Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia de Transportes

Luccas Henrique Moita

Pedro Henrique Almeida

ALOCAÇÃO DE PROFISSIONAIS DA SAÚDE A PACIENTES EM UMA EMPRESA DE ASSISTÊNCIA MÉDICA

Trabalho de Formatura do Curso de Engenharia Civil apresentado à Escola Politécnica da Universidade de São Paulo na disciplina “0313502 – Trabalho de Formatura em Engenharia Civil II”

Orientador: Profº. Dr. Cassiano Augusto Isler

São Paulo

2019

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta uma proposta de uma ferramenta computacional para a operação de uma empresa que atua no setor de *home care*, setor da saúde focado no atendimento domiciliar de pacientes com especificidades de cuidados. A partir de um algoritmo computacional existente, o Jsprit, foi executada a atribuição otimizada de profissionais da saúde em analogia ao problema de roteirização de veículos levando em consideração a minimização do tempo de deslocamento entre os profissionais e pacientes da empresa estudada. Todos os dados utilizados para criação da ferramenta computacional e para sua verificação foram fornecidos pela própria empresa. A partir do algoritmo utilizado foi possível desenvolver uma ferramenta computacional para otimizar o processo de atribuição de profissionais da empresa, além de apresentar uma ferramenta que permitisse automatizar uma distribuição que hoje é feita manualmente considerando as especificidades de cada paciente e atendente.

1. INTRODUÇÃO

O conceito de *home care* é bastante abrangente, assim como os serviços médicos e de enfermagem prestados aos pacientes em sua residência. Neste trabalho, *home care* refere-se ao atendimento domiciliar a pacientes, sejam eles de internação, ambulatorial ou de assistência, por pessoal de enfermagem especializada.

Segundo Edvaldo de Oliveira Leme (2015)[1], o setor de *home care* surgiu nos Estados Unidos nos anos 1940 no contexto pós-Segunda Guerra Mundial, porém, somente nos anos 1960 o setor começou a ter destaque no mercado. Nesse contexto, os hospitais apresentavam uma taxa de ocupação elevada devido ao crescimento e longevidade populacional, surgindo então as *Nursing Homes*, cujo enfoque é o atendimento de pacientes crônicos terminais. Percebeu-se, porém, a possibilidade de atendimento de outros tipos de pacientes nesses locais que por vezes ocupavam leitos hospitalares desnecessariamente.

Ainda segundo Edvaldo de Oliveira Leme (2015), dada a situação de crescimento populacional e saturação dos hospitais, buscou-se alternativas para o atendimento dos pacientes com diferentes enfermidades e surgiram então as primeiras instituições que se propunham a tratá-los em domicílio. Essas instituições apresentaram bons resultados, realizando atendimentos de qualidade elevada e com custos entre 20% e 70% menores do que os hospitais tradicionais (Falcão, 1999). Apesar dessa eficiência, o setor não apresentou grande popularidade pois não era coberto por planos de saúde e apenas uma pequena parcela da população poderia pagar por um atendimento com esse custo.

A empresa MetLife, seguradora de saúde fundada em 1868, foi a primeira a utilizar nos Estados Unidos, em 1960, as visitas de enfermagem para reduzir seus custos de sinistralidade, isto é, custos assistenciais diretos. Nesse caso, a seguradora

[1] https://portalhomecare.com.br/historia-do-home-care/

percebeu que, ao fornecer um atendimento periódico para alguns tipos de pacientes, os atendimentos hospitalares e utilização de serviços mais caros fornecidos eram substancialmente menores, sendo, portanto, uma oportunidade para ganho de eficiência e margem operacional da empresa.

No Brasil, o contexto de utilização de *home care* também tem ganhado espaço nos últimos anos. A utilização do *home care* se mostra como uma alternativa viável, sendo interessante para a seguradora dados os menores custos para atendimento dos seus pacientes. Também é interessante para os hospitais quando estes apresentam taxa de ocupação elevada, já que um paciente de perfil de *home care* muitas vezes não irá consumir no hospital um volume grande de materiais e medicamentos. Por fim, essa situação pode ser interessante para a família do paciente, que pode contar com um atendimento sem a necessidade de estar em um ambiente hospitalar.

Na empresa objeto de estudo neste trabalho final de graduação, os clientes são, de maneira geral, planos de saúde que fornecem para os seus segurados a possibilidade de atendimento ou internação domiciliar, quando recomendada pelo médico. Na operação da empresa, as etapas consistem no credenciamento dos planos de saúde, captação e atendimento de pacientes. Na etapa do atendimento, a empresa responsável pelo *home care* envia profissionais para a casa do paciente para prestação do serviço contratado.

Atualmente, profissionais da empresa fazem a alocação dos profissionais de saúde aos pacientes de forma totalmente manual, com base na experiência e percepção de qual seria o melhor cenário. Esse procedimento é feito diariamente de modo a programar a escala do dia seguinte. Contudo, não há uma avaliação quantitativa de quais são os custos envolvidos e quais seriam, eventualmente, as economias se o processo contasse com um procedimento automatizado. Com o crescimento da base de pacientes (já próxima de 500), o procedimento manual acaba perdendo sua eficiência e por vezes pode não atingir o melhor cenário.

Os pacientes apresentam especificidades, cujas necessidades de atendimento variam conforme o tipo de doença que possuem. Os atendimentos podem ter duração desde uma hora, como a realização de uma troca de curativo ou administração de um medicamento intravenoso, até um plantão completo, de doze horas, nos casos de pacientes de internação domiciliar, em que praticamente se monta um leito hospitalar na casa do paciente e há um profissional presente 24 horas.

Observa-se dificuldade significativa na atribuição de profissionais a pacientes na cidade de São Paulo dadas as características dos atendimentos, a localização dos pacientes e as distâncias que os profissionais devem percorrer entre atendimentos. Assim, a aplicação de um método de alocação de profissionais a pacientes pode contribuir para maximizar o potencial operacional da empresa em análise.

Este estudo considera o acesso à base de endereços e perfil dos profissionais contratados e pacientes atendidos pela empresa, os quais poderão ser utilizados para solução do problema de alocação de profissionais estabelecido em analogia ao problema de roteirização comumente observado em situações reais e que, em geral, é resolvido sob os conceitos da Pesquisa Operacional.

Assim, espera-se que o estudo possa ser utilizado como um protótipo no âmbito empresarial, viabilizando uma potencial redução do quadro de funcionários, além de viabilizar maior agilidade e confiabilidade nas operações da empresa.

1.1. Objetivos

O objetivo geral deste trabalho de conclusão de curso de graduação é propor uma ferramenta computacional capaz de alocar profissionais de saúde a pacientes no contexto de uma empresa de assistência médica domiciliar na região metropolitana de São Paulo.

Para atingir o objetivo principal são estabelecidos os seguintes objetivos específicos:

i. Estabelecer uma analogia do problema de alocação de profissionais a pacientes ao problema de roteirização de veículos (*Vehicle Routing Problem* - VRP);

ii. Executar uma ferramenta computacional que resolve um problema de roteirização a partir de uma base de dados de profissionais e pacientes, e de um algoritmo existente, o Jsprit, de modo a obter uma solução otimizada para o problema de alocação dos profissionais.

1.2. Justificativa

Durante o curso de graduação em Engenharia Civil na Escola Politécnica da Universidade de São Paulo foram estudados conceitos e modelos no contexto da Pesquisa Operacional em disciplinas do Departamento de Engenharia de Transportes, em que se buscam soluções otimizadas para problema de Logística e Transportes.

O problema a ser resolvido neste trabalho consiste na designação de auxiliares e técnicos de enfermagem para atendimento de seus pacientes de uma forma eficiente, uma vez que essa atribuição é realizada de maneira manual atualmente.

A empresa possui uma base de profissionais aptos a atender pacientes que precisam de cuidados médicos específicos, tal que a atribuição dos técnicos não é uma tarefa trivial quando se busca a otimização dos recursos disponíveis para realização das tarefas.

Além disso, existem casos em que não é possível atender todos os pacientes com os recursos humanos disponíveis, sendo necessária a contratação emergencial de empresas terceiras para os atendimentos, acarretando em maiores custos operacionais para a empresa.

Assim, espera-se que a aplicação de conceitos no âmbito da Pesquisa Operacional e roteirização viabilize a identificação de soluções otimizadas ao problema de atribuição de profissionais a seus pacientes no contexto prático da empresa, atendendo às restrições dos profissionais e às necessidades dos pacientes.

2. REVISÃO BIBLIOGRÁFICA

2.1. Problema de alocação de profissionais

O problema de alocação de pessoal e planejamento operacional de escalas de trabalhadores é estudado em diversos contextos e, segundo Causmaecker e Berghe (2010), recebeu ampla atenção nos últimos anos devido à complexidade e relevância social e econômica.

Warner (1975) propôs uma solução matemática para a escala de profissionais de enfermagem em um hospital, definindo horizontes de planejamento de quatro ou seis semanas e levando em consideração na modelagem os custos, a qualidade do serviço, a estabilidade do serviço, a flexibilidade, além de uma medida de justiça, que representa o quanto os profissionais sentem que tem a mesma influência que seus pares no processo decisório dos turnos. Esse modelo, apesar de aplicável, à princípio, em hospitais, apresenta motivações que são semelhantes àquelas buscadas no segmento de *home care*, ou seja, busca-se essencialmente a redução de custos e a melhoria da qualidade e percepção de qualidade do serviço prestado.

Os problemas de alocação de profissionais, em geral, são semelhantes àquele estudado por Graham et al. (1979) em que se desejava programar um número de máquinas para executar trabalhos específicos, otimizando determinado fator de produção ou eficiência. Para Causmaecker e Berghe (2010), pode-se classificar o problema de alocação de pessoal de enfermagem utilizando a mesma notação α/β/γ proposta por Graham et al. (1979), em que α representa o ambiente em que as máquinas estão, β as características do trabalho e suas restrições, e γ os objetivos de otimização. Para trazer essa notação para o contexto de profissionais de saúde, deve-se substituir as máquinas por profissionais. No contexto do atendimento *home care* estudado nesse trabalho, os conceitos de α/β/γ podem ser novamente adaptados para o tipo de profissionais, as restrições do trabalho e os objetivos a serem alcançados.

Cunha (2003) também propôs uma metodologia para atribuição de profissionais à clientes de forma semelhante ao que se procura fazer neste trabalho, porém, a atribuição foi de gerentes de banco à potenciais clientes. Um diferencial do trabalho de Cunha foi que, dados as especificidades do tipo de trabalho que os gerentes fariam e as restrições do problema, foi possível obter uma solução exata para um problema do tipo NP-difícil, que em situações normais somente poderia ser resolvido por algoritmos, de maneira parecida à que será proposta nesste trabalho. Além disso, o trabalho é e foi aplicado à um problema real enfrentado por uma instituição financeira, assim como se pretende fazer para a área de saúde.

Li e King (1998) consideraram os efeitos do treinamento dos profissionais para realizar diferentes tarefas na modelagem de alocação. Essas considerações são importantes para o modelo de estudo do *home care* pois a base de profissionais disponíveis para alocação teria menos restrições quanto ao perfil dos pacientes, tornando necessária a utilização menos frequente de profissionais extras. Para efeitos de modelagem, seria possível considerar que uma parte da base que antes não estava apta para, por exemplo, tratar de crianças, poderia agora ser alocada nesses pacientes.

Writh et al. (2006) e Fowler et al. (2006) também realizaram análises de turnos e rotação de trabalhadores considerando seus impactos nos custos de mão de obra. Wright propôs uma análise de quanto o fator mínimo de profissionais para pacientes estipulado pela legislação impacta a atribuição de profissionais e o custo total agregado no caso de hospitais. Novamente, para o setor de *home care*, esse fator não pode ser considerado no atendimento pois sempre haveria um para um. Porém, um profissional que tem uma produtividade diária mais alta colaboraria para o fator analisado no dia ser mais eficiente, acarretando em menores custos para o prestador do serviço.

Fowler et al. (2006), por outro lado, apesar de terem analisado profissionais fora da área da saúde, propuseram restrições parecidas com o que se observa no mercado de *home care*. Propôs-se no modelo a consideração de diferenças de habilidades

dos seus profissionais, considerando que determinados profissionais da saúde são mais capacitados para atender determinados casos do que outros. Essa medida pode ser entendida como uma limitação nas habilidades do profissional.

Maenhout e Vanhoucke (2012) discutiram e desenvolveram um método integrado para solução de problemas de alocação de enfermeiros em grandes hospitais e como as políticas de alocação do hospital e as características dos profissionais influenciam a solução fornecida pelo modelo. Com aquele trabalho, concluíram que integrar as especificidades dos profissionais no processo decisório leva ao aumento da qualidade da distribuição e isso pode ser observado em reduções de custos, aumento de qualidade do serviço e satisfação da equipe, podendo resultar em um menor *turn-over*[2] e, portanto, menor esforço e tempo gastos na contratação de novos profissionais para executar os serviços.

Gilgen et al. (2018) propuseram um modelo de caracterização de escala de trabalho de operadores de equipamentos em um terminal portuário de contêineres, visando a redução de custos de contratação de recursos humanos. Nesse problema, buscou-se otimizar a alocação de operadores a equipamentos em turnos, enquanto que no problema em estudo busca-se otimizar a disposição de auxiliares e técnicos de enfermagem a pacientes, ou seja, são problemas semelhantes em sua essência.

Observa-se pela literatura tratada acima que o problema de atribuição de profissionais já foi estudado em diversos contextos, porém, poucas são as abordagens sobre a otimização de escala especificamente no setor de *home care*.

[2] É um conceito frequentemente utilizado na área de Recursos Humanos (RH) para designar a rotatividade de pessoal em uma organização, ou seja, as entradas e saídas de funcionários em determinado período de tempo.

2.2. Problema de roteirização

A análise preliminar do problema prático descrito neste texto permite afirmar que a sua solução pode ser análoga ao problema de roteirização de veículos no contexto da logística. No caso tratado, os profissionais correspondem aos veículos, que devem realizar visitas programadas aos pacientes, analogamente aos locais de entrega de produtos. Pode-se também considerar a carga horária total possível de trabalho do profissional como a capacidade de um veículo fazendo uma entrega, e o tempo que este profissional passa nos atendimentos e fazendo seu deslocamento, o consumo dessa carga.

O problema de roteirização derivou do problema clássico e amplamente estudado do caixeiro viajante. No problema de VRP deseja-se determinar um conjunto de trajetos iniciando e terminando em um depósito e visitando diversos clientes, ou seja, pontos intermediários ao longo do trajeto. Cada cliente apresenta uma demanda e cada veículo que sai de um depósito não pode fornecer mais que a sua demanda permite. Em geral, o objetivo é minimizar o tempo de trajeto, ou o número de veículos, ou ainda a distância percorrida. Tais parâmetros são associados ao custo do percurso.

Serão apresentadas as formulações matemáticas para dois problemas derivados do VRP, o primeiro é chamado de CVRP, *Capacitaded Vehicle Routing Problem*, em que cada veículo apresenta uma capacidade máxima, que é consumida conforme a demanda dos clientes. Esse problema se assemelha bastante ao que será estudado neste trabalho, sendo que, para o caso da roteirização de profissionais, a capacidade é o tempo de trabalho máximo por dia e a demanda é o tempo de atendimento de cada cliente e de trajeto entre clientes.

Outro problema tradicional é o VRPTW, *Vehicle Routing Problem With Time Windows,* em que se acrescenta uma restrição em relação ao intervalo de tempo em que cada cliente deve ser atendido. Neste trabalho, foi considerado que não haveria essa restrição para os clientes, porém a ferramenta computacional desenvolvida

suporta esse tipo de condição que poderia vir a ser acrescentada, dependendo do tipo de cliente a ser atendido.

As formulações apresentadas a seguir foram extraídas do artigo *A generalized formulation for vehicle routing problems* de Munari et al. (2017).

Considera-se um conjunto de clientes, denotado por $C = \{1, \ldots, n\}$, tal que uma demanda positiva (no caso, tempo) é associada a cada cliente $i \in C$. Para atender esses clientes, designa-se uma frota com K veículos em um único depósito. Cada rota deve se iniciar e terminar no depósito, visitando um subconjunto de clientes. Todos os clientes devem ser visitados uma vez. Cada veículo apresenta uma capacidade máxima Q, que limita o número de clientes que ele pode atender antes de retornar ao depósito. Define-se ainda $N = C \cup \{0, n+1\}$, o conjunto de nós associado aos clientes C e ao depósito em 0 e $n+1$. Deve-se impor que todas as rotas devem se iniciar em 0 e retornar para $n+1$. Define-se $\mathcal{E}$ como o conjunto que contém os trajetos (i, j) para cada par de nós $i, j \in N$. O custo desse trajeto é denotado por c_{ij}. Cada nó tem ainda uma demanda $q_i > 0$ para cada $i \in C$ e $q_o = q_{n+1} = 0$. Deseja-se determinar o conjunto de trajetos que resulta no mínimo custo e satisfaz todas as condições determinadas acima.

Define-se a variável de decisão x_{ij} que assume o valor 1 se, e somente se, existe uma rota que parte do cliente i para o cliente j com $i, j \in N$. Adicionalmente y_j é uma variável de decisão contínua correspondendo à demanda acumulada no trajeto que visita o nó $j \in N$ até a visita i. Com esses parâmetros, restrições e variáveis, a formulação matemática é dada por:

$$\text{Minimizar} \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \qquad (1)$$

sujeito a

$$\sum_{\substack{j=1 \\ j\neq i}}^{n+1} x_{ij} = 1, \qquad i = 1,\dots,n \qquad (2)$$

$$\sum_{\substack{i=0 \\ i\neq h}}^{n} x_{ih} - \sum_{\substack{j=1 \\ j\neq h}}^{n+1} x_{hj} = 0, \qquad h = 1,\dots,n \qquad (3)$$

$$\sum_{j=1}^{n} x_{0j} \leq K, \qquad (4)$$

$$y_j \geq y_i + q_j * x_{ij} - Q * \left(1 - x_{ij}\right), \qquad i,j = 0,\dots,n+1 \qquad (5)$$

$$q_i \leq y_i \leq Q, \qquad i = 0,\dots,n+1 \qquad (6)$$

$$x_{ij} \in \{0,1\}. \qquad i,j = 0,\dots,n+1 \qquad (7)$$

A função objetivo é definida por (1) e impõe que o custo total de viagem deve ser minimizado. A restrição (2) garante que todos os clientes são visitados exatamente uma vez, a restrição (3) garante o fluxo correto de veículos através dos trajetos, afirmando que se um veículo chegar a um nó h, então ele deve partir também desse nó. A restrição (4) limita o máximo número de rotas para ser igual a K, ou seja, o número de veículos. Por fim, as restrições (5) e (6) garantem que a capacidade do veículo não será excedida.

Pode-se estender o problema CVRP para um problema do tipo VRPTW definindo-se a janela de tempo $[w_i^a, w_i^b]$ que impõe que o nó i não pode começar a ser atendido anteriormente a w_i^a e nem após w_i^b. Atribui-se a cada arco (i,j) um intervalo de tempo t_{ij} e a cada nó um intervalo t_i que corresponde ao intervalo de tempo que cada veículo passará no nó visitado.

Seja w_i uma variável de decisão contínua representando o instante de tempo que o serviço começa no nó i. Pode-se estender o modelo para uma solução do problema VRPTW adicionando-se as seguintes restrições à formulação previamente apresentada.

$$w_j \geq w_i + \left(s_i + t_{ij}\right) * x_{ij} - M_{ij} * \left(1 - x_{ij}\right), \quad i = 0, \ldots, n; \; j = 1, \ldots, n+1, \quad (8)$$

$$w_i^a \leq w_i \leq w_i^b, \quad i = 0, \ldots, n+1 \qquad (9)$$

em que M_{ij} é um valor suficientemente grande, que pode ser definido como $M_{ij} = \max\{w_i^b - w_j^a, 0\}$.

2.2.1. Analogia do Problema de Roteirização ao de Alocação de Profissionais de *Home Care*

Dekhici et. al (2019) estenderam o problema de roteirização à aplicação ao problema de alocação de profissionais ao setor de *home care* e o resolveram utilizando um algoritmo específico. Esta analogia também foi admitida válida neste trabalho de conclusão de curso. A variação estudada no artigo daqueles autores é conhecida amplamente como VRPTW (*Vehicle Routing Problem with Time Windows*), ou seja, um problema de roteirização em que cada ponto deve ser visitado dentro de uma específica janela de tempo.

As restrições abordadas no artigo seguem classificações análogas às consideradas neste trabalho, sendo elas:

i. Restrições de atribuição: são aquelas que levam em consideração a continuidade do atendimento ou lealdade profissional-paciente. No atual trabalho, essas restrições especificamente não precisaram ser consideradas, já que a roteirização foi feita somente para pacientes do tipo “Procedimentos”. Restrições desse tipo são comuns em pacientes de internação, em que o profissional passa um

plantão de 12 horas na casa do paciente atendido e as alterações de profissional são vistas como piora de qualidade do atendimento;

ii. Restrições geográficas: são chamadas de restrições por Dekhici et. Al (2019), porém podem ser melhor entendidas como critérios ou definições. Deve-se definir se serão utilizadas distâncias euclidianas ou distâncias efetivas de percurso na malha viária. Na maioria dos artigos publicados até hoje, inclusive no citado, considera-se a distância euclidiana para otimização de distâncias. No presente trabalho será utilizada para a atribuição de profissionais da empresa em análise a distância na malha viária por meio de uma matriz do tipo O/D (Origem e Destino) gerada por um software de mapas;

iii. Restrições de tempo: nessa categoria, podem surgir diversos tipos de restrições relacionadas ao tempo, como por exemplo o tempo total de trabalho por dia por profissional, o tempo em função do tipo de atendimentos que cada paciente necessita, a janela de tempo em que os pacientes devem ser atendidos, as durações de cada atendimento, entre diversos outros aspectos que podem ser considerados, dependendo do que é relevante para o problema específico a ser resolvido. Dekhici et. al (2019) consideraram que os profissionais devem trabalhar no máximo 8 horas por dia, porém, consideraram essa uma restrição fraca, ou seja, que pode ser violada com aplicação de penalidade, havendo incremento no custo total para o serviço.

Estabelecidas as restrições, Dekhici et. al (2019) fizeram uso de um algoritmo meta-heurístico chamado Firefly para a determinação da melhor rota. O processo é iterativo, em que para cada etapa, o profissional em uma localidade busca o próximo ponto de atendimento com base em critérios de atratividade. Para o trabalho de conclusão de curso foi escolhido outro algoritmo computacional para identificação de soluções ao problema, a qual será melhor detalhada na seção seguinte.

2.2.2. Meta-heurística do Jsprit

A meta-heurística utilizada pelo Jsprit foi proposta inicialmente por Schrimpf et al. (2000). O processo proposto consiste no método chamado de *Ruin-and-Recreate*. Nesse processo, toma-se uma malha que apresenta uma solução factível para o problema proposto, sendo essa solução não necessariamente boa. A partir da solução, aplica-se uma rotina de melhora para a solução previamente encontrada.

A etapa de ruína da malha escolhe, com base em um critério pré-determinado, uma sub-malha do problema para retirar da solução, ou seja, cria-se um subproblema que contém pontos da solução original e pontos que agora passaram a não ser atendidos pela solução proposta. A escolha de quais pontos retirar da rota de atendimento pode ser feita de 3 maneiras:

i. Ruína Radial: nesse tipo de estratégia de ruína, escolhe-se um nó aleatório, que no caso deste trabalho representará um paciente a ser atendido, removendo-o. Além disso, determina-se outro número aleatório N, necessariamente menor que o número total de nós do problema e remove-se os N nós mais próximos do primeiro nó removido. Pode-se determinar qual será o critério de proximidade, como por exemplo distância euclidiana;

ii. Ruína Aleatória: nesse tipo de estratégia, remove-se um número aleatório de nós, sendo que a única restrição é que o número de nós removidos deve ser menor que o número total de nós do problema;

iii. Ruína Sequencial: Remove-se um número aleatório de nós sequenciais em uma rede circular.

A partir da ruína, deve ser recriada uma solução que atenda todas as restrições do problema. A forma de fazer isso, proposta por Schrimpf et al. (2000) consiste no método da melhor inserção, ou seja, escolhe-se de forma aleatória um ponto que foi removido do roteiro original e coloca-o novamente no sistema de forma a atender

todas as restrições e, das possibilidades de reinserção, escolhe-se aquela que tem um menor tempo total, partindo-se então para o próximo ponto, escolhido também aleatoriamente.

Ao final do processo de recriar o problema, deve-se decidir se a solução recriada é adotada como nova solução ou descartada, tal decisão também é baseada em algum determinado critério, por exemplo, aceita-se a nova solução se ela for melhor que a anterior, ou então a solução é aceita se ela for melhor ou levemente pior que a solução anterior, ou até mesmo que qualquer solução nova é aceita como nova solução. Tal algoritmo pode ser configurado com base na necessidade de uso e, no caso do Jsprit, utiliza-se o algoritmo conhecido como *Simulated Annealing*, conforme proposto por Schrimpf et al. (2000) que determina que qualquer solução melhor que a solução atual deverá ser aceita, além de, com uma determinada probabilidade, também serão aceitas soluções piores que a solução atual.

Além da meta-heurística proposta por Schrimpf el al. (2000), o Jsprit também se baseia na estratégia de busca por soluções proposta por Pisinger e Ropke (2007). Na proposta dos autores, os cinco problemas clássicos de roteirização, isso é, VRPTW (*Vehicle routing problem with time windows)*, CVRP (*Capacitated vehicle routing problem)*, OVRP (*Open vehicle routing problem)*, MDVRP (*Multi-depot vehicle routing problem)* e SDVRP (*Site-dependent vehicle routing problem)* são caracterizados em um único problema do tipo RPDPTW (*Rich pickup and delivery problem with time windows)*.

Fazendo essa transformação, as funções do Jsprit podem ser implementadas na linguagem de programação Java com o apoio de uma IDE (*Integrated Development Environment)* e aplicado para uma variedade grande de problemas de roteirização.

3. O PROBLEMA

O problema da empresa objeto de estudo neste trabalho de conclusão de curso é um gargalo operacional pois cada profissional é designado manualmente por um funcionário da empresa que atua no *back-office* para atender um paciente. Assim, conforme aumenta o número de atendimentos necessários, mais pessoas são requisitadas para fazer as atribuições.

Além disso, não se sabe se o processo hoje é otimizado, uma vez que as atribuições são feitas conforme a experiência passada e a percepção do profissional sobre qual seria o melhor cenário para realizar o maior número de atendimentos possível.

Como os pacientes não necessariamente são atendidos todos os dias, a escala de trabalho deve ser feita diariamente, tornando o trabalho das pessoas que devem fazer a escala repetitivo e pouco produtivo.

Deseja-se, portanto, resolver o problema de atribuição de profissionais à pacientes de forma automatizada, levando em consideração critérios objetivos para definir qual profissional deverá atender qual paciente e ainda em que ordem os atendimentos deverão ser feitos.

Há ainda algumas especificidades do problema. No caso da empresa, considera-se que todos os profissionais fazem os trajetos até os pacientes utilizando ônibus. Além disso, profissionais partem de suas casas e retornam para elas ao final dos atendimentos.

4. MÉTODO

Primeiramente é necessário contextualizar o cenário atual da empresa estudada. A empresa hoje atua na região metropolitana de São Paulo, Baixada Santista, Rio de Janeiro, Rio Grande do Sul e Espírito Santo oferecendo quatro tipos de atendimento:

- Assistência supervisionada: consiste em atendimentos em que há uso somente de mão-de-obra para auxílio em atividades diárias do paciente, sendo que eventuais materiais ou medicamentos são cobrados à parte. Hoje, a empresa conta com 252 pacientes;

- Internação Domiciliar: são casos de atendimentos de alta e média complexidade, sendo que os de alta complexidade necessitam de assistência 24 horas por dia e os de média complexidade, somente 12 horas. Hoje, a empresa conta com 132 pacientes nessa categoria;

- Procedimentos: são fornecidos técnicos para auxílios pontuais, como por exemplo a aplicação de uma medicação ou troca de curativos. Hoje, a empresa possui 86 pacientes;

- Programa de medicação: somente são entregues medicamentos controlados e o acompanhamento é feito mensalmente por enfermeiros especializados. Hoje, a empresa possui 43 pacientes.

Os profissionais alocados a cada tipo de paciente podem variar, ou seja, um profissional que atende a pacientes do tipo “Procedimentos” poderia estar alocado a um paciente de “Internação Domiciliar” e vice-versa. Porém, para o escopo do trabalho, a empresa separou um grupo de profissionais que usualmente atendem a pacientes de “Procedimentos”.

Para o problema em questão, deseja-se executar a alocação somente dos casos de pacientes do tipo “Procedimentos” para a região metropolitana de São Paulo (atualmente 71 pacientes). Como os casos de “Internação Domiciliar” são atendimentos com plantões fixos e com baixa rotatividade de profissionais, não há um valor prático para a aplicação, já que a empresa não tem interesse em realizar a troca dos profissionais atendendo esse perfil de paciente. Para os pacientes de perfil “Assistência Supervisionada”, grande parte da mão-de-obra consiste em fisioterapeutas e fonoaudiólogos, que apresentam um outro modelo de contratação (normalmente empresas terceiras), não sendo de interesse da empresa estudar uma melhor distribuição desses profissionais, já que isso não teria efeito significativo em seu custo operacional.

Além da questão de “Procedimentos” ser um tipo de atendimento com possibilidade de melhorias quanto à forma de alocação de profissionais, esse atendimento representa, até o momento em 2019, 14% do faturamento total da empresa. Porém, ao ser removida da análise os pacientes do tipo “Internação Domiciliar” que, como previamente mencionado, não apresentam margem para melhora devido ao tipo de serviço fornecido, esse percentual passa a ser 35% do faturamento total da empresa. Outros 34% estão concentrados em “Programas de Medicação”, já que os medicamentos entregues são de alto custo, porém, as entregas ocorrem com uma rotina mensal e não diária, como nos “Procedimentos”.

Por esses motivos, escolheu-se como objeto de análise deste trabalho o perfil de “Procedimentos”, pacientes com atendimentos pontuais de profissionais, que devido ao seu perfil de consumo de materiais e medicamentos em domicílio, são responsáveis por uma parcela considerável do faturamento e da margem operacional da empresa. Assim, a possibilidade de ganho de escala está no atendimento do maior número possível de pacientes, por dia e por profissional, nesse setor.

4.1. Analogia ao Problema de Roteirização

Esta seção descreve o procedimento utilizado para alocação dos profissionais da saúde como uma analogia ao problema de roteirização de veículos, contemplando os dados utilizados no modelo e a descrição do algoritmo computacional utilizado na aplicação.

O problema básico de roteirização é um problema de distribuição em que veículos localizados em um depósito, ou vários, devem ser designados para visitar clientes geograficamente dispersos para coleta de itens, sendo que uma limitação é a capacidade do veículo. No presente trabalho, os depósitos são os pontos de partida de cada profissional separadamente e os clientes que receberiam a entrega são os pacientes. Os profissionais partem de suas respectivas casas e se deslocam entre pontos de atendimento, devendo permanecer determinado tempo em cada um dos endereços e, ao final do dia, retornar ao ponto de partida.

Para a roteirização de profissionais, o horário comercial é uma limitação uma vez que o profissional pode atender uma quantidade máxima de pacientes nas suas oito horas de trabalho diárias. Além disso, será considerado para esse trabalho que todos os profissionais se deslocam de ônibus para atender os pacientes designados.

4.1.1. Dados do problema

A empresa tem uma base de dados de profissionais e pacientes a serem atendidos, porém, em um primeiro momento, deseja-se resolver o problema para os pacientes classificados como “Procedimentos”, conforme já explicitado.

O primeiro passo para a resolução do problema proposto é a conversão dos dados dos pacientes e profissionais em um formato de coordenadas geográficas (latitude e longitude), formato que pode ser lido pelo Jsprit, utilizado para a determinação da

melhor rota dos profissionais conforme descrito em maiores detalhes nas seções subsequentes.

Para a conversão dos dados foi utilizado um *add-in* do Excel chamado GeodesiX. Esse *add-in* funciona de maneira integrada ao Google Maps e trabalha com um processo de geocodificação, de modo que se pode obter coordenadas geográficas e distâncias na rede viária entre pontos.

A função “Geocode” utilizada pelo GeodesiX funciona com um código API[3] fornecido pelo Google. Essa função tem restrições de consulta por IP[4], podendo ser realizadas apenas 2.500 consultas de endereços por dia.

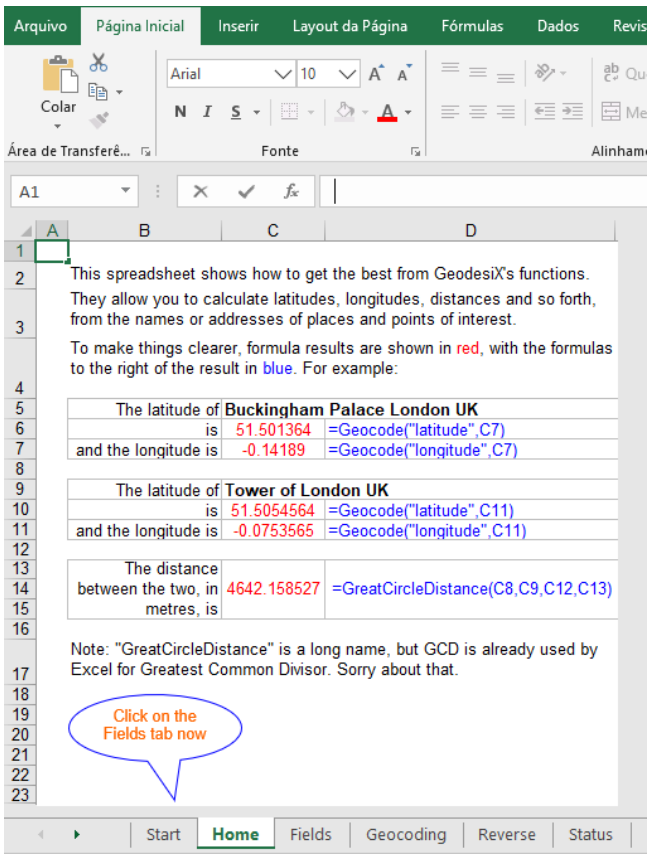

Figura 1 – Exemplo do processo de Geocodificação e cálculo de distâncias do GeodesiX

[3] API (*Application Programming Interfaces*) é um conjunto de estruturas de programação para acesso a um determinado sistema. As APIs foram criadas para permitir que máquinas pudessem se comunicar entre si.
[4] O IP (*Internet Protocol*) é uma identificação única para cada computador conectado a uma rede.

Para obter as coordenadas de cada profissional e paciente foram utilizados seus endereços, no formato de endereço seguido pela respectiva cidade, uma vez que o processo de geocodificação não apresenta erros, diferentemente do que ocorre caso fossem utilizados os códigos de endereçamento postal. Ainda, para que as coordenadas pudessem ser lidas pelo Jsprit, o arquivo resultante de coordenadas foi salvo no formato de texto (.txt) com o computador configurado para representação de número decimal por “ponto” para possibilitar a correta leitura do arquivo.

Por fim, novamente, impondo que cada profissional deve trabalhar por até oito horas diárias, foi considerado um tempo médio de atendimento de uma hora e vinte minutos, baseado no tempo médio de um procedimento real, conforme informado pela empresa.

Em seguida, para a determinação das distâncias utilizou-se novamente o GeodesiX, agora com a função “Travel("distance","Origem","Destino","driving")”. A partir do endereço de cada paciente, foi possível a determinação das distâncias na malha viária de São Paulo entre os pontos de atendimento, isto é, um vetor com uma coluna e 5041 linhas (o produto da quantidade de pacientes por ela mesma) de distâncias entre os pacientes, além de um vetor com uma coluna e 2627 linhas (o produto da quantidade de pacientes pela quantidade de profissionais) da distância entre as casas dos profissionais e o local de atendimento de cada um dos pacientes. Foi novamente gerado um arquivo .txt com essas informações com as distâncias registradas em metros.

Foi necessário também produzir matrizes de tempo de deslocamento entre os pontos. Todas as distâncias da base de dados e do código elaborado foram admitidas em metros e o tempo, em segundos. Para definição do tempo de deslocamento entre pontos foi considerada velocidade de 16 km/h (ou aproximadamente 4,44 m/s) conforme divulgado pela Companhia de Engenharia de Tráfego como a velocidade média de tráfego de ônibus (modo de transporte majoritariamente utilizado pelos profissionais) na cidade de São Paulo para o ano de 2018.

É importante destacar que as matrizes de distância e tempo no Jsprit tiveram suas informações inseridas no formato de vetores, ou seja, cada linha das matrizes seguiu o padrão “origem, destino, distância ou tempo”.

4.1.2. Algoritmo Computacional Jsprit

Como já mencionado, o problema de alocação de profissionais de saúde foi resolvido de maneira análoga ao problema de roteirização de veículos (VRP). Para isso foi usado o algoritmo disponibilizado sob o nome de Jsprit, um pacote computacional codificado em Java para resolver problemas de caixeiro viajante (*Traveling Salesman Problems* - TSP) e problemas do tipo VRP.

O algoritmo pode ser encontrado em uma comunidade *open source*[5] chamada GraphHopper[6]. Este site permite que um usuário utilize o Jsprit online, para resolver problemas simples e verificar como o algoritmo funciona. Além disso, há um repositório do Jsprit com diversos dados e informações sobre o algoritmo, bem como uma pasta repleta de exemplos de aplicação e das funções usadas, que podem ser adaptadas caso necessário. Para este trabalho, o arquivo Simple Example serviu para orientar o grupo inicialmente sobre como adaptar o algoritmo para o problema dos profissionais de saúde.

É importante entender que para resolução dos problemas impostos ao Jsprit, o algoritmo sempre otimiza os custos totais conforme for especificado no código. Assim, pode-se especificar um custo por distância, por tempo de deslocamento, por serviço prestado ou outras especificidades que cada problema tenha, dentro das funções existentes no algoritmo.

---

[5] *Open source* é um termo em inglês que significa código aberto. Isso diz respeito ao código-fonte de um software, que pode ser adaptado para diferentes fins.
[6] https://www.graphhopper.com/open-source/

O IDE utilizado no trabalho para execução do Jsprit foi o Eclipse 2019-06, de download gratuito. As bibliotecas necessárias para execução do Jsprit no Eclipse são mathCommons3[7], jFreeChart[8], graphStream[9] e Simple Logging Facade for Java (SLF4J[10]).

O código SimpleExample é apresentado no Anexo I. Nele, são configurados quatro clientes em localizações distintas, que recebem cada um uma entrega, e um único depósito, de onde podem partir a quantidade de veículos necessária para que todas as entregas sejam feitas. Para esse exemplo, um limitante é a capacidade de cada veículo, limitada a dois itens por veículo. Além disso, os veículos devem fazer suas entregas e retornar ao ponto de partida.

Ao executar o código, a ferramenta computacional retorna uma tabela e duas imagens. Pela tabela é possível ver quantos veículos saem do único depósito (nesse caso dois veículos) e qual veículo atende cada cliente e em que ordem. Nas imagens verifica-se a localização de cada integrante do problema, e as rotas que cada veículo realiza. Enquanto na primeira imagem (Figura 2) o sentido das rotas é facilmente identificável pelas setas, na segunda (Figura 3) o início da rota é representado pelo triângulo.

[7] https://commons.apache.org/proper/commons-math/
[8] http://www.jfree.org/jfreechart/
[9] http://graphstream-project.org/
[10] http://www.slf4j.org/download.html

indicator	value
noJobs	4
noServices	4
noShipments	0
noBreaks	0
fleetsize	INFINITE

solution	
indicator	value
costs	35.3238075793812
noVehicles	2
unassgndJobs	0

detailed solution						
route	vehicle	activity	job	arrTime	endTime	costs
1	vehicle	start	-	undef	0	0
1	vehicle	service	2	6	6	6
1	vehicle	service	1	12	12	12
1	vehicle	end	-	18	undef	18
2	vehicle	start	-	undef	0	0
2	vehicle	service	4	6	6	6
2	vehicle	service	3	12	12	12
2	vehicle	end	-	18	undef	18

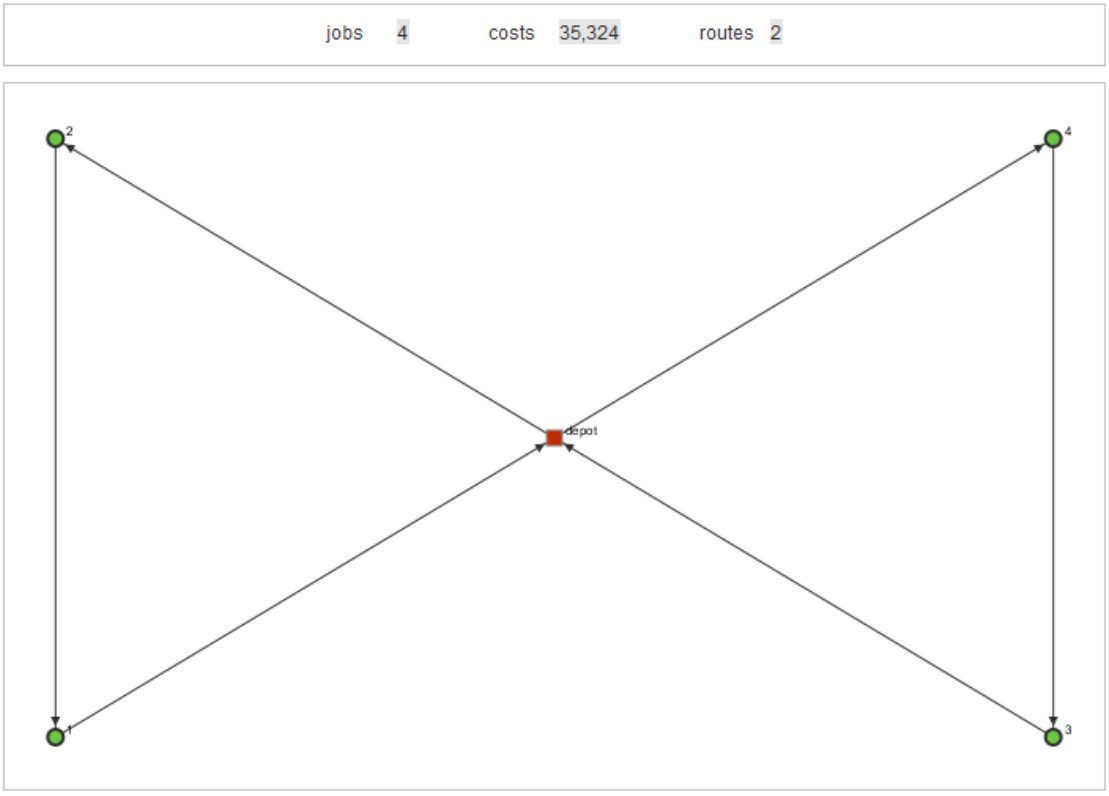

Figura 2 – Solução boa do SimpleExample

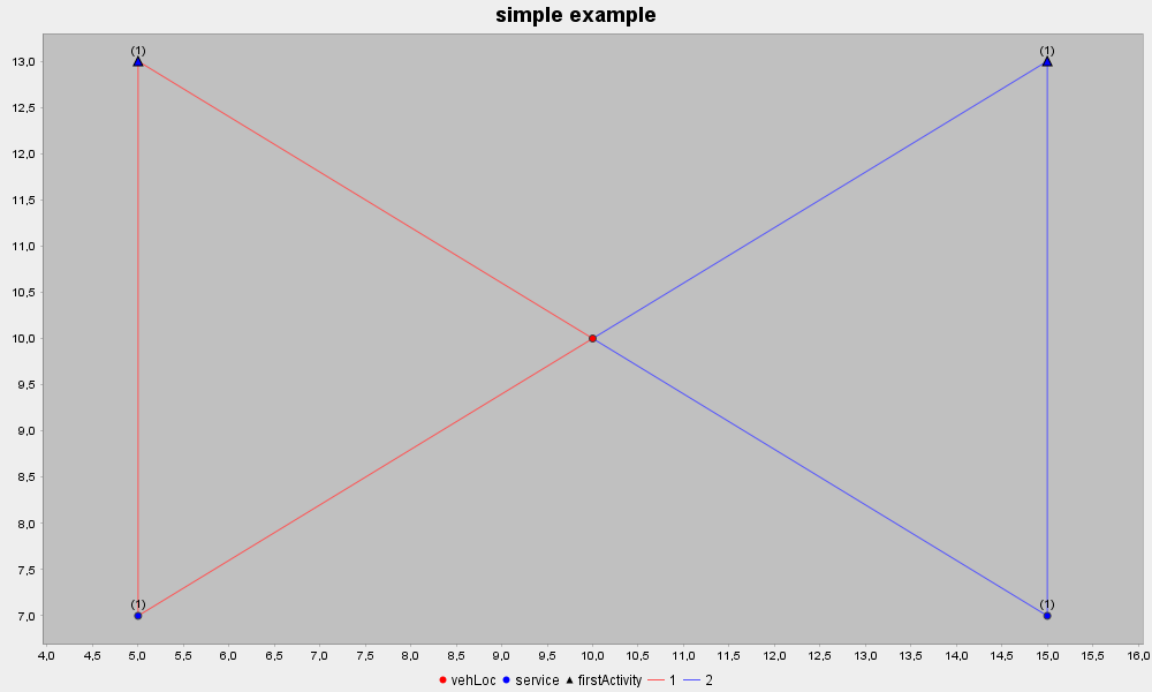

Figura 3 – Imagem de resultado do SimpleExample em formato .png

O SimpleExample auxiliou no entendimento do algoritmo, porém, como pode ser visto no seu código, ele é simples e não considera todas as especificidades de um problema real. Além disso, o código considera distâncias euclidianas, o que pode gerar respostas que não correspondem à realidade. Desse modo, foi necessário analisar outro exemplo que exemplificasse a leitura de distâncias reais entre pontos de um arquivo externo e que possibilitasse a alteração dos atributos de otimização do Jsprit.

Para considerar as distâncias reais entre os pontos e o tempo de deslocamento entre eles foi utilizada uma nova função que consta no código presente no exemplo CostMatrixExample, apresentado no Anexo II. O resultado do código é a quantidade de veículos utilizados e o custo da melhor solução apresentada.

Entendido o funcionamento do algoritmo, o código foi adaptado para que houvessem diversos depósitos dos quais partisse apenas um veículo (profissional de saúde). A posição geográfica de cada depósito foi obtida através do arquivo .txt criado após a conversão dos endereços pelo Geodesix.

Ainda no novo código, nomeado como VRP_HomeCare_Rede_Trafego, apresentado no Anexo III, o veículo foi configurado com custo de deslocamento de 0, custo de serviço de 0 e custo de tempo de deslocamento 1, desse modo, o algoritmo aloca os pacientes reduzindo o tempo de deslocamento total gasto por cada profissional em um dia. Levando isso em consideração, há as limitações de tempo já citadas de 8 horas (ou 28800 segundos) de atendimento por profissional e de 1h20min (ou 4800 segundos) para o tempo médio de serviço por paciente. Caso sejam considerados novos tipos de atendimentos, ou se queira otimizar a roteirização de outra maneira, aqueles custos devem ser adaptados.

Seguida da criação dos profissionais e dos pacientes no código, foi inserido o código para leitura das matrizes de distância e tempo, cujas informações foram gravadas dentro de uma matriz de custos em formato próprio do Jsprit próprio código. É importante ressaltar que no momento de criação dos profissionais e pacientes o nome de cada um deve corresponder ao nome dado na matriz para os pontos de origem e destino, seguindo o padrão “origem, destino, distância ou tempo”, conforme o exemplo da Figura 4, para que o algoritmo possa considerar corretamente as informações.

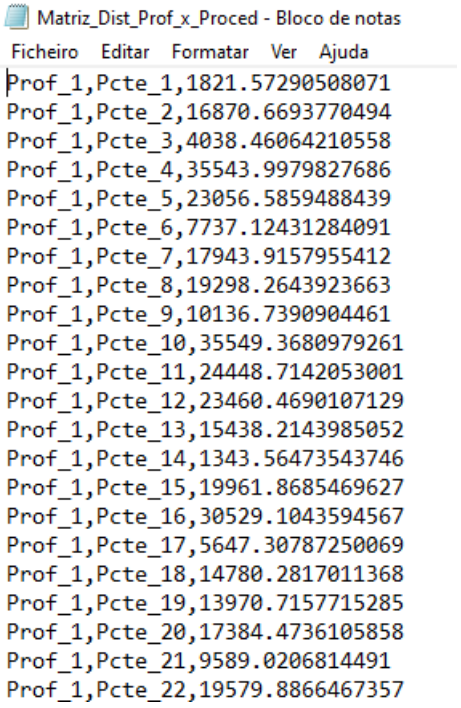

Figura 4 – Matriz de distâncias em metros entre profissional 1 e pacientes até o 22

Dentro do código cabe destacar algumas funções e preenchimentos. Com relação à criação dos tipos de veículos/profissionais disponíveis ou qualificados:

- A função “addCapacityDimension” contém duas variáveis: “WEIGHT_INDEX” se refere ao peso do pacote levado dentro do veículo, para o problema tratado no trabalho, esse valor é igual a zero uma vez que isso não é um limitante para o serviço; “capacity” se refere a quantos pacotes o veículo pode levar considerando seu limite de peso, para o problema tratado esse valor escolhido é um valor alto arbitrado (igual a 80) para que essa característica também não seja um limitante;

- As funções de “setCost” tiveram seus valores definidos conforme o objetivo da análise do trabalho, no caso minimizar o tempo total de deslocamento dos profissionais de modo a otimizar a quantidade de atendimentos por profissional, sendo justamente o motivo pelo qual “setCostPerTransportTime” é a única função com valor diferente de zero;

- Pode ser definido ainda no tipo de veículo uma velocidade máxima, desconsiderada neste trabalho pois na criação das matrizes de tempo já foi considerada a velocidade média de um ônibus em São Paulo, e o tempo de espera foi estabelecido como nulo.

Com relação à criação dos veículos/profissionais que serão considerados na definição da solução:

- Foi definida a localização de cada ponto de partida com base no documento de coordenadas criado no momento do tratamento de dados pelo Geodesix;

- Na função “newInstance” foi definido o nome de cada veículo correspondente ao nome do profissional indicado nas matrizes de distância e tempo;

- A função “setType” define o tipo de veículo, cujo padrão equivale às características dos profissionais nos seus deslocamentos;

- A função “setLatestArrival” define o instante máximo de retorno para o ponto de partida, no caso as 8 horas de trabalho diárias;

Com relação à criação dos clientes/pacientes considerados na definição da solução:

- Semelhante ao que é feito para cada profissional, primeiramente é definido o local de cada paciente a partir das respectivas coordenadas;

- A função “newInstance”, novamente, dá o nome a cada paciente, que deve corresponder àquele definido nas matrizes para ponto de chegada e de partida (quando o profissional parte de um paciente para o outro ou para a sua casa);

- A função “addSizeDimension” é similar à função “addCapacityDimension”;

- A função “setServiceTime” define o tempo de serviço para cada atendimento. Como a análise foi feita apenas sobre os “Procedimentos”, o tempo de serviço foi definido com uma hora e vinte minutos;

Nas funções seguintes, referentes à leitura e criação da matriz de custo, basta apenas inserir o endereço do arquivo das matrizes correspondentes, com os arquivos salvos seguindo o padrão de leitura do código. Com isso, o código finalizado apresentado no Anexo III resulta no código da ferramenta computacional proposta para a alocação de profissionais otimizada.

4.1.3. Procedimento de Roteirização

Para esse trabalho, foram realizados três processos de roteirização utilizando bases de dados diferentes. Em uma primeira análise buscou-se identificar o número

máximo de pacientes que poderiam ser atendidos em um dia com a base atual de profissionais que a empresa trabalha, supondo um atendimento por dia para cada paciente e o atendimento médio padrão de uma hora e vinte minutos para “Procedimentos”.

Como a base final que se deseja utilizar apresenta somente 71 pacientes, utilizou-se também a base de dados de pacientes de “Assistência Supervisionada”, totalizando então 250 pacientes, para que não fosse necessário a utilização de endereços fictícios. Como esses pacientes já são atendidos pela empresa, supor que atendimentos novos estejam nessa área de atuação é uma aproximação razoável para os propósitos de análise do número máximo de atendimentos possível.

Nesta primeira análise foi utilizada a distância real entre os pontos de interesse, isto é, a residência dos profissionais e pacientes uma vez que o objeto de estudo é a aproximação da quantidade máxima de pacientes atendidos pela base de 37 profissionais correspondente a “Procedimentos”.

A Figura 5 representa a localização das residências dos profissionais utilizados nesta primeira análise. De cada ponto partirá apenas um profissional.

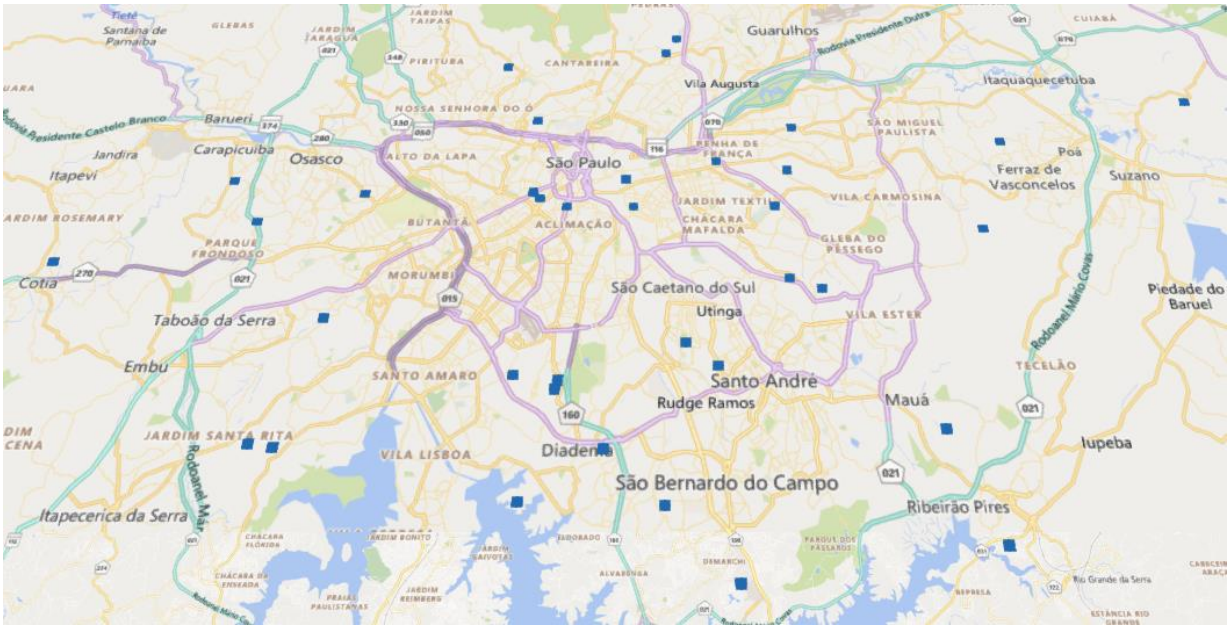

Figura 5 – Residência dos profissionais para análise do número máximo de pacientes atendidos

Em uma segunda etapa, a base de pacientes foi filtrada para contemplar apenas os 71 pacientes de “Procedimentos” para os quais se deseja resolver o problema de alocação de profissionais efetivamente.

Nessa fase espera-se obter uma solução melhor para a situação atual da empresa em termos de alocação dos profissionais, ou seja, a situação que resulta em menores tempos de deslocamento e menores distâncias percorridas pelos profissionais. A Figura 6 indica a localização dos 71 pacientes a serem atendidos pelos 37 profissionais. A Figura 7 mostra a localização de todos os envolvidos, tanto pacientes quanto profissionais.

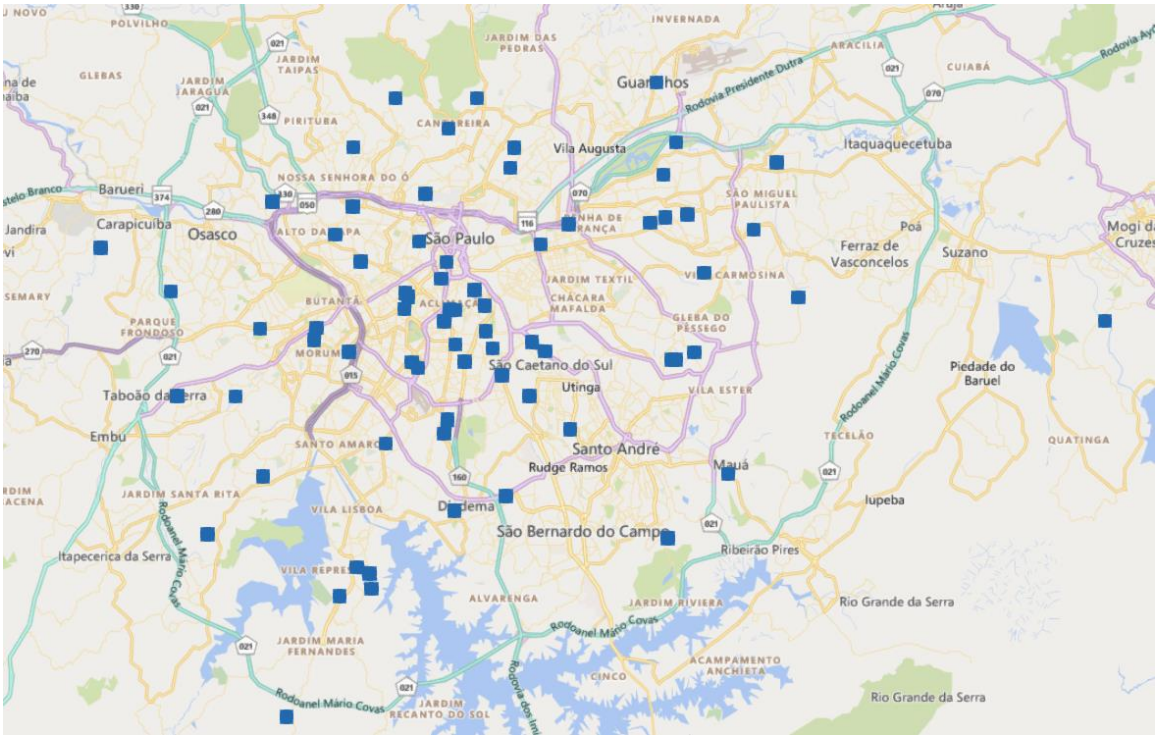

Figura 6 – Localização dos 71 pacientes a serem atendidos pelos 37 profissionais

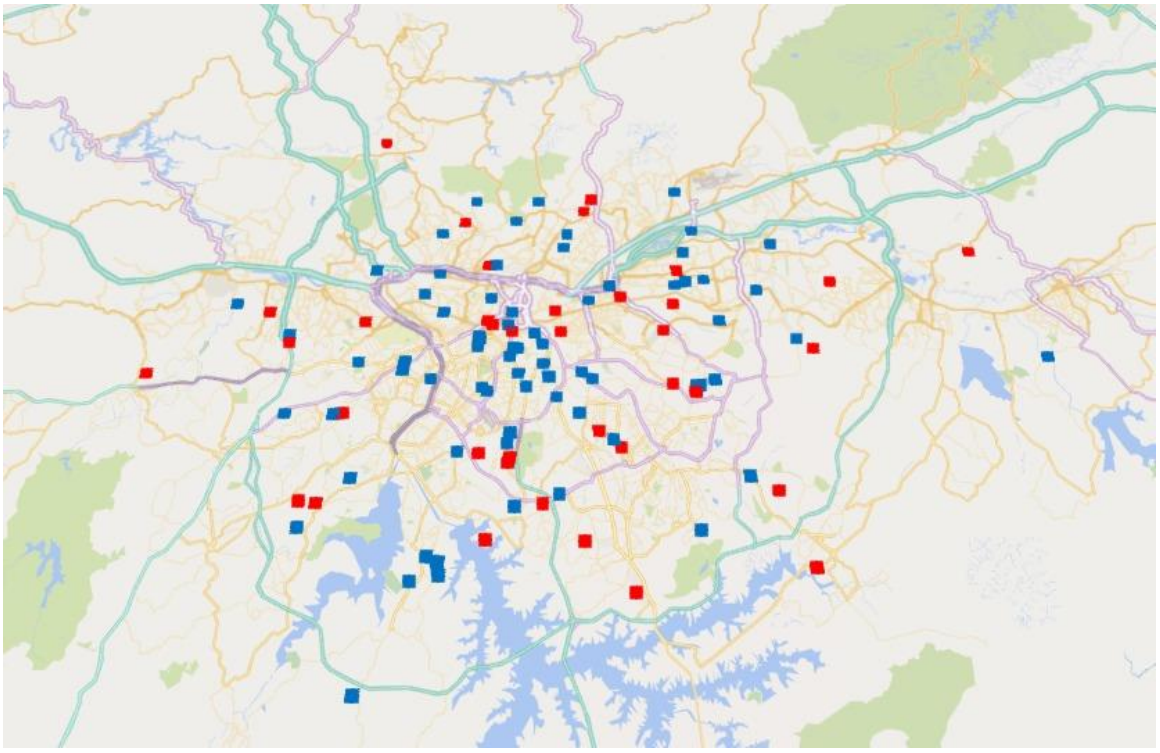

Figura 7 – Localização dos 71 pacientes (em azul) e dos 37 profissionais (em vermelho)

Após ser realizada a segunda análise, por se tratar de uma solução otimizada em relação ao que ocorre hoje na empresa, espera-se que alguns profissionais não sejam alocados a nenhum paciente, ou seja, nas condições propostas, haverá um número de profissionais acima do que o necessário para realizar os atendimentos de forma eficiente, garantindo que todos trabalhem próximo do tempo máximo possível no dia.

Com essa expectativa, deve ser determinado o número ótimo de profissionais para atender o total dos pacientes, ou seja, devem ser retirados profissionais de forma a chegar em uma equipe que realize todos os atendimentos sem que ninguém trabalhe mais horas do que o permitido ou ficasse ocioso, e sem que nenhum paciente fique sem atendimento de profissionais.

Determinar a equipe ótima analiticamente é uma tarefa difícil, já que seria necessário escolher um número de profissionais dentro de uma base de 37, ou seja, no melhor dos cenários, não havendo nenhum problema de deslocamento e cada profissional realizando uma média de 4 atendimentos (número médio informado pela empresa)

da base de 71 pacientes de “Procedimentos”, seriam necessários aproximadamente 18 profissionais para realizar todos os atendimentos sem que houvesse ociosidade de profissionais. Escolher 18 profissionais dentro de uma base de 37 é um problema de análise combinatória, sendo necessário analisar combinações em quantidade inviável para os propósitos do trabalho e, portanto, buscou-se uma alternativa que levasse em consideração os resultados da segunda análise.

Para determinar a equipe ótima de profissionais retiram-se da base todos os profissionais que não tiverem nenhum paciente alocado, esperando assim chegar em um primeiro resultado, já melhor que aquele obtido anteriormente. Este processo é repetido em sucessivas alocações até que todos os profissionais sem atendimentos designados estejam fora da base. Posteriormente, são retirados os profissionais que tiverem somente um atendimento e recalcula-se os trajetos e atendimentos. O processo deve ser repetido sucessivamente, retirando todos os profissionais com os menores números de atendimentos até que não seja possível realizar todos os atendimentos sob as restrições impostas.

5. RESULTADOS

Com a utilização do algoritmo Jsprit, em analogia ao problema VRP, foi possível então realizar a primeira atribuição de profissionais a pacientes de forma a garantir o atendimento de todos os clientes da base fornecida pela empresa.

Como explicado anteriormente, em um primeiro cenário, utilizou-se a base completa de profissionais para atender a base de pacientes da empresa para os atendimentos do tipo “Procedimentos” e “Assistência Supervisionada”. Como o foco dessa análise foi verificar o número máximo de pacientes que pudessem ser atendidos pelos 37 profissionais, independentemente de os pacientes serem de “Procedimentos” ou de “Assistência Supervisionada”, percebe-se que alguns pacientes da base não foram atendidos, porém isso não interfere nessa análise. Assim, pode-se perceber que os 37 profissionais da base podem atender até 149 pacientes em um único dia (Figura 8).

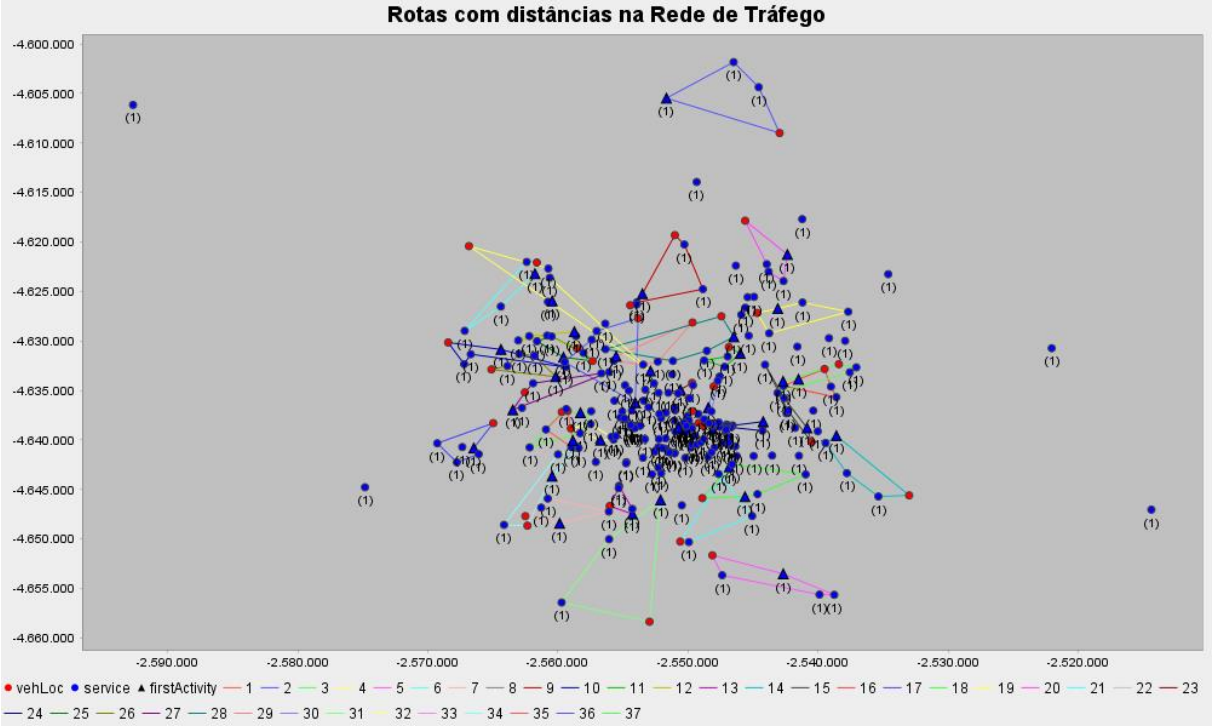

Figura 8 – Roteiro dos 37 profissionais para atender os 149 pacientes

Na segunda análise, o propósito foi verificar quantos profissionais eram necessários para atender os 71 pacientes de “Procedimentos”. Nessa análise, percebe-se, como

era esperado, uma ociosidade de profissionais de modo que para atender os 71 pacientes são necessários 26 funcionários. O resultado dessa análise pode ser verificado na Figura 9.

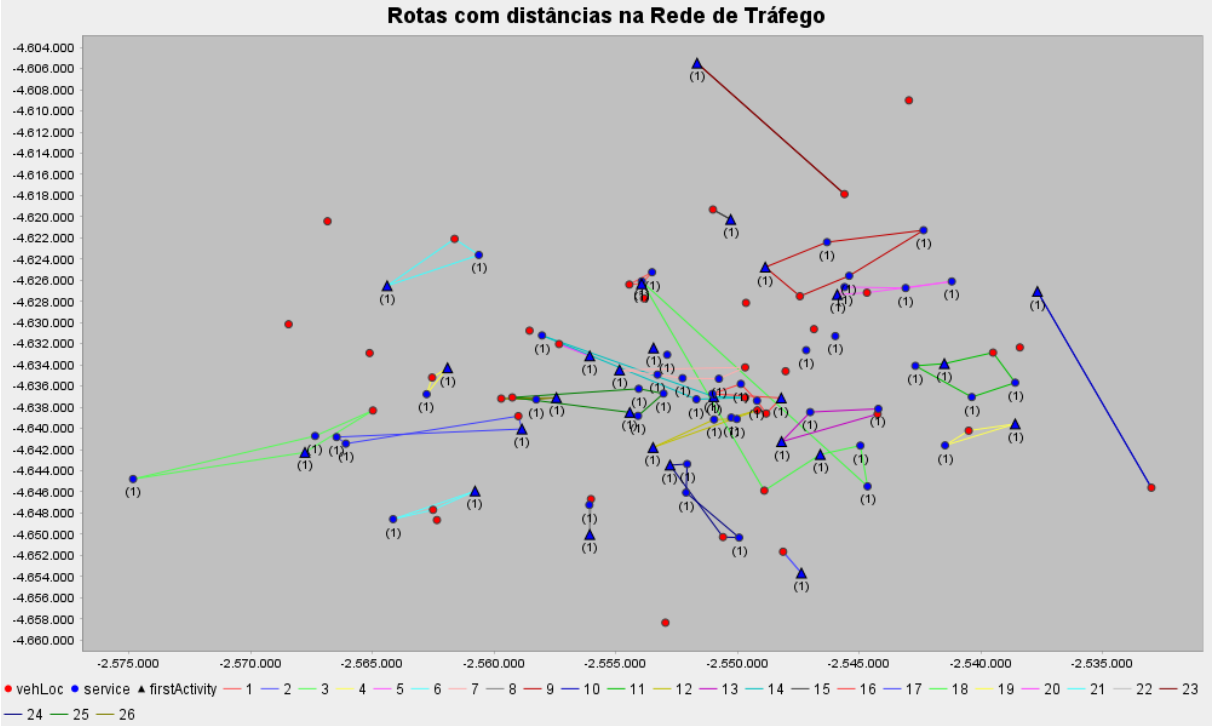

Figura 9 – Roteiro dos 26 profissionais para atender os 71 pacientes

Todos os pontos vermelhos sem conexão são profissionais que não estão realizando nenhum atendimento na simulação apresentada, ou seja, não são necessários todos esses profissionais, e a solução claramente não é ótima, já que 11 profissionais não atenderiam nenhum paciente. O resultado também pode ser visualizado como uma lista de pacientes atendidos por profissionais conforme apresentado no Anexo IV.

Cada numeração de profissional e paciente corresponde à numeração definida na própria base de dados, bem como nas matrizes de distância e tempo.

O passo seguinte foi obter o número mínimo de profissionais necessários para que todos os pacientes da base pudessem ser atendidos. Para chegar nesse resultado, foram retirados profissionais até que as condições de tempo máximo de trabalho impostas no modelo não pudessem ser satisfeitas, ou seja, não seria possível

atender a todos os pacientes, dadas as limitações colocadas, com aquele número de profissionais.

Para determinar a equipe ótima de profissionais, com base no resultado anterior, retiraram-se da base todos os profissionais que não tiveram nenhum paciente alocado, chegando assim em um primeiro resultado. Esse processo foi repetido até que todos os profissionais sem atendimentos designados estivessem fora da base. Posteriormente, foram retirados os profissionais que tiveram somente um atendimento até que não fosse possível realizar todos os atendimentos sob as restrições impostas.

Para essa situação limite, o número de profissionais necessários para atender os pacientes da base foi de 21 profissionais e o resultado pode ser observado na Figura 10.

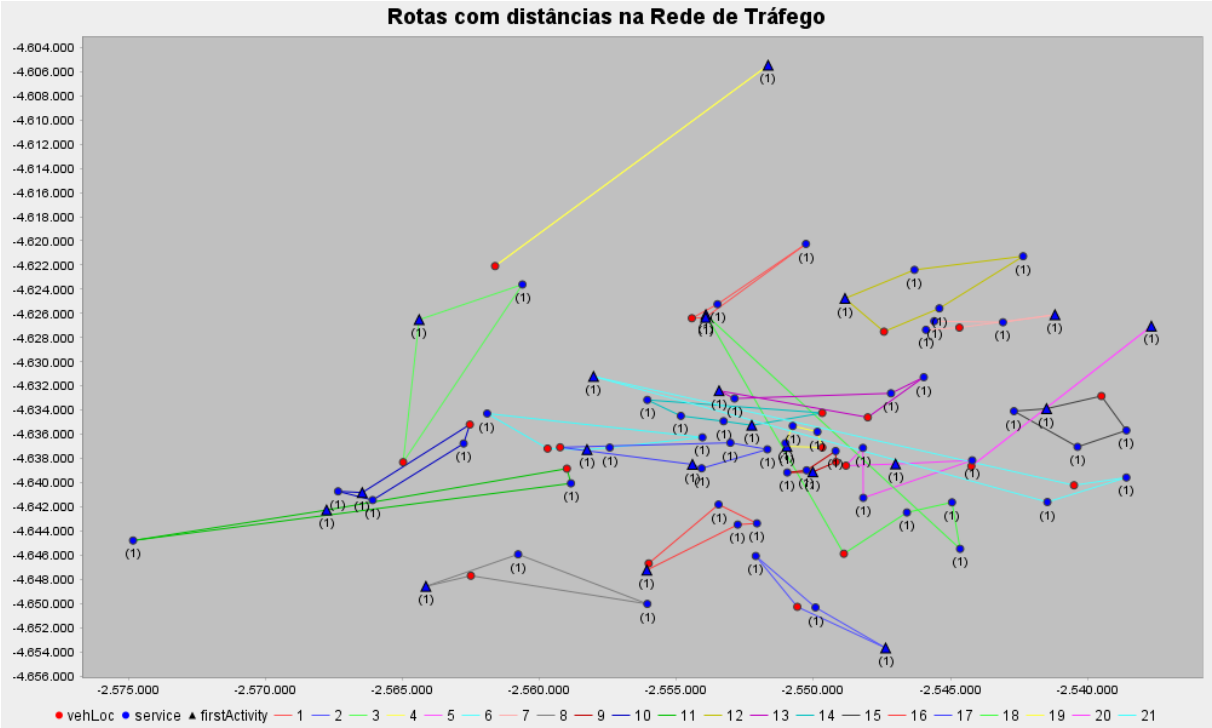

Figura 10 – Roteiro dos 21 profissionais para atender os 71 pacientes

Nesse caso, pode-se perceber que todos os profissionais (pontos vermelhos na Figura 10) realizam atendimentos de diversos pacientes. Essa é uma solução melhor

que aquela apresentada previamente por dois motivos: a não existência de profissionais ociosos e a otimização do trabalho dos profissionais durante o dia. No Anexo V é apresentada a lista de profissionais atribuídos aos pacientes.

Esses resultados podem ser comparados com o que realmente ocorre na empresa atualmente. Segundo informações da gerência de operações da empresa, hoje os enfermeiros que trabalham até 8 horas por dia realizam entre 3 e 4 atendimentos por dia. Para a solução ótima proposta, obteve-se uma média de 3,4 atendimentos por profissional, ou seja, um valor próximo daquele informado. Dessa forma, o ganho de eficiência não parece ser substancial ao tentar maximizar o número de atendimentos por profissional.

Nota-se, porém, que hoje a empresa conta com 10 pessoas no time de definição de escala de profissionais de enfermagem, tanto para procedimentos quanto para os outros produtos que a empresa oferece. O ganho real de eficiência estaria, portanto, em uma maior produtividade desses profissionais. Não necessariamente tais profissionais deveriam ser desligados, pois a escala funciona 24 horas por dia, porém, outras tarefas poderiam ser desempenhadas por esses profissionais, tornando o tempo mais eficiente.

Como o algoritmo do Jsprit é facilmente executável e, para a quantidade de atendimentos avaliados é bastante rápido em termos de tempo computacional, eventuais ajustes ao longo do dia poderiam ser feitos simplesmente retirando os pacientes que já foram atendidos ou acrescentando novos atendimentos, ajustando os pontos de partida dos profissionais e executando novamente o algoritmo.

Pode-se também executar novamente o algoritmo retirando profissionais em caso de absenteísmo e, caso não seja possível realizar todos os atendimentos, o resultado será a incapacidade de cumprir com todas as restrições, sendo deixados alguns pacientes não atendidos, porém com o menor custo possível dado o conjunto de profissionais disponíveis.

6. CONCLUSÃO

O objetivo geral deste trabalho de conclusão de curso de graduação foi propor uma ferramenta computacional capaz de alocar profissionais de saúde a pacientes no contexto de uma empresa de assistência médica domiciliar na região metropolitana de São Paulo, a partir da utilização de um algoritmo computacional construído para resolver problemas de roteirização de veículos (VRP).

A utilização do algoritmo Jsprit considerou uma analogia do problema de alocação de profissionais a pacientes ao problema de roteirização de veículos. Com essa analogia definida, pode-se construir a ferramenta em Java utilizando aquele algoritmo. A ferramenta computacional foi criada para resolver a alocação de profissionais para o serviço de “Procedimentos” da empresa e as simulações realizadas permitem observar soluções satisfatórias com foco em minimizar o tempo total de percurso dos profissionais maximizando o número de atendimentos e também automatizar essa alocação de profissionais hoje feita de maneira manual.

Apesar da ferramenta proposta resolver problemas de alocação de profissionais para um tipo de serviço apenas, a lógica utilizada é replicável para os demais serviços oferecidos pela empresa. Para isso, basta apenas atualizar a base de dados de pacientes que requerem aqueles serviços e a quantidade de profissionais disponíveis. Para ser possível resolver mais de um serviço por vez é importante definir as habilidades (ou especificidades) apresentadas para cada tipo de serviço.

Desde que a base de dados seja inserida corretamente, a função de custos (“CostMatrix”) viabiliza a execução do algoritmo e obtenção de resultados de alocação de profissionais. Além disso, como já apontado, a ferramenta computacional proposta pode considerar diferentes tipos de custos como distância ou tempo de deslocamento, tempo de serviço, tempo de paradas. Cabe ao utilizador definir aquele a ser otimizado, considerando os interesses da empresa ou dos atendentes.

7. REFERÊNCIAS BIBLIOGRÁFICAS

BARD, J. F., & PURNOMO, H. W. (2005). A column generation-based approach to solve the preference scheduling problem for nurses with downgrading. *Socio-Economic Planning Sciences*, *39*(3), 193-213.

BESTER, M. J., NIEUWOUDT, I., & VAN VUUREN, J. H. (2007). Finding good nurse duty schedules: a case study. *Journal of Scheduling*, *10*(6), 387-405.

BRUSCO, M. J., & SHOWALTER, M. J. (1993). Constrained nurse staffing analysis. *Omega*, *21*(2), 175-186.

DA CUNHA, C. B. (2003). Um modelo matemático para o problema de sequenciamento e programação de visitas de gerentes de banco. *Gestão e Produção, vol.10, no.2, São Carlos*.

DE CAUSMAECKER, P., & BERGHE, G. V. (2011). A categorization of nurse rostering problems. *Journal of Scheduling*, *14*(1), 3-16.

EASTON, F. F., & ROSSIN, D. F. (1996). A stochastic goal program for employee scheduling. *Decision Sciences*, *27*(3), 541-568.

EASTON, F. F., ROSSIN, D. F., & BORDERS, W. S. (1992). Analysis of alternative scheduling policies for hospital nurses. *Production and Operations Management*, *1*(2), 159-174.

FOWLER, J. W., WIROJANAGUD, P., & GEL, E. S. (2008). Heuristics for workforce planning with worker differences. *European Journal of Operational Research*, *190*(3), 724-740.

GILGEN, J., SILVA, V. & ISLER, C. (2018). Modelo de caracterização da escala de trabalho de operadores no terminal portuário de contêineres de Itapoá – SC. *32º Congresso de Pesquisa e Ensino em Transporte da ANPET*. Gramado - SC.

GNANLET, A., & GILLAND, W. G. (2009). Sequential and simultaneous decision making for optimizing health care resource flexibilities. *Decision Sciences*, *40*(2), 295-326.

HENDERSON, J. C., KRAJEWSKI, L. J., & SHOWALTER, M. J. (1982). An integrated approach for manpower planning in the service sector. *Omega*, *10*(1), 61-73.

HILLIER, F.S.; LIEBERMAN, G.J. (2013). Introdução à Pesquisa Operacional. *Nona Edição, Porto Alegre: AMGH.*

MUNARI, P., DOLLEVOET, T., & SPLIET, R. (2016). A generalized formulation for vehicle routing problems.

PISINGER, D., & ROPKE, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, *34*(8), 2403-2435.

SCHRIMPF, G., SCHNEIDER, J., STAMM-WILBRANDT, H., & DUECK, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*(2), 139-171.

SONG, H., & HUANG, H. C. (2008). A successive convex approximation method for multistage workforce capacity planning problem with turnover. *European Journal of Operational Research*, *188*(1), 29-48.

VENKATARAMAN, R., & BRUSCO, M. J. (1996). An integrated analysis of nurse staffing and scheduling policies. *Omega*, *24*(1), 57-71.

WARNER, D. M. (1976). Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, *24*(5), 842-856.

WINSTON, W.L. (2002). Introduction to Mathematical Programming: Applications and Algorithms. *Fourth Edition, Duxbury.*

WIROJANAGUD, P., GEL, E. S., FOWLER, J. W., & CARDY, R. (2007). Modelling inherent worker differences for workforce planning. *International journal of production research*, *45*(3), 525-553.

8. ANEXOS

8.1. Anexo I – Código SimpleExample

```
package com.graphhopper.jsprit.examples;
import com.graphhopper.jsprit.analysis.toolbox.GraphStreamViewer;
import com.graphhopper.jsprit.analysis.toolbox.GraphStreamViewer.Label;
import com.graphhopper.jsprit.analysis.toolbox.Plotter;
import com.graphhopper.jsprit.core.algorithm.VehicleRoutingAlgorithm;
import com.graphhopper.jsprit.core.algorithm.box.Jsprit;
import com.graphhopper.jsprit.core.problem.Location;
import com.graphhopper.jsprit.core.problem.VehicleRoutingProblem;
import com.graphhopper.jsprit.core.problem.job.Service;
import com.graphhopper.jsprit.core.problem.solution.VehicleRoutingProblemSolution;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleImpl;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleImpl.Builder;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleType;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleTypeImpl;
import com.graphhopper.jsprit.core.reporting.SolutionPrinter;
import com.graphhopper.jsprit.core.util.Solutions;
import com.graphhopper.jsprit.io.problem.VrpXMLWriter;
import java.io.File;
import java.util.Collection;

public class SimpleExample {

   public static void main(String[] args) {
      //some preparation - create output folder
      File dir = new File("output");
      // if the directory does not exist, create it
      if (!dir.exists()) {
         System.out.println("creating directory ./output");
         boolean result = dir.mkdir();
         if (result) System.out.println("./output created");
      }

         //a vehicle type-builder and build a type with the typeId "vehicleType" and one capacity
dimension, i.e. weight, and capacity dimension value of 2
      final int WEIGHT_INDEX = 0;
      VehicleTypeImpl.Builder vehicleTypeBuilder = VehicleTypeImpl.Builder
         .newInstance("vehicleType").addCapacityDimension(WEIGHT_INDEX, 2);
      VehicleType vehicleType = vehicleTypeBuilder.build();

         //get a vehicle-builder and build a vehicle located at (10,10) with type "vehicleType"
      Builder vehicleBuilder = VehicleImpl.Builder.newInstance("vehicle");
      vehicleBuilder.setStartLocation(Location.newInstance(10, 10));
      vehicleBuilder.setType(vehicleType);
      VehicleImpl vehicle = vehicleBuilder.build();

         //build services at the required locations, each with a capacity-demand of 1.
      Service service1 = Service.Builder.newInstance("1").addSizeDimension(WEIGHT_INDEX, 1).
            setLocation(Location.newInstance(5, 7)).build();
      Service service2 = Service.Builder.newInstance("2").addSizeDimension(WEIGHT_INDEX, 1).
            setLocation(Location.newInstance(5, 13)).build();

      Service service3 = Service.Builder.newInstance("3").addSizeDimension(WEIGHT_INDEX, 1).
            setLocation(Location.newInstance(15, 7)).build();
```

```
        Service service4 = Service.Builder.newInstance("4").addSizeDimension(WEIGHT_INDEX, 1).
                setLocation(Location.newInstance(15, 13)).build();

        VehicleRoutingProblem.Builder vrpBuilder = VehicleRoutingProblem.Builder.newInstance();
        vrpBuilder.addVehicle(vehicle);
        vrpBuilder.addJob(service1).addJob(service2).addJob(service3).addJob(service4);

        VehicleRoutingProblem problem = vrpBuilder.build();

            //get the algorithm out-of-the-box.
        VehicleRoutingAlgorithm algorithm = Jsprit.createAlgorithm(problem);

            //and search a solution
        Collection<VehicleRoutingProblemSolution> solutions = algorithm.searchSolutions();

            //get the best
        VehicleRoutingProblemSolution bestSolution = Solutions.bestOf(solutions);
        SolutionPrinter.print(problem, bestSolution, SolutionPrinter.Print.VERBOSE);

            //plot
        new Plotter(problem,bestSolution).plot("output/plot.png","simple example");

            //render problem and solution with GraphStream
        new GraphStreamViewer(problem, bestSolution).labelWith(Label.ID).setRenderDelay(200).
                display();
    }
}
```

8.2. Anexo II – Código CostMatrixExample

```
import com.graphhopper.jsprit.analysis.toolbox.Plotter;
import com.graphhopper.jsprit.core.algorithm.VehicleRoutingAlgorithm;
import com.graphhopper.jsprit.core.algorithm.box.Jsprit;
import com.graphhopper.jsprit.core.problem.Location;
import com.graphhopper.jsprit.core.problem.VehicleRoutingProblem;
import com.graphhopper.jsprit.core.problem.VehicleRoutingProblem.FleetSize;
import com.graphhopper.jsprit.core.problem.cost.VehicleRoutingTransportCosts;
import com.graphhopper.jsprit.core.problem.job.Service;
import com.graphhopper.jsprit.core.problem.solution.VehicleRoutingProblemSolution;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleImpl;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleType;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleTypeImpl;
import com.graphhopper.jsprit.core.reporting.SolutionPrinter;
import com.graphhopper.jsprit.core.util.Solutions;
import com.graphhopper.jsprit.core.util.VehicleRoutingTransportCostsMatrix;
import java.util.Collection;

public class CostMatrixExample {

   public static void main(String[] args) {

      VehicleType type = VehicleTypeImpl.Builder.newInstance("type").addCapacityDimension(0, 2)
         .setCostPerDistance(1).build();
      VehicleImpl vehicle = VehicleImpl.Builder.newInstance("vehicle")
         .setStartLocation(Location.newInstance("0")).setType(type).build();

      Service s1 = Service.Builder.newInstance("1").addSizeDimension(0, 1).setLocation(Location
         .newInstance("1")).build();
      Service s2 = Service.Builder.newInstance("2").addSizeDimension(0, 1).setLocation(Location
         .newInstance("2")).build();
      Service s3 = Service.Builder.newInstance("3").addSizeDimension(0, 1).setLocation(Location
         .newInstance("3")).build();

                  /*
       * Assume the following symmetric distance-matrix
                   * from,to,distance
                   * 0,1,10.0
                   * 0,2,20.0
                   * 0,3,5.0
                   * 1,2,4.0
                   * 1,3,1.0
                   * 2,3,2.0
                   *
                   * and this time-matrix
                   * 0,1,5.0
                   * 0,2,10.0
                   * 0,3,2.5
                   * 1,2,2.0
                   * 1,3,0.5
                   * 2,3,1.0
                   */
      //define a matrix-builder building a symmetric matrix
      VehicleRoutingTransportCostsMatrix.Builder costMatrixBuilder = VehicleRoutingTransport
            CostsMatrix.Builder.newInstance(true);
      costMatrixBuilder.addTransportDistance("0", "1", 10.0);
      costMatrixBuilder.addTransportDistance("0", "2", 20.0);
```

```
        costMatrixBuilder.addTransportDistance("0", "3", 5.0);
        costMatrixBuilder.addTransportDistance("1", "2", 4.0);
        costMatrixBuilder.addTransportDistance("1", "3", 1.0);
        costMatrixBuilder.addTransportDistance("2", "3", 2.0);

        costMatrixBuilder.addTransportTime("0", "1", 10.0);
        costMatrixBuilder.addTransportTime("0", "2", 20.0);
        costMatrixBuilder.addTransportTime("0", "3", 5.0);
        costMatrixBuilder.addTransportTime("1", "2", 4.0);
        costMatrixBuilder.addTransportTime("1", "3", 1.0);
        costMatrixBuilder.addTransportTime("2", "3", 2.0);

        VehicleRoutingTransportCosts costMatrix = costMatrixBuilder.build();

        VehicleRoutingProblem vrp = VehicleRoutingProblem.Builder.newInstance().setFleetSize
            (FleetSize.INFINITE).setRoutingCost(costMatrix)
           .addVehicle(vehicle).addJob(s1).addJob(s2).addJob(s3).build();

        VehicleRoutingAlgorithm vra = Jsprit.createAlgorithm(vrp);

        Collection<VehicleRoutingProblemSolution> solutions = vra.searchSolutions();

        SolutionPrinter.print(Solutions.bestOf(solutions));

        new Plotter(vrp, Solutions.bestOf(solutions)).plot("output/yo.png", "po");
    }
}
```

8.3. Anexo III – Código VRP_HomeCare_Rede_Trafego

```
import java.util.Collection;
import com.graphhopper.jsprit.analysis.toolbox.GraphStreamViewer;
import com.graphhopper.jsprit.analysis.toolbox.GraphStreamViewer.Label;
import com.graphhopper.jsprit.analysis.toolbox.Plotter;
import com.graphhopper.jsprit.core.algorithm.VehicleRoutingAlgorithm;
import com.graphhopper.jsprit.core.algorithm.box.Jsprit;
import com.graphhopper.jsprit.core.algorithm.state.InternalStates;
import com.graphhopper.jsprit.core.algorithm.state.StateManager;
import com.graphhopper.jsprit.core.problem.Location;
import com.graphhopper.jsprit.core.problem.VehicleRoutingProblem;
import com.graphhopper.jsprit.core.problem.job.Service;
import com.graphhopper.jsprit.core.problem.solution.SolutionCostCalculator;
import com.graphhopper.jsprit.core.problem.solution.VehicleRoutingProblemSolution;
import com.graphhopper.jsprit.core.problem.solution.route.VehicleRoute;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleImpl;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleType;
import com.graphhopper.jsprit.core.problem.vehicle.VehicleTypeImpl;
import com.graphhopper.jsprit.core.reporting.SolutionPrinter;
import com.graphhopper.jsprit.core.reporting.SolutionPrinter.Print;
import com.graphhopper.jsprit.core.util.Solutions;
import com.graphhopper.jsprit.core.problem.VehicleRoutingProblem.FleetSize;
import com.graphhopper.jsprit.core.util.Coordinate;
import java.util.ArrayList;
import com.graphhopper.jsprit.core.util.VehicleRoutingTransportCostsMatrix;
import com.graphhopper.jsprit.io.algorithm.VehicleRoutingAlgorithms;
import com.graphhopper.jsprit.core.problem.cost.VehicleRoutingTransportCosts;
import java.io.*;

public class VRP_HomeCare_Rede_Trafego {

public static void main(String[] args) throws IOException {

    VehicleRoutingProblem.Builder vrpBuilder = VehicleRoutingProblem.Builder.newInstance();

   int numeroProfissionais = 37; // Max 37;
   int numeroPacientes = 71; // Max 71;
   double EARTH_RADIUS_M_2 = 6378137;
   ArrayList<Coordinate> coordenadasProfissionais = new ArrayList<Coordinate>();
   ArrayList<Coordinate> coordenadasPacientes = new ArrayList<Coordinate>();

   // Profissional
   int numeroVeiculos = 1; // Number vehicles
   int capacity = 80;
   final int WEIGHT_INDEX = 0;

   // Read professional coordinates and create as vehicles
   String arquivoProfissionais = "C:\\Users\\lucca\\Google Drive\\TF Home Care_Pedro_Luccas\\5.
        Base de dados\\Dados_Atendentes_Proced.txt";
   InputStream fileInputStreamProfissionais = new FileInputStream(arquivoProfissionais);
   InputStreamReader inputStreamReaderProfissionais = new InputStreamReader(fileInputStream
        Profissionais);
   BufferedReader bufferedReaderProfissionais = new BufferedReader(inputStreamReader
        Profissionais);

   for (int contProfissionais = 0; contProfissionais < numeroProfissionais; contProfissionais++) {
        String stringProfissionais = bufferedReaderProfissionais.readLine();
```

```
        String[] arrayStringProfissionais = stringProfissionais.split("\\s");
        double latitudeRad = Math.toRadians(Double.parseDouble(arrayStringProfissionais[0]));
        double longitudeRad = Math.toRadians(Double.parseDouble(arrayStringProfissionais[1]));
        double latitudeDistancia = Math.sin(latitudeRad) * EARTH_RADIUS_M_2;
        double longitudeDistancia = Math.sin(longitudeRad) * EARTH_RADIUS_M_2;
        Coordinate coordenadaProfissional = Coordinate.newInstance(latitudeDistancia, longitude
            Distancia);
        coordenadasProfissionais.add(coordenadaProfissional);
    }

VehicleType vehicleType = VehicleTypeImpl.Builder.newInstance("Profissional_Procedimentos")
                                .addCapacityDimension(WEIGHT_INDEX, capacity)
                                .setCostPerDistance(0)
                                .setCostPerServiceTime(0)
                                .setCostPerTransportTime(1)
                                .build();

    for (int contProfissionais = 0; contProfissionais < numeroProfissionais; contProfissionais++) {
        for (int i = 0; i < numeroVeiculos; i++) {
            Location locationProfissional = Location.newInstance(coordenadasProfissionais
                .get(contProfissionais).getX(),coordenadasProfissionais.get(contProfissionais).getY());
            VehicleImpl veiculoProfissional = VehicleImpl.Builder.newInstance("Prof _" +
                (contProfissionais +1)).setStartLocation(locationProfissional).setType(vehicleType)
                .setLatestArrival(28800).build();
            vrpBuilder.addVehicle(veiculoProfissional);
        }
    }

    //Read patient coordinates and create as picking points
    String arquivoPacientes = "C:\\Users\\lucca\\Google Drive\\TF Home Care_Pedro_Luccas\\5.
        Base de dados\\Dados_Pacientes_Proced.txt";
    InputStream fileInputStreamPacientes = new FileInputStream(arquivoPacientes);
    InputStreamReader inputStreamReaderPacientes = new InputStreamReader
        (fileInputStreamPacientes);
    BufferedReader bufferedReaderPacientes = new BufferedReader(inputStreamReaderPacientes);

    for (int contPacientes = 0; contPacientes < numeroPacientes; contPacientes++) {
        String stringPacientes = bufferedReaderPacientes.readLine();
        String[] arrayStringPacientes = stringPacientes.split("\\s");
        double latitudeRad = Math.toRadians(Double.parseDouble(arrayStringPacientes[0]));
        double longitudeRad = Math.toRadians(Double.parseDouble(arrayStringPacientes[1]));
        double latitudeDistancia = Math.sin(latitudeRad) * EARTH_RADIUS_M_2;
        double longitudeDistancia = Math.sin(longitudeRad) * EARTH_RADIUS_M_2;
        Coordinate coordenadaPaciente = Coordinate.newInstance(latitudeDistancia,
            longitudeDistancia);
        coordenadasPacientes.add(coordenadaPaciente);
    }

    for (int contPacientes = 0; contPacientes < numeroPacientes; contPacientes++) {
        Location locationPaciente = Location.newInstance(coordenadasPacientes
            .get(contPacientes).getX(),coordenadasPacientes.get(contPacientes).getY());
        Service atendimentoPaciente = Service.Builder.newInstance("Pcte_" + (contPacientes + 1))
            .addSizeDimension(WEIGHT_INDEX,1).setServiceTime(5440).setLocation(locationPaciente)
            .build();
        vrpBuilder.addJob(atendimentoPaciente);
    }

    // Close files
```

```
bufferedReaderProfissionais.close();
bufferedReaderPacientes.close();

// Build time and distance matrix
VehicleRoutingTransportCostsMatrix.Builder costMatrixBuilder = VehicleRouting
    TransportCostsMatrix.Builder.newInstance(true);

// Matrix between patients
String arquivoMatrizPacientePaciente = "C:\\Users\\lucca\\Google Drive\\TF Home Care
    _Pedro_Luccas\\5. Base de dados\\Matriz_Tempo_Proced_x_Proced.txt";
InputStream fileInputStreamMatrizPacientePaciente = new
    FileInputStream(arquivoMatrizPacientePaciente);
InputStreamReader inputStreamReaderMatrizPacientePaciente = new InputStreamReader(
                fileInputStreamMatrizPacientePaciente);
BufferedReader bufferedReaderMatrizPacientePaciente = new BufferedReader(
                inputStreamReaderMatrizPacientePaciente);

for (int i = 0; i < numeroPacientes; i++) {
    for (int j = 0; j < numeroPacientes; j++) {
        String stringMatrizPacientePaciente = bufferedReaderMatrizPacientePaciente.readLine();
        String[] arrayStringMatrizPacientePaciente = stringMatrizPacientePaciente.split(",");
        double distanciaPacientePaciente = Double
            .parseDouble(arrayStringMatrizPacientePaciente[2]);
        String coordPacienteOrigem = "[x=" + coordenadasPacientes.get(i).getX() + "][y="
            + coordenadasPacientes.get(i).getY() + "]";
        String coordPacienteDestino = "[x=" + coordenadasPacientes.get(j).getX() + "][y="
            + coordenadasPacientes.get(j).getY() + "]";
        costMatrixBuilder.addTransportDistance(coordPacienteOrigem, coordPacienteDestino,
            distanciaPacientePaciente);
    }
}

// Matrix between patients and professionals
String arquivoMatrizProfissionalPaciente = "C:\\Users\\lucca\\Google Drive\\TF Home
    Care_Pedro_Luccas\\5. Base de dados\\Matriz_Dist_Prof_x_Proced.txt";
InputStream fileInputStreamMatrizProfissionalPaciente = new
    FileInputStream(arquivoMatrizProfissionalPaciente);
InputStreamReader inputStreamReaderMatrizProfissionalPaciente = new InputStreamReader(
    fileInputStreamMatrizProfissionalPaciente);
BufferedReader bufferedReaderMatrizProfissionalPaciente= new BufferedReader(
    inputStreamReaderMatrizProfissionalPaciente);

for (int i = 0; i < numeroProfissionais; i++) {
    for (int j = 0; j < numeroPacientes; j++) {
        String stringMatrizProfissionalPaciente =
            bufferedReaderMatrizProfissionalPaciente.readLine();
        String[] arrayStringMatrizProfissionalPaciente = stringMatrizProfissionalPaciente.split(",");
        double distanciaProfissionalPaciente =
            Double.parseDouble(arrayStringMatrizProfissionalPaciente[2]);
        String coordPacienteOrigem = "[x=" + coordenadasProfissionais.get(i).getX() + "][y="
            + coordenadasProfissionais.get(i).getY() + "]";
        String coordPacienteDestino = "[x=" + coordenadasPacientes.get(j).getX() + "][y="
            + coordenadasPacientes.get(j).getY() + "]";
        costMatrixBuilder.addTransportDistance(coordPacienteOrigem, coordPacienteDestino,
            distanciaProfissionalPaciente);
    }
}
```

```
//Close files
bufferedReaderMatrizPacientePaciente.close();
bufferedReaderMatrizProfissionalPaciente.close();

// Time matrix between patients
String arquivoMatrizTempoPacientePaciente = "C:\\Users\\lucca\\Google Drive\\TF Home Care
    _Pedro_Luccas\\5. Base de dados\\Matriz_Tempo_Proced_x_Proced.txt";
InputStream fileInputStreamMatrizTempoPacientePaciente = new
    FileInputStream(arquivoMatrizTempoPacientePaciente);
InputStreamReader inputStreamReaderMatrizTempoPacientePaciente = new
    InputStreamReader(fileInputStreamMatrizTempoPacientePaciente);
BufferedReader bufferedReaderMatrizTempoPacientePaciente = new
    BufferedReader(inputStreamReaderMatrizTempoPacientePaciente);
for (int i = 0; i < numeroPacientes; i++) {
    for (int j = 0; j < numeroPacientes; j++) {
        String stringMatrizPacientePaciente =
            bufferedReaderMatrizTempoPacientePaciente.readLine();
        String[] arrayStringMatrizPacientePaciente = stringMatrizPacientePaciente.split(",");
        double tempoPacientePaciente =
            Double.parseDouble(arrayStringMatrizPacientePaciente[2]);
        String coordPacienteOrigem = "[x=" + coordenadasPacientes.get(i).getX() + "][y="
                + coordenadasPacientes.get(i).getY() + "]";
        String coordPacienteDestino = "[x=" + coordenadasPacientes.get(j).getX() + "][y="
                + coordenadasPacientes.get(j).getY() + "]";
        costMatrixBuilder.addTransportTime(coordPacienteOrigem, coordPacienteDestino,
            tempoPacientePaciente);
    }
}

// Matriz de tempo entre profissionais e pacientes
String arquivoMatrizTempoProfissionalPaciente = "C:\\Users\\lucca\\Google Drive\\TF Home
    Care_Pedro_Luccas\\5. Base de dados\\Matriz_Tempo_Prof_x_Proced.txt";
InputStream fileInputStreamMatrizTempoProfissionalPaciente = new
    FileInputStream(arquivoMatrizTempoProfissionalPaciente);
InputStreamReader inputStreamReaderMatrizTempoProfissionalPaciente = new
    InputStreamReader(fileInputStreamMatrizTempoProfissionalPaciente);
BufferedReader bufferedReaderMatrizTempoProfissionalPaciente= new BufferedReader(
     inputStreamReaderMatrizTempoProfissionalPaciente);
for (int i = 0; i < numeroProfissionais; i++) {
    for (int j = 0; j < numeroPacientes; j++) {
        String stringMatrizProfissionalPaciente =
            bufferedReaderMatrizTempoProfissionalPaciente.readLine();
         String[] arrayStringMatrizProfissionalPaciente =stringMatrizProfissionalPaciente.split(",");
        double tempoProfissionalPaciente =
            Double.parseDouble(arrayStringMatrizProfissionalPaciente[2]);
        String coordPacienteOrigem = "[x=" + coordenadasProfissionais.get(i).getX() + "][y="
                + coordenadasProfissionais.get(i).getY() + "]";
        String coordPacienteDestino = "[x=" + coordenadasPacientes.get(j).getX() + "][y="
                + coordenadasPacientes.get(j).getY() + "]";
        costMatrixBuilder.addTransportTime(coordPacienteOrigem, coordPacienteDestino,
            tempoProfissionalPaciente);
    }
}

//Close files
bufferedReaderMatrizTempoPacientePaciente.close();
bufferedReaderMatrizTempoProfissionalPaciente.close();
```

```
    VehicleRoutingTransportCosts costMatrix = costMatrixBuilder.build(); // Create cost matrix
    vrpBuilder.setRoutingCost(costMatrix); // Set cost matrix

    vrpBuilder.setFleetSize(FleetSize.FINITE);

    //Create Problem
    VehicleRoutingProblem problem = vrpBuilder.setRoutingCost(costMatrix).build(); // Cria problema

    //Solve
    VehicleRoutingAlgorithm algorithm = Jsprit.createAlgorithm(problem); // Cria algoritmo

    Collection<VehicleRoutingProblemSolution> solutions = algorithm.searchSolutions(); //Execute
    VehicleRoutingProblemSolution bestSolution = Solutions.bestOf(solutions); //Best solution

    // Plot Solution and save to file
    SolutionPrinter.print(problem, bestSolution, Print.VERBOSE); // Plot to console and graph
    new Plotter(problem, bestSolution).plot("Rotas_Rede.png", "Rotas com distâncias na Rede de
        Tráfego"); // Plot to file
    new                                                    GraphStreamViewer(problem,
        bestSolution).setRenderDelay(50).labelWith(Label.ID).display();
       }
    }
```

8.4. Anexo IV – Resultado da segunda simulação

indicator	value
noJobs	71
noServices	71
noShipments	0
noBreaks	0
fleetsize	FINITE

indicator	value
costs	133785
noVehicles	26
unassgndJobs	0

route	vehicle	activity	job	arrTime	endTime	costs
1	Profissional_1	start	-	undef	0	0
1	Profissional_1	service	Paciente_14	493	5933	493
1	Profissional_1	service	Paciente_38	6255	11695	815
1	Profissional_1	service	Paciente_1	12500	17940	1620
1	Profissional_1	service	Paciente_3	19614	25054	3294
1	Profissional_1	end	-	26928	undef	5168
2	Profissional_2	start	-	undef	0	0
2	Profissional_2	service	Paciente_37	8346	13786	8346
2	Profissional_2	end	-	22133	undef	16693
3	Profissional_5	start	-	undef	0	0
3	Profissional_5	service	Paciente_35	582	6022	582
3	Profissional_5	service	Paciente_27	8377	13817	2937
3	Profissional_5	service	Paciente_63	14208	19648	3328
3	Profissional_5	end	-	22165	undef	5845
4	Profissional_6	start	-	undef	0	0
4	Profissional_6	service	Paciente_52	567	6007	567
4	Profissional_6	service	Paciente_15	6474	11914	1034
4	Profissional_6	end	-	12216	undef	1336
5	Profissional_8	start	-	undef	0	0
5	Profissional_8	service	Paciente_55	997	6437	997
5	Profissional_8	service	Paciente_70	7666	13106	2226
5	Profissional_8	service	Paciente_40	14460	19900	3580
5	Profissional_8	service	Paciente_53	21782	27222	5462
5	Profissional_8	end	-	28318	undef	6558
6	Profissional_9	start	-	undef	0	0

6	Profissional_9	service	Paciente_18	506	5946	506
6	Profissional_9	service	Paciente_33	9478	14918	4038
6	Profissional_9	service	Paciente_13	18340	23780	7460
6	Profissional_9	end	-	24530	undef	8210
7	Profissional_10	start	-	undef	0	0
7	Profissional_10	service	Paciente_47	544	5984	544
7	Profissional_10	end	-	6528	undef	1088
8	Profissional_11	start	-	undef	0	0
8	Profissional_11	service	Paciente_58	1689	7129	1689
8	Profissional_11	service	Paciente_49	8431	13871	2991
8	Profissional_11	service	Paciente_50	14966	20406	4086
8	Profissional_11	end	-	20680	undef	4360
9	Profissional_12	start	-	undef	0	0
9	Profissional_12	service	Paciente_60	1613	7053	1613
9	Profissional_12	service	Paciente_8	7839	13279	2399
9	Profissional_12	service	Paciente_11	14856	20296	3976
9	Profissional_12	service	Paciente_9	21891	27331	5571
9	Profissional_12	end	-	28258	undef	6498
10	Profissional_13	start	-	undef	0	0
10	Profissional_13	service	Paciente_42	2583	8023	2583
10	Profissional_13	service	Paciente_7	10209	15649	4769
10	Profissional_13	end	-	16492	undef	5612
11	Profissional_14	start	-	undef	0	0
11	Profissional_14	service	Paciente_66	1401	6841	1401
11	Profissional_14	service	Paciente_34	7866	13306	2426
11	Profissional_14	end	-	13910	undef	3030
12	Profissional_15	start	-	undef	0	0
12	Profissional_15	service	Paciente_16	1135	6575	1135
12	Profissional_15	service	Paciente_4	8323	13763	2883
12	Profissional_15	end	-	14453	undef	3573
13	Profissional_16	start	-	undef	0	0
13	Profissional_16	service	Paciente_51	77	5517	77
13	Profissional_16	service	Paciente_61	6147	11587	707
13	Profissional_16	end	-	12248	undef	1368
14	Profissional_18	start	-	undef	0	0
14	Profissional_18	service	Paciente_23	661	6101	661

14	Profissional_18	service	Paciente_43	7116	12556	1676
14	Profissional_18	end	-	13219	undef	2339
15	Profissional_20	start	-	undef	0	0
15	Profissional_20	service	Paciente_54	758	6198	758
15	Profissional_20	service	Paciente_36	7119	12559	1679
15	Profissional_20	service	Paciente_56	13167	18607	2287
15	Profissional_20	service	Paciente_24	19346	24786	3026
15	Profissional_20	end	-	25316	undef	3556
16	Profissional_21	start	-	undef	0	0
16	Profissional_21	service	Paciente_64	1705	7145	1705
16	Profissional_21	service	Paciente_46	7672	13112	2232
16	Profissional_21	service	Paciente_59	13377	18817	2497
16	Profissional_21	service	Paciente_71	19518	24958	3198
16	Profissional_21	end	-	25716	undef	3956
17	Profissional_22	start	-	undef	0	0
17	Profissional_22	service	Paciente_68	1816	7256	1816
17	Profissional_22	service	Paciente_20	8618	14058	3178
17	Profissional_22	service	Paciente_32	14333	19773	3453
17	Profissional_22	service	Paciente_2	19974	25414	3654
17	Profissional_22	end	-	25708	undef	3948
18	Profissional_24	start	-	undef	0	0
18	Profissional_24	service	Paciente_21	700	6140	700
18	Profissional_24	service	Paciente_67	6491	11931	1051
18	Profissional_24	service	Paciente_19	13352	18792	2472
18	Profissional_24	service	Paciente_41	19692	25132	3372
18	Profissional_24	end	-	26304	undef	4544
19	Profissional_25	start	-	undef	0	0
19	Profissional_25	service	Paciente_26	753	6193	753
19	Profissional_25	service	Paciente_22	7603	13043	2163
19	Profissional_25	end	-	13762	undef	2882
20	Profissional_26	start	-	undef	0	0
20	Profissional_26	service	Paciente_44	3063	8503	3063
20	Profissional_26	service	Paciente_25	10602	16042	5162
20	Profissional_26	service	Paciente_62	18738	24178	7858
20	Profissional_26	end	-	26983	undef	10663
21	Profissional_28	start	-	undef	0	0
21	Profissional_28	service	Paciente_30	537	5977	537
21	Profissional_28	end	-	6514	undef	1074

22	Profissional_29	start	-	undef	0	0
22	Profissional_29	service	Paciente_39	1492	6932	1492
22	Profissional_29	service	Paciente_45	7045	12485	1605
22	Profissional_29	service	Paciente_6	14653	20093	3773
22	Profissional_29	service	Paciente_17	21007	26447	4687
22	Profissional_29	end	-	28127	undef	6367
23	Profissional_30	start	-	undef	0	0
23	Profissional_30	service	Paciente_12	2638	8078	2638
23	Profissional_30	service	Paciente_5	8223	13663	2783
23	Profissional_30	service	Paciente_65	15179	20619	4299
23	Profissional_30	service	Paciente_29	22533	27973	6213
23	Profissional_30	end	-	28501	undef	6741
24	Profissional_31	start	-	undef	0	0
24	Profissional_31	service	Paciente_28	5583	11023	5583
24	Profissional_31	end	-	16606	undef	11166
25	Profissional_36	start	-	undef	0	0
25	Profissional_36	service	Paciente_69	1893	7333	1893
25	Profissional_36	service	Paciente_31	7463	12903	2023
25	Profissional_36	service	Paciente_57	13766	19206	2886
25	Profissional_36	service	Paciente_48	19793	25233	3473
25	Profissional_36	end	-	26987	undef	5227
26	Profissional_37	start	-	undef	0	0
26	Profissional_37	service	Paciente_10	991	6431	991
26	Profissional_37	end	-	7423	undef	1983

8.5. Anexo V – Resultado da terceira simulação

indicator	value
noJobs	71
noServices	71
noShipments	0
noBreaks	0
fleetsize	FINITE

indicator	value
costs	169595.55
noVehicles	21
unassgndJobs	0

route	vehicle	activity	job	arrTime	endTime	costs
1	Profissional_1	start	-	undef	0	0
1	Profissional_1	service	Paciente_3	1874	7314	1874
1	Profissional_1	service	Paciente_1	8988	14428	3548
1	Profissional_1	service	Paciente_38	15234	20674	4354
1	Profissional_1	service	Paciente_14	20996	26436	4676
1	Profissional_1	end	-	26928	undef	5168
2	Profissional_2	start	-	undef	0	0
2	Profissional_2	service	Paciente_44	3024	8464	3024
2	Profissional_2	service	Paciente_25	10563	16003	5123
2	Profissional_2	service	Paciente_35	20638	26078	9758
2	Profissional_2	end	-	26660	undef	10340
3	Profissional_3	start	-	undef	0	0
3	Profissional_3	service	Paciente_15	302	5742	302
3	Profissional_3	service	Paciente_52	6208	11648	768
3	Profissional_3	service	Paciente_48	13168	18608	2288
3	Profissional_3	service	Paciente_23	21171	26611	4851
3	Profissional_3	end	-	28387	undef	6627
4	Profissional_4	start	-	undef	0	0
4	Profissional_4	service	Paciente_55	997	6437	997
4	Profissional_4	service	Paciente_70	7666	13106	2226
4	Profissional_4	service	Paciente_40	14460	19900	3580
4	Profissional_4	service	Paciente_53	21782	27222	5462
4	Profissional_4	end	-	28318	undef	6558
5	Profissional_5	start	-	undef	0	0
5	Profissional_5	service	Paciente_18	506	5946	506
5	Profissional_5	service	Paciente_36	6163	11603	723
5	Profissional_5	service	Paciente_71	12165	17605	1285
5	Profissional_5	service	Paciente_56	18039	23479	1719
5	Profissional_5	end	-	24029	undef	2269

6	Profissional_6	start	-	undef	0	0
6	Profissional_6	service	Paciente_37	10782	16222	10782
6	Profissional_6	end	-	27004	undef	21564
7	Profissional_7	start	-	undef	0	0
7	Profissional_7	service	Paciente_9	927	6367	927
7	Profissional_7	service	Paciente_11	7962	13402	2522
7	Profissional_7	service	Paciente_8	14979	20419	4099
7	Profissional_7	service	Paciente_60	21205	26645	4885
7	Profissional_7	end	-	28258	undef	6498
8	Profissional_8	start	-	undef	0	0
8	Profissional_8	service	Paciente_28	8793	14233	8793
8	Profissional_8	end	-	23027	undef	17587
9	Profissional_9	start	-	undef	0	0
9	Profissional_9	service	Paciente_34	604	6044	604
9	Profissional_9	service	Paciente_12	8227	13667	2787
9	Profissional_9	service	Paciente_5	13812	19252	2932
9	Profissional_9	service	Paciente_68	20151	25591	3831
9	Profissional_9	end	-	28050	undef	6290
10	Profissional_10	start	-	undef	0	0
10	Profissional_10	service	Paciente_4	690	6130	690
10	Profissional_10	service	Paciente_16	7877	13317	2437
10	Profissional_10	service	Paciente_66	15796	21236	4916
10	Profissional_10	end	-	23790	undef	7470
11	Profissional_11	start	-	undef	0	0
11	Profissional_11	service	Paciente_51	77	5517	77
11	Profissional_11	service	Paciente_47	8333	13773	2893
11	Profissional_11	service	Paciente_61	16033	21473	5153
11	Profissional_11	end	-	22134	undef	5814
12	Profissional_12	start	-	undef	0	0
12	Profissional_12	service	Paciente_27	2863	8303	2863
12	Profissional_12	service	Paciente_62	8748	14188	3308
12	Profissional_12	service	Paciente_63	14895	20335	4015
12	Profissional_12	service	Paciente_43	22524	27964	6204
12	Profissional_12	end	-	28627	undef	6867
13	Profissional_13	start	-	undef	0	0
13	Profissional_13	service	Paciente_49	669	6109	669
13	Profissional_13	service	Paciente_50	7204	12644	1764
13	Profissional_13	service	Paciente_58	14557	19997	3677
13	Profissional_13	service	Paciente_54	21643	27083	5323
13	Profissional_13	end	-	27842	undef	6082
14	Profissional_14	start	-	undef	0	0

```
14 | Profissional_14 | service | Paciente_59 |  1392 |  6832 |  1392 |
14 | Profissional_14 | service | Paciente_46 |  7097 | 12537 |  1657 |
14 | Profissional_14 | service | Paciente_64 | 13064 | 18504 |  2184 |
14 | Profissional_14 | service | Paciente_30 | 19468 | 24908 |  3148 |
14 | Profissional_14 | end     | -           | 27008 | undef |  5248 |

15 | Profissional_15 | start   | -           | undef |     0 |     0 |
15 | Profissional_15 | service | Paciente_32 |   500 |  5940 |   500 |
15 | Profissional_15 | service | Paciente_20 |  6215 | 11655 |   775 |
15 | Profissional_15 | service | Paciente_2  | 11967 | 17407 |  1087 |
15 | Profissional_15 | service | Paciente_24 | 18057 | 23497 |  1737 |
15 | Profissional_15 | end     | -           | 24037 | undef |  2277 |

16 | Profissional_16 | start   | -           | undef |     0 |     0 |
16 | Profissional_16 | service | Paciente_21 |   700 |  6140 |   700 |
16 | Profissional_16 | service | Paciente_67 |  6491 | 11931 |  1051 |
16 | Profissional_16 | service | Paciente_19 | 13352 | 18792 |  2472 |
16 | Profissional_16 | service | Paciente_41 | 19692 | 25132 |  3372 |
16 | Profissional_16 | end     | -           | 26304 | undef |  4544 |

17 | Profissional_17 | start   | -           | undef |     0 |     0 |
17 | Profissional_17 | service | Paciente_33 |  5001 | 10441 |  5001 |
17 | Profissional_17 | service | Paciente_22 | 14847 | 20287 |  9407 |
17 | Profissional_17 | service | Paciente_26 | 21697 | 27137 | 10817 |
17 | Profissional_17 | end     | -           | 27890 | undef | 11570 |

18 | Profissional_18 | start   | -           | undef |     0 |     0 |
18 | Profissional_18 | service | Paciente_42 |  6975 | 12415 |  6975 |
18 | Profissional_18 | service | Paciente_7  | 14601 | 20041 |  9161 |
18 | Profissional_18 | end     | -           | 28472 | undef | 17592 |

19 | Profissional_19 | start   | -           | undef |     0 |     0 |
19 | Profissional_19 | service | Paciente_39 |  1492 |  6932 |  1492 |
19 | Profissional_19 | service | Paciente_45 |  7045 | 12485 |  1605 |
19 | Profissional_19 | service | Paciente_6  | 14653 | 20093 |  3773 |
19 | Profissional_19 | service | Paciente_17 | 21007 | 26447 |  4687 |
19 | Profissional_19 | end     | -           | 28127 | undef |  6367 |

20 | Profissional_20 | start   | -           | undef |     0 |     0 |
20 | Profissional_20 | service | Paciente_10 |  2115 |  7555 |  2115 |
20 | Profissional_20 | service | Paciente_29 |  9496 | 14936 |  4056 |
20 | Profissional_20 | service | Paciente_65 | 16850 | 22290 |  5970 |
20 | Profissional_20 | end     | -           | 23897 | undef |  7577 |

21 | Profissional_21 | start   | -           | undef |     0 |     0 |
21 | Profissional_21 | service | Paciente_69 |  1893 |  7333 |  1893 |
```

21	Profissional_21	service	Paciente_31	7463	12903	2023
21	Profissional_21	service	Paciente_13	13716	19156	2836
21	Profissional_21	service	Paciente_57	19848	25288	3528
21	Profissional_21	end	-	27048	undef	5288