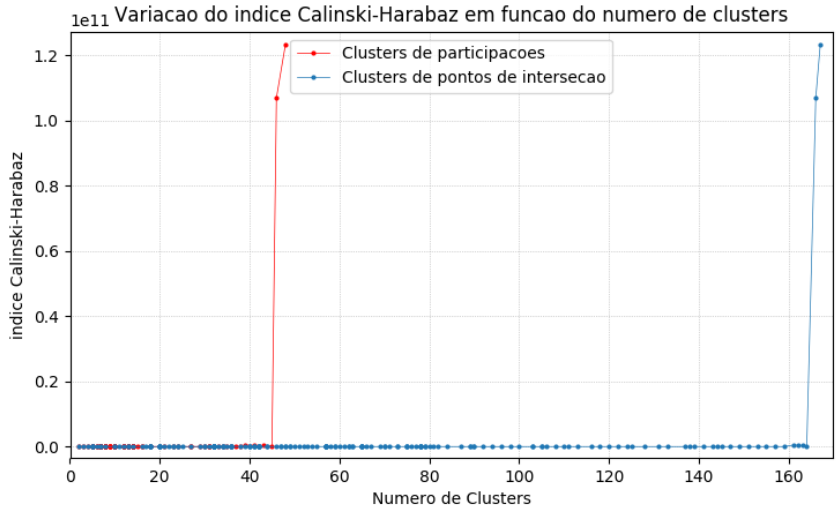
(a) Visão geral



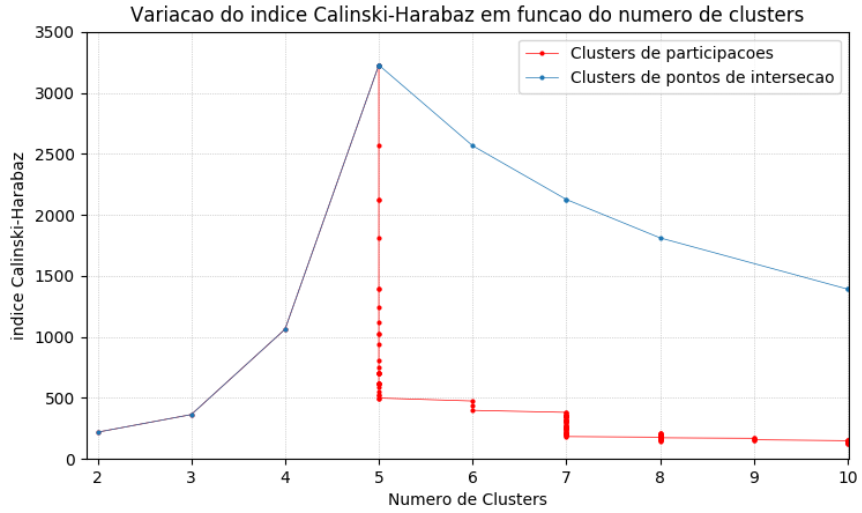
(b) Visão detalhada

Figura 5.19: Variação do índice *Silhouette* do cenário 5

A Figura 5.20 representa a variação do valor do índice *Calinski-Harabasz* em função do número de *clusters*. Verifica-se que, existe um máximo local em 5 *clusters*, o mesmo resultado obtido no índice *Silhouette*.



(a) Visão geral



(b) Visão detalhada

Figura 5.20: Variação do índice *Calinski-Harabasz* do cenário 5

Utilizando o valor de dissemelhança correspondente a 5 *clusters* como sendo o valor de dissemelhança máximo, obtiveram-se os seguintes resultados:

- O *cluster* constituído pelas participações associadas ao evento E_1;
- O *cluster* constituído pelas participações associadas ao evento E_2 e as participações P_{52},P_{55} e P_{510};
- O *cluster*constituído pelas participações associadas ao evento E_3;
- O *cluster* constituído pelas participações associadas ao evento E_4;
- O *cluster* constituído pelas participações associadas ao evento E_5, exceto as participações P_{52},P_{55} e P_{510}.

Para 4 *clusters* de participações também se obteve os *clusters* associados aos eventos E_1, E_3 e E_4, e houve a fusão dos *clusters* associados aos eventos E_2 e E_5.

Na Tabela 5.2 estão apresentados os valores obtidos para o índice de Fowlkes-Mallows para o intervalo de 2 a 5 *clusters* de participações, para ambos os métodos de *clustering*. Concluí-se que o método desenvolvido neste relatório obteve, em geral, melhores resultados do que aqueles obtidos em [10].

Além disso, verifica-se que o índice obtido para 5 *clusters* é o mais alto, o que significa uma boa precisão do resultado. Portanto, o índice indica o número 5 como sendo o número de *clusters* "ideal".

Clusters	**Índice Fowlkes-Mallows**	
	***Clustering* de Participações**	***Clustering* de POI's**
2	0.492	0.6
3	0.690	0.655
4	0.645	0.832
5	0.718	0.889

Tabela 5.2: Resultados do índice de Fowlkes-Mallows do cenário 5

Como três dos indicadores apontam para 5 *clusters*, identifica-se este como sendo o número "ideal" de *clusters*, contrariando a hipótese inicial de 7 *clusters*.

Em relação aos indicadores, pode-se concluir que os índices *Silhouette* e *Calinski-Harabasz* foram mais fiáveis do que o índice *Dunn*, uma vez que este último não conseguiu apresentar resultados esperados.

6 | Conclusão

Durante o período de estágio foram desenvolvidos dois trabalhos distintos: aprendizagem de ferramentas de desenvolvimento *web* e modelação com técnicas de *clustering* de participações geoespaciais.

Nos primeiros três meses, foi feita a aprendizagem das ferramentas com o objetivo de apresentar os resultados do tema inicial numa *dashboard*. Apesar disso não ter acontecido e do tema inicial ter sido alterado, esta parte do estágio foi muito importante. Por um lado, houve a oportunidade de estagiar numa empresa da área de informática e perceber de que forma as pequenas empresas abordam e implementam a análise de dados e, mais especificamente, técnicas de *machine learning*. Por outro lado, permitiu a aprendizagem de ferramentas transversais a várias áreas e que são uma mais valia dada a sua importância nos dias de hoje.

Nos meses seguintes foi desenvolvido o novo tema de estágio. O objetivo passava por identificar participações semelhantes, no contexto da aplicação *JuntarAJunta*, fazendo uso de novas variáveis, para além da geolocalização. Esta abordagem surgiu como uma forma de colmatar alguns problemas encontrados numa primeira fase do estágio, como por exemplo, a dificuldade em diferenciar participações muito próximas geograficamente mas que na realidade não reportam o mesmo evento. Este tipo de problemas deve-se à natureza dos dados e há possibilidade de se fotografar um evento de diferentes ângulos, posições e distâncias.

Para tal foi desenvolvido um modelo de processamento dos dados, a partir do método de *compass clustering*, e uma versão do método de *clustering* hierárquico aglomerativo adaptado ao contexto do problema e ao tipo de dados. O modelo de processamento de dados permite transformar a informação acerca de cada participação em novos dados que refletem a relação entre as diferentes participações, produzindo assim, dados mais relevantes para a resolução do problema proposto. O método de *clustering* hierárquico foi adaptado de forma a permitir o uso dos novos dados, descartar informação enganosa e transformar os *clusters* obtidos em possíveis grupos de participações semelhantes.

Concluiu-se que o método proposto neste relatório permite identificar de forma mais correta grupos de participações semelhantes, comparativamente ao método proposto em [10].

Além disso, verificou-se a importância de informação completar principalmente na presença de cenários anisotrópicos, onde as participações relativas a um mesmo evento não estão distribuídas apenas numa zona com maior frequência de participações. Neste tipo de cenários também se obteve resultados melhores em relação aos resultados obtidos usando o método de *clustering* hierárquico apenas da geolocalização.

Um dos problemas encontrados durante o período de estágio foi a falta de dados na base de dados da aplicação *JuntarAJunta*, o que implicou o uso apenas das variáveis relativas às coordenadas geográficas, impossibilitando o uso da informação relativa à fotografia e ao comentário. A introdução das novas variáveis também foi uma forma de substituir tal informação.

Em relação ao método desenvolvido, verificou-se que a escolha dos intervalos influencia a qualidade dos resultados. Por exemplo, com o aumento do número de participações a uma distância do evento superior a 50 metros, maior é o número de interseções enganosas. Portanto, um intervalo sem limite, como foi considerado no cenário 1, não é adequado. Assim, uma análise do tipo de eventos reportados e da frequência do uso de cada intervalo é importante para o ajustamento desses, de forma a ser possível identificar intervalos mais adequados e, consequentemente, resultados mais precisos.

Em relação aos indicadores de validação utilizados, concluiu-se que o índice *Silhouette* foi o que produziu resultados mais informativos em ambos os cenários e, por isso, é o mais fiável. Verificou-se que o índice Calinski-Harabasz apresenta melhores resultados quando se está perante *clusters* com o mesmo número de elementos.

Como trabalho futuro, seria importante otimizar o método proposto, por exemplo, no sentido de se fazer um *clustering* prévio dos dados para reduzir o número de interseções enganosas. Além disso, uma análise aos intervalos de distâncias também seria importante para esse problema. Por exemplo, uma análise aos intervalos de distância de participações relativas a diferentes categorias. Isto porque, há certas categorias onde é mais provável ser reportado um evento a uma menor distância deste, do que em outras categorias. A análise das descrições de cada participação também seria importante para extrair conteúdo relevante do texto. Ao combinar os métodos desenvolvidos neste relatório e métodos de processamento de texto, aumentaria a informação útil de cada participação e consequentemente, poderia produzir resultados mais

precisos.

Depois de realizadas as otimizações, o método ficaria nas condições de ser implementado e integrado na aplicação, ao nível da gestão dos dados. Futuramente, seria interessante a implementação de uma nova versão que conseguisse interagir com o utilizador. Por exemplo, uma versão que sugerisse ao utilizador as participações semelhantes à que o está a submeter. Assim, se de facto existir uma participação igual à do utilizador, este pode apoiar uma já existente de forma a que este tenha mais visibilidade e credibilidade.

Uma vez que os dados e variáveis utilizadas neste trabalho são comuns a muitas outras aplicações móveis e que a informação geoespacial é cada vez mais importante, o método desenvolvido neste relatório também poderia ser adaptado para esse tipo de aplicações móveis.

Bibliografia

[1] Calculate distance, bearing and more between latitude/longitude points. http://www.movable-type.co.uk/scripts/latlong.html. Acessado a 23/10/2017.

[2] Cluster validation statistics: Must know methods. http://www.sthda.com/english/articles/29-cluster-validation-essentials/97-cluster-validation-statistics-must-know-methods/, 2017. Acessado a 23/10/2017.

[3] E. Alpaydin. Introduction to machine learning. MIT press, 2010.

[4] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. Pattern Recognition, 46(1):243–256, 2013.

[5] H. Chen, R. H. Chiang, and V. C. Storey. Business intelligence and analytics: From big data to big impact. MIS quarterly, 36(4), 2012.

[6] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. Cluster analysis. Wiley Online Library, 5th edition, 2011.

[7] Y. A. Lacerda, R. G. F. Feitosa, G. A. R. M. Esmeraldo, C. d. S. Baptista, and L. B. Marinho. Compass clustering: A new clustering method for detection of points of interest using personal collections of georeferenced and oriented photographs. In Proceedings of the 18th Brazilian Symposium on Multimedia and the Web, WebMedia '12, pages 281–288, New York, NY, USA, 2012. ACM.

[8] D. D. R. Meneghetti. dunn-sklearn. https://gist.github.com/douglasrizzo/cd7e792ff3a2dcaf27f6, 2016.

[9] G. W. Milligan. Clustering validation: results and implications for applied analyses. In Clustering and classification, pages 341–375. World Scientific, 1996.

[10] M. Soares. *Clustering* hierárquico numa plataforma de *Smart Cities*. Relatório de Estágio (Submetido), Universidade do Minho, 2017.

[11] S. Theodoridis and K. Koutroumbas. Pattern Recognition. Elsevier, 2nd edition, 2003.

A | Anexo

A.1 Função de processamento dos dados

intersectionPoints.py

```
# -*- coding: utf-8 -*-
from matplotlib import pyplot as plt
import numpy as np
from matplotlib import style
from scipy.cluster.hierarchy import dendrogram, linkage,
fcluster,fclusterdata
from geographiclib.constants import Constants
from geographiclib.geodesic import Geodesic
import gpxpy.geo

#--- Dado um ponto (latitude e longitude), uma orientação e uma distancia,
calcula o ponto final ---
def getEndpoint(lat1, lon1, bearing, d):
    geod = Geodesic(Constants.WGS84_a, Constants.WGS84_f)
    d = geod.Direct(lat1, lon1, bearing, d )
    return [d['lat2'], d['lon2']]

#--- Verifica se duas retas se intersetam, nesse caso devolve o ponto de
interseção ---
def intersection(r1_i, r1_f, m1, b1, semi1, r2_i, r2_f, m2, b2, semi2):
    #Input:
    #   - r1_i / r2_i : ponto inicial da reta 1/ reta 2
    #   - r1_f / r2_f : ponto final da reta 1 / reta 2
    #   - m1 / m2 : declive da reta 1 / reta 2
    #   - b1 /b2 : ordenada na origem da reta 1 / reta 2
    #   - semi1 / semi2 : 0 ou 1 consoante é um segmento ou uma semi reta
    if m1 == m2: #Retas paralelas
        return False
    else: #As retas intersectam-se
        I = float(b2-b1)/(m1-m2)
        intersectionPoint = [I, I*m1+b1]
        # interseção de dois segmentos de reta
        if not semi1 and not semi2:
            if I >= max(min(r1_i, r1_f),min(r2_i,r2_f)) and I <=
            min(max(r1_i,r1_f), max(r2_i,r2_f)):
                return intersectionPoint
            else:
                return False

        # interseção de duas semirretas
        elif semi1: #r1 é semirreta
```

```
38              if semi2: # as duas são semirretas
39                  if r1_i<=r1_f and r2_i<=r2_f:
40                      if I>=max(r1_i,r2_i):
41                          return intersectionPoint
42                      else:
43                          return False
44                  if r1_i>=r1_f and r2_i>=r2_f:
45                      if I<=min(r1_i,r2_i):
46                          return intersectionPoint
47                      else:
48                          return False
49                  else:
50                      if I>=min(r1_i,r2_i) and I<=max(r1_i,r2_i):
51                          return intersectionPoint
52                      else:
53                          return False
54              else:  # r1 é semirreta e r2 é segmento de reta
55                  if I>=min(r2_i,r2_f) and I<=max(r2_i,r2_f):
56                      if r1_i<= r1_f:
57                          if I>=r1_i:
58                              return intersectionPoint
59                          else:
60                              return False
61                      else:
62                          if I<=r1_i:
63                              return intersectionPoint
64                          else:
65                              return False
66                  else:
67                      return False
68
69          else:  # r2 é semirreta e r1 é segmento de reta
70              if I>=min(r1_i,r1_f) and I<=max(r1_i,r1_f):
71                  if r2_i<= r2_f:
72                      if I>=r2_i:
73                          return intersectionPoint
74                      else:
75                          return False
76                  else:
77                      if I<=r2_i:
78                          return intersectionPoint
79                      else:
80                          return False
81              else:
82                  return False
83
84  #--- Determina os pontos de interseção (intersecao), as retas que lhes dão
    origem (origem), os pontos intermédios (Perturbacoes_mov) e a matriz com a
    informação de todas as retas intersetadas (matriz_intersecao) ---
85  def intersectionPoints(Perturbacoes):
86      #Input:
87      #   - Perturbacoes: lista com os dados [lat, long, orientacao,
        distancia] onde a distancia é um intervalo
88
89      # r_i = [lat, long] é o ponto do inicio da reta
90      # r_m = [lat, long] é o ponto médio da reta
91      # r_f = [lat, long] é o ponto do fim da reta
92      r1_i = Perturbacoes[0]
```

```
93      d1 = r1_i[3] #intervalo distancia [dmin, dmax]
94      #Verificar se estamos perante o caso do  intervalo [0,5]
95      if d1[1] ==5:
96          r1_m = r1_i
97      else:
98          r1_m =getEndpoint(r1_i[0], r1_i[1], r1_i[2], d1[0])
99      r1_f = getEndpoint(r1_i[0], r1_i[1], r1_i[2], d1[1])
100     Perturbacoes_mov = [[r1_m,r1_f]]
101     # Determinar equação da reta
102     m1 = float(r1_m[1]-r1_f[1]) /(r1_m[0]-r1_f[0])
103     b1 = r1_m[1]-m1*r1_m[0]
104     equacao = [[m1,b1]]
105     nPerturbacoes= len(Perturbacoes)
106     origem = []
107     intersecao=[]
108     matriz_intersecao = np.zeros((nPerturbacoes, nPerturbacoes),
        dtype=np.int)
109
110     for i in range(nPerturbacoes):
111         r1_i = Perturbacoes[i]
112         r1_m = Perturbacoes_mov[i][0]
113         r1_f = Perturbacoes_mov[i][1]
114         b1 = equacao[i][1]
115         m1 =  equacao[i][0]
116         d1 = r1_i[3]
117         if int(d1[0])==50: #reta r1 é semirreta ?
118             semi1 = 1
119         else:
120             semi1 = 0
121         for j in range(i+1,nPerturbacoes):
122             r2_i = Perturbacoes[j]
123             d2 = r2_i[3]
124             if int(d2[0])==50: #reta r2 é semirreta ?
125                 semi2 = 1
126             else:
127                 semi2 = 0
128             if i==0:
129                 if d2[0]==0:
130                     r2_m = r2_i
131                 else:
132                     r2_m = getEndpoint(r2_i[0], r2_i[1], r2_i[2], d2[0])
133                 r2_f = getEndpoint(r2_i[0], r2_i[1], r2_i[2], d2[1])
134                 m2 = float(r2_m[1]-r2_f[1]) /(r2_m[0]-r2_f[0])
135                 b2 = r2_m[1]-m2*r2_m[0]
136                 Perturbacoes_mov.append([r2_m,r2_f])
137                 equacao.append([m2,b2])
138             else:
139                 r2_m = Perturbacoes_mov[j][0]
140                 r2_f = Perturbacoes_mov[j][1]
141                 m2 = equacao[j][0]
142                 b2 = equacao[j][1]
143
144             existeIntersecao = intersection(r1_m[0], r1_f[0], m1, b1,
                semi1, r2_m[0], r2_f[0], m2, b2, semi2)
145             if existeIntersecao!=False:
146                 matriz_intersecao[i][j]=1
147                 intersecao.append(existeIntersecao)
148                 origem.append([i,j])
```

```
149
150         if not(1 in matriz_intersecao[i,:]) and not(1 in
            matriz_intersecao[:,i]):
151             if d1[0] == 50:
152                 intersecao.append(r1_m)
153                 origem.append([i])
154             else:
155                 d_media = d1[0] + float(d1[1]-d1[0])/2
156                 ponto_medio = getEndpoint(r1_i[0], r1_i[1], r1_i[2],
                    d_media)
157                 intersecao.append(ponto_medio)
158                 origem.append([i])
159     Perturbacoes_mov = np.array(Perturbacoes_mov)
160     return intersecao, origem, Perturbacoes_mov[:,1],Perturbacoes_mov[:,0],
        matriz_intersecao
161
162 def intersectionPoints_segmentos(Perturbacoes):
163     #Semelhante a intersectionPoints mas apenas lida com segmentos de reta
164     r1_i = Perturbacoes[0]
165     d1 = r1_i[3] #[dmin, dmax]
166     if r1_i[3][1] ==5:
167         r1_m = r1_i
168     else:
169         r1_m =getEndpoint(r1_i[0], r1_i[1], r1_i[2], d1[0])
170     r1_f = getEndpoint(r1_i[0], r1_i[1], r1_i[2], d1[1])
171     Perturbacoes_mov = [[r1_m,r1_f]]
172     m1 = float(r1_m[1]-r1_f[1]) /(r1_m[0]-r1_f[0])
173     b1 = r1_m[1]-m1*r1_m[0]
174     equacao = [[m1,b1]]
175     nPerturbacoes= len(Perturbacoes)
176     origem = []
177     intersecao=[]
178     matriz_intersecao = np.zeros((nPerturbacoes, nPerturbacoes),
        dtype=np.int)
179
180     for i in range(nPerturbacoes):
181         r1_i = Perturbacoes[i]
182         r1_m = Perturbacoes_mov[i][0]
183         r1_f = Perturbacoes_mov[i][1]
184         b1 = equacao[i][1]
185         m1 =  equacao[i][0]
186         d1 = r1_i[3]
187         semi1 =0
188         for j in range(i+1,nPerturbacoes):
189             r2_i = Perturbacoes[j]
190             d2 = r2_i[3]
191             semi2 = 0
192             if i==0:
193                 if d2[0]==0:
194                     r2_m = r2_i
195                 else:
196                     r2_m = getEndpoint(r2_i[0], r2_i[1], r2_i[2], d2[0])
197                 r2_f = getEndpoint(r2_i[0], r2_i[1], r2_i[2], d2[1])
198                 m2 = float(r2_m[1]-r2_f[1]) /(r2_m[0]-r2_f[0])
199                 b2 = r2_m[1]-m2*r2_m[0]
200                 Perturbacoes_mov.append([r2_m,r2_f])
201                 equacao.append([m2,b2])
202             else:
```

```
203                 r2_m = Perturbacoes_mov[j][0]
204                 r2_f = Perturbacoes_mov[j][1]
205                 m2 = equacao[j][0]
206                 b2 = equacao[j][1]
207             existeIntersecao = intersection(r1_m[0], r1_f[0], m1, b1,
                semi1, r2_m[0], r2_f[0], m2, b2, semi2)
208
209             if existeIntersecao!=False:
210                 matriz_intersecao[i][j]=1
211                 intersecao.append(existeIntersecao)
212                 origem.append([i,j])
213         if not(1 in matriz_intersecao[i,:]) and not(1 in
            matriz_intersecao[:,i]):
214             if d1[0] == 50:
215                 intersecao.append(r1_m)
216                 origem.append([i])
217             else:
218                 d_media = d1[0] + float(d1[1]-d1[0])/2
219                 ponto_medio = getEndpoint(r1_i[0], r1_i[1], r1_i[2],
                    d_media)
220                 intersecao.append(ponto_medio)
221                 origem.append([i])
222     Perturbacoes_mov = np.array(Perturbacoes_mov)
223     return intersecao, origem, Perturbacoes_mov[:,1],Perturbacoes_mov[:,0],
    matriz_intersecao
```

A.2 Função de identificação de *clusters* e de pontos enganosos

getClusters.py

```
1  # -*- coding: utf-8 -*-
2  import numpy as np
3  import math
4
5  # Função que identifica os clusters de participações, os clusters de pontos
   de interseção e os pontos enganosos
6  def getClusters(linkage, IntersecaoRetas, nPart, pontosIntersecao,
   distMaxima):
7    # linkage - matriz obtida usando o método linkage com pontos de
     interseção
8    # IntersecaoRetas - lista que contém as retas associadas a cada ponto de
     interseção
9    # nPart - número de participações
10   # pontosIntersecao - coordenadas dos pontos de interseção
11   # distMaxima - valor de dissemelhança máximo
12
13   # Numero de pontos de interseção
14   nIntersecoesRetas = len(IntersecaoRetas)
15   # Lista que contem a informação acerca do cluster que está a participação
     correspondente à posição i
16   clusters = np.zeros(nPart, dtype=np.int) + (-1)
17   # Lista que contem a informação acerca do cluster que está o ponto de
     intersecao correspondente à posição i
```

```
18    clusters_pI = np.zeros(nIntersecoesRetas, dtype=np.int) + (-1)
19    # Lista com as participações de cada cluster
20    clusters_participacoes=[]
21    # Lista com os pontos de interseção de cada cluster
22    clusters_pontosIntersecao =[]
23    # Lista com o número do cluster a que foi associado cada linha da matriz
      linkage
24    linkage_clusters = np.zeros(len(linkage), dtype=np.int) + (-1)
25    # Lista de pontos enganosos
26    pontos_enganosos=[]
27    # Nome do  cluster
28    novo_c = 0
29
30    i=0
31    while i<len(linkage) and linkage[i][2]<=distMaxima:
32      p1 = int(linkage[i][0])
33      p2 = int(linkage[i][1])
34
35      # União de pontos de interseção p1 e p2
36      if (p1 < nIntersecoesRetas) and (p2 < nIntersecoesRetas):
37        # As retas que dão origem ao ponto de interseção p1
38        participacoes_p1 = IntersecaoRetas[p1]
39        # As retas que dão origem ao ponto de interseção p2
40        participacoes_p2 = IntersecaoRetas[p2]
41        novoCluster = []
42        novoCluster_PI = []
43        pontos = [p1, p2]
44        lista_participacoes = [participacoes_p1,participacoes_p2]
45
46        # Analisar cada ponto de interseção para saber se as participações
          ficam ou nao no cluster
47        for j in range(0,2):
48          participacoes_p = lista_participacoes[j]
49          #o ponto de interseção é formado por duas retas
50          if len(participacoes_p) == 2:
51            # as retas associadas pertencem a dois clusters diferentes
52            if clusters[participacoes_p[0]] != -1 and
              clusters[participacoes_p[1]] != -1 and
              clusters[participacoes_p[0]] != clusters[participacoes_p[1]]:
53              #Ponto de interseção enganoso
54              pontos_enganosos.append(pontos[j])
55            # uma reta pertence ao cluster novo_c e a outra pertence ao
              cluster novo_c ou não está associada a nenhum
56            elif  (clusters[participacoes_p[0]] != -1 and
              clusters[participacoes_p[0]] != novo_c) and
              ((clusters[participacoes_p[1]] != -1 and
              clusters[participacoes_p[1]] != novo_c and
              clusters[participacoes_p[0]] == clusters[participacoes_p[1]]) or
              clusters[participacoes_p[1]] == -1):
57              for p in participacoes_p:
58                c = clusters[participacoes_p[0]]
59                clusters[p] = c
60                if p not in clusters_participacoes[c]:
61                  clusters_participacoes[c].append(p)
62              clusters_pontosIntersecao[c].append(pontos[j])
63              clusters_pI[pontos[j]] = c
64
```

```
65            elif (clusters[participacoes_p[1]] != -1 and
              clusters[participacoes_p[1]] != novo_c) and
              ((clusters[participacoes_p[0]] != -1 and
              clusters[participacoes_p[0]] != novo_c and
              clusters[participacoes_p[0]] == clusters[participacoes_p[1]]) or
              clusters[participacoes_p[0]] == -1):
66              for p in participacoes_p:
67                c = clusters[participacoes_p[1]]
68                clusters[p] = c
69                if p not in clusters_participacoes[c]:
70                  clusters_participacoes[c].append(p)
71              clusters_pontosIntersecao[c].append(pontos[j])
72              clusters_pI[pontos[j]] = c
73
74            else:
75              #o ponto de interseção é um ponto válido e poderá pertencer ao
                novo cluster
76              for p in participacoes_p:
77                clusters[p] = novo_c
78                if p not in novoCluster:
79                  novoCluster.append(p)
80              novoCluster_PI.append(pontos[j])
81              clusters_pI[pontos[j]] = novo_c
82
83          else:
84            if (clusters[participacoes_p[0]] != -1 and
              clusters[participacoes_p[0]] != novo_c):
85              #Ponto de interseção enganoso
86              pontos_enganosos.append(pontos[j])
87            else:
88              #o ponto de interseção é um ponto válido e poderá pertencer ao
                novo cluster
89              clusters[participacoes_p[0]] = novo_c
90              if participacoes_p[0] not in novoCluster:
91                novoCluster.append(participacoes_p[0])
92              novoCluster_PI.append(pontos[j])
93              clusters_pI[pontos[j]] = novo_c
94        linkage_clusters[i]=novo_c
95        clusters_participacoes.append(novoCluster)
96        clusters_pontosIntersecao.append(novoCluster_PI)
97        novo_c+=1
98
99      # União de dois clusters
100     elif (p1 > nIntersecoesRetas) and (p2 > nIntersecoesRetas):
101       p1_cluster = linkage_clusters[p1 - nIntersecoesRetas]
102       p2_cluster = linkage_clusters[p2 - nIntersecoesRetas]
103       if clusters_participacoes[p1_cluster] == []:
104         linkage_clusters[i]=p2_cluster
105       elif clusters_participacoes[p2_cluster] == []:
106         linkage_clusters[i]=p1_cluster
107       else:
108         # Novo cluster resultante de dois clusters ja existentes
109         linkage_clusters[i] = p1_cluster
110         #Associar todas as participacoes do cluster p2 ao cluster p1
111         for p in clusters_participacoes[p2_cluster]:
112           clusters[p] = p1_cluster
113         #Associar os pontos de interseção do cluster p2 ao cluster p1
114         for p in clusters_pontosIntersecao[p2_cluster]:
```

```
115             clusters_pI[p] = p1_cluster
116
117           #unir as participações de p2 na lista de participações de p1
118           clusters_participacoes[p1_cluster]+=clusters_participacoes[p2_cluster]
119           clusters_participacoes[p2_cluster] = []
120           #unir pontos de interseção de p2 na lista de pontos de interseção
              de p1
121           clusters_pontosIntersecao[p1_cluster]+=clusters_pontosIntersecao[p2_cluster]
122           clusters_pontosIntersecao[p2_cluster] = []
123
124       # União de uma interseção com um cluster ja existente
125       # o objetivo é verificar se as participações ficam no cluster ou nao
126       else:
127         # p1 é um cluster e p2 é uma interseçao
128         if p1 >= nIntersecoesRetas:
129           posicao = p1 - nIntersecoesRetas
130           #Obter participações da interseção
131           p_participacoes = IntersecaoRetas[p2]
132           # Obter o nome do cluster
133           p_cluster = linkage_clusters[posicao]
134           posicao_ponto = p2
135         elif p2 >= nIntersecoesRetas:
136           posicao = p2 - nIntersecoesRetas
137           p_participacoes = IntersecaoRetas[p1]
138           p_cluster = linkage_clusters[posicao]
139           posicao_ponto = p1
140         # Associar esta linha da matriz linkage ao cluster
141         linkage_clusters[i]=p_cluster
142
143         # Analisar cada participação para saber se fica ou não no cluster
144         if len(p_participacoes) == 2: #o ponto de interseção e formado por
            duas retas
145           # as retas associadas pertencem a dois clusters diferentes
146           if clusters[p_participacoes[0]] != -1 and
              clusters[p_participacoes[1]] != -1 and clusters[p_participacoes[0]]
              != clusters[p_participacoes[1]]:
147             #Ponto de interseção enganoso
148             pontos_enganosos.append(posicao_ponto)
149           #as duas retas pertencem ao mesmo cluster ou uma delas não está
              associada a nenhum cluster
150           elif  clusters[p_participacoes[0]]!=-1 and
              ((clusters[p_participacoes[1]]!=-1 and
              clusters[p_participacoes[0]]==clusters[p_participacoes[1]]) or
              clusters[p_participacoes[1]]==-1):
151             for p in p_participacoes:
152               c = clusters[p_participacoes[0]]
153               clusters[p] = clusters[p_participacoes[0]]
154               if p not in clusters_participacoes[c]:
155                 clusters_participacoes[c].append(p)
156             clusters_pontosIntersecao[c].append(posicao_ponto)
157             clusters_pI[posicao_ponto] = c
158
159           elif clusters[p_participacoes[1]]!=-1 and
              ((clusters[p_participacoes[0]]!=-1 and
              clusters[p_participacoes[0]]==clusters[p_participacoes[1]]) or
              clusters[p_participacoes[0]]==-1):
160             for p in p_participacoes:
161               c = clusters[p_participacoes[1]]
```

```
162             clusters[p] = clusters[c]
163             if p not in clusters_participacoes[c]:
164               clusters_participacoes[c].append(p)
165           clusters_pontosIntersecao[c].append(posicao_ponto)
166           clusters_pI[posicao_ponto] = c
167
168         else:
169           #o ponto de interseção é um ponto válido e poderá pertencer ao
              cluster p_cluster
170           if p_cluster != -1:
171             for p in p_participacoes:
172               clusters[p] = p_cluster
173               if p not in clusters_participacoes[p_cluster]:
174                 clusters_participacoes[p_cluster].append(p)
175           else:
176             novoCluster=[]
177             for p in p_participacoes:
178               clusters[p] = novo_c
179               novoCluster.append(p)
180             clusters_participacoes.append(novoCluster)
181             clusters_pontosIntersecao.append([posicao_ponto])
182             clusters_pI[posicao_ponto] = novo_c
183             novo_c +=1
184
185       else:
186         #o ponto de interseção é um ponto válido e poderá pertencer ao novo
            cluster
187         if p_cluster != -1:
188           clusters[p_participacoes[0]] = p_cluster
189           if p_participacoes[0] not in clusters_participacoes[p_cluster]:
190             clusters_participacoes[p_cluster].append(p_participacoes[0])
191           clusters_pI[posicao_ponto] = p_cluster
192           clusters_pontosIntersecao[p_cluster].append(posicao_ponto)
193         else:
194           novoCluster=[]
195           clusters[p_participacoes[0]] = novo_c
196           clusters_participacoes.append([p_participacoes[0]])
197           clusters_pontosIntersecao.append([posicao_ponto])
198           clusters_pI[posicao_ponto] = novo_c
199           novo_c +=1
200     i+=1
201
202   clusters_participacoes = [l for l in clusters_participacoes if l != []]
203   clusters_pontosIntersecao = [l for l in clusters_pontosIntersecao if l !=
      []]
204
205   for i in range(len(clusters)):
206     p = clusters[i]
207     if p == -1:
208       clusters_participacoes.append([i])
209       clusters[i] = novo_c
210       novo_c+=1
211
212   for i in range(len(clusters_pI)):
213     pI = clusters_pI[i]
214     if pI == -1:
215       clusters_pontosIntersecao.append([i])
216       clusters_pI[i] = novo_c
```

```
217         novo_c+=1
218     res = {'clusterP':clusters, 'list_clusterP':clusters_participacoes,
        'clusterPI':clusters_pI, 'list_clusterPI':clusters_pontosIntersecao,
        'PIenganosos':pontos_enganosos}
219     return res
```

A.3 Criação e Análise do cenário 1

cenario1.py

```
1   # -*- coding: utf-8 -*-
2   from matplotlib import pyplot as plt
3   import numpy as np
4   from scipy.cluster.hierarchy import dendrogram, linkage, cophenet, fcluster
5   from scipy.spatial.distance import pdist,squareform
6   from intersectionPoints import intersectionPoints
7   import gpxpy.geo
8   from getClusters import getClusters
9   from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
10  from mpl_toolkits.axes_grid1.inset_locator import mark_inset
11  from matplotlib.pyplot import figure, show
12  from matplotlib.ticker import MaxNLocator
13  from sklearn.metrics import silhouette_score, calinski_harabaz_score
14  from sklearn.metrics.cluster import fowlkes_mallows_score
15  from dunn_sklearn import dunn, min_cluster_distances, diameter
16
17  # -------- Definir eventos e perturbações --------
18  eventos = [[41.550097,-8.429125], [41.55004,-8.429191],
    [41.549966,-8.429569], [41.5501,-8.42941], [41.5505, -8.4296],
    [41.5501,-8.42951], [41.5518, -8.42913]]
19  Perturbacoes=[[41.550094,-8.429171,85.02,[0,10]],
    [41.550087,-8.429106,305.12,[0,10]], [41.550065,-8.42971,85.81,[30,50]],
    [41.550109,-8.429120,197.32,[0,10]], [41.549979,-8.429152,334.43,[0,10]],
    [41.550085,-8.429168,200.93,[0,10]], [41.550041,-8.429168,266.68,[0,10]],
    [41.550018,-8.429654,129.26,[0,10]], [41.549945,-8.429624,62.97,[0,10]],
    [41.549833,-8.429576,2.26,[10,30]], [41.549977,-8.429703,96.26,[10,30]],
    [41.549931,-8.430859,81.14,[50,500]],
    [41.551104,-8.429057,194.74,[50,500]],
    [41.549313,-8.428499,319.1,[50,500]], [41.5504,-8.42957,347.35,[10,30]],
    [41.55012,-8.42951,180,[0,10]], [41.5502,-8.42944,8.25,[50,300]]]
20  Perturbacoes_nome = ['P11', 'P12', 'P13', 'P14', 'P21', 'P22', 'P23',
    'P31', 'P32', 'P33', 'P34', 'P41', 'P42', 'P43', 'P51', 'P61', 'P71']
21  labels_true = [1 for i in range(0,4)] + [2 for i in range(0,3)] + [3 for i
    in range(0,4)] + [4 for i in range(0,3)] +[5, 6, 7]
22  # -------- Representacao grafica do cenario --------
23  print 'A apresentar a  representacao grafica do cenario ....'
24  fig, ax = plt.subplots() # create a new figure with a default 111 subplot
25  eventos = np.array(eventos)
26  colors = ['red', 'blue', 'green', 'pink', 'brown', 'orange', 'yellow']
27  labels = ['E1','E2','E3','E4','E5','E6','E7']
28  PertporEvento = [[0,4], [4,7], [7,11], [11,14], [14,15], [15,16], [16,17]]
29
30  for i in range(7):
31      ax.scatter(eventos[i,0], eventos[i,1],c=colors[i], marker ='*',
        label=labels[i], s=80)
```

```
32    lim_inf = PertporEvento[i][0]
33    lim_sup = PertporEvento[i][1]
34    ax.scatter([p[0] for p in Perturbacoes[lim_inf : lim_sup]], [p[1] for p
      in Perturbacoes[lim_inf : lim_sup]],c=colors[i],marker='o', s=25)
35
36  axins = zoomed_inset_axes(ax, 5, loc=3)
37  axins.scatter(eventos[0,0],eventos[0,1],c='red',marker='*', s=100)
38  axins.scatter(eventos[1,0],eventos[1,1],c='blue',marker='*', s=100)
39  axins.scatter([p[0] for p in Perturbacoes[0:4]], [p[1] for p in
    Perturbacoes[0:4]],c='red',marker='o', s=25)
40  axins.scatter([p[0] for p in Perturbacoes[4:7]], [p[1] for p in
    Perturbacoes[4:7]],c='blue',marker='o', s=25 )
41  axins.set_xlim(41.54995, 41.55015)
42  axins.set_ylim(-8.42925, -8.42905)
43  axins.xaxis.set_visible(False)
44  axins.yaxis.set_visible(False)
45  mark_inset(ax, axins, loc1=2, loc2=4, fc="none", ec="0.5")
46  ax.set_xlim(41.5485, 41.5522)
47  ax.set_ylim(-8.43115, -8.4282)
48  ax.legend()
49  ax.set_xlabel("latitude", fontsize=13)
50  ax.set_ylabel("longitude", fontsize=13)
51  plt.gca().set_aspect('equal', adjustable='box')
52  plt.rcParams.update({'font.size': 10})
53  plt.show()
54
55  # -------- Obter pontos de interseção --------
56  print '\n'
57  print 'A obter pontos de intersecao...'
58
59  pontosIntersecao, retasIntersetadas, pontosFinais,
    pontosMedios,matriz_intersecao = intersectionPoints(Perturbacoes)
60  # pontosIntersecao - coordenadas dos pontos de interseção
61  # retasIntersetadas - as correspondentes retas que formam os pontos de
    intersecao
62  # pontosFinais (apenas usado para a representacao grafica) - coordenadas do
    ponto final da reta
63  # pontosMedios (apenas usado para a representacao grafica) - coordenadas do
    ponto médio da reta
64  # matriz_intersecao - matriz que contem informacao acerca de todas as retas
    intersetadas
65
66  pontosIntersecao_nomes=[]
67  for pInt in retasIntersetadas:
68    if len(pInt) == 1:
69      pontosIntersecao_nomes.append(Perturbacoes_nome[pInt[0]])
70    else:
71      pontosIntersecao_nomes.append(Perturbacoes_nome[pInt[0]]+',
      '+Perturbacoes_nome[pInt[1]])
72
73  # -------- Representacao grafica das retas e intersecoes --------
74  print 'A apresentar a  representacao grafica dos pontos de intersecao ....'
75  for i in range(7):
76    plt.scatter(eventos[i,0], eventos[i,1],c=colors[i], marker ='*',
      label=labels[i], s=80)
77    lim_inf = PertporEvento[i][0]
78    lim_sup = PertporEvento[i][1]
```

```
79    plt.scatter([p[0] for p in Perturbacoes[lim_inf : lim_sup]], [p[1] for p
      in Perturbacoes[lim_inf : lim_sup]],c=colors[i],marker='o', s=25)
80
81  pontosIntersecao = np.array(pontosIntersecao)
82  plt.scatter(pontosIntersecao[:,0],
    pontosIntersecao[:,1],c='black',marker='+', s=50 )
83  for i in range(len(Perturbacoes)):
84    plt.plot([pontosMedios[i][0],pontosFinais[i][0]],
      [pontosMedios[i][1],pontosFinais[i][1]], c='grey', linewidth=0.2)
85    plt.plot([Perturbacoes[i][0], pontosMedios[i][0]], [Perturbacoes[i][1],
      pontosMedios[i][1]], c='grey', linestyle='--', linewidth=0.5)
86  plt.xlim(41.5485, 41.5522)
87  plt.ylim(-8.43115, -8.4282)
88  plt.rcParams.update({'font.size': 10})
89  plt.xlabel("latitude", fontsize=13)
90  plt.ylabel("longitude", fontsize=13)
91  plt.gca().set_aspect('equal', adjustable='box')
92  plt.legend()
93  plt.show()
94
95  print ' ----- Informacoes -----'
96  print 'Numero de participacoes: ' + str(len(Perturbacoes))
97  print 'Numero de pontos de intersecao : ' + str(len(pontosIntersecao))
98  print 'Medida de dissemelhanca : average, distancia de haversine'
99
100 # -------- Metodo de clustering --------
101 # Definicao da metrica
102 def distancia(u,v):
103     return gpxpy.geo.haversine_distance(u[0], u[1], v[0], v[1])
104
105 print '\n'
106 print 'A calcular a matriz linkage para o criterio average e distancia de
    haversine ...'
107
108 # Matriz linkage dos pontos de intersecao
109 Z_intersectionPoints = linkage(pontosIntersecao, 'average',
    metric=distancia)
110 # Matriz linkage das participacoes
111 Z_Points = linkage([p[0:2] for p in Perturbacoes] , 'average',
    metric=distancia)
112 #  Matriz de distancias
113 matrizDistancias_PI=pdist(pontosIntersecao, metric = distancia)
114 # Calculo do coeficiente cofenetico
115 c_intersectionPoints_average, coph_dists = cophenet(Z_intersectionPoints,
    matrizDistancias_PI)
116 print 'Coeficiente cofenetico: '+str(c_intersectionPoints_average)
117
118 # Dendrogramas
119 print 'A apresentar dendrogramas dos pontos de intersecao e das
    participacoes ....'
120 plt.subplot(1, 2, 1)
121 plt.title('Clustering Hierarquico de pontos de intersecao')
122 plt.xlabel('POI')
123 plt.ylabel('Dissemelhanca')
124 d_intersectionPoints = dendrogram(
125     Z_intersectionPoints,
126     leaf_rotation=90.,
127     leaf_font_size=12.,
```

```
128     labels=pontosIntersecao_nomes)
129
130 plt.subplot(1, 2, 2)
131 plt.title('Clustering Hierarquico de participacoes')
132 plt.xlabel('Participacoes')
133 plt.ylabel('Dissemelhanca')
134 d_Points = dendrogram(
135     Z_Points,
136     leaf_rotation=90.,
137     leaf_font_size=12.,
138     labels=Perturbacoes_nome)
139 plt.show()
140
141 # Aplicacao do algoritmo de identificacao de clusters
142 # 1) Aplicar o método a toda a hierarquia para identificar todos os pontos
    enganosos
143
144 resultado_identPI = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(Perturbacoes), pontosIntersecao, 70)
145 # dicionario:
146 # 'clusterP' - lista onde cada indice representa a participacao e o valor
    nesse indice indica o cluster a que esta associado
147 # 'list_clusterP' - lista onde cada indice representa um cluster e o valor
    nesse indice e uma lista com as participacoes do cluster
148 # 'clusterPI' - lista onde cada indice representa o ponto de intersecao e o
    valor nesse indice indica o cluster a que esta associado
149 # 'list_clusterPI' - lista onde cada indice representa um cluster e o valor
    nesse indice e uma lista com os pontos de intersecao do cluster
150 # 'PIenganosos' - lista de pontos enganosos
151 # info
152
153 pontos_enganosos = sorted(resultado_identPI['PIenganosos'], reverse=False)
154 print '\n'
155 print 'A identificar todos os pontos enganosos ....'
156 print 'Numero de pontos enganosos identificados:
    '+str(len(pontos_enganosos))
157
158 # Atualizar matriz das distancias
159 removidos = 0
160 matrizDistancias_PI_square = squareform(matrizDistancias_PI)
161 pontosIntersecao_sEnganosos = pontosIntersecao
162 for index in pontos_enganosos:
163   # apagar linha
164   matrizDistancias_PI_square = np.delete(matrizDistancias_PI_square,
    index-removidos,0)
165   # apagar coluna
166   matrizDistancias_PI_square =
    np.delete(matrizDistancias_PI_square,index-removidos,1)
167   pontosIntersecao_sEnganosos =
    np.delete(pontosIntersecao_sEnganosos,index-removidos,0)
168   removidos +=1
169
170 # 2) Analise da variacao da dissemelhanca para identificar um numero ideal
    de clusters
171 print '\n'
172 print 'A representar a variacao da dissemelhanca ...'
173 alturas_dendrograma_sEnganosos = []
174 alturas_dendrograma = []
```

```
175 uniao_enganosos = []
176 for index in range(len(Z_intersectionPoints)):
177   z = Z_intersectionPoints[index]
178   alturas_dendrograma = [z[2]]+ alturas_dendrograma
179   if z[0] not in pontos_enganosos and z[1] not in pontos_enganosos:
180     alturas_dendrograma_sEnganosos = [z[2]]+ alturas_dendrograma_sEnganosos
181   else:
182     uniao_enganosos.append(len(Z_intersectionPoints)-index-1)
183
184 plt.subplot(1, 2, 1)
185 x = range(1, len(alturas_dendrograma)+1)
186 plt.plot(x,alturas_dendrograma, marker='o', linewidth=0.5, markersize=2)
187 for i in range(len(uniao_enganosos)):
188   index = uniao_enganosos[i]
189   if i == 0:
190     plt.plot(x[index],alturas_dendrograma[index], marker='o',
      linewidth=0.5, markersize=2, c='red', label = 'Uniao de ponto
      enganoso')
191   else:
192     plt.plot(x[index],alturas_dendrograma[index], marker='o',
      linewidth=0.5, markersize=2, c='red')
193 plt.plot(x, np.array([0.07 for i in xrange(len(x))]), c='black',
    linewidth=0.5, linestyle = 'dashed', label='Dissemelhanca = 0.07' )
194 plt.title('Variacao da dissemelhanca')
195 plt.xlabel('Numero de Clusters')
196 plt.ylabel('Dissemelhanca')
197 plt.grid(linestyle=':', lw=0.5)
198 plt.legend()
199 plt.xticks(range(0, len(Z_intersectionPoints)+1))
200
201 plt.subplot(1, 2, 2)
202 x_sEnganosos = range(1, len(alturas_dendrograma_sEnganosos)+1)
203 plt.plot(x_sEnganosos,alturas_dendrograma_sEnganosos, marker='o',
    linewidth=0.5, markersize=2)
204 plt.plot(x_sEnganosos, np.array([0.07 for i in xrange(len(x_sEnganosos))]),
    c='black', linewidth=0.5, linestyle = 'dashed', label='Dissemelhanca =
    0.07' )
205
206 print 'A apresentar a Variacao da dissemelhanca em funcao do numero de
    clusters'
207 plt.title('Variacao da dissemelhanca')
208 plt.xlabel('Numero de Clusters')
209 plt.ylabel('Dissemelhanca')
210 plt.grid(linestyle=':', lw=0.5)
211 plt.legend()
212 plt.xticks(range(0, len(Z_intersectionPoints)+1))
213 plt.show()
214
215 # 3) Aplicar o metodo para identificar clusters e analisar resultados
216 lista_silhouette = []
217 lista_dunn = []
218 lista_CH = []
219 lista_fm_score = []
220 x_pI = []
221 x_p = []
222 for i in range(1,len(alturas_dendrograma_sEnganosos)):
223   d = alturas_dendrograma_sEnganosos[i]
```

```
224     r = getClusters(Z_intersectionPoints, retasIntersetadas,
        len(Perturbacoes), pontosIntersecao, d)
225     # Eliminar pontos enganosos
226     clusters_pontosIntersecao = r['clusterPI']
227     clusters_participacoes = r['clusterP']
228     removidos = 0
229     for index in pontos_enganosos:
230       clusters_pontosIntersecao = np.delete(clusters_pontosIntersecao,
        index-removidos)
231       removidos +=1
232     # numero de clusters identificados
233     x_p.append(len(set(clusters_participacoes)))
234     x_pI.append(len(set(clusters_pontosIntersecao)))
235     # Analise do coeficiente silhouette
236     indice_silhouette =
        silhouette_score(matrizDistancias_PI_square,clusters_pontosIntersecao,
        metric="precomputed")
237     lista_silhouette.append(indice_silhouette)
238     # Analise do indice dunn
239     indice_dunn = dunn(clusters_pontosIntersecao, matrizDistancias_PI_square)
240     lista_dunn.append(indice_dunn)
241     # Analise do indice Calinski-Harabaz
242     indice_CH = calinski_harabaz_score(pontosIntersecao_sEnganosos,
        clusters_pontosIntersecao)
243     lista_CH.append(indice_CH)
244     # fowlkes_mallows_score
245     fm_score = fowlkes_mallows_score(labels_true, clusters_participacoes)
246     lista_fm_score.append(fm_score)
247   #representação grafica dos indicadores
248   print 'A apresentar a variacao do indice silhouette ...'
249   plt.plot(x_p, lista_silhouette, marker='o', linewidth=0.5, markersize=2, c
      ='red' , label='Clusters de participacoes')
250   plt.plot( x_pI, lista_silhouette,marker='o', linewidth=0.5, markersize=2,
      label='Clusters de pontos de intersecao')
251   plt.title('Variacao do indice silhouette em funcao do numero de clusters')
252   plt.xlabel('Numero de Clusters')
253   plt.ylabel('indice silhouette')
254   plt.xticks(range(1,len(x)+2))
255   plt.grid(linestyle=':', lw=0.5)
256   plt.legend()
257   plt.show()
258
259   print 'A apresentar a variacao do indice Dunn ...'
260   plt.plot(x_p,lista_dunn, marker='o', linewidth=0.5, markersize=2, c ='red'
      , label='Clusters de participacoes')
261   plt.plot(x_pI, lista_dunn, marker='o', linewidth=0.5, markersize=2,
      label='Clusters de pontos de intersecao')
262   plt.title('Variacao do indice Dunn em funcao do numero de clusters')
263   plt.xlabel('Numero de Clusters')
264   plt.ylabel('indice Dunn')
265   plt.xticks(range(1,len(x)+2))
266   plt.legend()
267   plt.grid(linestyle=':', lw=0.5)
268   plt.show()
269
270   print 'A apresentar a variacao do indice Calinski-Harabaz ...'
271   plt.plot(x_p,lista_CH, marker='o', linewidth=0.5, markersize=2, c ='red' ,
      label='Clusters de participacoes')
```

```
plt.plot(x_pI, lista_CH, marker='o', linewidth=0.5, markersize=2, label='Clusters de pontos de intersecao')
plt.title('Variacao do indice Calinski-Harabaz em funcao do numero de clusters')
plt.xlabel('Numero de Clusters')
plt.ylabel('indice Calinski-Harabaz')
plt.xticks(range(1,len(x)+2))
plt.legend()
plt.grid(linestyle=':', lw=0.5)
plt.show()

print 'A apresentar o indice de Fowlkes-Mallows...'

fm_score_part = []

for d in Z_Points[2:,2]:
  r = fcluster(Z_Points, d, criterion='distance')
  r_fm = fowlkes_mallows_score(labels_true, r)
  fm_score_part.append(r_fm)

plt.plot(x_p,lista_fm_score, marker='o', linewidth=0.5, markersize=2, c='red' , label='Clustering de pontos de intersecao')
plt.plot(range(2,len(Z_Points[2:,2])+2),fm_score_part, marker='o', linewidth=0.5, markersize=2, label='Clustering de participacoes')
plt.title('Variacao do indice de Fowlkes-Mallows em funcao do numero de clusters')
plt.xlabel('Numero de Clusters de Participacoes')
plt.ylabel('Indice de Fowlkes-Mallows')
plt.xticks(range(1,len(x)+2))
plt.legend()
plt.grid(linestyle=':', lw=0.5)
plt.show()

# para 7 clusters
resultado_7c = getClusters(Z_intersectionPoints, retasIntersetadas, len(Perturbacoes), pontosIntersecao, alturas_dendrograma_sEnganosos[6])
print 'Resultado para 7 clusters'
print resultado_7c['list_clusterP']

#indice fowlkes_mallows
print fowlkes_mallows_score(labels_true, resultado_7c['clusterP'])
```

A.4 Criação e Análise do cenário 5

cenario5.py

```
# -*- coding: utf-8 -*-
from matplotlib import pyplot as plt
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet
from scipy.spatial.distance import pdist,squareform
from intersectionPoints import intersectionPoints,intersectionPoints_segmentos
import gpxpy.geo
from getClusters import getClusters
```

```
from sklearn.metrics import silhouette_score, calinski_harabaz_score,
fowlkes_mallows_score
from dunn_sklearn import dunn
# -------- Definir eventos e perturbações --------
eventos=[[41.550097,-8.429125], [41.550789,-8.429578], [41.54985,-8.43085],
[41.5490,-8.429580], [41.5513,-8.4302]]
cenario5 = [[41.549774293288522,-8.4297126461034502,53.73,[50,200]],
[41.549725190844491,-8.429172005181222,5.4,[30,50]],
[41.549978235754729,-8.4290983849352621,350.48,[10,30]],
[41.549967286506529,-8.4291581122160508,10.82,[10,30]],
[41.550222782827852,-8.4293374064349731,128.35,[10,30]],
[41.550279566700212,-8.4290460815920145,197.93,[10,30]],
[41.550245167401791,-8.428801131117627,238.56,[30,50]],
[41.550047684960532,-8.4292019605774655,49.43,[0,10]],
[41.54961672883384,-8.4293990752195924,23.13,[50,200]],
[41.549790178026363,-8.4294187204162299,35.62,[30,50]],
[41.550862298523299,-8.4295517146409331,195.02,[0,10]],
[41.55119188963203,-8.4300106184180503,141.22,[50,200]],
[41.550970192164968,-8.4297619151106957,142.78,[10,30]],
[41.550659670597035,-8.4297118446362891,37.76,[10,30]],
[41.550511201273906,-8.4295782458065833,0.04,[30,50]],
[41.55102216251678,-8.4294094038644491,208.42,[10,30]],
[41.551090961564796,-8.4301765266160942,123.99,[50,200]],
[41.551352641570745,-8.4295966771918227,178.58,[50,200]],
[41.550781012278748,-8.4297924775059876,87.15,[10,30]],
[41.550771265307269,-8.4300387745995575,87.06,[30,50]],
[41.550678183973687,-8.4318584922849684,137.66,[50,200]],
[41.549978233833116,-8.4305985228421765,235.73,[10,30]],
[41.549951262035428,-8.4317596000206496,98.46,[50,200]],
[41.549620176128165,-8.431721117133355,70.58,[50,200]],
[41.550218801223068,-8.4308858416163037,175.84,[30,50]],
[41.549605662077397,-8.4306435931980346,327.7,[30,50]],
[41.549527301941502,-8.4310611980688872,26.1,[30,50]],
[41.549131404765525,-8.430636943902666,347.49,[50,200]],
[41.549169154328595,-8.4308503666186549,0.02,[50,200]],
[41.549240904168094,-8.4294901947428595,300.9,[50,200]],
[41.549573849581954,-8.4298915471710991,157.89,[50,200]],
[41.54900120334316,-8.4291172065638076,269.8,[30,50]],
[41.548991708297393,-8.4295747569439889,334.68,[0,10]],
[41.548883983183508,-8.4296659230617568,29,[10,30]],
[41.548739992285874,-8.429782053681814,30.18,[30,50]],
[41.548776358691406,-8.429345808534034,321.91,[30,50]],
[41.548897914855125,-8.4296657412670797, 32.15, [10,30]],
[41.548964400140754,-8.4298184793613107,78.72,[10,30]],
[41.548950887026884,-8.4289779854822697,276.22,[50,200]],
[41.548702084050767,-8.4291660427361617,313.88,[30,50]],
[41.551503431429488,-8.4299306528484639,224.74,[30,50]],
[41.551576124180556,-8.4302380367687029,174.11,[30,50]],
[41.550995670507689,-8.4307555337686537,53.8,[50,200]],
[41.550908454026953,-8.4299239116880837,332.18,[30,50]],
[41.551652164601428,-8.4306280383099814,137.71,[50,200]],
[41.551355600533732,-8.430039555019798,245.15,[10,30]],
[41.550995711926276,-8.4301355164113954,350.99,[30,50]],
[41.551745646660315,-8.4300804403941498,191.35,[50,200]],
[41.551061971823373,-8.4303260072375004,21.61,[10,30]],
[41.551581583179583,-8.4304101527200359,150.82,[30,50]]]
```

```
14 cenario5_nome = ['P1'+str(i+1) for i in range(0,10)] + ['P2'+str(i+1) for i
   in range(0,10)] + ['P3'+str(i+1) for i in range(0,10)] + ['P4'+str(i+1) for
   i in range(0,10)] + ['P5'+str(i+1) for i in range(0,10)]
15 labeltrue=[1 for i in range(0,10)]+[2 for i in range(0,10)]+[3 for i in
   range(0,10)]+[4 for i in range(0,10)]+[5 for i in range(0,10)]
16
17 # -------- Representacao grafica do cenario --------
18 print 'A apresentar a  representacao grafica do cenario ....'
19 eventos = np.array(eventos)
20 colors = ['red', 'blue', 'green', 'brown', 'orange']
21 labels = ['E1','E2','E3','E4','E5']
22 PertporEvento = [[0,9], [10,19], [20,29], [30,39], [40,49]]
23 for i in range(5):
24   plt.scatter(eventos[i,0], eventos[i,1],c=colors[i], marker ='*',
     label=labels[i], s=80)
25   lim_inf = PertporEvento[i][0]
26   lim_sup = PertporEvento[i][1]
27   plt.scatter([p[0] for p in cenario5[lim_inf : lim_sup]], [p[1] for p in
     cenario5[lim_inf : lim_sup]],c=colors[i],marker='o', s=25)
28 plt.legend()
29 plt.rcParams.update({'font.size': 10})
30 plt.xlabel("latitude", fontsize=13)
31 plt.ylabel("longitude", fontsize=13)
32 plt.gca().set_aspect('equal', adjustable='box')
33 plt.show()
34
35 # -------- Obter pontos de interseção --------
36 print '\n'
37 print 'A obter pontos de intersecao...'
38 pontosIntersecao_sLimite, retasIntersetadas_sLimite, pontosFinais_sLimite,
   pontosMedios_sLimite,matriz_intersecao_sLimite =
   intersectionPoints(cenario5)
39 pontosIntersecao, retasIntersetadas, pontosFinais,
   pontosMedios,matriz_intersecao = intersectionPoints_segmentos(cenario5)
40 # pontosIntersecao - coordenadas dos pontos de interseção
41 # retasIntersetadas - as correspondentes retas que formam os pontos de
   intersecao
42 # pontosFinais (apenas usado para a representacao grafica) - coordenadas do
   ponto final da reta
43 # pontosMedios (apenas usado para a representacao grafica) - coordenadas do
   ponto médio da reta
44 # matriz_intersecao - matriz que contem informacao acerca de todas as retas
   intersetadas
45
46 pontosIntersecao_nomes=[]
47 for pInt  in retasIntersetadas:
48   if len(pInt) == 1:
49     pontosIntersecao_nomes.append(cenario5_nome[pInt[0]])
50   else:
51     pontosIntersecao_nomes.append(cenario5_nome[pInt[0]]+',
     '+cenario5_nome[pInt[1]])
52
53 # -------- Representacao grafica das retas e intersecoes --------
54 print 'A apresentar a  representacao grafica dos pontos de intersecao ....'
55 for i in range(5):
56   plt.scatter(eventos[i,0], eventos[i,1],c=colors[i], marker ='*',
     label=labels[i], s=80)
57   lim_inf = PertporEvento[i][0]
```

```
58    lim_sup = PertporEvento[i][1]
59    plt.scatter([p[0] for p in cenario5[lim_inf : lim_sup]], [p[1] for p in
      cenario5[lim_inf : lim_sup]],c=colors[i],marker='o', s=25)
60  pontosIntersecao = np.array(pontosIntersecao)
61  plt.scatter(pontosIntersecao[:,0],
    pontosIntersecao[:,1],c='black',marker='+', s=50 )
62  for i in range(len(cenario5)):
63    plt.plot([pontosMedios[i][0], pontosFinais[i][0]], [pontosMedios[i][1],
      pontosFinais[i][1]], c='grey', linewidth=0.2)
64    plt.plot([cenario5[i][0], pontosMedios[i][0]], [cenario5[i][1],
      pontosMedios[i][1]], c='grey', linestyle='--', linewidth=0.5)
65  plt.rcParams.update({'font.size': 10})
66  plt.xlabel("latitude", fontsize=13)
67  plt.ylabel("longitude", fontsize=13)
68  plt.gca().set_aspect('equal', adjustable='box')
69  plt.legend()
70  plt.show()
71
72  print ' ----- Informacoes -----'
73  print 'Numero de participacoes: ' + str(len(cenario5))
74  print 'Numero de pontos de intersecao : ' + str(len(pontosIntersecao))
75  print 'Numero de pontos de intersecao usando semirretas: '+
    str(len(pontosIntersecao_sLimite))
76  print 'Medida de dissemelhanca : average, distancia de haversine'
77
78  # -------- Metodo de clustering --------
79  # Definicao da metrica
80  def distancia(u,v):
81      return gpxpy.geo.haversine_distance(u[0], u[1], v[0], v[1])
82
83  print '\n'
84  print 'A calcular a matriz linkage para o criterio average e distancia de
    haversine ...'
85
86  # Matriz linkage dos pontos de intersecao
87  Z_intersectionPoints = linkage(pontosIntersecao, 'average',
    metric=distancia)
88  # Matriz linkage das participacoes
89  Z_Points = linkage([p[0:2] for p in cenario5] , 'average',
    metric=distancia)
90  #  Matriz de distancias
91  matrizDistancias_PI=pdist(pontosIntersecao, metric = distancia)
92  # Calculo do coeficiente cofenetico
93  c_intersectionPoints_average, coph_dists = cophenet(Z_intersectionPoints,
    matrizDistancias_PI)
94  print 'Coeficiente cofenetico: '+str(c_intersectionPoints_average)
95
96  # Dendrogramas
97  print 'A apresentar dendrogramas dos pontos de intersecao e das
    participacoes ....'
98  plt.title('Clustering Hierarquico de pontos de intersecao',fontsize=15)
99  plt.xlabel('POI',fontsize=14)
100 plt.ylabel('Dissemelhanca', fontsize=14)
101 d_intersectionPoints = dendrogram(
102     Z_intersectionPoints,
103     leaf_rotation=90.,
104     leaf_font_size=5.,
105     labels=pontosIntersecao_nomes)
```

```
106 plt.show()
107
108 plt.title('Clustering Hierarquico de participacoes')
109 plt.xlabel('Participacoes')
110 plt.ylabel('Dissemelhanca')
111 d_Points = dendrogram(
112     Z_Points,
113     leaf_rotation=90.,
114     leaf_font_size=12.,
115     labels=cenario5_nome)
116 plt.show()
117
118 # Aplicacao do algoritmo de identificacao de clusters
119 # 1) Aplicar o método a toda a hierarquia para identificar todos os pontos
    enganosos
120 resultado_identPI = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, 200)
121 # dicionario:
122 # 'clusterP' - lista onde cada indice representa a participacao e o valor
    nesse indice indica o cluster a que esta associado
123 # 'list_clusterP' - lista onde cada indice representa um cluster e o valor
    nesse indice e uma lista com as participacoes do cluster
124 # 'clusterPI' - lista onde cada indice representa o ponto de intersecao e o
    valor nesse indice indica o cluster a que esta associado
125 # 'list_clusterPI' - lista onde cada indice representa um cluster e o valor
    nesse indice e uma lista com os pontos de intersecao do cluster
126 # 'PIenganosos' - lista de pontos enganosos
127 # info
128
129 pontos_enganosos = sorted(resultado_identPI['PIenganosos'], reverse=False)
130 print '\n'
131 print 'A identificar todos os pontos enganosos ....'
132 print 'Numero de pontos enganosos identificados:
    '+str(len(pontos_enganosos))
133
134 # Atualizar matriz das distancias
135 removidos = 0
136 matrizDistancias_PI_square = squareform(matrizDistancias_PI)
137 pontosIntersecao_sEnganosos = pontosIntersecao
138 for index in pontos_enganosos:
139   # apagar linha
140   matrizDistancias_PI_square = np.delete(matrizDistancias_PI_square,
    index-removidos,0)
141   # apagar coluna
142   matrizDistancias_PI_square =
    np.delete(matrizDistancias_PI_square,index-removidos,1)
143   pontosIntersecao_sEnganosos =
    np.delete(pontosIntersecao_sEnganosos,index-removidos,0)
144   removidos +=1
145
146 # 2) Analise da variacao da dissemelhanca para identificar um numero ideal
    de clusters
147 print '\n'
148 print 'A representar a variacao da dissemelhanca em funcao do numero de
    clusters ...'
149
150 alturas_dendrograma_sEnganosos = []
151 alturas_dendrograma = []
```

```
152 uniao_enganosos = []
153 for index in range(len(Z_intersectionPoints)):
154   z = Z_intersectionPoints[index]
155   alturas_dendrograma = [z[2]]+ alturas_dendrograma
156   if z[0] not in pontos_enganosos and z[1] not in pontos_enganosos:
157     alturas_dendrograma_sEnganosos = [z[2]]+ alturas_dendrograma_sEnganosos
158   else:
159     uniao_enganosos.append(len(Z_intersectionPoints)-index-1)
160 plt.subplot(1, 2, 1)
161 x = range(1, len(alturas_dendrograma)+1)
162 plt.plot(x,alturas_dendrograma, marker='o', linewidth=0.5, markersize=2)
163 for i in range(len(uniao_enganosos)):
164   index = uniao_enganosos[i]
165   if i == 0:
166     plt.plot(x[index],alturas_dendrograma[index], marker='o',
      linewidth=0.5, markersize=2, c='red', label = 'Uniao de ponto
      enganoso')
167   else:
168     plt.plot(x[index],alturas_dendrograma[index], marker='o',
      linewidth=0.5, markersize=2, c='red')
169 plt.title('Variacao da dissemelhanca em funcao do numero de clusters')
170 plt.xlabel('Numero de Clusters')
171 plt.ylabel('Dissemelhanca')
172 plt.grid(linestyle=':', lw=0.5)
173 plt.legend()
174
175 plt.subplot(1, 2, 2)
176 x_sEnganosos = range(1, len(alturas_dendrograma_sEnganosos)+1)
177 plt.plot(x_sEnganosos,alturas_dendrograma_sEnganosos, marker='o',
    linewidth=0.5, markersize=2)
178 plt.title('Variacao da dissemelhanca em funcao do numero de clusters')
179 plt.xlabel('Numero de Clusters')
180 plt.ylabel('Dissemelhanca')
181 plt.grid(linestyle=':', lw=0.5)
182 plt.show()
183
184 lista_silhouette = []
185 lista_dunn = []
186 lista_CH = []
187 x_pI = []
188 x_p = []
189 for i in range(1,len(alturas_dendrograma_sEnganosos)):
190   d = alturas_dendrograma_sEnganosos[i]
191   r = getClusters(Z_intersectionPoints, retasIntersetadas, len(cenario5),
    pontosIntersecao, d)
192   # Eliminar pontos enganosos
193   clusters_pontosIntersecao = r['clusterPI']
194   clusters_participacoes = r['clusterP']
195   removidos = 0
196   for index in pontos_enganosos:
197     clusters_pontosIntersecao = np.delete(clusters_pontosIntersecao,
      index-removidos)
198     removidos +=1
199   # numero de clusters identificados
200   x_p.append(len(set(clusters_participacoes)))
201   x_pI.append(len(set(clusters_pontosIntersecao)))
202   # Analise do coeficiente silhouette
```

```
203     indice_silhouette =
        silhouette_score(matrizDistancias_PI_square,clusters_pontosIntersecao,
        metric="precomputed")
204     lista_silhouette.append(indice_silhouette)
205     # Analise do indice dunn
206     indice_dunn = dunn(clusters_pontosIntersecao, matrizDistancias_PI_square)
207     lista_dunn.append(indice_dunn)
208     # Analise do indice Calinski-Harabaz
209     indice_CH = calinski_harabaz_score(pontosIntersecao_sEnganosos,
        clusters_pontosIntersecao)
210     lista_CH.append(indice_CH)
211
212 #representação grafica dos indicadores
213
214 print 'A apresentar a variação do indice Silhouette em funcao do numero de
    clusters...'
215 plt.plot(x_p,lista_silhouette, marker='o', linewidth=0.5, markersize=2,
    c='red',  label='Clusters de participacoes')
216 plt.plot(x_pI,lista_silhouette, marker='o', linewidth=0.5, markersize=2,
    label='Clusters de pontos de intersecao')
217 plt.title('Variacao do indice silhouette em funcao do numero de clusters')
218 plt.xlabel('Numero de Clusters')
219 plt.legend()
220 plt.ylabel('indice silhouette')
221 plt.grid(linestyle=':', lw=0.5)
222 plt.show()
223
224 print 'A apresentar a variação do indice Dunn em funcao do numero de
    clusters...'
225 plt.plot(x_p,lista_dunn, marker='o', linewidth=0.5, markersize=2,c='red',
    label='Clusters de participacoes')
226 plt.plot(x_pI,lista_dunn, marker='o', linewidth=0.5, markersize=2,
    label='Clusters de pontos de intersecao')
227 plt.title('Variacao do indice Dunn em funcao do numero de clusters')
228 plt.xlabel('Numero de Clusters')
229 plt.ylabel('indice Dunn')
230 plt.grid(linestyle=':', lw=0.5)
231 plt.legend()
232 plt.show()
233
234 print 'A apresentar a variação do indice Calinski-Harabaz em funcao do
    numero de clusters...'
235 plt.plot(x_p,lista_CH, marker='o', linewidth=0.5, markersize=2,c='red',
    label='Clusters de participacoes')
236 plt.plot(x_pI,lista_CH, marker='o', linewidth=0.5,
    markersize=2,label='Clusters de pontos de intersecao')
237 plt.title('Variacao do indice Calinski-Harabaz em funcao do numero de
    clusters')
238 plt.xlabel('Numero de Clusters')
239 plt.ylabel('indice Calinski-Harabaz')
240 plt.legend()
241 plt.grid(linestyle=':', lw=0.5)
242 plt.show()
243
244 # para 5 clusters
245 resultado_5c = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, alturas_dendrograma_sEnganosos[4])
246 print 'Resultado para 5 clusters'
```

```
247 print resultado_5c['list_clusterP']
248 resultado_4c = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, alturas_dendrograma_sEnganosos[3])
249 print 'Resultado para 4 clusters'
250 print resultado_4c['list_clusterP']
251 resultado_2c = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, alturas_dendrograma_sEnganosos[1])
252 resultado_3c = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, alturas_dendrograma_sEnganosos[2])
253 resultado_7c = getClusters(Z_intersectionPoints, retasIntersetadas,
    len(cenario5), pontosIntersecao, alturas_dendrograma_sEnganosos[6])
254
255
256 print 'Indice Fowlkes-Mallows para 2 clusters: '+
    str(fowlkes_mallows_score(labeltrue, resultado_2c['clusterP']))
257 print 'Indice Fowlkes-Mallows para 3 clusters: '+
    str(fowlkes_mallows_score(labeltrue, resultado_3c['clusterP']))
258 print 'Indice Fowlkes-Mallows para 4 clusters: '+
    str(fowlkes_mallows_score(labeltrue, resultado_4c['clusterP']))
259 print 'Indice Fowlkes-Mallows para 5 clusters: '+
    str(fowlkes_mallows_score(labeltrue, resultado_5c['clusterP']))
260 print 'Indice Fowlkes-Mallows para 7 clusters: '+
    str(fowlkes_mallows_score(labeltrue, resultado_7c['clusterP']))
```