

ESCALONAMENTO DE EQUIPES DE CAMPO DA ESCELSA RELATÓRIO PARCIAL

RAUL H.C. LOPES

1. INTRODUÇÃO

Este relatório parcial apresenta os componentes do *SEECE*, descrevendo as funcionalidades por eles apresentadas. São introduzidas três classes de funcionalidades: configuração do sistema, localização de equipes e simulação de despacho. A figura 1 apresenta um diagrama das funcionalidades do sistema.

2. CONFIGURAÇÃO DO SISTEMA

O *SEECE* admite dois tipos básicos de funções de configuração:

- configuração geral de bases de dados, discutidas nesta seção;
- configurações locais dos componentes de localização de equipes e simulação de despachos, que dizem respeito fundamentalmente a intervalos de tempo e número de equipes consideradas.

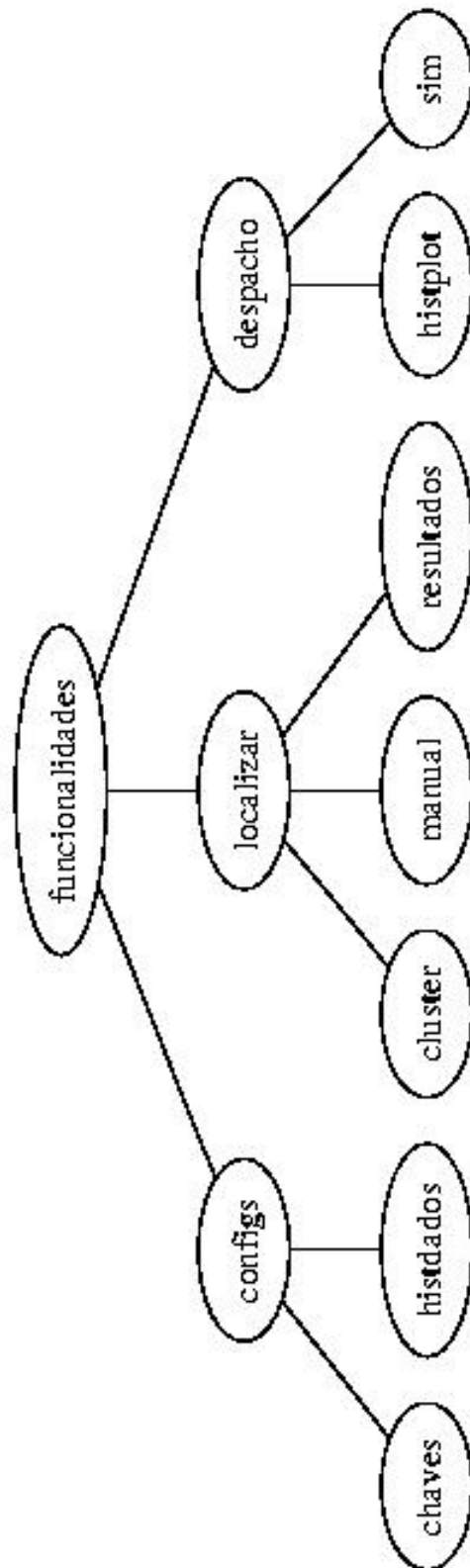
O componente de configuração do sistema, em particular, é ativado na configuração de início de execução ou diretamente via menu. Ele define a base de dados sobre a qual o sistema trabalhará. Essa base de dados permite definir:

- configuração de chaves, nó *chaves* da figura 1;
- dados históricos em consideração, nó *hisdados* da figura 1.

Por configuração de chaves, entende-se a divisão do estado em regiões, que são divididas elas mesmas em centrais, que, por sua vez, são conjuntos de chaves. Cada chave tem sua coordenada geográfica, sendo o conjunto completo de chaves apresentado em arquivo texto, contendo uma chave por linha composta de:

- identificação da chave;
- coordenadas.

Um arquivo texto também é usado para definir a regionalização do estado, apresentando uma linha por chave, composta de campos com identificações da chave, do conjunto, da central e da região.



A configuração de dados históricos de ocorrência permite selecionar as amostras de dados de registros de ocorrências e atendimentos que serão usadas pelos algoritmos de simulação. Esses dados definem:

- o período de tempo coberto pelos dados;
- cada ocorrência registrada com respectivos:
 - data e hora de registro;
 - equipe de atendimento;
 - data e hora de início de deslocamento da equipe e chegada ao local;
 - tempo do reparo;
 - conjuntos e consumidores afetados.

Escolhidas a configuração de chaves e o histórico de ocorrências, todas as operações de localização de equipes e simulação serão realizadas em relação a essa base específica.

3. COMPONENTE DE LOCALIZAÇÃO DE EQUIPES

A partir desta seção, chamar-se-á *base de equipes* ao local (chave) onde as equipes de atendimento ficarão estacionadas e de onde partirão ao início de um turno de trabalho.

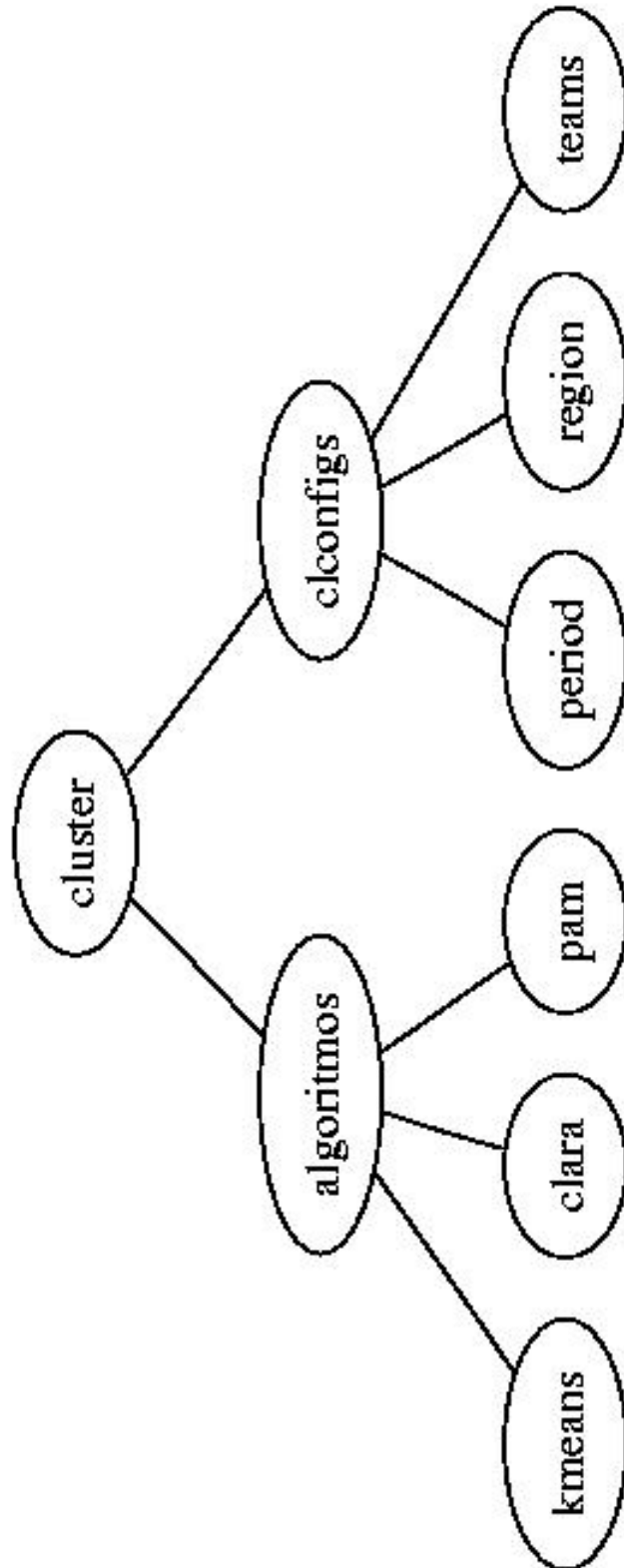
O objetivo final do componente de localização de equipes, nó *cluster* da figura 2, consiste em oferecer alternativas para o cálculo das bases onde residirão as equipes de atendimento. Esse cálculo pode ser feito via algoritmo, através de um conjunto de algoritmos heurísticos baseados em *clustering*, ou de forma interativa, pela alteração de alguma configuração de bases previamente calculada.

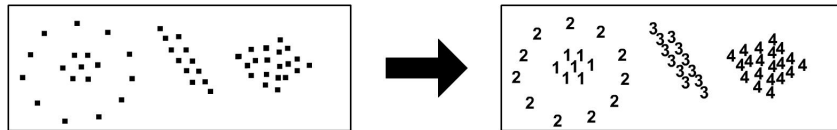
3.1. Via algoritmo. O cálculo do posicionamento de bases via algoritmo consiste em usar dados históricos de ocorrências e as posições geográficas de chaves para calcular a melhor distribuição de equipes de atendimento para uma região. Os algoritmos de cálculo usam os seguintes dados como entrada:

- base previamente configurada, ver seção 2, definindo configuração de chaves (coordenadas, conjuntos, regiões);
- base de dados históricos previamente definidos, ver seção 2;
- número de equipes a utilizar, dado fornecido pelo usuário;
- período de tempo dentro do histórico a considerar.

Em adição o usuário escolhe um algoritmo para calcular a melhor distribuição de equipes.

3.2. Algoritmos para distribuição de equipes. O problema de distribuição de equipes pelo estado vem sendo identificado neste projeto




 FIGURA 3. Um exemplo de *clustering*

como o problema de se agrupar os possíveis focos de problema e associar uma ou mais equipes de atendimento a cada grupo reduzindo assim o tempo de deslocamento de cada equipe e sua implementação realiza o componente *cluster* do diagrama 1.

Uma das técnicas clássicas de agrupamento é identificada na literatura [5, 3] como *clustering*: a classificação não-supervisionada de padrões (observações, itens de dados ou vetores de características) em grupos (*clusters*)” [6].

Seja o exemplo de agrupar os pontos mais similares entre si da primeira parte da figura 3: na segunda parte da figura, cada ponto foi substituído pelo número do *cluster* a que ele pertence, indentificado pelo processo 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 características que os diferenciem. Segundo Guha *et al.* [4], 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. É importante salientar que, quando a função de similaridade é dada pela distância Euclidiana entre os pontos, freqüentemente soluções da geometria computacional, como diagramas de Voronoi e triangulação revelam-se mais precisos e eficientes.

3.2.1. *Agrupamento e distribuição de equipes.* No problema estudado os objetos a serem agrupados serão as chaves da empresa, 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 sugerir os novos posicionamentos das equipes, diversos algoritmos de *clustering*, correspondentes aos nós *kmeans*, *clara* e *pam* do diagrama 1, foram implementados e comparados com o objetivo de determinar os que melhor se adaptam ao problema. A validação das distribuições dos algoritmos é feita por simulações de despachos de equipes com dados históricos da empresa.

Uma definição formal do problema de *clustering* é encontrada em [1, 2] 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.

Os algoritmos conhecidos de *clustering* atendem a diferentes tipos de requisitos, tais como [2]:

- encontrar ou não um número adequado de *clusters* (alguns algoritmos precisam de um valor previamente determinado 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.

3.2.2. *k-means.* O algoritmo *k-means* [5] é o mais conhecido método por particionamento. Nesse algoritmo, os k centros iniciais são escolhidos aleatoriamente. No final de cada iteração, esses centros passam a

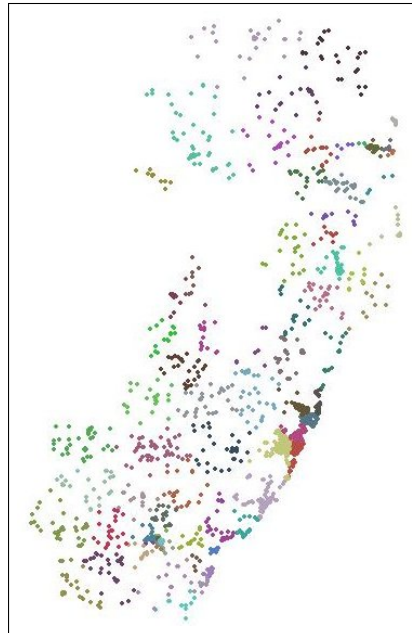


FIGURA 4. Um resultado do *kmeans*

ser o centro de gravidade de cada *cluster* (média dos pontos do grupo). Abaixo, apresenta-se um pseudo-código do algoritmo.

```
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 é 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 *kmeans* 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* [2], normalmente $n \gg k$ e $n \gg t$.

O fato de o algoritmo *kmeans* exigir que definamos inicialmente quantos grupos serão formados não foi prejudicial no nosso problema. Isso porque utilizamos k = número de equipes de atendimento que a empresa 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*.

Uma variação deste método utilizada foi: ao recalcularmos os novos centros a cada iteração, foi utilizado em cada grupo o objeto mais próximo do seu centro de gravidade, ao invés do próprio centro de gravidade. Essa alteração não representou muita variação no formato dos *clusters* obtidos em relação ao algoritmo original.

Na figura 4, vemos um resultado do algoritmo *kmeans* com a adaptação descrita acima. As chaves de mesma cor foram agrupadas em um mesmo *cluster*. Foram gerados 75 grupos, que é uma média da quantidade de equipes da empresa no Estado.

Vale lembrar que o *kmeans* 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.

3.2.3. PAM. *PAM* também é um método por partição. *Carlantonio* [2] explica o algoritmo: depois de uma seleção aleatória inicial de k medoids, o algoritmo repetidamente tenta fazer a melhor escolha de medoids. Todos os pares possíveis de objetos são analisados, onde um objeto em cada par é considerado um medoid e o outro um não-medoid. A qualidade do agrupamento resultante é calculada para cada uma de tais combinações. Um objeto, O_j , é substituído pelo objeto que causa a maior redução no erro-quadrado. O conjunto dos melhores objetos para cada cluster em uma iteração forma os medoids para a próxima iteração. Para valores muito grandes de n e k , tal computação torna-se muito custosa. Segue um pseudo-código 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 diferença entre ele e o *kmeans* é a forma como os novos centros são escolhidos. Em *reCalcMedoids()*, é calculado o novo centros de cada *cluster* como sendo o primeiro que tiver custo total menor que o custo do centro atual. Custo total de um ponto é a soma de todas as distâncias entre ele e os demais pontos do *cluster*.

3.2.4. *DBSCAN*. O algoritmo *DBSCAN* é um exemplo de algoritmo por densidade. Esses métodos são adequados para descobrirmos agrupamentos de formas arbitrárias. *Carlantonio* [2] esclarece que a idéia chave 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 (*implementado em Python*).

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

O algoritmo supõe que, inicialmente, todos os objetos de *objects* estão não-classificados. Abaixo, segue a função *expandCluster()* utilizada em *dbscan()*.

```
def expandCluster(self, objects, Eps, MinPts, id):
    N1 = self.epsNeighbourhood(Eps)
    if len(N1) < MinPts:
        self.isNoise()
        return
    else:
        setID(N1,id)
        N1.remove(self)
        for o in N1:
            N2 = o.epsNeighbourhood(Eps)
            if len(N2) ≥ MinPts:
                # Seleccionamos todos os objetos ainda
                # não classificados ou ruidos;
                # adicionamos os não classificados em N1;
                # marcamos todos com o cluster-id atual
        return
```

A função *epsNeighbourhood(Eps)* retorna uma lista com todos os objetos da vizinhança *Eps* de um objeto. A função *isNoise()* marca um

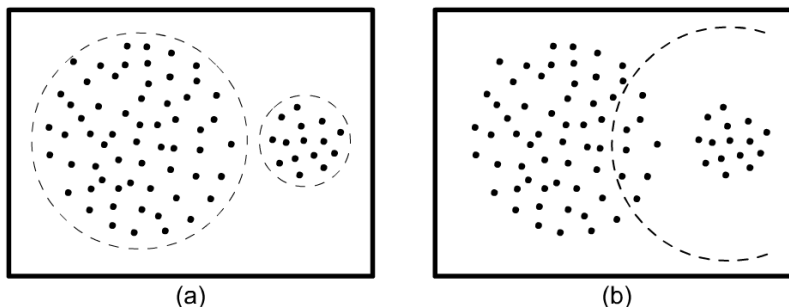


FIGURA 5. Agrupamento por Densidade (a) X Agrupamento por Partição (b)

objeto como sendo um ruído. A função $setID(N, id)$ atribui a todos o objetos da lista N um certo id . A figura 5 mostra uma comparação entre o resultado de um *clustering* por particionamento e um por densidade, para uma mesma base de dados.

3.3. Interativa. Uma alternativa ao cálculo automático de bases de equipes consiste em usar uma distribuição previamente calculada e permitir que o usuário defina de forma interativa uma nova posição para uma ou mais bases, implementando a funcionalidade *manual* do diagrama 2.

3.4. O resultado. No cálculo de posicionamento de equipes, seja via algoritmo ou interativamente, o resultado produzido pelo sistema consiste em:

- definição de agrupamentos de chaves;
- identificação, para cada agrupamento de chaves, da base das equipes que atenderão a tal agrupamento.

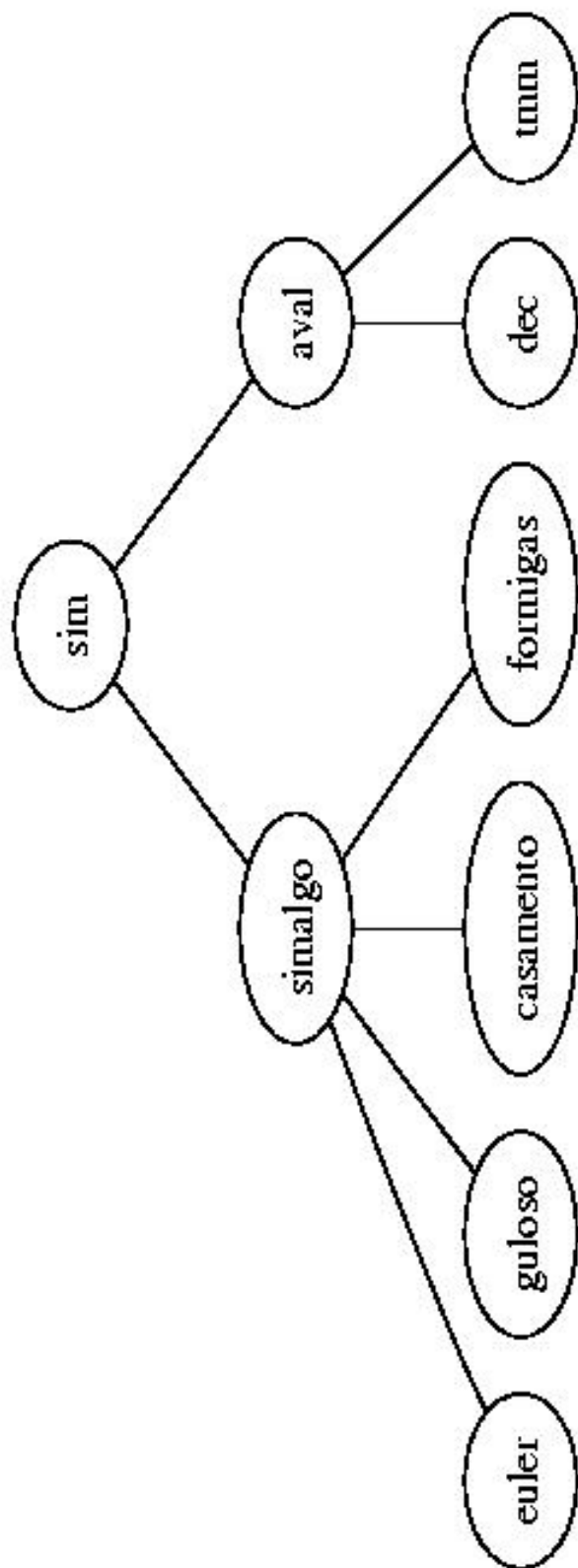
Esses resultados, que correspondem ao componente *resultados* do diagrama 2, podem ser visualizados via:

- interface gráfica que exibe o mapa do estado com todos os agrupamentos e suas respectivas base;
- relatórios definindo as chaves dos agrupamentos e suas bases.

4. COMPONENTE DE DESPACHO DE EQUIPES

Por despacho de equipes entende-se o processo que cobre:

- identificação do local da ocorrência;
- definição de equipe de atendimento a ser deslocada para o local.



Os objetivos do componente de despachos, nó *sim* do diagrama 6, consistem em permitir a avaliação de qualidade de distribuição de equipes via simulação de atendimentos de emergência e a visualização e avaliação de despacho reais.

4.1. Validação via despacho. A validação dos algoritmos de distribuição de equipes no presente estágio é feita através de duas famílias de algoritmos: algoritmos de aproximação, que oferecem garantias com respeito à qualidade dos priores despacho e algoritmos herísticos que tentam rapidamente apresentar soluções para o problema.

Um algoritmo de simulação de despacho objetiva identificar o cálculo realizado por um operador ao receber uma notificação de uma ocorrência e avaliar a qualidade desse despacho em termos de:

- tempo de deslocamento da equipe alocada para o atendimento;
- tempo de desconexão dos consumidores afetados.

O tempo de deslocamento da equipe alocada é calculado de forma aproximada, usando:

- distância entre as chaves do deslocamento da equipe;
- velocidade aproximada do deslocamento, que calculada a partir de dados históricos da empresa.

O tempo de desconexão dos consumidores afetados é dado pelo tempo do deslocamento calculado para a simulação em questão somado ao tempo real de reparo obtido dos dados históricos da ocorrência.

O período da simulação é definida interativamente pelo usuário. Também são definidos por ele: os dados históricos a considerar e a distribuição de equipes.

Os seguintes algoritmos já foram implementados para avaliação de qualidade simulação:

- **TimeWindow:** deslocamento total das equipes de atendimento usando distribuição de equipes proposta por algoritmo de agrupamento e algoritmo de despacho por aproximação de **TSP**, correspondente ao nó *euler* do diagrama 6.
- **Guloso:** deslocamento total das equipes de atendimento usando distribuição de equipes proposta por algoritmo de agrupamento e algoritmo de despacho, que simula o trabalho de um operador, despachando a equipe mais próxima da ocorrência, nó *guloso* da figura 6.

4.2. Visualização de despachos. O caminho percorrido em um atendimento é abstraído no *SEECE* como um segmento ligando a chave em que a equipe se encontrava antes do despacho à chave onde o atendimento ocorre.

O componente de visualização de despachos, nó *histplot* do diagrama 6, permite exibir um mapa do Estado com todos os despachos de equipes ocorridos em um intervalo de tempo, incluindo os caminhos percorridos pelas respectivas equipes. Os caminhos mostrados em tal visualização são formados pelos segmentos ligando as chaves de origem e destino do deslocamento da equipe. Como em toda a interface gráfica do sistema existe um mecanismo de *zoom* que permite uma visualização mais acurada de uma porção restrita do espaço geográfico em consideração.

4.3. Avaliação de qualidade de despacho. A qualidade dos despachos, reais ou simulados, pode ser avaliada através de relatórios sobre o desempenho do atendimento em relação aos índices de **DEC**, nó *dec*, e **TMM**, nó *tmm* do diagrama 6.

A figura 7 mostra os dados de deslocamentos obtidos usando o histórico de ocorrências da Escelsa de setembro de 2003 a maio de 2004. A tabela mostra três colunas:

- **TimeWindow:** deslocamento total das equipes de atendimento usando distribuição de equipes proposta por algoritmo de agrupamento *kmeans* e algoritmo de despacho por aproximação de **TSP**.
- **Guloso:** deslocamento total das equipes de atendimento usando distribuição de equipes proposta por algoritmo de agrupamento *kmeans* e algoritmo de despacho, que simula o trabalho de um operador, despachando a equipe mais próxima da ocorrência.
- **Real:** dados reais de deslocamento, obtidos do histórico da empresa.

Na figura 8 são apresentados apenas dados de simulação referentes a março de 2004. São comparadas as somas de **TMM**, usando o algoritmo de simulação **TimeWindow**. São comparadas simulações realizadas com os algoritmos **Clara** (variação do algoritmo **PAM** descrito acima) e **K-means**. Em cada caso, foram realizadas distribuições de equipes considerando que o Estado tem uma única região (barra azul), e considerando que o Estado apresenta as divisões atuais da Escelsa (barra vinho.) A barra amarela representa os dados reais.

A figura 9 apresenta comparações de **DEC** obtido para o mesmo mês: março de 2004. As simulações seguem o mesmo padrão das apresentadas no caso do **TMM**, acima.

O resumo dos dados de **DEC** segue na tabela 1.

Podem ser obtidos gráficos e tabelas similares para os índices de **DEC** e **TMM** de qualquer período coberto pelos dados históricos apresentados no processo de configuração do sistema, ver seção 2.

| Mês | Médias | | |
|---------|------------|--------|---------|
| | TimeWindow | Guloso | Real |
| 09/2003 | 151965 | 196589 | 304598 |
| 10/2003 | 257025 | 318542 | 572934 |
| 11/2003 | 207189 | 224749 | 427747 |
| 12/2003 | 569689 | 567475 | 1063008 |
| 01/2004 | 494560 | 690378 | 1137527 |
| 02/2004 | 456469 | 446799 | 917201 |
| 03/2004 | 781769 | 967730 | 1192993 |
| 04/2004 | 281364 | 287498 | 682415 |
| 05/2004 | 182790 | 181683 | 318638 |

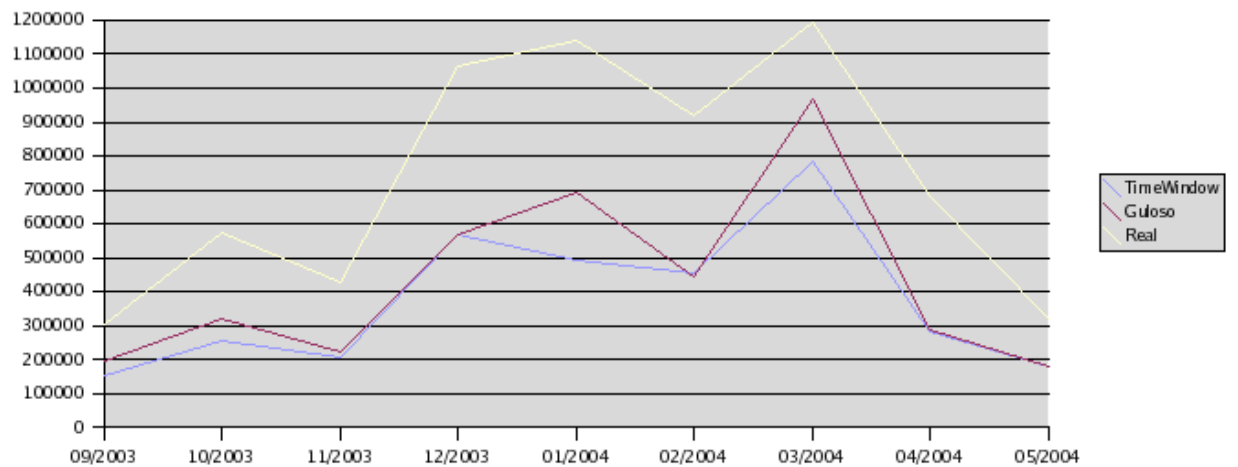


FIGURA 7. Comparação de TMM mês a mês

| | Regional | Estado | Real |
|----------------|----------|--------|--------|
| K-means | 157.74 | 143.07 | 183.63 |
| Clara | 127.96 | 169.48 | 183.63 |

TABELA 1. Soma de **DEC** para março de 2004

5. PRÓXIMOS PASSOS

No estágio atual, estão sendo testados dois novos algoritmos para o componente de despachos de equipes:

- simulação de despachos via teoria de casamentos estáveis;
- simulação de despachos via meta-heurística de colônia de formigas.

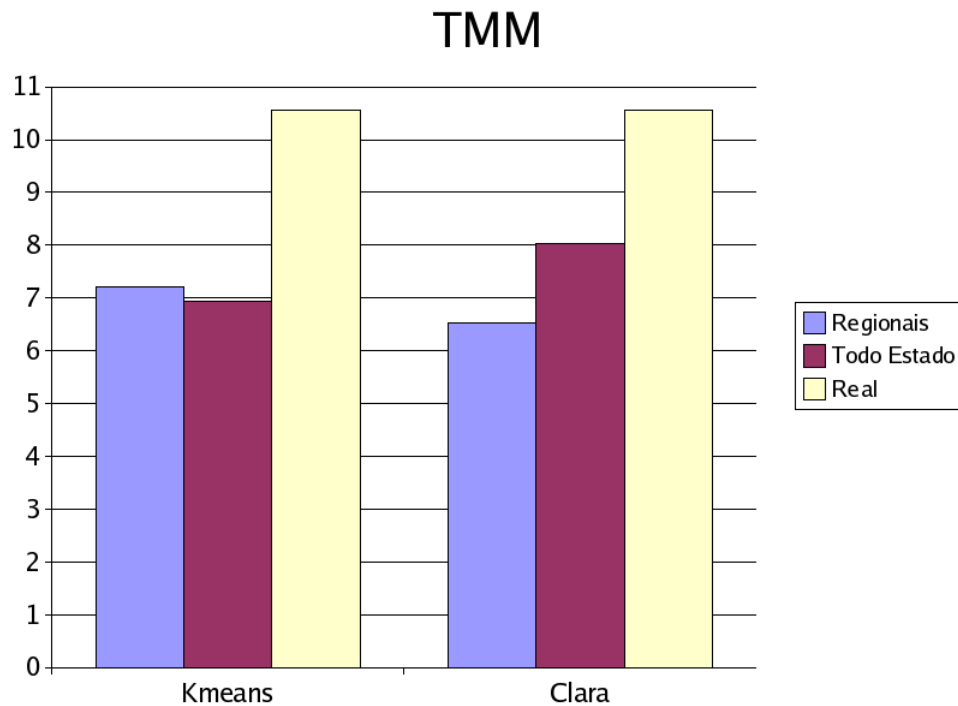


FIGURA 8. Comparação de **TMM** para março de 2004

Além disso, o componente de configuração do sistema está sendo aprimorado para permitir a definição de escalas de atendimento.

REFERÊNCIAS

1. Charles Jay Alpert, *Multi-way graph and hypergraph partitioning*, Ph.D. thesis, University of California, 1996.
2. Lando Mendonça di Carlantonio, *Novas metodologias para clusterização de dados*, Master's thesis, COPPE/UFRJ, 2001.
3. Daniel Fasulo, *Ana analysis of recent work on clustering algorithms*, Tech. Report 01-03-02, Department of Computer Science, University of Washington, 1999.
4. Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim, *CURE: an efficient clustering algorithm for large databases*, In Proceedings of ACM SIGMOD International Conference on Management of Data, ACM, 1998, pp. 73–84.
5. John Hartigan, *Clustering algorithms*, JOHN WILEY & SONS, 1975.
6. A. K. Jain, M. N. Murty, and P. J. Flynn, *Data clustering: a review*, ACM Computing Surveys **31** (1999), no. 3, 264–323.

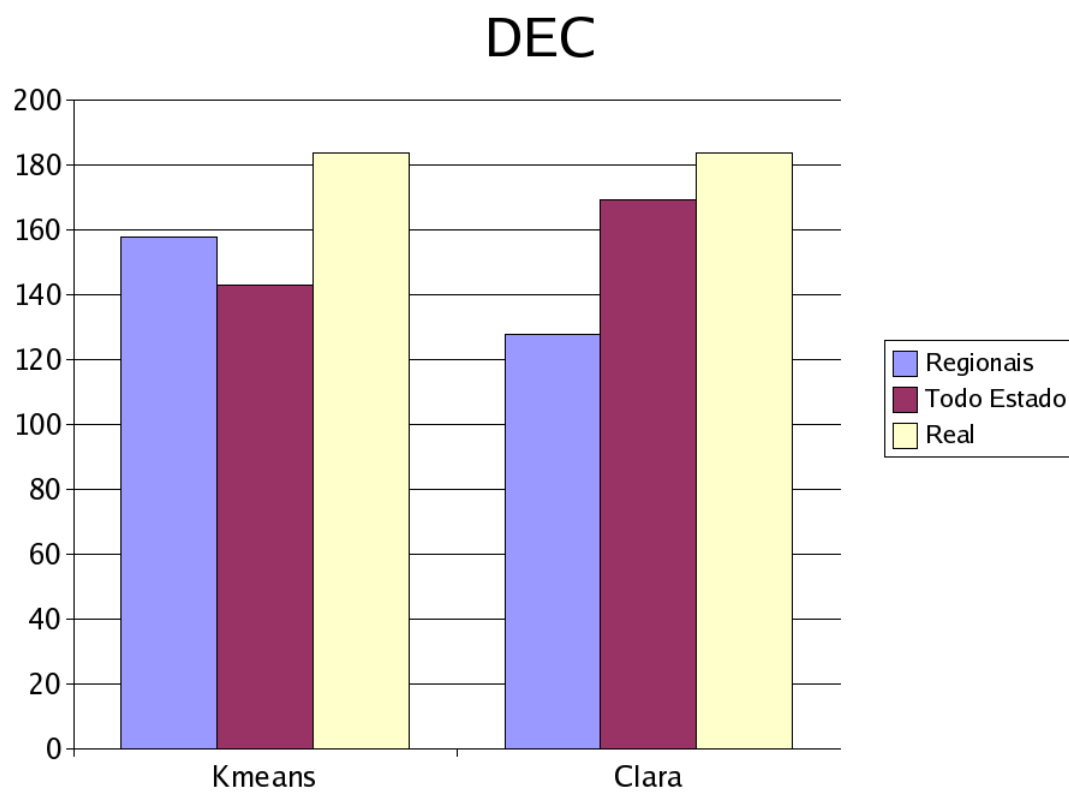


FIGURA 9. Comparação de **DEC** para março de 2004