

ESCALONAMENTO DE EQUIPES DE CAMPO DA ESCELSA

RAUL H.C. LOPES

1. INTRODUÇÃO

Este documento apresenta uma proposta inicial de arquitetura para o desenvolvimento do *Sistema de Escalonamento Equipes de Campo da ESCELSA*, de agora em diante denominado **SEECE**. É pressuposto básico que o sistema computacional a ser desenvolvido deverá possuir ao menos duas funcionalidades:

- propor a distribuição de equipes a partir de dados históricos de atendimentos, incluindo:
 - localização geográfica das equipes;
 - definição do número de equipes;
- realizar simulações a partir de localizações de equipes e dados históricos de ocorrências emergenciais previamente fornecidos.

Neste documento são abordados:

- na seção 2, funcionalidades mínimas do sistema;
- na seção 3, abordagem clássica da literatura para o problema de despacho de equipes;
- na seção 4, pressupostos teóricos para o estudo do problema de distribuição de equipes;
- na seção 5, arquitetura geral do sistema, incluindo arquitetura de hardware proposta e resumo das funcionalidades do sistema a desenvolver.

2. DAS FUNCIONALIDADES BÁSICAS DO SISTEMA

O *SEECE* deve fundamentalmente estabelecer ambiente computacional que permita realizar simulações para fornecer subsídios para a alocação e despacho de equipes para atendimentos de emergência. Como tal, os dados fundamentais das simulação serão:

- a localização das chaves onde podem ocorrer as emergências;
- a distribuição das bases onde equipes de atendimento se encontram estacionadas;
- o histórico de ocorrências, incluindo:

- data e hora de registro da ocorrência;
- tempo e distância de deslocamento da equipe;
- tempo de espera pela chegada da equipe ao local do serviço;
- tempo de reparo;
- definição de conjuntos de consumidores afetados.
- a definição dos conjuntos de consumidores do Estado;
- os dados históricos de **DEC** e **TMM**.

A partir desses dados, o *SEECE* deverá permitir ao usuário:

- realizar simulações de despacho de equipes, gerando relatórios de **DEC** e **TMM** e usando dados de:
 - distribuições de equipes de atendimento, obtidas automaticamente ou não;
 - dados históricos de ocorrências;
- propor distribuição de equipes de campo, incluindo localização geográfica e definição do número de equipes.

A figura 1 mostra mapa com todas as posições de todas as chaves da empresa no Estado do Espírito Santo. São mais de 2^{16} chaves. Cada chave representa potencialmente um ponto onde pode ocorrer a qualquer momento uma falha, demandando um atendimento emergencial. Reduzir os tempos de atendimento demanda reduzir essencialmente o tempo de espera entre a detecção da falha, possivelmente via registro de ocorrência procedente de reclamação de usuário, e a chegada da equipe de atendimento à chave correspondente. Isso é em essência um problema de despacho e roteamento de equipes de manutenção, ver [13], que se tratado por algoritmos exaustivos poderia demandar, em casos extremos, milênios de processamento de dados, como veremos a seguir.

3. O PROBLEMA DE DESPACHO DE EQUIPES

Nesta seção, o problema do despacho de equipes de campo é abordado. São discutidos:

- os pressupostos teóricos que estabelecem a dificuldade computacional do problema;
- o mapeamento das restrições específicas da empresa para problemas clássicos da literatura, que permite definir o conjunto inicial de algoritmos considerados relevantes para a solução deste problema;
- a necessidade de se comparar algoritmos que tratem o problema na sua natureza *online* e algoritmos que assumem conhecimento prévio de todas as requisições de um turno de trabalho.

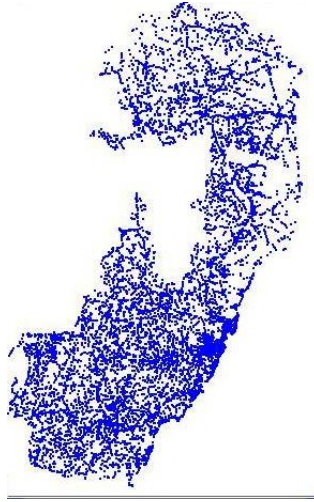


FIGURA 1. Mapa de chaves

3.1. O problema do caixeiro viajante. O *problema do caixeiro viajante*, de agora em diante referenciado como **TSP**, ver [9], fornece um modelo matemático clássico que permite antever a complexidade computacional do problema de despacho de equipes de campo. Neste modelo, assume-se a existência de:

- uma equipe de atendimento, na terminologia clássica, o caixeiro viajante (*salesman*), estacionada em alguma chave;
- um conjunto de chaves com requisições de atendimento associadas;
- uma medida de distância para as possíveis conexões entre as chaves.

Um programa de computador deve neste caso calcular o caminho mais curto que permita à equipe de atendimento sair da chave inicial, visitar todas as chaves com requisições associadas e voltar à chave inicial. Esse caminho mais curto serviria, então, de roteiro de atendimento.

Podem ser consideradas como medidas de distâncias entre pontos:

- distância euclideana, calculada a partir das coordenadas geográficas dos mesmos;
- distância percorrida pelas equipes da empresa e registrada nos dados históricos de ocorrências.

Como salientado antes, é um caso clássico de **TSP**. Embora, a simplificação do problema para a modelagem via **TSP** fuja da realidade da empresa em vários aspectos, como será visto a seguir, ainda assim

ele serve para fornecer base para definir expectativas de desempenho de um provável sistema computacional que reduza os tempos de viagem das equipes e, conseqüentemente, de espera do usuário.

Dado que a empresa possui hoje $\Omega(2^{16})$ chaves e que as soluções conhecidas de **TSP** demandam $\Omega(n!)$ transições para obtenção de melhor solução para n pontos, pode-se esperar que no extremo um dia movimentado demande para cálculo do melhor roteiro para uma equipe de atendimento

$$\Omega(2^{20}) \text{ anos.}$$

Esse cálculo assume a existência de um computador capaz de executar com um clock de 64GHz , podendo gerar 2^{26} soluções possíveis em um ano. Além disso, esse computador deverá apresentar um memória com mais de 10^{3000} bytes.

É importante salientar que paralelismo por si só não reduz essa demanda de tempo e recursos de memória. Mesmo que se dispusesse de uma rede de oito bilhões de computadores, ainda seriam necessários $\Omega(2^{20})$ anos para resolver o problema. O projeto de arquitetura para o *SEECE* deve, então, levar em conta que algoritmos exaustivos, mesmo que executados em hardwares paralelos inimaginavelmente rápidos ainda demandariam tempos excessivamente elevados na gerações de soluções para o despacho de equipes.

A inspeção visual da figura 2, no entanto, mostra as distribuição de chaves (pontos em cinza) e de ocorrência (pontos em vermelho) no Estado, no mês de janeiro de 2004. Essa inspeção visual já é suficiente para mostrar que um grande número de chaves não apresentou ocorrência alguma. No entanto, o número de ocorrências mensal ainda está na casa de 10^4 , trazendo o número de ocorrências diárias para $\mathcal{O}(10^3)$. Mesmo que se considere apenas uma região como Vitória, o número de ocorrências ainda pode chegar a algumas centenas em um curto período do dia, o que, baseado nos cálculos apresentados acima, ainda representa um peso exagerado para um algoritmo de busca exaustiva de melhor solução para o **TSP**.

Nas próximas seções, o modelo de **TSP** é refinado para admitir restrições inerentes ao despacho de equipes de atendimento.

3.2. Múltiplos TSP. Mesmo na modelagem via **TSP**, deve-se levar em conta que a empresa possui mais de uma equipe de atendimento e que essas equipes estão previamente distribuídas por diversas chaves espalhadas pelo Estado. A modelagem clássica para este problema, levando ainda em conta a situação em que todas requisições são conhecidas antecipadamente, é a de roteamento de veículos com múltiplos depósitos: ver [9, 13]. Assume-se agora que existem diversos pontos

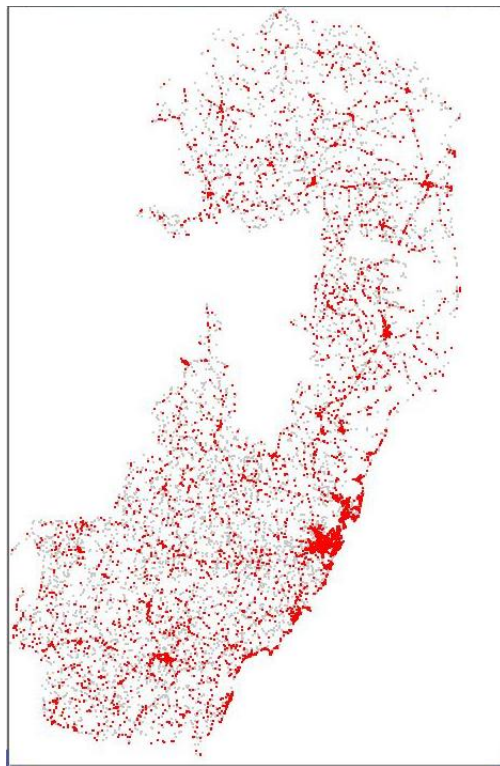


FIGURA 2. Mapa de ocorrências

no Estado, doravante chamados de depósitos, onde estão inicialmente, estacionadas equipes de atendimento, sendo possível um número arbitrário, logo diferente, de equipes para cada depósito.

Vale notar, no entanto, que, embora a dificuldade de solução exata do problema seja reduzida com esta subdivisão em instâncias menores, ainda assim trata-se de problema de grande dificuldade dado que:

- um depósito atenderá tipicamente a centenas de chaves;
- soluções genéricas do **TSP** (e, por consequência do roteamento de veículos também) para instâncias com centenas de pontos podem demandar séculos, mesmo se usada toda a capacidade computacional do planeta hoje disponível.

3.3. Roteamento de veículos com janelas de tempo. Uma restrição importante no escalonamento de equipes está na necessidade de reduzir os coeficientes **DEC** e **TMM** para cada conjunto de consumidores do Estado. O coeficiente **TMM** pode ser usado, então, como uma restrição que estabelecerá que idealmente nenhum tempo de espera deveria ser superior ao atual tempo médio de espera do conjunto em

questão, ou a um tempo médio de espera ideal fornecido como restrição por algum usuário especialista do sistema. Isso estabeleceria ao longo do tempo uma tendência de queda de tempo de espera, com conseqüente redução de **DEC** e **TMM**. Isso demanda nova modelagem para o problema, desta vez como roteamento de veículos com janelas de tempo, ver [13, 14].

O projeto dos algoritmos deve levar em conta agora:

- o tempo de chegada de uma equipe a um local deve atender às restrições de redução de **DEC** e **TMM**;
- o tempo gasto em um reparo impõe restrições de precedência, impedindo, por exemplo, o deslocamento de uma equipe ocupada em um reparo para atendimento a nova requisição.

3.4. O problema offline. Nos modelos propostos acima, assume-se que ao início de um turno de trabalho uma equipe, ou conjunto de equipes, conhece todas as requisições a que atenderá: essas são chamadas soluções *offline*, ou estáticas para problemas de escalonamento. Em oposição, existem as soluções *online*, ou dinâmicas, que admitem que o conjunto de requisições vai sendo conhecido durante o próprio processo de despacho e atendimento.

A proposta de uso de algoritmos *offline*, embora distante da realidade do conhecimento de requisições do dia-a-dia da empresa, encontra respaldo na teoria clássica de soluções para problemas *online*, ver [3], porque as soluções para os problemas *offline* sempre servem de *lower bound* e padrão de comparação para as soluções aproximadas para algoritmos que tratam o problema em tempo real.

No entanto, dadas as dificuldades computacionais envolvidas com o problema até mesmo na sua abordagem estática, alternativas devem ser buscadas que reduzam o tempo de computação necessário para se ter uma solução para escalonamento e despacho de um conjunto de equipes. O sistema computacional de simulação de escalonamento de equipes prevê a implementação e avaliação das seguintes classes de algoritmos:

- algoritmos de aproximação com heurísticas que garantam a qualidade das soluções obtidas, até duas vezes a solução ótima, ver [5], para aproximação de Euler e Chritofides;
- heurísticas de melhora local com Lin-Kernighan e Held-Karp, ver [5];
- meta-heurísticas, que realizem buscas aleatórias controladas em um espaço local de pesquisa, ver [1, 4].

3.5. O problema online. A modelagem completa do problema de escalonamento de equipes deve levar em conta que:

- as equipes poderão estar inicialmente estacionadas em múltiplos depósitos espalhados pelo Estado;
- as requisições de serviço não são completamente conhecidas ao início de um turno de trabalho e nem mesmo no momento de despacho de qualquer equipe para um serviço.

Por outro lado, deve-se levar em conta que o objetivo final consiste em reduzir o tempo de espera ligado a cada requisição. Estabelecidas essas restrições tem-se um problema de roteamento de veículos ou escalonamento de servidores em tempo real.

3.5.1. Roteamento de veículos em tempo real. No roteamento de veículos em tempo real existe um conjunto de veículos, um depósito onde estão estacionados, localizações de requisições e restrições de tempo. No roteamento em tempo real, ver [15], admite-se que:

- uma rota pré-atribuída a um veículo eventualmente não pode ser cumprida;
- que rotas podem ser alteradas dinamicamente.

Neste caso, de cada vez que uma rota se torna inviável, por uma obstrução, por exemplo, todo o roteamento deve ser recalculado e pode-se ter a situação em que veículos deixarão de cumprir algumas requisições e ganharão novas requisições. Duas classes de algoritmos têm sido apresentadas para resolver esse problema:

- exploração exata, possivelmente em paralelo, de todas as opções possíveis, via algoritmos *branch and bound* e *branch and cut*;
- exploração concorrente de soluções via agentes com heurísticas de otimização local.

No problema em que se admite mais de um depósito, cada um tendo uma ou mais equipes estacionadas, as soluções via algoritmos *branch and bound* ou via agentes concorrentes estendem-se naturalmente desde que se admitam as restrições de que:

- veículos podem agora “roubar” requisições de outros veículos localizados em depósitos diferentes;
- existem restrições de tempo, que são dadas pelo limite de **TMM** do conjunto em questão pelos tempos de reparo;

3.5.2. K -servidores. A alocação de equipes para atendimento a ocorrências emergenciais pode ser também modelada como um problema geral de k servidores. É um modelo teórico [3, 2, 11] em que se admite um espaço métrico (\mathcal{M}, d) , onde \mathcal{M} é um conjunto de pontos, e

$k \geq |\mathcal{M}|$ e d definem uma métrica sobre \mathcal{M} . Dada uma sequência de requisições $a = [r_0, r_1, \dots, r_n]$, cada r_i define um ponto de \mathcal{M} em que um serviço é requisitado. Uma requisição r_j é imediatamente atendida se existir um servidor no ponto r_j , no momento em que a requisição é apresentada. A não existência de servidor no ponto r_j demanda o deslocamento de algum servidor de outro ponto $m_i \in \mathcal{M}$ para r_j , (m_j). O custo de tal deslocamento, denotado por d_{ij} , é dado pela função d , que codifica uma função de distância ($d : m_i \times m_j \in M \rightarrow \mathcal{R}$). É importante observar, no entanto, que essa função de distância não precisa e, no contexto colocado, não deve ser Euclidiana: o custo de deslocamento de m_i a m_j não é necessariamente igual ao custo de deslocamento de m_j para m_i e não tem, também, relação apenas com distância no sentido geográfico.

4. O PROBLEMA DE LOCALIZAÇÃO DE EQUIPES

Nesta seção, discutem-se possíveis soluções para o problema de distribuição de equipes pelo estado.

A definição das posições iniciais das equipes, ou depósitos na terminologia da literatura clássica de roteamento de veículos, é especialmente complicado pela:

- quantidade de pontos de atendimento possíveis;
- natureza *online* do problema.

Soluções encontradas na literatura clássica da área e que vêm sendo consideradas no âmbito deste projeto incluem:

- definição de *clusters* de chaves, que definiriam regiões que poderiam ser preferencialmente percorridas pelas equipes de um depósito, via técnicas de planos de corte usadas no contexto do **TSP**, ver [9].
- algoritmos de clusterização com medidas de proximidade parametrizadas, como explorados no trabalho fundamental de Hartigan [10] e também por Fasulo [8] e Eppstein[7], que estuda clusterização com aplicações em computação geométrica e **TSP**.
- algoritmos de geração de *meshes*, tradicionalmente explorados na literatura de computação geométrica, ver Edelsbrunner [6] e Preparata [12], em conexão, por exemplo, com dissipação de calor em circuitos integrados.

Finalmente, não se pode descartar que um especialista possa definir uma alocação arbitrária de depósitos e equipes e a partir dela gerar simulações com dados históricos do sistema.

O *SEECE* oferecerá alternativas de definição automática e localização interativa de equipes e depósitos.

5. ARQUITETURA GERAL DO SISTEMA

O *SEECE* oferecerá as seguintes funções, como discutido ao longo deste relatório.

- realização de simulação de despacho;
- computação de localização de equipes.

5.1. Simulação de despacho de equipes. As simulações serão realizadas sobre dados históricos da empresa permitindo que o usuário selecione para uma dada simulação:

- período e região do Estado da simulação em questão;
- restrições sobre tempos de espera de determinados conjuntos que influenciarão nos resultados obtidos de **DEC**, **TMM** e distribuição de equipes;
- configuração de distribuição e número de equipes de atendimento, incluindo para cada equipe:
 - definição de posição inicial;
 - definição de turno de trabalho.
- escolha de algoritmo de simulação, definindo simulação e modelagem *offline* ou *online*.

A configuração de distribuição e número de equipes selecionada pelo usuário será obtida de configuração previamente gerada por algoritmo de localização de equipes, ver seção 4, ou através de interação com usuário especialista.

O sistema oferecerá como resultado das simulações:

- relatório de **DEC** e **TMM**, resultante da simulação em questão;
- recomendações sobre possíveis alocações de novas equipes.

5.2. Localização de equipes. O sistema oferecerá ao menos duas modalidades de definição de localização de equipes:

- localização automática via algoritmos de clusterização;
- localização via interação com usuário especialista.

5.3. Arquitetura de hardware e software. O sistema vem sendo desenvolvido como uma plataforma de computação distribuída configurável pelo usuário do sistema. Os parâmetros principais de configuração da arquitetura são:

- topologia da rede de processamento que definirá também o número de processadores simultaneamente disponíveis;

- número de processos concorrentes a usar nas simulações paralelas.

A alocação e balanceamento de processos ficará a cargo do sistema subjacente que usará software de propriedade pública, sem ônus para a empresa. Os softwares usados no desenvolvimento são:

- sistema operacional Linux;
- linguagens de programação para construção de protótipos: Lisp, Haskell e Python;
- linguagens de implementação final do sistema: C, Ada, LAM/MPI e Perl.

REFERÊNCIAS

1. J. Beasley, K. Dowsland, F. Glover, M. Laguna, C. Peterson, C.R. Reeves, and B. Söderberg, *Modern heuristic techniques for combinatorial problems*, Colin R. Reeves, 1992.
2. A. Borodin, N. Linial, and M. Saks, *An optimal online algorithm for metrical task systems*, Journal of the ACM **39** (1992), 745–763.
3. Alan Borodin and Ran El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.
4. Olli Bräysy, *Genetic algorithms for the vehicle routing problem with time windows*.
5. William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver, *Combinatorial optimization*, John Wiley & Sons, Inc., 1998.
6. Herbert Edelsbrunner, *Algorithms in combinatorial geometry*, Springer-Verlag, 1987.
7. David Eppstein, *Fast hierarchical clustering and other applications of dynamic closest pair*, Proceedings of SODA, 1998.
8. Daniel Fasulo, *Ana analysis of recent work on clustering algorithms*, Tech. Report 01-03-02, Department of Computer Science, University of Washington, 1999.
9. Gregory Gutin and Abraham P. Punen, *Traveling salesman problem and its variations*, Kluwer Academic Publishers, 2002.
10. John Hartigan, *Clustering algorithms*, JOHN WILEY & SONS, 1975.
11. Rajeev Motwani and Prabhakar Raghavan, *Randomized algorithms*, Cambridge University Press, 1995.
12. Franco P. Preparata and Michael Ian Shamos, *Computational geometry: An introduction*, Springer-Verlag, 1985.
13. Jennifer L. Rich, *A computational study of vehicle routing applications*, Ph.D. thesis, Rice University, 1999.
14. Jürgen Schulze and Torsten Fahle, *A parallel algorithm for the vehicle routing problem with time window constraints*.
15. Kenny Qili Zhu and Kar-Loon Ong, *A reactive method for real time dynamic vehicle routing problem*.