



Posicionamento das Equipes de Campo da ESCELSA

Hilário Seibel Júnior

Setembro de 2004

Resumo

Há tempos, a Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) precisou definir um posicionamento para suas equipes de campo ao longo do Estado. Este posicionamento não foi escolhido de uma forma ótima, porém foi mantido desde então. Através de técnicas de *clustering*, sugeriremos novos posicionamentos para tais equipes, de modo que o custo de elas percorrerem o estado para atender as ocorrências de atendimento emergenciais da empresa deverá ser mínimo.

Dedico este projeto a todas as pessoas maravilhosas que me apoiaram incondicionalmente em tudo que já fiz. A meus pais, minhas irmãs lindas, avós, tios, tias e toda a família que ainda está aqui ou que já se foi. Todos foram essenciais no meu crescimento.

Aos irmãos de coração, Thiago e Rafael, por nunca terem me deixado só quando precisava.

Aos amigos e amigas que fiz pela vida e aos da UFES que não se livrarão de mim tão cedo (Mari, Daniel, Márcio, Alex, Jociel, Diogo, Francisco, Arthur).

E à minha Binha linda, que me faz esquecer os problemas do mundo e por ter me tornado uma pessoa mais feliz.

Valeu, galera! Amo todos vocês! Muito!

Não poderia terminar minha graduação sem agradecer àqueles grandes mestres que fazem seus alunos crescerem e, acima de tudo, gostarem de estudar.

Ao “tio” Raul, por ter confiado em mim desde o começo e por ter se tornado um grande amigo.

À Rô e à Boeres, por levantarem meu astral diariamente.

Ao Sérgio, pelo grande professor e ser humano que é.

E a Arlindo, Thomas, Cristina, Lucia, Andrea e outros poucos que, além de professores, são pessoas fenomenais.

Vou sentir saudades daqui!!

; -)

Sumário

1	Introdução	7
1.1	Motivação	7
1.2	Objetivo	7
2	O Problema Estudado	8
2.1	O Problema da Distribuição de Equipes pelo Estado	8
2.2	A Velocidade Média das Equipes	9
2.3	Terminologias	11
2.3.1	Terminologia da ANEEL	11
2.3.2	Demais termos	12
3	<i>Clustering</i>	13
3.1	O Problema de <i>Clustering</i>	13
3.2	Definição Formal	15
3.3	Algoritmos Tradicionais de <i>Clustering</i>	15
3.3.1	Métodos por Particionamento	16
3.3.2	Métodos Hierárquicos	16
3.4	Aplicação ao nosso problema	18
4	Algoritmos Estudados	19
4.1	<i>k-means</i>	19
4.2	PAM	21
4.3	CLARA	22
4.4	DBSCAN	23
5	Resultados Computacionais	25
5.1	Validação dos Resultados	25
5.2	<i>Time Window</i>	26
5.2.1	<i>Regionais</i>	27
5.2.2	<i>Todo Estado</i>	28
5.3	<i>Regionais</i> versus <i>Todo Estado</i>	29

6	Conclusão	30
A	Cálculos e Fórmulas	32
A.1	Cálculo do DEC	32
A.2	Cálculo do FEC	33
B	Interface Gráfica	34
B.1	Tela Inicial	35
B.2	Resultado do <i>clustering</i>	36
B.3	O percurso das equipes	37
C	Implementação	38
C.1	Por que Python?	38
C.2	Implementação da Interface	39

Lista de Figuras

2.1	Todas as chaves do Estado	8
3.1	Um exemplo de <i>clustering</i>	13
3.2	Etapas do <i>Clustering</i>	14
3.3	Exemplo de um <i>dendograma</i>	17
3.4	<i>Clusters</i> formados por um método hierárquico	17
4.1	Um resultado do <i>k-means</i>	20
4.2	<i>Clustering</i> por Densidade (a) X <i>Clustering</i> por Partição (b)	24
5.1	Algoritmos - Regionais	27
5.2	Algoritmos - Todo Estado	28
5.3	Comparação entre <i>Regionais</i> e <i>Todo Estado</i>	29
A.1	Fórmula de cálculo do DEC	32
A.2	Fórmula de cálculo do FEC	33
B.1	Tela inicial da Interface	34
B.2	Exemplo de Ampliação do Painel	35
B.3	Resultado do <i>clustering</i>	36
B.4	Caminho percorrido por uma equipe num certo dia	37

Lista de Tabelas

2.1	Cálculo das velocidades médias	10
5.1	Algoritmos - Regionais	27
5.2	Algoritmos - Todo Estado	28
5.3	Comparação entre <i>Regionais</i> e <i>Todo Estado</i>	29

Capítulo 1

Introdução

1.1 Motivação

A Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) possui equipes de campo responsáveis por atender as ocorrências de atendimento emergenciais que surgem aleatoriamente ao longo do Estado. Como não é possível prever onde haverá ocorrências, a empresa definiu um posicionamento para estas equipes ficarem à espera dos problemas que surgirem. No entanto, este posicionamento não parece ter sido feito de uma forma satisfatória.

1.2 Objetivo

O objetivo deste projeto é propor um posicionamento para as equipes da Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) ao longo do estado. Este posicionamento deverá ser tal que, através da análise de dados históricos de atendimentos fornecidos pela empresa, o custo de estas equipes percorrerem o estado para atender tais ocorrências deverá ser mínimo.

Capítulo 2

O Problema Estudado

2.1 O Problema da Distribuição de Equipes pelo Estado

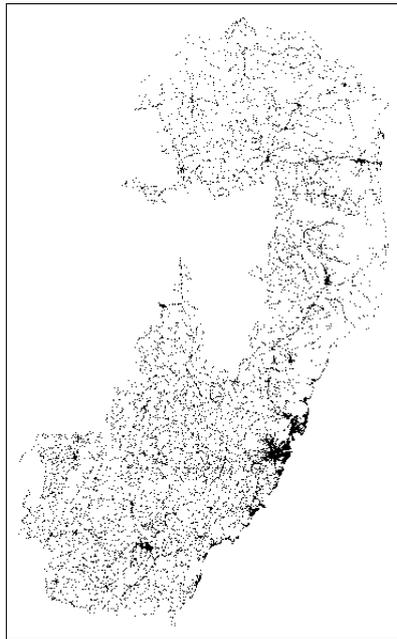


Figura 2.1: Todas as chaves do Estado

A Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) dispõe hoje de aproximadamente 20.000 chaves (figura 2.1) ao longo do estado. Estas chaves caracterizam os locais onde pode haver problemas que causem queda de energia, afetando os clientes da empresa. Esses clientes foram di-

vididos geograficamente em conjuntos, e para cada um desses conjuntos é calculado mensalmente um valor (DEC) que medirá quão seus consumidores ficaram sem energia no mês estudado. O cálculo do DEC (ver A.1) leva em conta o número de consumidores atingidos por uma determinada ocorrência, o intervalo de tempo desta falta de energia desde o momento do conhecimento do problema até o final do serviço, e a quantidade de consumidores do conjunto.

Pode-se notar que um fator relevante neste cálculo que poderia ser minimizado é o intervalo de tempo descrito anteriormente (diferença entre o tempo que a equipe chegou ao local da ocorrência e o tempo em que a empresa tomou conhecimento da mesma). Para tal, é proposto neste projeto uma melhor distribuição das equipes de atendimento, de modo que elas fiquem posicionadas mais perto de onde os problemas supostamente ocorrerão. Através dos dados históricos de atendimento fornecidos pela empresa, serão estudados diversos algoritmos de *clustering* [1, 2, 3, 4] na tentativa de minimizar o custo citado.

2.2 A Velocidade Média das Equipes

Nem todas as chaves do Estado possuem conexões entre si, e não é difícil compreender por quê. Existem limitações geográficas¹ que não permitem que haja uma ligação entre quaisquer pares de chaves. Algo prejudicial na nossa análise é, portanto, o fato de não termos o mapeamento de todos os caminhos possíveis entre as chaves e suas respectivas distâncias. O número de rotas conhecidas no histórico da empresa é ínfimo, e esses caminhos seriam necessários para simularmos de forma realística os percursos feitos pelos veículos ao longo do Estado. Para contornar o problema, consideraremos que todas as chaves estão conectadas entre si, adotando a Distância Euclidiana entre dois pontos para determinar quanto os carros percorrerão para ir de uma chave à outra. Essa distância, multiplicada por uma velocidade média *vm* abaixo da real, dará o tempo de deslocamento de um veículo entre duas chaves quaisquer.

Houve, então, um estudo sobre esta velocidade média da seguinte forma: Para cada ocorrência do histórico, sabemos o horário que a equipe chegou ao local da chave problemática (*i*) e o horário no qual a equipe saiu da última ocorrência antes da atual.

A diferença entre estes tempos de deslocamento chamaremos de t_i . A Distância Euclidiana entre as chaves em questão chamaremos de d_i . O

¹Rios, montanhas, etc.

número de ocorrências do período chamaremos de n . De posse de t_i , d_i e n , calculamos dois tipos de velocidade média:

1. SD/ST é a razão entre a soma de todas as Distâncias Euclidianas percorridas num determinado mês e a soma de todos os tempos de deslocamento deste período.

$$SD/ST = \frac{\sum_{i=1}^n d_i}{\sum_{i=1}^n t_i}$$

2. vm_m é a média de todas as velocidades médias das equipes entre tais pares de chaves no período analisado.

$$vm_m = \frac{\sum_{i=1}^n (d_i/t_i)}{n}$$

Mês	SD/ST	vm_m
09/2003	20.43	21.11
10/2003	21.14	21.54
11/2003	21.95	23.09
12/2003	46.2	31.33
01/2004	41.49	26.3
02/2004	33.71	25.32
03/2004	36.62	30.39
04/2004	36.84	24.9
05/2004	21.73	21.4
Todos os meses	33.35	25.56

Tabela 2.1: Cálculo das velocidades médias

O estudo gerou os resultados presentes na tabela 2.1. Pode-se notar na tabela que $20km/h < vm < 50km/h$. Foi adotada então, em nosso estudo, $vm = 20km/h$ para quaisquer pares de chaves. Intuitivamente, pode-se perceber que esta velocidade adotada é bem mais baixa que a real². O objetivo de utilizarmos esta velocidade média bem baixa é que a consistência dos nossos resultados obtidos, se melhores que os reais, não seja afetada por esta falta de dados.

²A velocidade média tão baixa encontrada pode ter sido causada por erros de medição ou de cadastro dos dados.

2.3 Terminologias

Usaremos alguns termos ao longo deste trabalho bastante específicos do nosso problema. Alguns deles serão, portanto, descritos nesta seção.

2.3.1 Terminologia da ANEEL

A Agência Nacional de Energia Elétrica (*ANEEL*), nas Resoluções nº 24 de 27 de janeiro de 2002 e nº 520 de 17 de setembro de 2002, definiu os termos a seguir:

- Consumidor: Pessoa física ou jurídica, ou comunhão de fato ou de direito, legalmente representada, que assumir a responsabilidade pelo pagamento das faturas de energia elétrica e pelas demais obrigações fixadas em normas e regulamentos da Agência Nacional de Energia Elétrica (*ANEEL*), assim vinculando-se ao contrato de fornecimento, de uso e de conexão ou de adesão, conforme cada caso.
- Conjunto de Unidades Consumidoras: Qualquer agrupamento de unidades consumidoras, global ou parcial, de uma mesma área de concessão de distribuição, definido pela concessionária ou permissionária e aprovado pela Agência Nacional de Energia Elétrica (*ANEEL*).
- Interrupção de Energia Elétrica: Descontinuidade do neutro ou da tensão disponível em qualquer uma das fases de um circuito elétrico que atende a unidade consumidora.
- Ocorrência Emergencial: Evento na rede elétrica que prejudique a segurança e/ou a qualidade do serviço prestado ao consumidor, com conseqüente deslocamento de equipes de atendimento de emergência.
- Período de Apuração: Intervalo de tempo estabelecido para registro e apuração das ocorrências emergenciais de um determinado conjunto de unidades consumidoras.
- Tempo de Preparação (TP): Intervalo de tempo para o atendimento da ocorrência emergencial, expresso em minutos, compreendido entre o conhecimento da existência de uma ocorrência e o instante da autorização para o deslocamento da equipe de emergência.
- Tempo de Deslocamento (TD): Intervalo de tempo, expresso em minutos, compreendido entre o instante da autorização para o deslocamento da equipe de atendimento de emergência até o instante de chegada no local da ocorrência.

- Tempo de Mobilização (TM): Intervalo de tempo, expresso em minutos, compreendido entre o conhecimento da existência de uma ocorrência emergencial, o deslocamento e o instante de chegada da equipe de atendimento de emergência no local da ocorrência, correspondendo à soma dos tempos TP e TD.
- Tempo Médio de Mobilização (TMM): Valor médio correspondente aos tempos de mobilização (TM) das equipes de emergência, para o atendimento às ocorrências emergenciais verificadas em um determinado conjunto de unidades consumidoras, no período de apuração considerado.
- DEC - Duração Equivalente de Interrupção por Unidade Consumidora: Intervalo de tempo que, em média, no período de observação, em cada unidade consumidora do conjunto considerado ocorreu descontinuidade da distribuição de energia elétrica, expresso em horas e centésimos de hora, ver A.1.
- FEC - Frequência Equivalente de Interrupção por Unidade Consumidora: Número de interrupções ocorridas, em média, no período de observação, em cada unidade consumidora do conjunto considerado, expresso em números de interrupções centésimos de números de interrupções, ver A.2.

2.3.2 Demais termos

Além dos termos descritos em 2.3.1, ainda consideraremos os seguintes:

- Chave: Unidade física que caracteriza um local onde pode haver problemas que causem queda de energia, afetando os clientes da empresa.
- *Cluster* (pl.: *clusters*): Do inglês, significa grupo.
- *Clustering*: Substantivo derivado do verbo inglês *to cluster*, que significa agrupar. Neste documento, *clustering* se referirá ao ato de agrupar dados ou objetos.

Capítulo 3

Clustering

“Clustering é a classificação não-supervisionada de padrões (observações, itens de dados ou vetores de características) em grupos (clusters)” [1].

3.1 O Problema de *Clustering*

Imagine como escolher onde localizar k hospitais numa cidade de modo que nenhum habitante more mais longe que o necessário do hospital mais próximo. Este problema pode não ser tão facilmente resolvido dependendo do tamanho da cidade em questão. *Alpert e Kahng* citam-no como exemplo de um problema de *clustering* em [3]. Outro exemplo é o de agrupar os pontos mais similares entre si da primeira parte da figura 3.1. Na segunda parte da figura, cada ponto foi substituído pelo número do *cluster* a que ele pertence após o *clustering*.

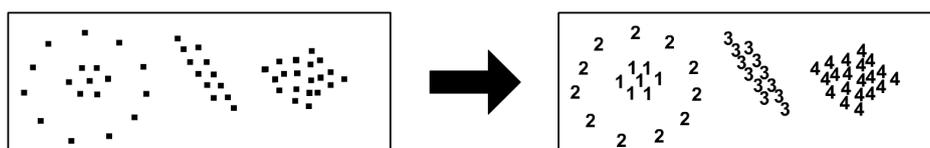


Figura 3.1: Um exemplo de *clustering*

Bastante estudado na literatura, *clustering* trata-se basicamente de agrupar determinados objetos em grupos de acordo com características comuns a eles, e separá-los de acordo com suas características que os diferenciem. Segundo Guha *et al.* [2], o problema de *clustering* pode ser definido como a seguir: dados n objetos num espaço métrico de d dimensões, particiona-se estes objetos em k grupos (*clusters*), tais que os objetos em um grupo são mais similares entre si do que em relação a objetos de outros grupos.

Os objetos analisados precisam ter características que possam vir a ser usadas para agrupá-los ou separá-los. Eles costumam ter, portanto, um vetor de d dimensões com essas características. Para analisá-las, é necessário uma função de similaridade que leve em conta as características mais relevantes para a classificação. Por exemplo, pontos num espaço bidimensional poderiam ser agrupados de acordo com suas coordenadas cartesianas, e a função de similaridade poderia usar apenas a Distância Euclidiana como fator de classificação.

Jain *et al.* [1] citam 5 passos típicos envolvidos num processo de agrupamento:

1. Representação dos Padrões: Leva em consideração o número de padrões, número de classes e número, tipo e escala das características. Nesta etapa, opcionalmente, pode haver a *Seleção das Características* mais apropriadas a serem analisados no agrupamento. Pode haver também a *Extração das Características*, que é o ato de transformar as características dos objetos em algo mensurável.
2. Definição da medida de proximidade dos padrões apropriada ao domínio de dados (normalmente, uma função de distância entre pares de objetos).
3. Agrupamento (*Clustering*).
4. Abstração dos dados: Trata-se de simplificar e compactar a representação de um conjunto de dados, para uma análise automática posterior por alguma máquina ou para ser compreendida por humanos.
5. Geração de uma saída (*output*).

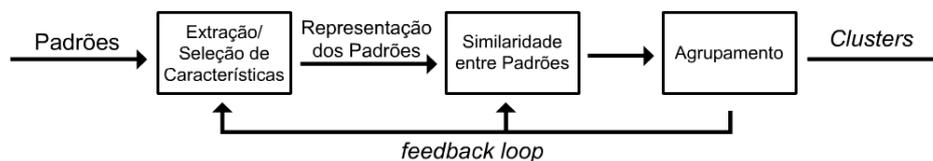


Figura 3.2: Etapas do *Clustering*

Além destas etapas, pode haver ainda uma última de validação dos grupos gerados, onde será feita uma análise do que foi obtido com o algoritmo.

A figura 3.2 mostra uma seqüência típica das três primeira etapas, incluindo ainda um *feedback loop*, no qual a saída do processo de agrupamento pode afetar a próxima extração de características ou as computações de similaridade.

3.2 Definição Formal

Uma definição formal do problema de *Clustering* é encontrada em [3, 4] e descrita a seguir:

- $X = \{X_1, X_2, \dots, X_n\}$ é o conjunto de todos os n objetos da base de dados a serem classificados. Cada $X_i \in \mathbb{R}^d$ é um vetor de dimensão d (as d características do objeto).
- $C = \{C_1, C_2, \dots, C_k\}$ são os k grupos de objetos (*clusters*) formados pelo algoritmo. As seguintes propriedades devem ser respeitadas no processo de formação dos *clusters*.
 1. $C_1 \cup C_2 \cup \dots \cup C_k = X$;
 2. $C_i \neq \emptyset, \forall i, 1 \leq i \leq k$;
 3. $C_i \cap C_j = \emptyset, \forall i \neq j, 1 \leq i \leq k, 1 \leq j \leq k$

Em suma, as propriedades acima significam que cada grupo precisa ter ao menos um objeto e que cada objeto deve estar contido em exatamente um grupo.

3.3 Algoritmos Tradicionais de *Clustering*

Os algoritmos conhecidos de *clustering* atendem a diferentes tipos de requisitos, tais como [4]:

- encontrar ou não um número adequado de *clusters* (alguns algoritmos precisam de um valor fixado de *clusters*).
- ser capazes de desprezar ou não os ruídos.
- descobrir *clusters* com formas arbitrárias.
- identificar *clusters* de tamanhos variados.
- trabalhar com objetos de diversos números de atributos.
- fornecer resultados interpretáveis e utilizáveis.
- apresentar o resultado num tempo satisfatório.

Como não há um método que atenda a todas estas características da melhor forma, foram criados algoritmos que se adaptam melhor a alguns conjuntos dessas características. Ao utilizar um certo método, deve-se analisar quais dos aspectos citados acima ele satisfaz e escolher aquele que se adaptar melhor às suas necessidades. De acordo com essas características, os algoritmos de *clustering* foram divididos em categorias [4]:

- Métodos por particionamento;
- Métodos hierárquicos;
- Métodos baseados em densidade;
- Métodos baseados em grade;
- Métodos baseados em modelos;

Os algoritmos mais tradicionais de *clustering* existentes são os métodos por particionamento e hierárquicos, descritos a seguir.

3.3.1 Métodos por Particionamento

A idéia principal deste tipo de algoritmo é que ele divide o conjunto de objetos em k grupos, sendo que k é dado como entrada pelo usuário. Na maioria dos problemas este número não é conhecido, tornando o algoritmo inutilizável em alguns casos.

Primeiramente, o algoritmo escolhe k objetos da base de dados que serão os centros iniciais de k *clusters*. Cada objeto é então atribuído ao *cluster* cujo centro lhe é mais similar, de acordo com a função de similaridade adotada. Quando todo objeto já estiver em um grupo, o centro de cada *cluster* é recalculado e os objetos são reatribuídos aos seus centros mais próximos. O algoritmo termina quando um critério de parada é atingido. Esse critério pode ser, por exemplo, um número máximo de iterações ou quando não há mais mudança nos centros dos grupos.

Os algoritmos de *clustering* por particionamento se distinguem pela forma como os k centros iniciais são escolhidos e como tais centros são recalculados em cada iteração.

3.3.2 Métodos Hierárquicos

Ao contrário dos métodos por partição, os hierárquicos não requerem um número k de grupos fixado *a priori*. Assim como os de partição, são também bastante populares, apesar de serem geralmente mais custosos.

Num método hierárquico, é formada uma árvore na qual cada nível representa uma subdivisão no conjunto de objetos, conhecida como *dendograma* (figura 3.3). O *dendograma* irá, então, representar os *clusters* gerados pelo algoritmo. No fim da árvore temos cada objeto em apenas um grupo.

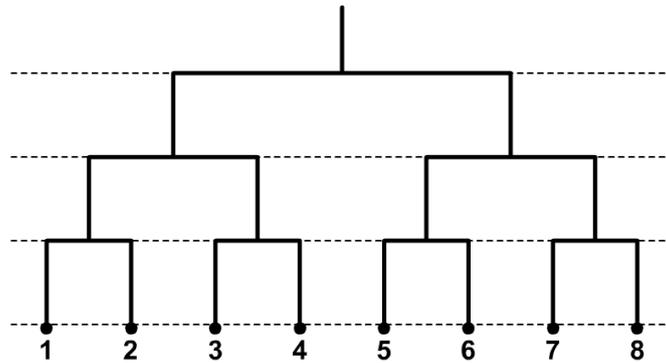


Figura 3.3: Exemplo de um *dendograma*

Se a árvore for sendo formada de cima para baixo, o *clustering* é chamado de aglomerativo. Dessa forma, começamos com cada objeto num *cluster* e, a cada passo, juntamos um par de grupos até que haja apenas um grande grupo que contenha todos os objetos (raiz de árvore). Uma etapa intermediária deste processo é mostrada na figura 3.4.

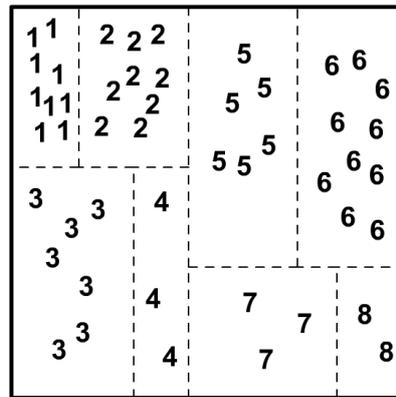


Figura 3.4: *Clusters* formados por um método hierárquico

Caso contrário, o *clustering* é chamado de divisivo. Começa-se com todos os objetos em um só grupo, e são sendo formados subgrupos a partir dele, até que se isole cada objeto num só *cluster*.

3.4 Aplicação ao nosso problema

No problema estudado os objetos a serem agrupados serão as chaves da Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*), que estão distribuídas geograficamente pelo estado. As características destas chaves a serem analisadas no processo de *clustering* são, além de suas posições geográficas, suas densidades de ocorrências de atendimentos retiradas do histórico e a quantidade de consumidores atingidos por elas.

Para sugerirmos os novos posicionamentos das equipes, implementaremos diversos algoritmos de *clustering* e compararemos quais se adaptaram melhor ao problema. Essa adaptividade será medida através do custo que seria gerado (DEC, ver A.1) em cada conjunto caso os posicionamentos das equipes sugeridos por tais algoritmos tivessem sido utilizados. Além disso, tais custos também serão comparados com os custos reais obtidos nos conjuntos.

Capítulo 4

Algoritmos Estudados

4.1 *k-means*

O algoritmo *k-means* [1, 4] é o mais conhecido dos métodos por particionamento. Nele, os k centros iniciais dos *clusters* são escolhidos aleatoriamente entre os objetos da base de dados e, depois de feita a escolha, todos os demais objetos são atribuídos ao grupo cujo centro lhe for mais similar. No final de cada iteração, cada grupo passa a ter como centro o seu próprio centro de gravidade (média dos pontos do grupo). Abaixo, um código resumido do algoritmo, implementado em *Python*:

```
def kmeans(k, objects):
    means = sample(objects, k)
    changed = True
    while changed:
        clusters = group(means, points)
        newMeans = reCalcMeans(points)
        changed = (means != newMeans)
        means = newMeans
    return clusters
```

A função *sample()* retorna uma lista de k objetos escolhidos aleatoriamente. A função *group()* inclui cada objeto no *cluster* cujo centro lhe for mais similar, retornando uma lista com os k *clusters* e seus respectivos objetos. Por fim, *reCalcMeans()* servirá para calcular os novos centros de gravidade de cada grupo. O algoritmo termina quando os centros não se modificam mais.

O *k-means* é bastante popular por ser fácil de implementar e por ter um bom desempenho em base de dados grandes, já que sua complexidade

computacional é $O(n \cdot k \cdot t)$, onde n é o total de objetos da base de dados, k é o número de grupos formados e t é o número de iterações. Segundo *Carlantonio* [4], normalmente $n \gg k$ e $n \gg t$.

O fato de o algoritmo *k-means* exigir que definamos inicialmente quantos grupos serão formados não foi prejudicial ao nosso problema, visto que utilizamos k = número de equipes de atendimento que a Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) possui no Estado. Em contrapartida, o algoritmo não permitiu que levássemos em conta o número de consumidores afetados por uma interrupção de energia na formação dos *clusters*.

Na figura 4.1, vemos um resultado do algoritmo *k-means*. As chaves de mesma cor foram agrupadas em um mesmo *cluster*. Foram gerados 75 grupos, que é uma média da quantidade de equipes da Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) no Estado.

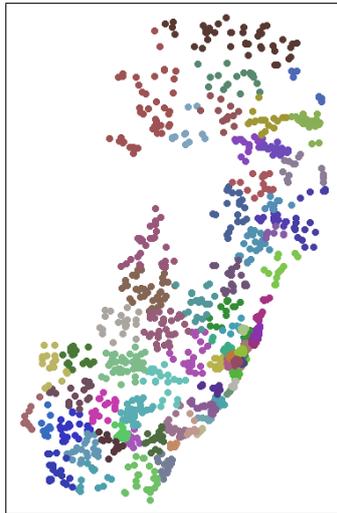


Figura 4.1: Um resultado do *k-means*

Vale lembrar que o *k-means* é aleatório e, por isso, a qualidade de seus resultados depende muito dos k centros iniciais escolhidos. O ideal é rodarmos o algoritmo várias vezes em busca de um melhor solução. Além disso, o algoritmo *k-means* é muito sensível a ruídos, visto que um objeto bem distante dos demais em um certo *cluster* pode afetar substancialmente o cálculo do centro de gravidade do grupo a que ele pertence. No nosso caso, esses ruídos não representam erros de medição nem dados inconsistentes. Eles se tratam, sim, de chaves isoladas geograficamente em áreas pouco habitadas do interior do Estado. Essas chaves não podem ser descartadas, mas também não devem influenciar tanto o cálculo dos centros dos grupos.

4.2 PAM

Como descrito em 4.1, o algoritmo *k-means* é muito sensível a ruídos, por usar o centro de gravidade como ponto representativo de um *cluster*. Existe uma abordagem chamada *k-medoids* na qual, ao invés de utilizarmos o valor médio dos objetos em um *cluster* como seu ponto de referência, usamos o *medoid*, que é o objeto localizado mais ao centro do grupo. Neste caso, o centro do *cluster* será, necessariamente, um objeto do próprio grupo.

PAM (Partitioning around Medoids) [4] foi um dos primeiros algoritmos *k-medoids* apresentados. A diferença básica entre ele e o *k-means* é a forma como os centros dos *clusters* são escolhidos.

O algoritmo também começa com a escolha aleatória de *k* objetos da base de dados, que serão os primeiros centros dos *k clusters*. Tais centros são chamados, neste método, de medóides. Cada objeto é, então, atribuído ao *cluster* cujo medóide lhe é mais similar. Temos portanto, neste momento, os primeiros *k clusters* formados pelo algoritmo. A partir daí, repetidamente, são escolhidos *k* novos medóides para cada *cluster*, gerando *k* novos grupos. Assim como no *k-means*, o processo termina quando não há mais mudança nos centros dos grupos.

Para escolhermos os novos medóides, entretanto, todos os objetos O_i possíveis são analisados e é escolhido como novo centro de cada grupo aquele objeto que causa maior redução no custo total do *cluster*. Esse custo é a soma de todos os custos entre o objeto O_i e os demais objetos O_j do grupo a que ele pertence. Para tal cálculo, no nosso problema, levamos em conta o custo de deslocamento entre O_i e O_j , a densidade de ocorrências de atendimentos emergenciais de tais objetos no período analisado, e a quantidade de consumidores afetados por eles. Vale lembrar que não pudemos analisar essas duas últimas características no *k-means*.

A seguir, mostramos um código resumido do PAM:

```
def pam(objects, k):
    medoids = sample(objects, k)
    changed = True
    while changed:
        clusters = group(medoids, points)
        newMedoids = reCalcMedoids(points)
        changed = (medoids != newMedoids)
        medoids = newMedoids
    return clusters
```

A função *sample()* retorna uma lista de *k* objetos escolhidos aleatoriamente. A função *group()* inclui cada objeto no *cluster* cujo centro lhe for

mais similar, retornando uma lista com os k *clusters* e seus respectivos objetos. Por fim, em *reCalcMedoids()*, é calculado o novo centro de cada *cluster* como sendo o objeto que possui menor custo total, como descrito anteriormente.

Apesar de produzir *clusters* de boa qualidade, o *PAM* tem um grande problema. Assim como o método *k-means*, seu resultado ainda é aleatório, pois depende dos k centros escolhidos inicialmente. Ele também precisa, portanto, ser executado várias vezes para termos mais chances de conseguirmos obter uma boa qualidade dos *clusters*. Isso se agrava ainda mais no *PAM* pois, por sempre percorrermos todos os objetos de cada grupo em busca daquele que provocaria maior redução do custo total caso fosse eleito como seu medóide, pode-se notar que, para valores muito grandes de n e k , que é o nosso caso, a computação deste método torna-se muito custosa.

4.3 CLARA

A alta ordem de complexidade do algoritmo *PAM*, descrito em 4.2, torna-o proibitivo computacionalmente para grandes base de dados. Utilizamos, então, um método alternativo ao *PAM* chamado *CLARA* (*Clustering LARge Applications*) [4]. A idéia deste algoritmo é rodarmos o *PAM* apenas para uma pequena amostra da base de dados. Cada objeto restante, entre aqueles que não fizeram parte da amostra, é posteriormente incluído no cluster cujo centro lhe for mais similar.

São feitas algumas iterações, com diferentes amostras, e é retornado o melhor resultado obtido (aquele com menor custo total). Seguimos com um código simplificado do método *CLARA*, onde o *PAM* é executado apenas uma vez para uma amostra de tamanho s .

```
def clara(objects, k, s):
    objSample = sample(objects, s)
    others = [x for x in objects if x not in objSample]
    clusters = pam(objSample, k)
    return group(clusters, others)
```

A lista *others* conterà os objetos que estão em *objects* mas não estão em *objSample*, ou seja, todos aqueles que não foram selecionados para fazerem parte da amostra inicial. A função *group* irá associar cada objeto de *others* ao *cluster* cujo centro lhe for mais similar.

Vale ressaltar que a qualidade da solução do *CLARA* dependerá substancialmente da amostra escolhida no início, o que é feito aleatoriamente. Por

isso, quanto mais vezes o método *PAM* for executado, maior será a chance de obtermos um boa solução.

Em contrapartida, como já mencionado, o resultado do *PAM* também é aleatório pois, assim como o *k-means*, ele depende dos *k* centros escolhidos inicialmente. O método *CLARA*, por ter custo computacional bem menor, pode ser executado mais vezes. Com isso, foi verificado que, em geral, o *CLARA* encontrou resultados ainda melhores que o próprio *PAM*.

4.4 DBSCAN

O algoritmo *DBSCAN* é um exemplo de algoritmo baseado em densidade. Esses métodos são adequados para descobrirmos *clusters* de formas arbitrárias. A principal idéia dos métodos baseados em densidade é que para cada objeto de um cluster, sua vizinhança, para algum dado raio (*Eps*), tem que conter ao menos um número mínimo de objetos (*MinPts*). *Eps* e *MinPts* são parâmetros de entrada destes métodos. Abaixo, um código resumido do algoritmo *DBSCAN* (ver [5]).

```
def dbscan(objects, Eps, MinPts):
    id = 1
    for obj in objects:
        if unclassified(obj):
            expandCluster(obj, objects, Eps, MinPts, id)
            id += 1
```

O algoritmo supõe que, inicialmente, todos os objetos de *objects* estão não-classificados. A seguir, uma explicação da função *expandCluster()* utilizada em *dbscan()*.

```
expandCluster(obj, objects, Eps, MinPts, id):
    N recebe a lista com os vizinhos de obj num raio Eps
    se  $|N| < MinPts$ :
        Marcamos obj como um ruído
    senão:
        Cada objeto de N recebe o id atual
        para cada objeto em N (exceto obj):
            N2 recebe os vizinhos do objeto num raio Eps
            se  $|N2| \geq MinPts$ :
                Selecionamos todos os objetos ainda
                não classificados ou ruidos de N2;
                Adicionamos os não classificados em N1;
                Marcamos todos com o cluster-id atual
```

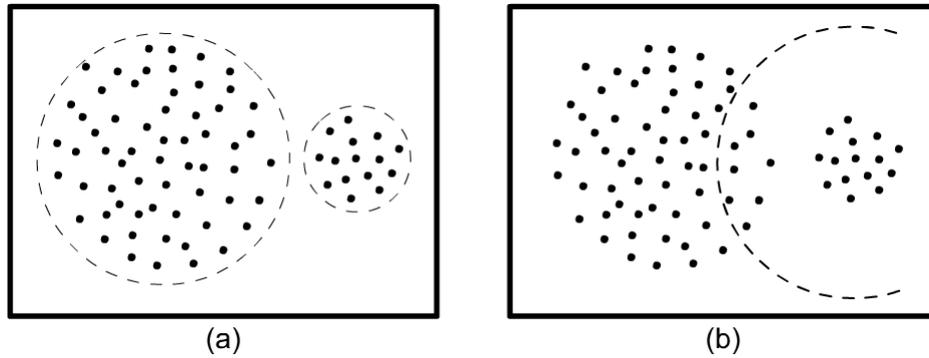


Figura 4.2: *Clustering* por Densidade (a) X *Clustering* por Partição (b)

A figura 4.2 mostra uma comparação entre o resultado de um *clustering* por particionamento e um por densidade, para uma mesma base de dados.

O método *DBSCAN* não se adequou bem ao nosso problema. Os parâmetros *Eps* e *MinPts* ideais variavam em diferentes regiões do Estado. Além disso, no nosso caso, não podemos desprezar os ruídos pois todas as chaves que tiverem ocorrências devem ser levadas em conta mesmo se estiverem isoladas geograficamente. Mesmo atribuindo os ruídos ao *cluster* mais próximo, não foram obtidos bons resultados. Por isso, ele não entrará em nossas tabelas de resultados computacionais e comparações.

Capítulo 5

Resultados Computacionais

5.1 Validação dos Resultados

Nosso projeto buscou a definição de locais ao longo do Estado do Espírito Santo, nos quais as equipes de campo responsáveis por atender as ocorrências de atendimento emergenciais da Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) devessem ficar posicionadas. Através de técnicas de *clustering*, foram obtidos alguns resultados para tal propósito.

Esse resultados, porém, precisavam ser validados. Como provar que os posicionamentos encontrados gerariam ganhos para a empresa?

Os retornos dos algoritmos foram, então, submetidos a uma etapa de validação desses resultados.

Primeiramente, optamos por executar o processo de *clustering* de duas formas: por *Regionais* e por *Todo Estado*. Existem 3 regionais atualmente no Estado, definidas pela Agência Nacional de Energia Elétrica (*ANEEL*): Norte, Sul e Centro. Cada uma delas possui uma quantidade de equipes de atendimento disponíveis.

Optar pelo processo de *clustering* por *Regionais* significa que será respeitada essa divisão na formação dos *clusters*, mantendo-se em cada uma delas a quantidade de equipes que cada uma possui. Caso contrário, escolher que o processo de *clustering* seja feito por *Todo Estado*, significa que a divisão das regionais será ignorada, e o algoritmo distribuirá todas as equipes pelo Estado da forma que achar conveniente.

Após escolhida uma das opções acima e terminado o processo de *clustering*, cada centro de cada grupo retornado pelo algoritmo passa a ser um dos locais onde as equipes devem se localizar e passamos, então, a simular o percurso das equipes de atendimento durante o período analisado.

Numa primeira abordagem, partindo desses locais, cada equipe seguirá

uma rota durante o dia atendendo as ocorrências pertencentes apenas a seu *cluster*, nunca invadindo a área de outro grupo. A cada intervalo de tempo, chamado de *Time Window* [6] ou *Janela de Tempo*, uma certa equipe atenderá as ocorrências surgidas no intervalo de tempo anterior em seu *cluster*. Se houver mais de uma ocorrência num dado intervalo de tempo, um simples algoritmo de roteamento [7, 8, 9, 10, 11, 12, 13] será usado para definir tal ordem de atendimento. Essa abordagem foi chamada, em nossas tabelas de comparação, de *Time Window*.

Numa outra abordagem, a partir do mesmo posicionamento inicial das equipes, uma equipe E_i atenderá a uma certa ocorrência, independentemente de esta última estar ou não no seu grupo, desde que tal ocorrência esteja mais perto de E_i do que de qualquer outra equipe no momento de seu surgimento. Se uma equipe tiver mais de uma ocorrência a atender ao mesmo tempo, não há roteamento: as ocorrências são atendidas na ordem que elas surgem. Essa abordagem foi chamada de *Método Guloso*.

Ao fim de cada dia do período analisado nas simulações, calculamos para cada conjunto o seu custo (DEC ver A.1) obtido, utilizando o posicionamento que fora sugerido.

Após todos os dias do período analisado, temos o resultado obtido do DEC de cada conjunto, e a soma de todos eles. Esses são os resultados que compararemos com os custos reais obtidos pela Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) no mesmo período, retirados do histórico fornecido pela empresa.

Nosso intuito é que, mesmo sem calcularmos o melhor roteamento possível para os veículos, o custo obtido ainda seja menor que o real. Isso porque nós queremos encontrar apenas o posicionamento das equipes, e não o direcionamento das mesmas.

5.2 *Time Window*

Todas as comparações que mostraremos a seguir foram feitas utilizando a primeira abordagem descrita anteriormente (*Time Window*). Optamos por executar cada algoritmo (*k-means*, *PAM* e *CLARA*) vinte vezes, sendo dez iterações utilizando a opção *Regionais* e as demais com a opção *Todo Estado*, para cada mês do histórico fornecido pela Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) (Setembro de 2003 a maio de 2004). Os resultados comparados foram a soma dos tempos de mobilização (TM, ver 2.3.1) gastos em cada conjunto num certo mês, em minutos.

5.2.1 Regionais

Os resultados obtidos optando pelo processo de *clustering* por *Regionais* estão na tabela 5.1

Mês	<i>k-means</i>	<i>PAM</i>	<i>CLARA</i>	Real
09/2003	185127	212136	205457	304598
10/2003	188403	218988	201994	572934
11/2003	314124	389564	326067	427747
12/2003	265182	327714	271028	1063008
01/2004	586287	693351	600244	1137527
02/2004	766086	776446	766185	917201
03/2004	524998	642584	551510	1192993
04/2004	842426	877365	864835	682415
05/2004	381503	391175	381032	318638

Tabela 5.1: Algoritmos - Regionais

Para melhor visualização desses resultados, a figura 5.1 mostra um gráfico extraído da tabela 5.1.

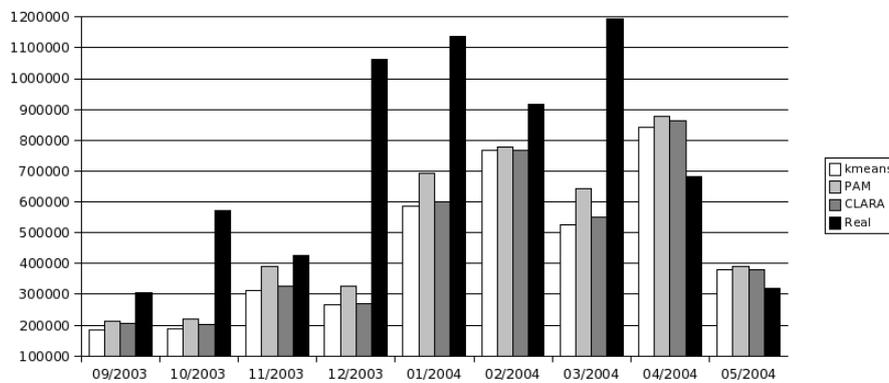


Figura 5.1: Algoritmos - Regionais

Pode-se notar que o método *k-means*, apesar de ser o mais simples, obteve os melhores resultados. Além disso, no último mês analisado, nossos resultados não foram melhores que os reais. Porém, nosso objetivo final é analisarmos o DEC, e não apenas o TM (que não leva em conta o número de consumidores atingidos por cada ocorrência de atendimento emergencial).

5.2.2 *Todo Estado*

Os resultados obtidos optando pelo processo de *clustering* por *Todo Estado* estão na tabela 5.2

Mês	<i>k-means</i>	<i>PAM</i>	<i>CLARA</i>	Real
09/2003	150440	177748	161187	304598
10/2003	151121	172919	165796	572934
11/2003	224742	246391	245303	427747
12/2003	207234	249161	215500	1063008
01/2004	489029	626551	508200	1137527
02/2004	540738	599038	568244	917201
03/2004	393966	476164	423771	1192993
04/2004	592588	723860	661722	682415
05/2004	249755	306063	274426	318638

Tabela 5.2: Algoritmos - Todo Estado

Para melhor visualização desses resultados, a figura 5.2 mostra um gráfico extraído da tabela 5.2.

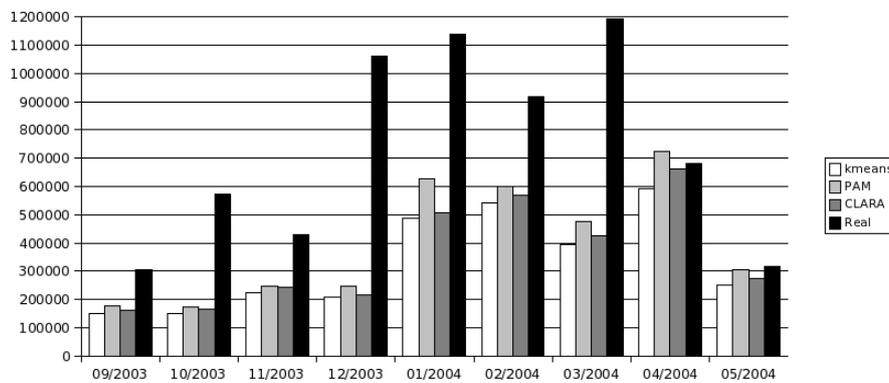


Figura 5.2: Algoritmos - Todo Estado

Novamente, o método *k-means*, apesar de ser o mais simples, obteve os melhores resultados. Agora, entretanto, nossos resultados no último mês analisado *foram* melhores que os reais.

5.3 Regionais versus Todo Estado

Fizemos também uma comparação entre *Regionais* e *Todo Estado*. A tabela 5.3 e a figura 5.3 mostram tais dados

Mês	<i>Regionais</i>	<i>Todo Estado</i>	Real
09/2003	185127	150440	304598
10/2003	188403	151121	572934
11/2003	314124	224742	427747
12/2003	265182	207234	1063008
01/2004	586287	489029	1137527
02/2004	766086	540738	917201
03/2004	524998	393966	1192993
04/2004	842426	592588	682415
05/2004	381032	249755	318638

Tabela 5.3: Comparação entre *Regionais* e *Todo Estado*

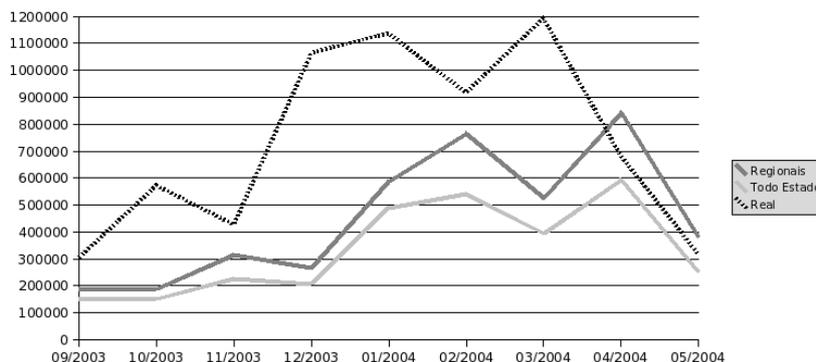


Figura 5.3: Comparação entre *Regionais* e *Todo Estado*

Os resultados mostram que são obtidos melhores resultados quando o sistema tem a liberdade de alocar as todas as equipes pelo Estado da forma que achar melhor, e não limitando certas quantidades de equipes entre as regionais. Esses resultados, no entanto, podem ser modificados quando compararmos o DEC ao invés de tempo de mobilização, já que o último não leva em conta o número de consumidores atingidos por cada ocorrência de atendimento emergencial. Vale lembrar também que os dados fornecidos pela empresa contêm muitos erros (alguns chegaram a ser corrigidos), e que a velocidade usada pelos veículos na simulação é muito baixa, 20km/h , o que é irreal, por exemplo, na região da Grande Vitória.

Capítulo 6

Conclusão

Como já explicado anteriormente, a Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) possui equipes de campo responsáveis por atender as ocorrências de atendimento emergenciais que surgem aleatoriamente ao longo do Estado. Este posicionamento não parece ter sido feito de uma forma satisfatória e, portanto, o objetivo deste projeto (ver capítulo 1) era propor um posicionamento para as equipes desta empresa de forma que o custo de elas percorrerem o Estado para atender tais ocorrências fosse mínimo.

Foram estudados vários algoritmos de *clustering* [1, 2, 3, 4] (ver capítulo 3) em busca dos que se adaptassem bem ao nosso problema. Dos algoritmos estudados (ver capítulo 4), 4 foram implementados e 3 deles (*k-means*, *PAM* e *CLARA*) corresponderam melhor ao nosso objetivo e fizeram parte de nossos testes comparativos.

Os resultados obtidos (ver capítulo 5) foram animadores. Com todos os algoritmos usados nas comparações, nosso custo gerado foi menor do que o obtido pela empresa. Vale lembrar que a velocidade média (ver 2.2) utilizada pelos veículos das equipes em nossas simulações (20km/h) foi muito baixa. Na região da Grande Vitória, por exemplo, a velocidade real é bem maior do que a adotada. Além disso, os dados fornecidos pela empresa continham muitos erros de medição e/ou de cadastramento absurdos¹.

Surpreendentemente, o algoritmo que melhor se comportou dentre os estudados foi o método *k-means*, o mais simples dos que foram implementados.

Entre os demais, o *CLARA* obteve resultados melhores que o *PAM*, apesar de que o primeiro é apenas o segundo rodado para uma pequena amostra

¹Exemplos: (1) O momento em que uma equipe terminava o atendimento de uma ocorrência era menor que o momento em que ela chegara ao local. (2) O momento em que uma equipe partia para o atendimento de uma ocorrência era menor que o momento do surgimento do mesmo. (3) A distância percorrida entre duas chaves era menor que a Euclidiana.

da base de dados. Isso foi conseguido porque o *CLARA* tem custo de processamento bem menor e conseguimos rodá-lo mais vezes. Como o resultado de todos é aleatório (dependem da escolha dos centros iniciais de cada grupo), quanto mais rodarmos um algoritmo, maior a chance de encontrarmos um bom resultado.

Nosso projeto, no entanto, não pode ainda ser considerado terminado, pois ainda não fizemos a comparação entre os nossos resultados e os reais em termos de DEC (ver 2.3.1 e A.1), que é nosso alvo maior. Isso porque os dados fornecidos pela Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) não estavam completos até o fechamento desta versão do trabalho. Só então, quando de posse de todos os dados necessários, poderemos determinar qual dos algoritmos se comportou realmente melhor em relação ao nosso problema, determinando assim, com mais coerência, os posicionamentos adequados para as equipes de campo de atendimento às ocorrências emergenciais da empresa.

Caso seja feita uma atualização posterior neste documento, a nova versão estará disponível em:

<http://www.inf.ufes.br/~hsjunior/projetofinal>

O apêndice A contém, ainda, as fórmulas utilizadas no projeto e citadas neste documento.

O apêndice B descreve a Interface Gráfica que foi implementada para uma melhor visualização dos resultados obtidos.

Por fim, o apêndice C trata da implementação do sistema.

Para maiores informações sobre este projeto, contate-me em *hsjunior at gmail.com*.

Apêndice A

Cálculos e Fórmulas

A.1 Cálculo do DEC

DEC - Duração Equivalente de Interrupção por Unidade Consumidora: Intervalo de tempo que, em média, no período de observação, em cada unidade consumidora do conjunto considerado ocorreu descontinuidade da distribuição de energia elétrica, expresso em horas e centésimos de hora, e calculado da seguinte forma:

$$DEC = \frac{\sum_{i=1}^k Ca(i) \cdot t(i)}{Cc}$$

Figura A.1: Fórmula de cálculo do DEC

Onde:

- i : Índice de eventos ocorridos no sistema que provocam interrupções em uma ou mais unidades consumidoras;
- $Ca(i)$: Número de unidades consumidoras interrompidas em um evento (i), no período de apuração;
- $t(i)$: Duração de cada evento (i), no período de apuração;
- k : Número máximo de eventos no período considerado;
- Cc : Número total de unidades consumidoras, do conjunto considerado, no final do período de apuração.

A.2 Cálculo do FEC

FEC - Frequência Equivalente de Interrupção por Unidade Consumidora: Número de interrupções ocorridas, em média, no período de observação, em cada unidade consumidora do conjunto considerado, expresso em números de interrupções e centésimos de números de interrupções, e calculado da seguinte forma:

$$FEC = \frac{\sum_{i=1}^k Ca(i)}{Cc}$$

Figura A.2: Fórmula de cálculo do FEC

Onde:

- i : Índice de eventos ocorridos no sistema que provocam interrupções em uma ou mais unidades consumidoras;
- $Ca(i)$: Número de unidades consumidoras interrompidas em um evento (i), no período de apuração;
- k : Número máximo de eventos no período considerado;
- Cc : Número total de unidades consumidoras, do conjunto considerado, no final do período de apuração.

Apêndice B

Interface Gráfica

Como implementamos vários algoritmos de *clustering* em busca de um que se adaptasse melhor ao nosso problema, fica difícil para um usuário do sistema saber como controlar tantos algoritmos. Além disso, nossos resultados, por lidarmos com uma grande base de dados, são difíceis de serem visualizados. Seria interessante que as regiões formadas nos métodos de *clustering* e o posicionamento sugerido para as equipes pudessem ser traduzidos para uma linguagem visual, simples de ser entendida. Devido a tais dificuldades, foi feita uma Interface Gráfica para o projeto, que será mostrada neste capítulo.

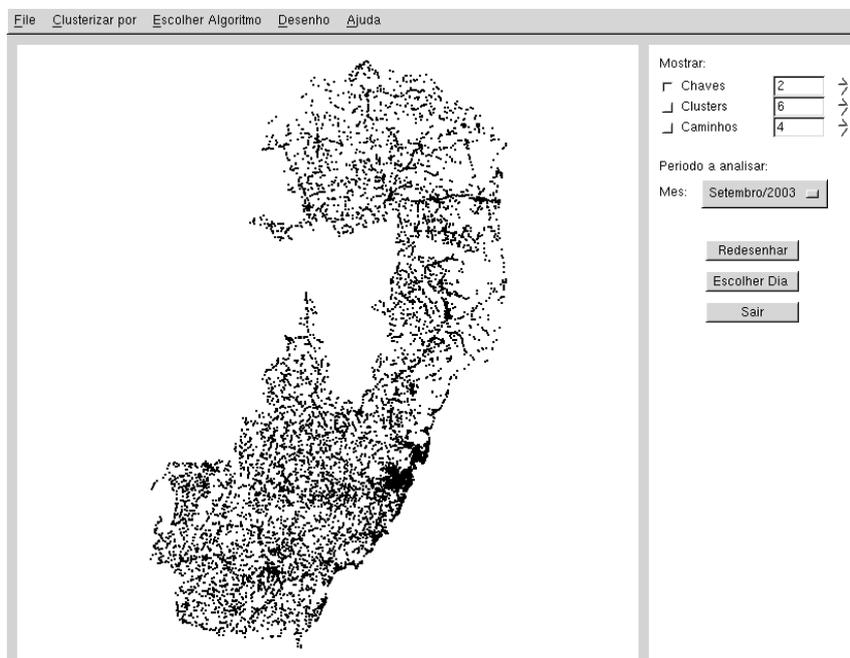


Figura B.1: Tela inicial da Interface

B.1 Tela Inicial

Ao abrirmos a nossa Interface Gráfica, já visualizamos todas as chaves do Estado. Os posicionamentos geográficos dessas chaves foram obtidos pela Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) através de GPS (*Global Positioning System*) e, a partir destes posicionamentos, pode-se observar nitidamente que o conjunto de todas as chaves apresenta o formato do mapa do Espírito Santo (figura B.1).

Através do *menu* pode-se chamar uma janela de ajuda, com informações básicas sobre como utilizar a Interface. Ainda a partir do *menu*, pode-se optar por escolher em qual período será feito o processo de *clustering* e qual algoritmo será utilizado para tal. Além disso, o painel onde as chaves são mostradas pode ser movido, ampliado ou reduzido (*Zoom In*, *Zoom Out*) e pode-se obter os dados¹ de uma determinada chave ao clicar-se sobre ela. Um exemplo do painel ampliado pode ser visto na figura B.2.

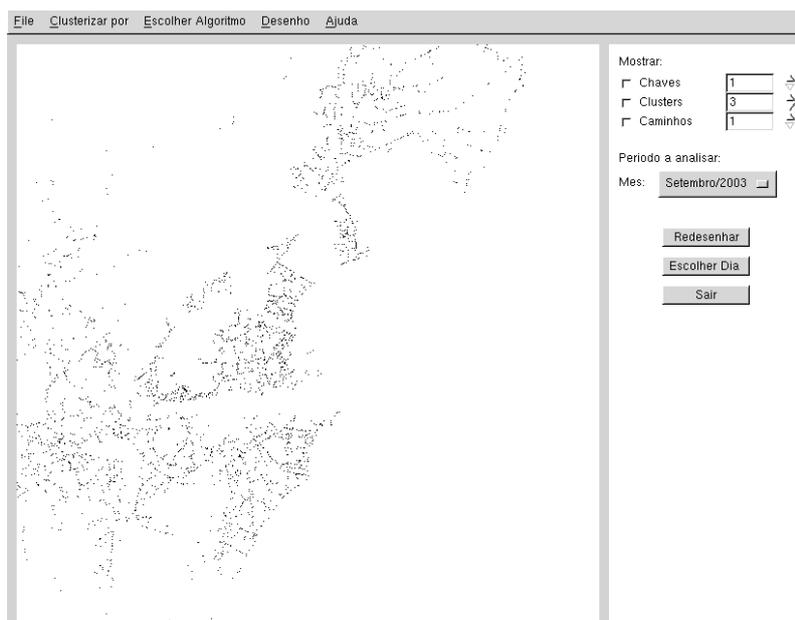


Figura B.2: Exemplo de Ampliação do Painel

Outra opção é escolher se o processo de *clustering* será feito pelas *Regionais* ou por *Todo o Estado*. Existem 3 regionais atualmente no Estado, definidas pela Agência Nacional de Energia Elétrica (*ANEEL*): Norte, Sul e Centro. Cada uma delas possui uma quantidade de equipes de atendimento disponíveis. Optar pelo processo de *clustering* por *Regionais* significa

¹Posição geográfica, conjunto ao qual ela pertence, etc.

que será respeitada essa divisão na formação dos *clusters*, mantendo-se em cada uma delas a quantidade de equipes que cada uma delas possui. Optar por *Todo o Estado* significa que essa divisão será ignorada, e o algoritmo distribuirá todas essas equipes pelo Estado da forma que achar conveniente.

Outra opção é escolher o número de equipes em cada regional ou, analogamente, em todo o estado. Caso não seja feita essa escolha, será adotado um número de equipes que foi definido através de uma média da quantidade de equipes presentes ao longo do dia pelo Estado, em cada regional, obtido do histórico fornecido pela Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*). Na opção *Todo o Estado*, utiliza-se a soma do número de equipes das regionais.

O próximo passo é, então, optar por um algoritmo de *clustering*.

B.2 Resultado do *clustering*

Escolhido o algoritmo de *clustering*, o resultado do processo de agrupamento é mostrado como no figura B.3. As chaves vizinhas mostradas com uma mesma cor foram incluídas num mesmo *cluster*. Analogamente, as de cores diferentes foram atribuídas a grupos distintos. O centro de cada grupo é onde cada equipe deve ser posicionada.

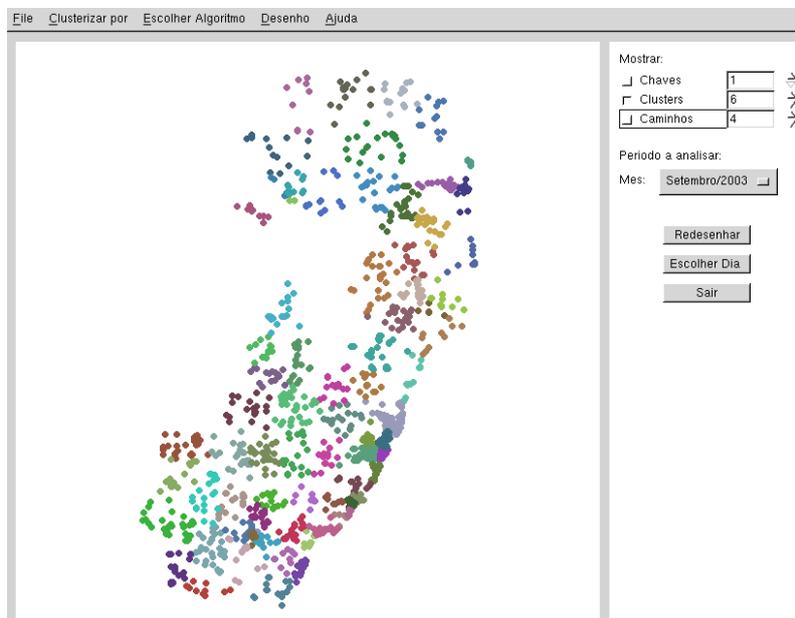


Figura B.3: Resultado do *clustering*

B.3 O percurso das equipes

Como explicamos em 5.1, para calcularmos qual seria o DEC gerado se nosso posicionamento sugerido fosse utilizado, simulamos o caminho que as equipes teriam percorrido com tais configurações. A Interface mostra, além do que já foi citado, esse caminho feito pelas equipes de atendimento da Companhia de Energia Elétrica do Espírito Santo (*ESCELSA*) num certo dia do período analisado. Após escolhido um certo dia, o caminho é mostrado como na figura B.4. Na figura, o painel foi ampliado para visualizarmos apenas o percurso de uma certa equipe durante todo o dia escolhido.

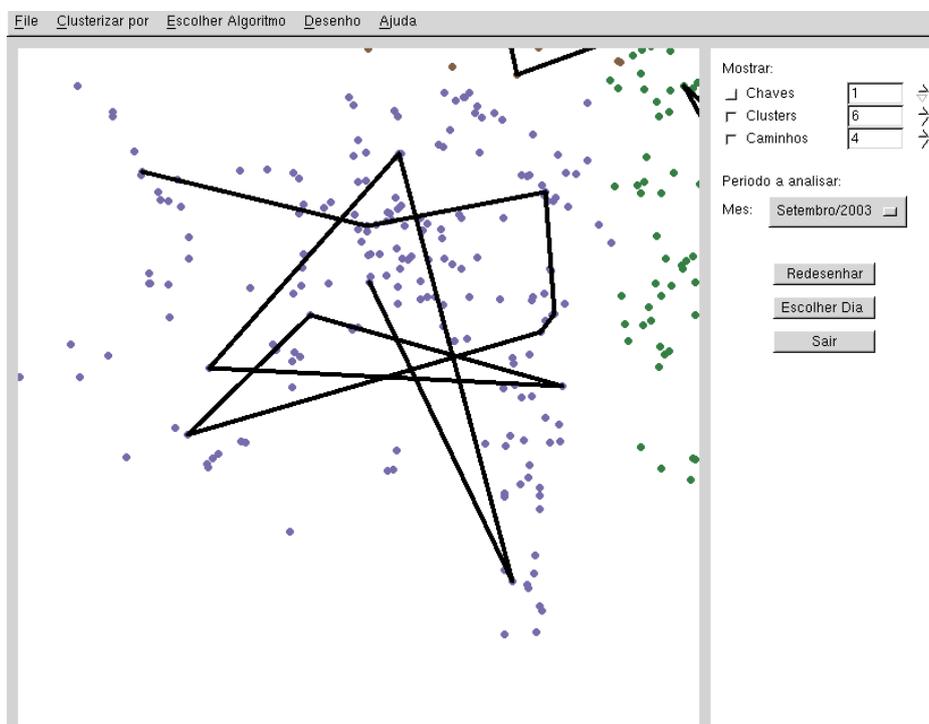


Figura B.4: Caminho percorrido por uma equipe num certo dia

Apêndice C

Implementação

Os códigos completos dos programas implementados (todos em Python) não foram incluídos neste documento por pura questão de espaço, mas estarão disponíveis em:

<http://www.inf.ufes.br/~hsjunior/projetofinal>

Neste endereço pode ser encontrada, também, a versão original deste documento. Para que os programas sejam executados, é necessário que a máquina tenha instalados:

- *Python* 2.3 ou superior;
- *wxPython* 2.5 ou superior.

C.1 Por que Python?

Python é uma linguagem de programação interpretada e interativa bem alto nível que se caracteriza pela clareza e simplicidade de expressão. Ela permite que programadores construam suas soluções computacionais usando modos humanos de pensamento e lógica. Além de simples e fácil de usar e aprender, possui uma extensa biblioteca nativa e é portátil.

Python foi originalmente criada por Guido van Rossum em 1989, como um meio de ensinar a não-programadores os mais poderosos e sofisticados conceitos de programação de computadores. Atualmente, é mantida pela “Python Software Foundation”, liderada por Guido van Rossum, com contribuições de voluntários de todo o mundo.

Mas por que utilizar Python? O misto de qualidades desta linguagem não é alcançado por qualquer outra linguagem. Ela combina flexibilidade com

graça, lógica com clareza, velocidade de desenvolvimento com facilidade de manutenção, etc. Python é um linguagem híbrida, que aceita paradigmas de programação procedurais, funcionais e orientados a objetos, oferecendo aos programadores a opção de usar a abordagem certa para a construção de cada parte de sua solução.

Python roda em quase todas as plataformas, de *palm-tops* a *mainframes*. Suporta todos os principais sistemas operacionais, incluindo quase todas as variações de *UNIX*, *Windows* e *Mac OS*. Ela é facilmente integrada com C, C++ e Java e, o melhor de tudo, é divertida de usar.

Além disso tudo, Python é liberada sob uma licença totalmente *Open Source* (código aberto), o que permite que o usuário use, modifique e até a revenda no contexto de um projeto ou uma aplicação proprietária.

A descrição acima foi inspirada no texto original contido em:

<http://pbf.strakt.com/python>

Para obter mais informações sobre a linguagem, visite:

<http://www.python.org>

C.2 Implementação da Interface

A linguagem de programação Python possui um conjunto de ferramentas para Interface Gráfica chamado *wxPython*. Ele permite que programadores de Python criem programas com uma interface gráfica robusta e muito funcional, de uma forma simples e fácil. Por usar a linguagem Python, *wxPython* é bem fácil de escrever e de entender. Para implementarmos a interface do problema, mostrada em B, utilizamos *wxPython*.

Como Python, *wxPython* é *Open Source* (código aberto), o que significa que é livre para que qualquer um use, e seu código é disponível para que qualquer um veja, modifique e até contribua com melhoramentos para o projeto.

Além disso, um mesmo programa feito em *wxPython* rodará em várias plataformas sem que qualquer modificação necessite ser feita. Atualmente, as plataformas suportadas são:

- A maioria dos sistemas *Unix* e seus similares;
- *Macintosh OS X*;
- *Microsoft Windows 32-bit*

Referências Bibliográficas

- [1] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. 9, 13, 14, 19, 30
- [2] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, citeseer.ist.psu.edu/guha98cure.html, 1998. ACM. 9, 13, 30
- [3] Charles Jay Alpert and Andrew B. Kahng. *Multi-way graph and hypergraph partitioning*. PhD thesis, 1996. 9, 13, 15, 30
- [4] Lando Mendonça di Carlantonio. Novas metodologias para clusterização de dados. Master’s thesis, COPPE/UFRJ, Coordenação de Programas de Pós-Graduação em Engenharia, Aug 2001. 9, 15, 16, 19, 20, 21, 22, 30
- [5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 323–333, 24–27 1998. 23
- [6] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macsvrptw: A multiple colony system for vehicle routing problems with time windows. 26
- [7] D. S. Johnson and L. A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. Draft of November 20, 1995. To appear as a chapter in the book *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, New York., 1995. 26
- [8] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial optimization*. Wiley, 1998. Cook. 26

- [9] Nicos Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem, 1976. 26
- [10] Chi lok Chan and Gilbert H. Young. Single-vehicle scheduling problem on a straight line with time window constraints. In *Computing and Combinatorics*, pages 617–626, 1995. 26
- [11] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter. the capacitated vehicle routing problem. 26
- [12] S. Krumke, J. Rambau, and L. Torres. Real-time dispatching of guided and unguided automobile service units with soft wime windows, 2001. 26
- [13] M. Gendreau and J. Potvin. Dynamic vehicle routing and dispatching, 1998. 26