Lucas Teles Agostinho
Rodrigo Mendonça da Paixão

Algoritimos Geneticos Aplicados no Problema de Roteirização de Veículos com Janela de Tempo

São Paulo – Brasil

2017

Lucas Teles Agostinho
Rodrigo Mendonça da Paixão

Algoritimos Geneticos Aplicados no Problema de Roteirização de Veículos com Janela de Tempo

Centro Universitário Senac

Bacharelado em Ciência da Computação

Orientador: Eduardo Heredia

São Paulo – Brasil

2017

Lista de abreviaturas e siglas

AG	Algoritmos Genéticos
API	Application Programming Interface
IA	Inteligência Artificial
CPU	Central Processing Unit
PCV	Problema do Caixeiro Viajante
PRV	Problema de Roteirização de Veículos
PRVJT	Problema de Roteirização de Veículos com Janela de Tempo
TS	Têmpera Simulada

Sumário

1 Introdução

No meio empresarial é essencial pensar na área logística, essa é a área que gerencia os recursos, matérias-primas, componentes, equipamentos, serviços, informações necessárias para execução e controle das atividades da empresa. Ela tem como foco orquestrar esses itens de forma a encontrar melhores condições de operação no menor tempo possível (DIAS, 2010).

1.1 Motivação

Com o grande crescimento populacional, a descentralização dos pontos de venda e o aumento da variedade de produtos tem provocado o crescimento e aumento da complexidade da rede de distribuição de bens e serviços. Um serviço ou produto apenas tem valor quando ele está disponível para ser consumido (TSUDA, 2007).

Um dos principais pontos dentro da logística é o transporte, onde chega a custar até 60% de seu custo total (RODRIGUES, 2007). Logo é de interesse das empresas conseguir minimizar este custo de escoamento de seus produtos. Graças essa importância no processo produtivo, a logística se tornou um grande fator competitivo entre empresas. Isso se deve ao fato que a cadeia de suprimento está relacionada com agregação de valores e disponibilidade dos seus bens e serviços para os clientes, fornecedores da empresa e os demais interessados.

No planejamento estratégico de logística um dos problemas está relacionado a roteirização de veículos (TSUDA, 2007) também conhecido como PRV. A idéia do PRV baseia-se em um conjunto de rotas que será percorrido por veículos obedecendo que cada rota começa e termina no depósito, todos os endereços são visitados somente uma vez e a demanda total de qualquer rota não pode ultrapassar capacidade dos veículos para encontrar a rota de menor tempo e distância. A identificação da ordem dos destinos, quando há um número elevado de endereços, se torna complexa por se tratar de um problema combinatório, onde é preciso avaliar todas as combinações para encontrar uma rota de menor tempo e distância (KARP, 1975).

O PRV exige um alto esforço computacional, pertencendo a classe dos problemas NP-difíceis, não pode ser solucionado em tempo polinomial, sendo uma forma de combinação da solução do problema do Caixeiro Viajante e do Problema da Mochila (OLIVEIRA, 2005b), por isso torna-se importante uma boa escolha do método a ser usado para sua solução.

Existem varias variedades de tipos de PRV, uma delas é o problema de roteamento

de veículos com janela de tempo, o PRVJT, assim como o PRV também pertence a classe NP-difíceis, por que a usa solução em tempo polinomial resulta na solução do PRV, , nele deve-se considerar um intervalo de tempo para o atendimento dos consumidores nos locais das entregas a serem realizadas, se aproximando mais do mundo real, por exemplo, não pode realizar uma entrega para uma pessoa na madrugada ou empresas que só funcionam em horário comercial.

Como exemplo de aplicações podemos citar:

- Entrega postal;
- Entrega em domicílio de produtos comprados nas lojas de varejo ou pela internet;
- Distribuição de produtos dos centros de distribuição (CD) de atacadistas para lojas do varejo;
- Escolha de rotas para ônibus escolares ou de empresas;

Para se aproximar de uma situação mais real, deve-se levar em consideração que o trânsito das grandes cidades muda constantemente, e o tempo de percorrer uma certa distância dependendo do dia e horário da semana também muda, assim como acidentes, obras em vias e etc, tornando o trânsito uma variável importante para o cálculo da rota de entrega. Tendo destinos com horários de funcionamento delimitados, pode não existir uma rota que satisfaça as restrições de horário, tornando impossível de ser encontrado uma rota que passe por todos os destinos com apenas um entregador, somando o problema de se identificar a quantidade de entregadores necessária para realizar todas as entregas respeitando todas as restrições de horários partindo do depósito.

O custo da logística em empresas que precisam realizar entregas é significativo (RODRIGUES, 2007), logo é importante avaliar formas de minimizar esse custo e consequentemente aumentar o lucro das empresas que tenham de lidar com entregas.

Os atuais resultados encontrados na literatura referentes ao PRV e PRVJT comprovam que os algoritmos exatos restringem-se à resolução de problemas-teste com tamanho reduzido e janelas de tempo apertadas. Embora hoje podemos resolver problemas com um tamanho que seja ligeiramente maior que os de alguns anos atrás, o crescimento da capacidade dos computadores e da eficiência dos algoritmos está muito distante da curva exponencial representada por este problema. Pode-se dizer que os métodos exatos não são uma alternativa viável para situações onde a um número maior de consumidores, como ocorre na maioria dos casos reais (CHABRIER, 2006).

Por isso a utilização de algoritmos genéticos, é uma meta-heurística que pode conseguir resultados satisfatórios sem a necessidade de calcular todas as combinações possíveis de rotas.

1.2 Objetivos

Desenvolver uma solução computacional utilizando algoritmos genéticos para calcular rotas de entregas, que respeite os horários de janela de tempo pré-determinados para realizar cada entrega no endereço, sem considerar a capacidade de cada entregador.

Rotas com muitos destinos e com um curto período de tempo para serem realizadas, serão separadas em rotas menores, cada rota deve ser realizada por um entregador diferente, todas as rotas menores tem como endereço inicial o depósito e avançam até o máximo de endereços possíveis no horário limite determinado.

Sendo possível determinar o número máximo de entregadores e horário de saída do depósito e horário máximo para realizar todas as entregas, caso a rota não seja possível com esse número limite entregados até o horário limite, o usuário será sinalizado.

O trânsito é considerado como alterador de tempo entre os endereços, fazendo com que a responde de rota mude dependendo do dia da semana e horário. Todas as rotas são organizadas considerando o trânsito médio.

Neste trabalho tem como objetivo a minimização da distância total percorrida e tempo para realizar o percurso, os mais comuns na literatura.

1.2.1 Objetivos Específicos

- Realizar a integração com o Google Maps, considerando o trânsito utilizando o tempo médio entre os endereços.
- Organizar os destinos em rotas utilizando Algoritmos Genéticos.
- Dividir a rota principal, em rotas menos partindo do depósito sem que ultrapassa o tempo limite.
- O número de rotas é o número de entregadores necessário para realizar a todas as entregas da rota principal.
- Criação de uma interface web para definição dos destinos, indicação do depósito, exibição em tabelas das rotas calculadas e exibição de cada rota em um mapa interativo do Google Maps.

1.3 Método de trabalho

Diferentes situações serão criadas para a simulação computacional, onde os testes fazem parte do programa, podendo ser escolhido e executado de uma maneira simples. Os endereços são reais, escolhidos em diferentes pontos no mapa e horários de abertura

e fechamento são os indicados no Google Maps para cada endereço. Utilizando essa ambiente controlado situações impossíveis devem ser rejeitas e sabendo a melhor resposta os resultados validados.

1.4 Organização do trabalho

Este trabalho é dividido em 4 capítulos. O primeiro capitulo faz uma introdução geral do problema, com a descrição dos objetivos e a motivação para a resolução do problema proposto.

O segundo capitulo trata do problema de forma separada, mostrando o que existe na literatura para uma possível solução. Também explica de forma mais detalhada o funcionamento das heurísticas e aplicações dos algoritmos genéticos para problemas semelhantes.

O terceiro capitulo é a proposta apresentada para a criação deste trabalho.

O quarto capitulo detalha implementação do programa e métodos utilizados para o seu funcionamento.

O quinto capitulo exibe os testes executados, resultados encontrados e futuras melhorias que podem ser adicionadas ao projeto.

2 Revisão de Literatura

Nesse capítulo é feita uma revisão no estado da arte dos algoritmos de roteamento de veículos a aplicação de algoritmos genéticos para mesma finalidade.

2.1 Roteamento de Veículos com Janelas de Tempo

Um dos problemas mais importantes de otimização combinatória e mais estudados na literatura de pesquisa operacional é o problema de Roteamento de Veículos com Janelas de Tempo (PRVJT). Nele consiste que tendo uma frota de veículos que deve partir de um depósito, deve atender a demanda de N consumidores e retornar ao depósito de forma que o custo total de viagem seja o mínimo. Levando em consideração que o atendimento aconteça dentro de um intervalo de tempo especificado para cada consumidor. Também deve-se respeitar a capacidade dos veículos.

Na literatura existem vários objetivos abordados pelos autores para o PRVJT. Neste trabalho temos como objetivo a minimização da distancia total percorrida, que é o mais comum na literatura. (YVES ROCHAT, 1995)

2.1.1 Formulação matemática

O PRVJT pode ser definido a partir um grafo completo orientado $G = (V, A)$ em que $V = 0, \cdots, n+1$ é um conjunto de vértices e $G = (i, j)|i, j \in V$ é o conjunto de arcos. Cada arco (i,j) é associado a um tempo t_{ij} e um custo de travessia c_{ij}.

É necessário uma definição precisa do termo *custo de travessia*. Em casos práticos pode se considerar diversos fatores, tais como distancia, tempo, desgaste do veículo ao percorrer determinado caminho, entre outros fatores. Porém, quando se trata de problemas teóricos envolvendo janelas de tempo, é comum converter todas as medidas relevantes em unidades de tempo para fins de padronização e também para facilitar a comparação entre diferentes métodos. Por isso, adota-se aqui a mesma definição de custo que a maioria dos trabalhos teóricos da literatura, considerando que o custo de viagem consiste na distância convertida em unidades de tempo.

Podemos descrever o problema como sendo um conjunto K de veículos com capacidade Q, eles devem atender n clientes, representados pelos vértices $1, \cdots, n$. Considera-se que $N = V\{0, n+1\}$ representa o conjunto de clientes. Os veículos devem partir do depósito e, após visitar todos os clientes, devem retornar ao mesmo local de onde partiram. Por conveniência, o deposito é representado por dois vértices, o vértice 0, que representa a origem, e o vértice $n+1$ que representa o destino. A cada cliente i, uma demanda q_i

é associada, esta deve ser atendida por um único veículo. E todos os vértices possuem uma janela de tempo $[e_i, l_i]$, o serviço no vértice i deve ser iniciado dentro desse intervalo. Caso ocorra que a chegada ao cliente i aconteça antes do horário previsto e_i, ele deve esperar a abertura da janela. O veículo não poderá chegar a i depois do instante l_i, pois isso faria violar a restrição de tempo do problema. Esse tipo de restrição é conhecido na literatura como janela de tempo rígida. A cada vértice é também associado um tempo de serviço, denotado por s_i. O objetivo é encontrar uma solução s de custo mínimo, de forma a minimizar a soma de todos os custos de viagem $\sum_{(i,j)\in s} c_{ij}$ que são associados aos arcos (i, j) presentes nas rotas que compõem essa solução.

A formulação matemática do PRVJT, é apresentada pelas expressões:

$$\min \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ijk} \tag{2.1}$$

Sujeito as seguintes restrições:

$$\sum_{k\in K} \sum_{j\in V} x_{ijk} = 1, \forall i \in N \tag{2.2}$$

$$\sum_{j\in V} x_{0jk} = 1, \forall k \in K \tag{2.3}$$

$$\sum_{i\in V} x_{ijk} - \sum_{i\in V} x_{jik} = 0, \forall k \in K, \forall j \in N \tag{2.4}$$

$$\sum_{i\in V} x_{i(n+1)k} = 1, \forall k \in K \tag{2.5}$$

$$\sum_{i\in N} q_i \sum_{j\in V} x_{ijk} \leq Q, \forall k \in K \tag{2.6}$$

$$b_{ik} + s_i + t_{ij} - (1 - x_{ijk})M_{ij} \leq b_{jk}, \forall k \in K, \forall (i, j) \in A \tag{2.7}$$

$$e_i \leq b_{ik} \leq l_i, \forall k \in K, \forall i \in V \tag{2.8}$$

$$x_{ijk} \in \{0, 1\}, \forall k \in K, \forall (i, j) \in A \tag{2.9}$$

A variável binária x_{ijk} assume valor 1 se o veículo k passa pelo arco (i,j) e 0, caso contrário.

A função objetivo 2.1 expressa o custo total a ser minimizado. As restrições 2.2 asseguram que somente um veículo k sai de cada cliente i. As restrições 2.3, 2.4, 2.5 garantem a continuidade do caminho a ser percorrido pelo veículo k, ou seja, cada veículo parte do depósito, visita os clientes e em seguida retorna ao depósito. As restrições 2.6 fazem com que cada veículo k somente possa atender a um conjunto de clientes cuja demanda total não ultrapasse sua capacidade Q. As restrições 2.7, 2.8 asseguram a viabilidade das rotas no que diz respeito as restrições de janelas de tempo, em que b_{ik} representa o tempo em que o veículo k começa a atender o cliente i e M_{ij} são constantes de valor suficientemente grande. As restrições 2.9 definem o domínio das variáveis de decisão. (CORDEAU et al.,)

2.1.2 Complexidade

Encontrar a solução ótima do PRVJT implica em obter simultaneamente a solução de vários problemas NP-difíceis, dentre os quais citam-se o *Problema do Caixeiro Viajante* (PCV) e o *Problema da Mochila*. Sendo assim, tal tarefa é também NP-difícil. Além disso, encontrar uma simples solução viável para o PRVJT dispondo de um conjunto limitado de veículos é NP-difícil no sentido forte (KOHL, 1995). Porém, uma solução inicial viável é trivial caso o número de veículos seja ilimitado, bastando atender cada consumidor com um veículo.

Os atuais resultados encontrados na literatura referentes ao PRVJT comprovam que os algoritmos exatos restringem-se à resolução de problemas-teste com tamanho reduzido e janelas de tempo apertadas. Embora hoje podemos resolver problemas com um tamanho que seja ligeiramente maior que o de alguns anos atrás, o crescimento da capacidade dos computadores e da eficiência dos algoritmos esta muito distante da curva exponencial representada por este problema. Pode-se dizer que os métodos exatos não são uma alternativa viável para situações onde a um número maior de consumidores, como ocorre na maioria dos casos reais. (CHABRIER, 2006)

Abordagens heurísticas e algoritmos aproximativos também tem sido utilizadas na resolução do PRVJT. As Heurísticas buscam obter uma solução em tempo hábil. Este fato torna as estratégias heurísticas muito poderosas se comparadas com abordagens exatas, que focam exclusivamente na obtenção da solução ótima. Uma boa heurística deve ser capaz de encontrar soluções próximas da ótima, em tempo bem inferior ao necessário pelos métodos exatos. A qualidade da solução não deve variar demasiadamente ao aplicá-la em diferentes ou ao mesmo problemas-teste. Até 2006, 45 do total de 56 problemas de Solomon tiveram uma solução ótima. Alguns casos foram gastos mais que cinco horas de processamento na resolução de algumas instancias, enquanto em outras puderam ser resolvidas em menos de um minuto. (JEPSEN; SPOORENDONK, 2006)

Os métodos aproximativos vem ao encontro destas características. Um método

aproximativo é uma heurística com garantia de qualidade no resultado. A melhor solução encontrada por um algoritmo de aproximação esta sempre a uma distancia percentual previamente definida da solução ótima desconhecida. A "distancia do ótimo"é particular de cada algoritmo, podendo até não ser muito relevante em termos práticos. Um exemplo bem conhecido é o algoritmo PRIM, para árvore geradora mínima, que é capaz de oferecer uma solução viável para o PCV, que é no máximo duas vezes o ótimo em distancia total percorrida (ALVARENGA, 2005).

Dado essa complexidade, resolver esse problema utilizando de abordagens puramente exatas é uma tarefa extremamente árdua, demandando tempo computacional muito elevado. Por isso é motivado o desenvolvimento de novos algoritmos heurísticos com tempos mais reduzidos para a solução do PRVJT, mesmo que esses não garantam uma solução ótima.

2.1.3 Heurísticas

Heurísticas são procedimentos de busca que visam a obtenção de soluções com uma qualidade satisfatória em um tempo computacional aceitável. Porem tais procedimentos não garantem encontrar a solução ótima nem são capazes de mensurar o quão próxima a solução obtida está da ótima. Sera enumerado as ideias centrais de algumas heurísticas construtivas e de refinamento disponíveis na literatura.

2.1.3.1 Construção de rotas

Um dos trabalhos mais antigos sobre heurística para construção de rotas para PRVJT proposto por Baker (BAKER; SCHAFFER, 1986) em 1989, Foi criada a partir da ideia da heurística das economias de Clarke e Write (CLARKE; WRIGHT, 1964) que foi proposta para criação de soluções para o PRV. O algoritmo funciona primeiramente criando uma rota partindo do depósito para cada cliente i, para em seguida, executar varias iterações, em cada interação, o algoritmo calcula quais duas rotas que podem ser combinadas de forma a gerar a maior economia possível.

Outra heurística proposta por (LANDEGHEM, 1988) também baseada na heurística das economias é uma heurística de dois critérios, nesta as janelas de tempo são utilizadas para mensurar o quanto uma ligação entre dois clientes é boa em termos de tempo.

De forma semelhante (SOLOMON, 1987) desenvolveu um algoritmo baseado na ideia na heurística das economias para resolução do PRVJT. Devido a existência de janelas de tempo, deve-se considerar também a orientação da rora. Também deve-se checar as violações de janelas de tempo quando mais de uma rota é combinada. Ela de forma igual a heurística das Economias original possuem complexidade O^2logn^2. Nesta heurística toda rota é inicializada encontrando o cliente mais próximo ao depósito que ainda não pertença a nenhuma rota. A cada iteração subsequente o cliente mais próximo ao último adicionado

à rota é considerado para inserção ao final da rota que está sendo gerado. Quando a busca falha, uma nova rota é inicializada.

As heurísticas de (LANDEGHEM, 1988) e (SOLOMON, 1987) de forma geral conseguem encontrar uma solução rapidamente. Porém as soluções que suas heurísticas encontram são geralmente de baixa qualidade. Geralmente pouco a mais de 10% do ótimo (EL-SHERBENY, 2010).

Criar uma rota por vez traz uma desvantagem, usualmente as rotas geradas por ultimo são de baixa qualidade, uma vez que os clientes sem rota tendem estar distantes geograficamente (EL-SHERBENY, 2010).

É possível encontrar uma tentativa de solução deste problema de inserção no trabalho de Rousseau (POTVIN; ROUSSEAU, 1993) por meio de construção simultânea de varias rotas. A inicialização das rotas é feita usando a heurística de inserção de Solomon. Em cada rota o cliente mais distante do depósito é selecionado como semente. A partir desse ponto, computa-se a melhor inserção viável para cada cliente que ainda não foi visitado. Este método é melhor que a heurística de Solomon, porem as soluções geradas continuam distantes das ótimas.

Antes e Derigs (ANTES; DERIGS, 1997) evoluem as ideias clássicas de inserção. No seu trabalho, todo cliente sem rota designada recebe um custo de inserção de cada uma das rotas. A definição desse custo é semelhante ao adotado nas heurísticas de Solomon. Cada cliente sem rota envia uma proposta a rota com melhor oferta, cada rota aceita a melhor proposta dos clientes com menor número de alternativas. Vale observar que mais clientes podem ser inseridos em cada iteração. Se houver alguma violação nas rotas, um certo numero de veículos é removido e o processo é reiniciado.

Os resultados do trabalhos de Antes e Derigs (ANTES; DERIGS, 1997) são comparados aqueles apresentados por Potvin e Rousseau (POTVIN; ROUSSEAU, 1993). Segundo os autores, construir rotas paralelamente produz soluções de maior qualidade que construir rotas uma a uma.

2.1.3.2 Aprimoramento de rotas

Quase todas as heurísticas de melhoria de rotas tem a noção de vizinhança. A vizinhança de uma solução S é um conjunto de soluções N(s) que podem ser geradas pela aplicação de uma única alteração denominada *movimento* na solução S.

Checar algumas ou todas as soluções de uma vizinhança pode revelar soluções melhores em relação a uma determinada função objetivo. Esta ideia pode ser repetida partindo-se da melhor solução obtida até o momento. Se em algum momento nenhuma solução melhor for encontrada em uma vizinhança, um ótimo local foi obtido. Trata-se definitivamente de um ótimo local, porem este pode eventualmente ser um ótimo global. A

este algoritmo da-se o nome de Hill Climbing. (BRäYSY; GENDREAU, 2005). Na próxima seção serão introduzidas varias estruturas de vizinhança empregadas na literatura para melhorar soluções do PRVJT. Em seguida serão descritos alguns dos algoritmos que as utilizam.

2.1.3.3 Estruturas de vizinhança

Uma estrutura de vizinhança mais utilizada em roteamento é a *k-opt*, onde k arcos são removidos e substituídos por outros k arcos. Um ótimo local obtido utilizando-se a vizinhança *k-opt* é dita solução *k-optimal*. Normalmente, k é no máximo 3.

Para todas as possíveis trocas *2-opt* e algumas das permutações da vizinhança *3-opt*, parte da rota é invertida. Isto comumente acarreta violações nas janelas de tempo. No trabalho de (POTVIN; ROBILLARD, 1995) são apresentadas duas variantes, a *2-opt** e a *Or-opt*, que mantêm a direção da rota.

Na vizinhança *Or-opt*, um conjunto contíguo de até 3 clientes é realocado para outra posição na mesma rota. Uma vez que nessa vizinhança três arcos são trocados por outros três, é fácil observar que ela é um subconjunto da vizinhança *3-opt*. Desta forma, o tamanho da vizinhança é reduzido de $O(n^3)$ para $O(n^2$. De forma geral, o tamanho da vizinhança *k-opt* é da ordem de $O(n^k)$. A vizinhança 2-opt* consiste na troca de um segmento de uma rota por um segmento de outra rota. Esses operadores de vizinhança são muitas vezes denotados na literatura por *crossover* ou simplesmente *cross*.

Movimentos da vizinhança *exchange* alteram diferentes rotas através da troca simultânea de dois clientes. A vizinhança k-node, proposta no trabalho de (CHRISTOFIDES; BEASLEY, 1984) tem sido adaptada por alguns autores para que este leve em consideração aspectos referentes às janelas de tempo. Nesta estrutura, cada cliente i é considerado e os conjuntos M1 e M2 são identificados. Em M1 são alocados o cliente i e seu sucessor j. O conjunto M2 é formado pelos clientes mais próximos aos clientes i e j que não estejam na mesma rota que i e j (encontrados pela minimização do custo de inserção considerando distância euclidiana). A vizinhança é então definida pela remoção dos elementos desses conjuntos e posterior inserção em qualquer outra possível localização. Como trata-se de uma vizinhança de dimensões muito elevadas, apenas os k candidatos mais promissores são considerados.

Outra vizinhança explorada na literatura é a $\lambda - interchange$ desenvolvida em (OSMAN, 1993), originalmente para o PRV. Trata-se de uma generalização do operador *relocate*. Nesta estrutura, um subconjunto de clientes de uma mesma rota é trocado por outro conjunto de outra rota. O mecanismo de geração $\lambda - interchange$ pode ser descrito como segue. Dada uma solução para o problema, representada pelo conjunto de rotas $S = \{r_1, \cdots, r_p, \cdots, r_q, \cdots, r_k\}$, um $\lambda - interchange$ entre um par de rotas (r_p, r_q) consiste na troca dos clientes $S_1 \bigcup r_p$ de tamanho $|S_1| \leq \lambda$ por outro subconjunto $S_2 \bigcup r_q$

de tamanho $|S_2| \leq \lambda$ para gerar novas rotas $r_p^* = (r_p - S_1) \bigcup S_2, r_p^* = (r_q - S_2) \bigcup S_1$ e uma nova solução $S' = \{r_1, \cdots, r_p^*, \cdots, r_q^*, \cdots, r_k\}$. A vizinhança $N_\lambda(S)$ de uma dada solução S é o conjunto de todos os vizinhos S' gerados para um dado valor de λ.

A vizinhança denotada por *shift-sequence* é proposta em (SCHULZE; FAHLE, 1999). Nesta, um cliente é movido de uma rota para outra checando-se todas as possibilidades de inserção. Caso uma inserção possa se tornar viável pela remoção de outro cliente j, este é removido e inserido em outra rota. Este procedimento é repetido até que a viabilidade seja restabelecida.

A Tabela 1 apresenta uma breve descrição das estruturas de vizinhança comumente utilizadas na literatura por algoritmos de busca local para a resolução do PRVJT. Observa-se que algumas dessas são também utilizadas neste trabalho

Tabela 1 – Estruturas de vizinhança para o PRVJT

Vizinhança	**Descrição**
Relocate	Move um cliente de uma rota para outra.
Exchange	Troca dois clientes entre duas rotas
$2 - opt^*$	Troca um segmento de uma rota por um segmento de outra rota.
Or-opt	Um segmento contínuo de clientes é movido de uma posição em uma rota para outra posição da mesma rota.
k-node	Os clientes i, seu sucessor j e os dois clientes mais próximos que não estão na mesma rota são removidos. Tenta-se então inserir os quatro vértices em todas as possíveis localizações. Como trata-se de uma vizinhança de dimensões muito elevadas, apenas os k candidatos mais promissores são considerados.
$\lambda - interchange$	Um subconjunto S_1 de clientes de tamanho $\|S_1\| \leq \lambda$ de uma rota é trocado por um subconjunto S_2 de tamanho $\|S_2\| \leq \lambda$ de outra rota.
Shift-sequence	Um cliente é movido de uma rota para outra checando-se todas as possibilidades de inserção. Caso uma inserção se torne viável pela remoção de um consumidor j, este é removido e inserido em alguma outra rota. Este procedimento é repetido até que a viabilidade seja restabelecida.

2.1.4 Meta-heurísticas

Meta-heurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico (RIBEIRO, 1996) Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e providas de mecanismos para tentar escapar de ótimos locais. (SOUZA, M. J. F., 2011)

2.1.4.1 Busca Tabu (Tabu Search)

Essa meta-heurística foi introduzida por (GLOVER, 1986), mas foi (GARCIA; POTVIN; ROUSSEAU, 1994) proposa primeira aplicação para o problema de roteamento e programação de veículos com janela de tempo. O conceito básico da busca tabu (BT) é explorar o espaço solução, a cada iteração, movendo de uma dada solução para outra que pertença à sua vizinhança. Diferentemente dos métodos clássicos de descida, aceita-se soluções piores, o que pode gerar ciclos. Para evitar a ciclagem, as soluções já avaliadas são marcadas como proibidas e incluídas em uma lista tabu.

2.1.4.2 Têmpera Simulada (Simulated Annealing)

A têmpera simulada (TS) é uma técnica de relaxação estocástica baseada no processo térmico utilizado na metalurgia para obtenção de estados de baixa energia num sólido. No algoritmo da TS, uma nova solução x_t é aceita sempre que $f(xt) < f(x)$, onde x é a solução corrente. Para fugir dos mínimos locais, soluções com $f(x_t) \geq f(x)$ também são aceitas com uma probabilidade $e^{\delta/T}$, onde $\delta = f(x_t) - f(x)$ e T é um parâmetro (chamado temperatura) que varia ao longo das iterações, partindo de um número grande e terminando próximo de zero. A queda na temperatura ocorre gradativamente e costuma ser feita através da regra $T_k = \alpha T_(k-1)$ para $0 \leq \alpha \leq 1$. (CHIANG; RUSSELL, 1996)

2.1.4.3 Busca Local com Múltiplos Pontos Iniciais (Multi-Start Local Search)

Este método de busca local envolve a geração de um conjunto de soluções iniciais, seguida da aplicação de um procedimento de refinamento a cada solução gerada. As diferentes soluções iniciais permitem uma diversificação do espaço de busca, o que evita ótimos locais. A heurística de inserção mais barata é usada para criar a solução inicial, que é refinada por uma extensão da heurística de cadeia de ejeções, com o objetivo de reduzir o número de veículos. Finalmente, uma modificação da troca cruzada é empregada para reduzir a distância total percorrida em cada solução. (BRäYSY; HASLE; DULLAERT, 2004)

2.1.4.4 Algoritmos Genéticos / Evolutivos

Algoritmos geneticos e evolutivos são metaheuristica com boas aplicações no problema de PRV e PRVJT, (HOMBERGER, 2005) propuseram duas estratégias evolucionárias diferentes. As duas estratégias utilizam uma aproximação estocástica baseada na heurística das economias, ou seja, os clientes pertencentes a lista das economias são escolhidos aleatoriamente para constituir a rota. A função objetivo pondera o número de rotas, a distância total e um critério que determina a facilidade de eliminação da menor rota. A mutação é feita pelas heurísticas de refinamento *Or-opt*, $2-opt*$ e $\lambda-Interchange$. No primeiro algoritmo, o cruzamento não é executado. No segundo, o cruzamento é feito através de um procedimento uniforme.

Foi desenvolvido por (JUNG; MOON, 2002) um algoritmo genético híbrido, no qual a função objetivo é baseada na distância. O algoritmo começa com a aplicação da heurística de inserção de (SOLOMON, 1987) para a determinação de uma solução inicial. O primeiro cliente de cada rota é escolhido de forma aleatória entre o cliente mais distante do depósito, o cliente com o menor instante final da janela de tempo e um cliente também determinado aleatoriamente.

Em seu algoritmo a seleção é feita pelo torneio. Para o cruzamento, o grafo que contém o depósito, os clientes e as arestas utilizadas para formar as rotas de cada veículo são mapeados e a escolha dos pontos de corte é feita por meio de curvas ou figuras geométricas de diferentes tipos A figura 1 exemplifica o procedimento de cruzamento. As regiões formadas pela sobreposição das figuras geométricas definem conjuntos de nós. Cada conjunto pertence a um único pai. Primeiramente, definimos as rotas dentro desses conjuntos. Como essa divisão de nós é arbitrária, restarão várias rotas desconexas, de modo que é preciso utilizar algoritmos de reparação para reconstruir uma solução factível. Essa reconstrução é feita seguindo a regra do vizinho mais próximo.

Na mutação, são feitas mudanças de nós entre, no máximo, 3 rotas. As heurísticas de refinamento *or-opt*, *relocate* são aplicadas ao final da iteração para melhorar a solução.

Algoritmos evolucionários para o problema com janela de tempo foram analisados e comparados por Braysy (BRÄYSY; DULLAERT; GENDREAU, 2004).

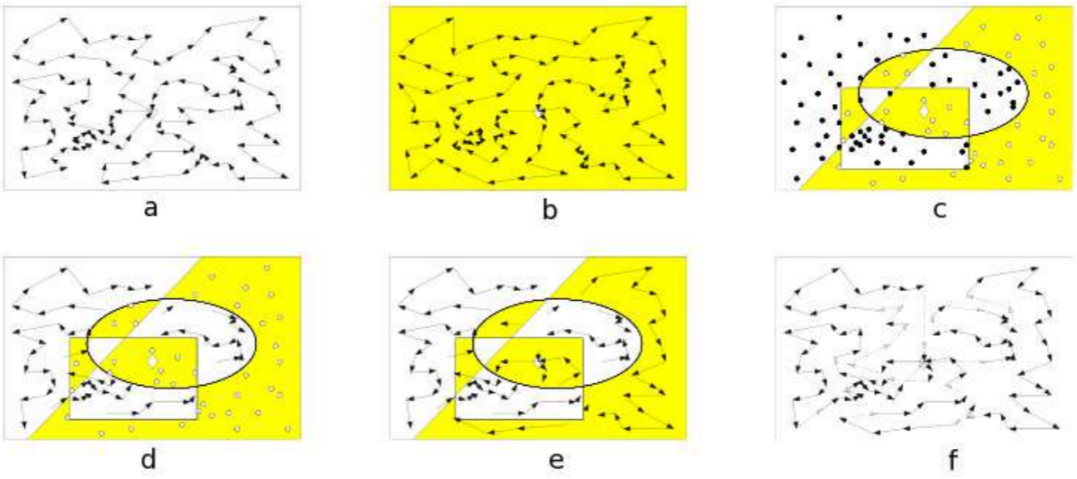

Figura 1 – O crossoover utilizado por Jung e Moon. As Figuras a e b representam, respectivamente, os pais 1 e 2. A Figura c mostra a divisão dos clientes com base em figuras geométricas. A Figura d mostra as ligações feitas nas regiões referentes ao primeiro pai, enquanto a Figura e mostra as rotas internas à região referente ao segundo pai. Finalmente, a Figura f mostra as rotas após a aplicação do algoritmo de reparação.

2.2 Algoritmos genéticos

AG é uma técnica amplamente utilizada de IA, que utilizam conceitos provenientes do princípio de seleção natural para abordar uma ampla série de problemas, geralmente de adaptação. (LUCAS, 2002)

2.2.1 Funcionamento

Inspirado na maneira como o seleção natural explica o processo de evolução das espécies, Holland (HOLLAND, 1975) decompôs o funcionamento dos AG em sete etapas, essa são *inicialização*, *avaliação*, *seleção*, *cruzamento*, *mutação*, *atualização* e *finalização* conforme a Figura 2.

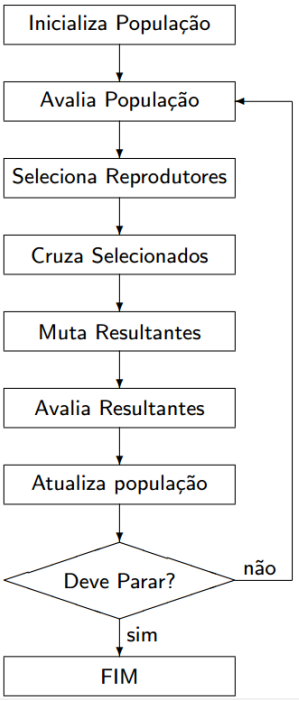

Figura 2 – Estrutura de um AG (LUCAS, 2002)

2.2.2 Inicialização

Criar uma população de possíveis respostas para um problema. É comum fazer uso de funções aleatórias para gerar os indivíduos, sendo este um recurso simples que visa fornecer maior diversidade.

2.2.3 Avaliação

Avalia-se a aptidão das soluções, os indivíduos da população, então é feita uma análise para que se estabeleça quão bem elas respondem ao problema proposto. A função de avaliação também pode ser chamada de função objetivo. Ela pode variar de acordo com problema, Calcular com exatidão completa o grau de adaptação dos indivíduos pode ser uma tarefa complexa em muitos casos, e se levarmos em conta que esta operação é repetida varias vezes ao longo do processo de evolução, seu custo pode ser consideravelmente alto. Em tais situações é comum o uso de funções não determinísticas, que não avaliam a totalidade das características do indivíduo, operando apenas sobre uma amostragem destas.

2.2.4 Seleção

Ela é a responsável pela perpetuação de boas características na espécie. Neste estágio que os indivíduos são escolhidos para posterior cruzamento, fazendo uso do grau de adaptação de cada um é realizado um sorteio, onde os indivíduos com maior grau de adaptação tem maior probabilidade de se reproduzirem. O grau adaptação é calculado a

partir da função de avaliação para cada individuo, determina o quão apto ele esta para reprodução relativo a sua população.

Selection Roulette Wheel: Faz a soma de todos os valores da função de aptidão da população, depois calcula a porcentagem de cada indivíduo referente ao total e guarda em um vetor. Então é gerado um valor X aleatório entre 0 a 1, e multiplicado pelo valor total dos pesos. Para selecionar o indivíduo é feito um loop no vetor de pesos e seus valores somados até que seja menor ou igual ao valor X aleatório calculado antes, a posição atual no vetor deste valor, será o utilizada para selecionar o individuo no vector de indivíduos. Desta forma aumentando a possibilidade de selecionar um indivíduo com melhor aptidão.

2.2.5 Cruzamento

Características das soluções escolhidas na seleção são recombinadas, gerando novos indivíduos.

Crossover OBX (Order-Based Crossover): Utiliza dois indivíduos escolhidos na seleção, então define dois números aleatórios, de 0 até menor tamanho da lista de cromossomos entre os dois, sendo que o primeiro tem que ser menor que o segundo e não podem ser iguais. O primeiro numero até o segundo numero, são definidas posições aleatórias e são salvas em uma lista. Faz um loop na lista e troca o cromossomo da posição do primeiro indivíduo para o segundo e do segundo para o primeiro.

Crossover PBX (Position-Based Crossover): Utiliza dois indivíduos selecionados, então define dois números aleatórios, de 0 até menor tamanho da lista de cromossomos entre os dois, sendo que o primeiro índice tem que ser menor que o segundo índice e os mesmos não podem ser iguais. Entre esse tamanho são definidas posições aleatórias e guardadas em uma lista. Os indivíduos resultantes são zerados, e para cada posição é trocado do cromossomo principal para o resultante de mesma posição outro da mesma posição. As posições não preenchidas são completadas com os cromossomos restante, seguindo a ordem do cromossomo e adicionado se ele não ja existir na lista.

2.2.6 Mutação

Características dos indivíduos resultantes do processo de reprodução são alteradas, acrescentando assim variedade a população. A mutação opera sobre os indivíduos resultantes do processo de cruzamento e com uma probabilidade pré-determinada efetua algum tipo de alteração em sua estrutura. A importância desta operação é o fato de que uma vez bem escolhido seu modo de atuar, é garantido que diversas alternativas serão exploradas.

MutateEM (Exchange Mutation): Define duas das posições aleatórias distintas do segundo cromossomo até o ultimo, e troca os cromossomos do indivíduo.

MutateSM (Scramble Mutation): Define duas das posições aleatórias distintas do segundo cromossomo até o ultimo, e uma quantidade aleatória. Então faz um loop da quantidade aleatória e mistura os cromossomos que estão entre a posição inicial e final trocando aleatoriamente dois pontos entre eles.

MutateDM (Displacement Mutation): Define duas das posições aleatórias distintas do segundo cromossomo até o ultimo, e remove todos os cromossomo entre essa posições e recoloca a partir de uma posição aleatória.

MutateIM (Insertion Mutation): Define uma posição aleatória, remove o cromossomo da posição, reorganiza os cromossomos e insere o cromossomo removido em uma nova posição aleatória.

MutateIVM (Inversion Mutation): Define duas das posições aleatórias distintas do segundo cromossomo até o ultimo, e inverte todos os cromossomos que está entre as posições.

MutateDIVM (Displaced Inversion Mutation):Define duas das posições aleatórias distintas do segundo cromossomo até o ultimo, e remove todos os cromossomo entre essa posições e recoloca a partir de uma posição aleatória de forma invertida.

2.2.7 Finalização

É testado se as condições de encerramento da evolução foram atingidas, retornando para a etapa de avaliação em caso negativo e encerrando a execução em caso positivo.

Os critérios para a parada podem ser vários, desde o número de gerações criadas até o grau de convergência da população atual.

Toda base dos AG se fundamenta nos indivíduos, eles são a unidade básica em qual o algoritmo se baseia, sua função é codificar as possíveis soluções do problema a ser tratado e partir de sua manipulação no processo evolutivo, a partir daí que são encontradas as respostas.

Esses indivíduos precisam de uma representação, essa será o principal responsável pelo desempenho do programa. É comum chamar de *genoma* ou *cromossomo* para se referir ao individuo. Por essa definição podemos resumir um indivíduo pelos genes que possui, ou seja seu *genótipo*.

Apesar de toda representação por parte do algoritmo ser baseada única e exclusivamente em seu genótipo, toda avaliação é baseada em seu fenótipo, o conjunto de características observáveis no objeto resultante do processo de decodificação dos genes do individuo, ver Tabela 2.

Para cada indivíduo é calculado o seu grau de adaptação, a partir de uma função objetivo, comumente denotada como na formula 2.10.

Tabela 2 – Exemplos de genótipos e fenótipos correspondentes em alguns tipos de problemas (LUCAS, 2002)

Problema	**Genótipo**	**Fenótipo**
Otimização numérica	0010101001110101	10869
Caixeiro viajante	CGDEHABF	Comece pela cidade C, depois passe pelas cidades G, D, E, H, A, B e termine em F
Regras de aprendizado para agentes	$C_1R_4C_2R_6C_4R_1$	Se condição 1 (C_1) execute regra 4 (R_4), se (C_2) execute (R_6), se (C_4) execute (R_1)

$$f_O(x) \tag{2.10}$$

Que vai representar o quão bem a resposta apresentada pelo individuo soluciona o problema proposto.

Também é calculado o grau de adaptação do indivíduo relativo aos outros membros da população a qual ele pertence, esse é chamado de grau de aptidão, para um indivíduo x temos seu grau de aptidão denotado pela fórmula 2.11.

$$f_A(x) = \frac{f_O(x)}{\sum_{i=1}^{n} f_O(i)} \tag{2.11}$$

Sendo n o tamanho da população.

A dinâmica populacional é a responsável pela evolução, ao propagar características desejáveis a gerações subsequentes no processo de cruzamento, enquanto novas são testadas no processo de mutação.

Algumas definições importantes relativo as populações de um AG são:

Geração: É o número de vezes em que a população passou pelo processo de seleção, reprodução, mutação e atualização.

Média de adaptação: É a taxa média que os indivíduos se adaptaram ao problema, é definida pela formula 2.12.

$$M_A = \frac{\sum_{i=1}^{n} f_O(i)}{n} \tag{2.12}$$

Grau de convergência: define o qual próxima esta a media de adaptação desta população relativo as anteriores. O objetivo dos AG é fazer a população convergir para uma valor de adaptação ótimo. Um estado negativo que pode ocorrer relativo a esta medida é a *convergência prematura*, a mesma ocorre quando a população converge em uma média de adaptação sub-ótima, e dela não consegue sair por causa de sua baixa diversidade.

Diversidade: Mede o grau de variação entre os genótipos da população. Ela é fundamental para o tamanho da busca. Sua queda esta fortemente ligada ao fenômeno de *Convergência prematura*.

Elite: São os indivíduos mais bem adaptados da população. Uma técnica comum nos AG é p *elitismo*, onde são selecionados k melhores indivíduos que serão mantidos a cada geração.

2.2.8 Aplicações

Existem vários aplicações para os algoritmo genéticos, por serem uma inteligência artificial não supervisionada, de rápido aprendizado e podendo ser paralelizado.

O modelo m-PRC(Problema de Rotas de Cobertura multi-veículo) é uma aplicação de algoritmos genéticos para construção de rotas em uma região mapeada, encontrando uma boa distribuição de viaturas para patrulhamento urbano, que pode ser utilizado por departamentos segurança, como a policia, guardas municipais ou segurança privada citeWashington. O Modelo é definido como um grafo não direcionado 2.13.

$$G = (V \cup W, E) \tag{2.13}$$

Onde 2.14:

$$V \cup W \tag{2.14}$$

Compõem o conjunto de vértices e E o conjunto de arestas, ou seja, o subgrafo induzido por E e um grafo completo cujo conjunto de nós é V. V são todos os vértices que podem ser visitados e é composto pelo subconjunto T, que são os vértices que devem ser visitados por algum veiculo. W é um conjunto de vértices onde todos os M veículos devem passar. M é o numero de rotas de veículos que começam no vértice base V_0.

O m-PRC atribui o conjunto de m rotas de veículos com as restrições: todas as m rotas de veículos começam e terminam na base V_0, Tem exatamente m rotas, cada vértice de V pertence a no máximo uma rota, cada vértice de T pertence a exatamente uma rota, com exceção a base, cada vértice de W deve ter uma rota que passa por ele e em uma distancia C de um vértice V visitado, O modulo da diferença entre o número de vértices

de diferentes rotas não pode exceder um determinado valor R. A Figura 3 mostra o grafo da relação de V com W.

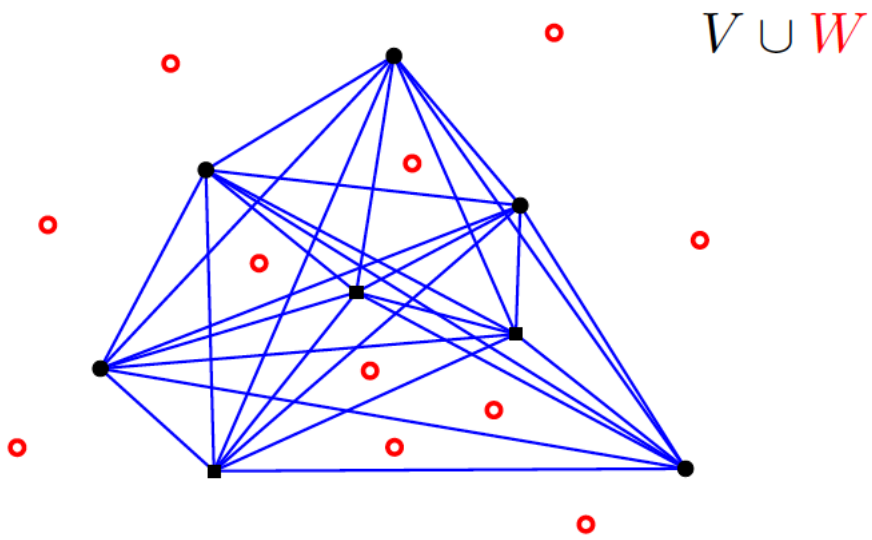

Figura 3 – Exemplo de gráfo não direcionado para V U W. (OLIVEIRA, 2009)

Para utilizar o algoritmos genéticos com o modelo m-PRC, o trabalho propõem dois modelos. O AGS (Algoritmo genético sequencial), que utiliza heurísticas GENIUS e 2-opt balanceada para ajustes finais para tentar melhor a solução; O AGH(Algoritmos genéticos H-1-PRC), que utiliza heurísticas H-1-PRC-MOD e 2-opt balanceada em todo o processo de resolução.

A conclusão de (OLIVEIRA, 2009) é que a utilização de algoritmos genéticos para a resolução de uma adaptação do problema de rotas de cobertura de veículos como bastante relevantes e de fácil manipulação. O modelo AGS resolve o problema de forma rápida e tem uma fácil implementação dentro dos critérios de comparação adotadas. O modelo AGH é mais lento e não conseguiu encontrar a solução para alguns exemplos.

Homberger, Jorg, Gehring e Hermann propuseram duas estratégias evolucionárias diferentes. As duas estratégias utilizam uma aproximação estocástica baseada na heurística das economias, ou seja, os clientes pertencentes a lista das economias são escolhidos aleatoriamente para constituir a rota. A função objetivo pondera o número de rotas, a distância total e um critério que determina a facilidade de eliminação da menor rota. A mutação é feita pelas heurísticas de refinamento Or-opt, 2-opt e λ -Interchange. No primeiro algoritmo, o cruzamento não é executado. No segundo, o cruzamento é feito através de um procedimento uniforme. (**??**)

O trabalho de Sabir Ribas na universidade federal de fluminense utilizou uma abordagem híbrida para resolver o Problema de Roteamento de Veículos, usando Algoritmos Genéticos com a metaheurística Iterated Local Search e o metodo Variable Neighborhood Descent. Chama de IILS-SP, desenvolvido em C++. O algoritmo foi submetido a 56

problemas teste de 100 clientes, para cada problema, executado 5 vezes em intervalos de 10min. Teve resultados melhores do que os encontrados pelo autor em sua pesquisa na literatura. (RIBAS, 2011).

Humberto,Germano e Guilherme também tentaram resolver o PRVJT utilizando GA, para a criação da população inicial utilizar a heurística Push-Forward Insertion Heuristic. Utilizando 56 problemas de teste e 100 clientes, chamadas instâncias de Solomon de 1987, mesma base de teste utilizadas em sua revisão de literatura para comparação de resultados, utiliza a mesma metodologia de teste do Sabir Ribas. (OLIVEIRA, 2005a)

No trabalho de Glaydston da UniAracruz e Luiz Antonio da Instituto Nacional de Pesquisas Espaciais utilizaram algoritmos genéticos para resolver Problemas de Roteamento de Veículos Dinâmico com Janelas de Tempo, em casos reais, podem existir mudanças que podem afetar as rotas, pensando nisso, o algoritmo recalcula a rota de todos os veículos. O GA deve um resultado equivalente ou superior em alguns casos se comparados com as heurísticas. (LORENA,)

3 Proposta

4 Metodologia

O software desenvolvido tem como objetivo demonstrar o problema, definindo caminho viáveis entres diferentes endereços, onde é possível realizar as entregas no período limitado, caminhos são impossíveis de ser entregues a tempo e caminhos que serão necessários mais de um entregador para ser realizado. Alertas de não possibilidade de realizar a entrega será comunicadona interface do usuário.

Utilizando a API do Google Maps como fonte de dados, informações reais de distância, tempo médio e localizações são utilizadas para uma simulação mais próxima de uma situação real. Por se tratar de entregas de pequenos porte, os testes foram criados com endereços dentro ou nas proximidades da cidade de São Paulo, um das maiores metrópoles do mundo, também tem um dos maiores índices de transito também segundo o TomTom Traffic Index (TOMTOM INTERNATIONAL BV, 2017), a menor distância de rota pode não ser a melhor escolha para certos horários do dia, as vezes escolher uma rota com maior distância que evita transito primeiro é a melhor escolha para economizar tempo.

Para preparar o calculo da rota, deve-se levar em consideração que todos os entregadores partem de uma única origem que chamamos de deposito,mas antes de começar a calcular as entregas, é calculada uma rota geral que passa por todos os endereços com apenas um entregador, essa rota é divida sempre que o horário de chega a um endereço, ultrapassar o horário limite para realizar todas as entregas. Quando uma rota é divida, primeiro entregador fica com a parte até o endereço que é possível realizar a entrega e é calculada uma nova rota geral com os endereços que sobraram, sendo passado para um novo entregador e partindo do deposito. Todas as vezes que não for possível entregador esse processo será repetido até que seja possível.

Existem situações onde não é possível entregar, mesmo com um numero alto de entregadores, essas situação podem depender da distancia, se for muito longe não tem como chegar no horário, do horário de abertura e fechamento, se forem muito próximos qualquer mudança no transito torna impossível realizar a entrega, também a situação do numero limitado de entregadores, que não pode ser um número infinito, muitos endereços para poucos entregadores pode ser impossível de se realizar a entrega.

Cada endereço tem um horário de abertura e fechamento para realizar entregas, por exemplo, um super mercado recebe nos produtos na madrugada, por que receber em seu horário normal de abertura irá atrapalhar as compras dos clientes, então esse horário deve ser considerado para a escolha do próximo endereço. Depois que chega ao endereço se verifica o horário de abertura, se chegar antes, deve-se esperar até abrir, se chegar no horário ou depois do horário, antes do fechamento, o tempo de descarga que é considerado.

Cada entrega também tem um tempo médio que demora para realizar a descarga do produto, produtos pequenos podem demorar minutos, muitos produtos demora mais para retirar do veiculo e produtos grande podem precisar ser levamos com mais demora.

Depois que uma entrega é feita, uma nova rota deve ser recalculada, agora como ponto inicial o endereço atual, todo o processo será refeito para verificar se o transito ou possível atrasos não afetaram a ordem dos próximos destinos, depois do recalculo o entregador deve seria o próximo destino indicado.

Se caso não for mais possível entregar no horário por motivos de piora de transito ou um grande tempo de atraso para descarregar, um alerta será emitido indicando que todos os destinos não podem ser visitados a tempo. Cada entregador tem seu próprio recalculo de rota, sendo que se um entregador concluir todos os destinos, os outros continuam pedindo novas rotas até que todos terminem suas entregas.

O fluxograma a baixo demostra o funcionamento do software.

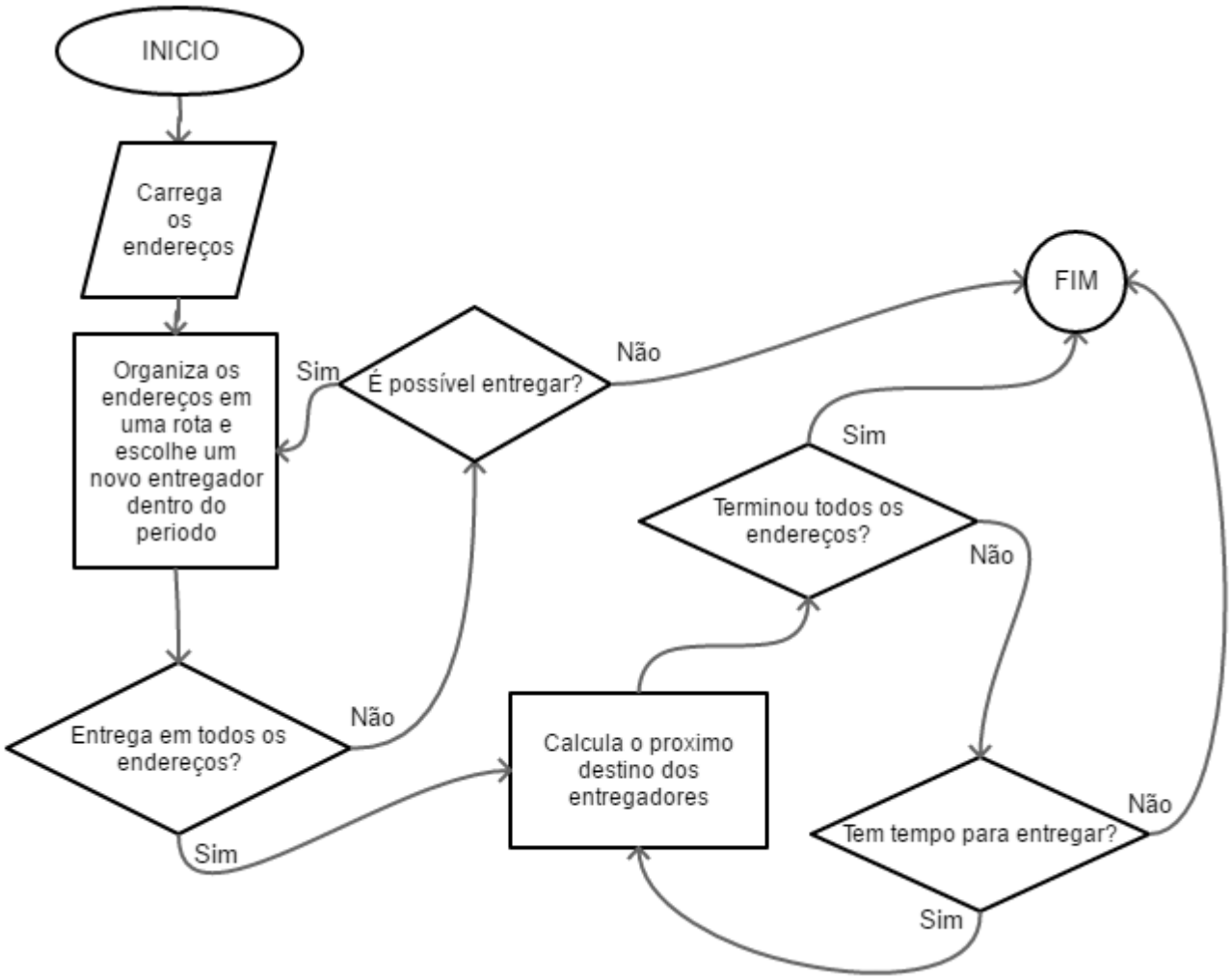

Figura 4 – Fluxograma macro do funcionamento do software.

Sempre que é necessário calcular a rota, o modulo de GA é chamado. Considerando

que um individuo é uma rota completa, gera uma população de varias rotas, onde a ordem da rota é aleatória somente mantendo o deposito fixo como primeiro endereço para criar a população inicial. Para cada rota da população de indivíduos a seleção determina dois para a realização do cruzamento, onde endereços das duas rotas são trocados de forma a criar duas novas rotas mantendo o deposito sempre como inicial. A mutação é executada individual em cada rota, mudando de posição um ou mais endereços da rota, mas sempre mantendo o deposito como ponto inicial.

Depois que todas as rotas dos indivíduos da população foram modificados, agora é hora de verificar quais são os melhores, o que define isso é a função de aptidão, todos os parâmetros da rota são agrupados em um único numero e a rota que tem o menor numero é a melhor rota da população. O valor de aptidão é definido com a soma da distância entre todos os endereços da rota, mais o tempo de cada um dos trajetos com o tempo de espera e descarga.

A função de aptidão do GA considera o horário de saída como parâmetros inicial, com isso, utiliza o tempo dado pelo Google Maps entre os pontos e soma ao horário verificando se está dentro da janela de tempo do destino. Se o horário calculo for menor que o de abertura, é somado o tempo restante de espera. Se o horário for maior que o tempo de fechamento, é somado o tempo restante de espera até a abertura no próximo dia. Sempre que não chegar em um horário entra a janela de tempo, a penalidade é o tempo de espera, fazendo o tempo total ganhe mais peso no valor de aptidão. O Valor de aptidão final é a soma da distancia em metros do percurso passando por todos os pontos, com o tempo total em minutos.

O GA foi configurado com o numero de gerações em 200, tamanho da população em 1000, melhores indivíduos por geração em 20, probabilidade de cruzamento em 50% e probabilidade de Mutação em 0,1%.

Os testes foram configurados com um numero de 6 roteiros diferentes, endereços dentro da cidade de São Paulo e cidades próximas. Utilizando 6 tipos de mutação, sendo SM, IVM, IM, EM, DM e DIVM, e 2 tipos de cruzamento, sendo PBX e OBX. Considerando o transito médio enviado pelo Google Maps, para poder demostrar o impacto do transito nos caminhos calculados. Tudo será rodado 10 vezes e será retirada uma média do valor de aptidão, por que, cada vez que roda o GA a resposta da solução pode mudar, por ele não ser determinístico.

E uma base pré-definidas rotas, para prevenir possíveis problemas sera ignorado o transito atual. Ja que o mesmo altera dependendo das condições do clima ou horário do dia. Então utilizando o Google Maps, um cache inicial foi preparado e o software utiliza simulando uma buscar ao Google Maps, com isso, a informação é obtida mais rapidamente e sempre fixa para garantir a resposta pré-determinada do teste.

O fluxograma a baixo demostra o funcionamento do software utilizando o GA.

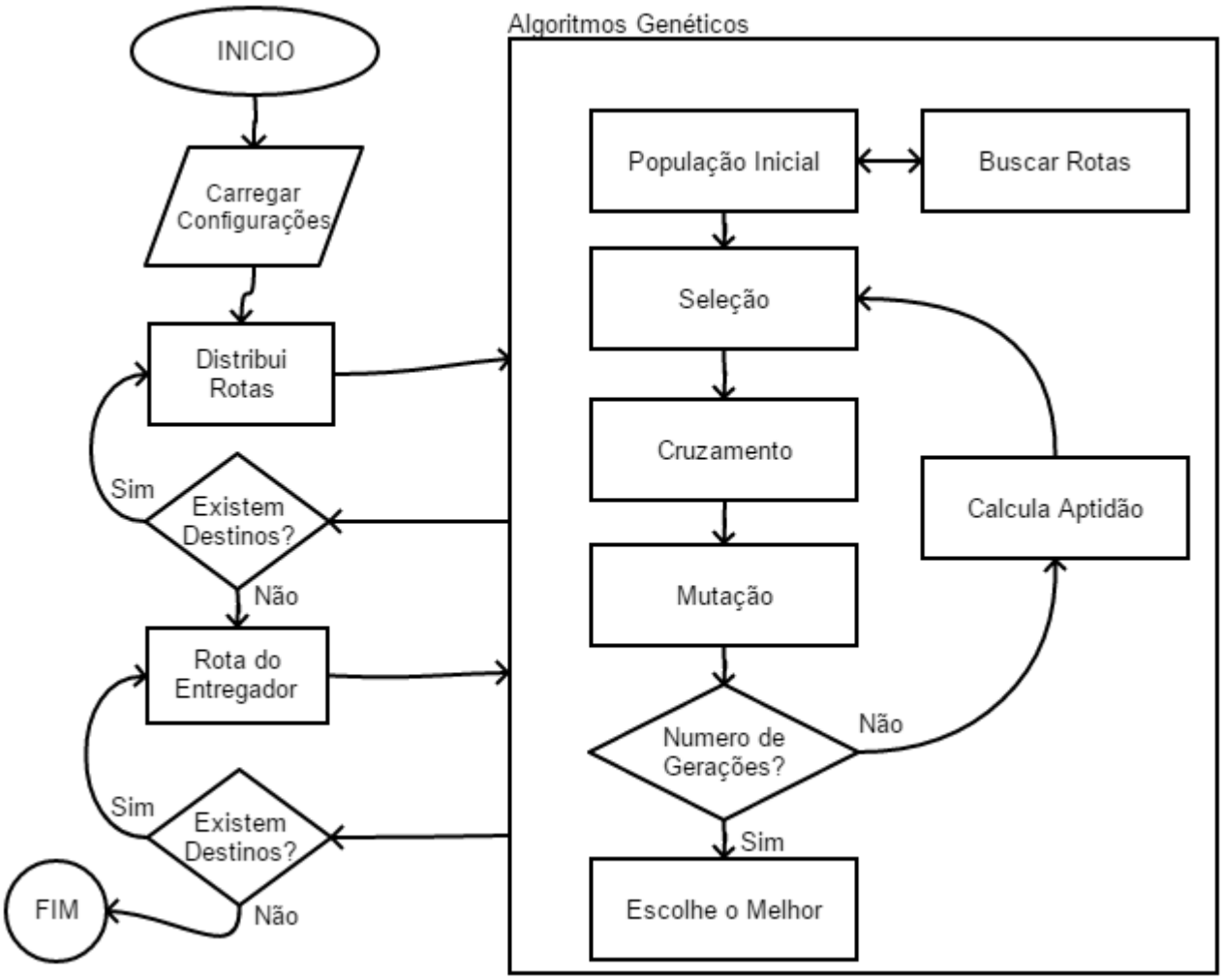

Figura 5 – Fluxograma macro da integração com o GA.

5 Implementação

Nesse capitulo será apresentado mais aprofundadamente as tecnologias, ferramentas e métodos que foram utilizados para a implementação do algoritmo genético para busca de rota com janela de tempo.

5.1 Tecnologias

O software foi desenvolvido na linguagem C# com Visual Studio 2017 com o Framework .Net Core 2.2 compatível com Windows, Linux e MacOs para o Back-End e ReactJs para a interface de utilização. Toda comunicação da interface com o calculador de rotas é por WebAPi Rest, onde são trocados dados em formato de Json.

5.2 Estrutura do Projeto

Para facilitar o entendimento inicial do projeto, é importante ter uma orientação de onde está cada parte de seu comportamento. A Solution do Visual Studio é divida em 4 projetos, onde eles são:

5.2.1 PathFinder

Projeto com função de utilização do software em geral, agrupamento informações de dados iniciais e organizando o retorno para a exibição.

Program: Implementação das união de todas as classes, executa a separação dos entregadores.

Entregador: Informação individual, armazena a rota completa do entregar e o genoma representante.

TimeMeasure: Configuração de registro/exibição do tempo de processamento.

5.2.2 RouteGA

Projeto de WebApi para a comunicação com a interface, seu objetivo é converter os dados recebidos pelo interface e chamar métodos do projeto **PathFinder**, convertendo o retorno para uma melhor estrutura que será exibida na interface.

RouteController: Implementação da comunicação da Api, recebe a requisição da interface, executa métodos do PathFinder e retorna uma lista da classe EntregadorView-Model para a interface.

EntregadorViewModel: Classe de retorno para a interface agrupando todas as informações da rota do entregador.

LocalViewModel: Classe de retorno para a interface agrupando todas as informações de um endereço.

RotaViewModel: Classe de retorno para a interface agrupando todas as informações de uma rota.

RoteiroViewModel: Classe de recebimento da interface agrupando todas as informações configuradas na interface para o calculo da rota.

5.2.3 PathFinder.Routes

Nesse projeto estão os algoritmos de comunicação com a API Google Maps, classes para serialização dos retornos da API e classes para o agrupamento dos dados das rotas. Esse projeto é utilizado pelo genoma do GA para o recalculo das rotas

GoogleMaps: Classes de estrutura para serialização do Json de retorno da API do Google Maps.

MapPoint: Classe com informações do ponto de destino com latitude, longitude e janela de tempo para a entrega.

Period: Classe para a definição da janela de tempo.

Route: Classe para definição da origem e destino de uma rota, contendo o tempo e a distância entra os pontos.

RouteMap: Classe para a definição da rota completa e o deposito. A definição de cada caminho completo é controlado nessa classe.

SearchRoute: Classe para a integração com a API do Google maps.

5.2.4 PathFinder.GeneticAlgorithm

Neste projeto são definidos todas as implementações referentes ao algoritmo genético, pela complexidade do algoritmo ele possui uma estrutura própria de pastas com algoritmos de seleção, cruzamento, mutação e a função de aptidão para definições e configurações de injeção de dependência.

5.2.5 Abstraction

Nesta pasta estão todos os arquivos a nível de abstração das etapas do algoritmo genético.

ISelection: Interface responsável por abstrair os algoritmos de seleção.

IGenome: Interface que tem como funcionalidade abstrair a definição de genoma.

IFitness: Interface que tem como objetivo abstrair o calculo de fitness.

IMutate: Interface que tem como objetivo abstrair os operadores de mutação.

ICrossover: Interface que tem como objetivo abstrair os operadores de cruzamento.

IRandom: Interface que tem como objetivo abstrair a implementação de geração de números aleatórios.

AbstractMutate: Implementação base para operador de mutação.

AbstractCrossover: Implementação base para operador de cruzamento.

5.2.6 Core

Enumerators: Contem as definições de enumerações, usados para usar nomes bem definidos ao invés de números avulsos no código.

RandomAdapter: Implementação responsável por gerar números aleatórios, implementa IRandom.

GASettings: Arquivo responsável por carregar configuração estática de GA.

Genome: Classe responsável por representar o genoma no algoritmo de GA, implementa a IGenome.

5.2.7 Selection

Nesta pasta estão todas as implementações dos algoritmos de seleção.

SelectionRouletteWheel: Implementação de seleção roleta.

5.2.8 Crossover

Nesta pasta estão todas as implementações dos algoritmos de cruzamento.

CrossoverOBX: Implementação do operador de cruzamento OBX.

CrossoverPBX: Implementação do operador de cruzamento PBX.

5.2.9 Mutation

Nesta pasta estão todas as implementações dos algoritmos de mutação.

MutateBitwise: Implementação do operador de cruzamento Bitwise.

MutateDIVM: Implementação do operador de cruzamento DIVM.

MutateDM: Implementação do operador de cruzamento DM.

MutateEM: Implementação do operador de cruzamento EM.

MutateIM: Implementação do operador de cruzamento IM.

MutateIVM: Implementação do operador de cruzamento IVM.

MutateSM: Implementação do operador de cruzamento SM.

5.2.10 Fitness

Nesta pasta estão todas as implementações dos algoritmos de fitness.

FitnessTimePath: Implementação do fitness que considera o menor caminho e o tempo, para chegar no destino dentro da janela de tempo como a melhor solução.

5.3 Funcionalidades

O software é composto por funcionalidade individuais que podem ser utilizadas separadamente se necessário e também entender cada uma, ajuda a entender o software por completo.

Integração com a API do Google Maps: Usando a classe SearchRoute do projeto PathFinder.Routes é possível procurar rotas e endereços do Google Maps e obter informação de coordenadas geográficas de um ponto usando o endereço, calcular rota entre dos dois usando o endereço ou com as coordenadas geográficas, obter um mapa estático de um ou vários pontos.

Algoritmos Genéticos O modulo de GA poderia ser utilizado separadamente, porém é preciso o desenvolvimento da função de aptidão para o caso que ele será utilizado, o resto da estrutura funcionaria.

5.4 Interface

A interface é utilizada para interagir com o calculador de rotas de uma forma mais simples, podendo procurar por endereços, adicionar a lista, definir o deposito, os horários de cada endereço, numero de entregadores e tempo de descarga de cada endereço. Para melhor visualização dos testes, é possível encontrar todos os roteiros de teste para fácil execução e verifica os resultados.

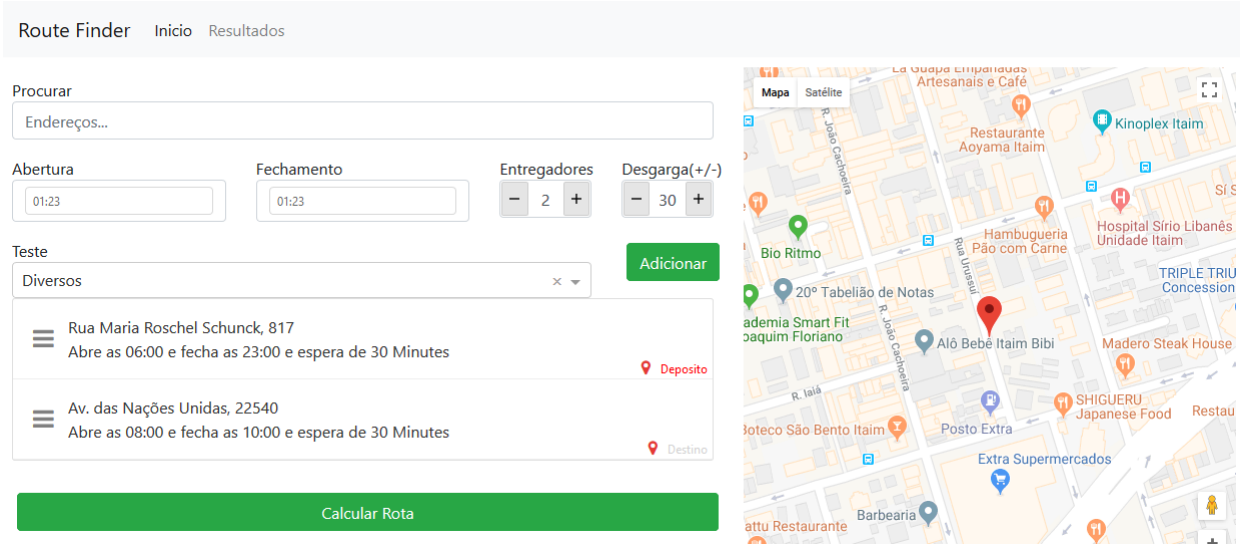

Figura 6 – Aparência da interface

- No campo procurar é preciso digitar o endereço para ser localizado no Google Maps, uma lista de possíveis escolhas aparece a baixo assim que começa a digitar, encontra o endereço e selecione.
- O mapa exibe o endereço selecionado.
- Abertura é o horário que que é permitida a entrada no local, no caso do deposito é o horário que o entregador sai para realizar as entregas.
- Fechamento é o horário limite para realizar a entrega, no caso do deposito, é o horário limite para terminar todas as entregas, horário que o deposito fecha.
- Entregadores é o numero de entregados disponível para realizar as entregas.
- Descarga o tempo médio para descarregar no endereço de entrega, no caso do deposito esse campo não é utilizado.
- Adicionar Agrupa os campos do endereço, abertura, fechamento e descarga para a lista.
- Lista de endereços reúne todos os endereços escolhidos para enviar para o calculador de rotas, o primeiro da lista é sempre o deposito
- Calcular Rota faz a chama da Api de calculo e envia a lista.

Figura 7 – Aparência da interface para exibição dos resultados

Cada entregador é exibido de forma separada, com sua própria tabela de endereço e recalculo para o próximo destino. A lista acima do mapa, é possível escolher qual rota será exibida no mapa, trocando entre os entregadores. A tabela tem 8 colunas com informações de cada percurso, essa colunas são:

- Endereço Saída: É o endereço que o entregador vai sair, na primeira vez será do deposito. Todos entregadores saem do mesmo endereço de deposito.
- Endereço Chegada: Endereço que o entregador chegará depois que sair do endereço de saída.
- KM: É a distancia em Quilômetros de cada percurso indicada pelo Google Maps.
- Saída: Horário que o entregador sairá para realizar a entrega.
- Percurso: Tempo de locomoção para chegar até o destino de entrega.
- Espera: Se chegar no destino antes do horário de abertura, esse é o tempo que o entregador ficará esperando.
- Entrada: Horário que o entregador consegui entrar para começar a realizar a descarga da entrega.
- Descarga: Tempo Médio para descarregador toda a encomenda.

Depois de finalizar a entrega o campo saída deve ser preenchido com o horário que está tudo pronto para fazer a próxima entrega e clicar no botão Próxima Rota, uma nova chamada para a Api será feita somente com o entregador pedido.

6 Conclusão

Nesse capítulo é apresentado como os testes foram organizados, as limitações do software, futuras melhorias e os resultados encontrados.

6.1 Limitações

Na criação do software os testes nos iniciais já possível identificar possíveis situações que o software não tem como dar uma resposta, ou seja, suas limitações. Identificamos 3 possíveis limitações nos testes iniciais:

Não é possível entregar a tempo: Levando em consideração o horário de saída e o horário limite para realizar todas as entregas, é possível que um percurso entre os endereços tem seu tempo de trajeto mais demorado que o tempo disponível para a realização de todas as entregas, com isso seria impossível entregar, mesmo com mais entregadores, com isso, o software não consegue definir uma rota por considerar a velocidade média das vias por onde ele passará.

Limite de Entregadores: Para pedir a definição de uma rota é preciso indicar quantos entregadores estão disponível para realizar as entregas, em algumas situações é possível que mesmo dividindo para todos os entregadores, mais entregadores seriam precisos para chegar a tempo em todos os endereços, com isso, o software não consegue definir uma rota.

Tempo limite de entrega excedido: Se a rota não tiver nenhum percurso muito demorado e também for possível determinar a divisão da rota principal entre o numero de entregadores, podemos encontrar outro problema, sempre que um entregador chega a um destino, é possível sofre atrasos na descarga, um maior tempo de espera ou até o transito piorando por causa de um acidente em uma via principal, por exemplo, isso muda o tempo dos próximos percursos, podendo elevar muito o tempo do trajeto. É possível encontrar a situação que seria preciso mais entregadores para finalizar a entrega, que não é mais possível por que o entregador já está em transito, também pode encontrar um percurso completamente parado, com isso, o software não consegue definir uma rota para finalizar as entregas.

6.2 Roteiros dos Testes

Os roteiros foram escolhidos de forma arbitraria com endereços dentro ou próximos da cidade de São Paulo. As listas de endereços e as rotas encontrada podem ser vista a

baixo:

O teste da tabela abaixo 3 começa o roteiro em Senac Nacões Unidas, endereço Av. Eng. Eusébio Stevaux, 823 - Santo Amaro, horário de saída 09:00, horário de volta 20:00, com 3 entregadores disponíveis e um tempo de espera médio em cada ponto 20 minutos.

Tabela 3 – Senac

#	Nome	Endereço	Aber.	Fech.
1	Senac Largo Treze	R. Dr. Antônio Bento, 393 - Santo Amaro	09:00	12:00
2	Senac Taboão da Serra	Rua Salvador Branco de Andrade, 182 - Jardim Sao Miguel	09:00	18:00
3	Senac Jabaquara	Av. do Café, 298 - Jabaquara	09:00	11:00
4	Senac Osasco	R. Dante Batiston, 248 - Centro	12:00	15:00
5	Senac Santana	R. Voluntários da Pátria, 3167 - Santana	10:00	19:00
6	Senac Tatuapé	R. Cel. Luís Americano, 130 - Tatuapé	15:00	19:00
7	Senac Vila Prudente	Rua do Orfanato, 316 - Vila Prudente	09:00	17:00
8	Senac - Campos do Jordão	Av. Frei Orestes Girardi, 3549 - Capivari, Campos do Jordão - SP	09:00	17:00

O calculo da rota dividiu o roteiro para os 3 entregadores, de forma que, o percurso mais distante deve ser feito por apenas um entregador e os outros dois realizam as entregas dentro e proximidades da cidade. Os resultados completos estão nas tabelas abaixo.

Entregador 1							
Endereço Saída	Endereço Chega...	KM	Saída	Percurso	Espera	Entrada	Descarga
Av. Eng. Eusébio Ste...	R. Dr. Antônio B...	3,946	09:00:00	00:13:29	00:00:00	09:13:29	00:20:00
R. Dr. Antônio Bent...	Rua Salvador Br...	13,276	09:33:29	00:32:54	00:00:00	10:06:23	00:20:00

Figura 8 – Resultado do Entregador 1

Entregador 2							
Endereço Saída	Endereço Chegada	KM	Saída	Percurso	Espera	Entrada	Descarga
Av. Eng. Eusébio Stev...	Av. do Café, 298 - Ja...	10,954	09:00:00	00:27:39	00:00:00	09:27:39	00:20:00
Av. do Café, 298 - Ja...	R. Dante Batiston, 24...	26,976	09:47:39	00:41:38	01:30:43	12:00:00	00:20:00
R. Dante Batiston, 24...	R. Voluntários da Pát...	21,859	12:20:00	00:32:08	00:00:00	12:52:08	00:20:00
R. Voluntários da Pát...	R. Cel. Luís American...	11,022	13:12:08	00:24:01	01:23:51	15:00:00	00:20:00
R. Cel. Luís American...	Rua do Orfanato, 31...	7,173	15:20:00	00:16:36	00:00:00	15:36:36	00:20:00

Figura 9 – Resultado do Entregador 2

Entregador 3							
Endereço Saída	Endereço Chegada	KM	Saída	Percurso	Espera	Entrada	Descarga
Av. Eng. Eusébio Stev...	Av. Frei Orestes Girar...	215,...	09:00:00	02:54:13	00:00:00	11:54:13	00:20:00

Figura 10 – Resultado do Entregador 3

O teste da Tabela 4 começa o roteiro no Extra Morumbi, endereço Av. das Nações Unidas, 16741 - Santo Amaro, horário de saída 06:00, horário de volta 22:00:00, com 4 entregadores disponíveis e um tempo de espera médio em cada ponto 60 minutos.

Tabela 4 – Extra

Nome	Endereço	Aber.	Fech.
Extra Hipermercado	R. João Batista de Oliveira, 47 - Centro, Taboão da Serra - SP	09:00	22:00
Extra João Dias	Av. Guido Caloi, 25 - Jardim São Luís, São Paulo - SP	09:00	22:00
Extra Aeroporto	Avenida Washignton Luís, 5859 - Jd. Aeroporto, São Paulo - SP	09:00	22:00
Extra - Itaim Bibi	R. João Cachoeira, 899 - Itaim Bibi, São Paulo - SP	06:00	22:00
Extra - Ricardo Jafet	Av. Dr. Ricardo Jafet, 1501 - Vila Mariana, São Paulo - SP	09:00	22:00
Extra	R. Nossa Sra. das Mercês, 29 - Vila das Merces, São Paulo - SP	09:00	22:00
Extra Hipermercado	Av. Brigadeiro Luís Antônio, 2013 - Bela Vista, São Paulo - SP	09:00	22:00
Extra	Rua Três Rios, 282 - Bom Retiro, São Paulo - SP	09:00	22:00
Extra Hiper Guarapiranga	Av. Guarapiranga, 752 - Socorro, São Paulo - SP	09:00	22:00
Extra - Jardim Angela	Estrada Velha do M'Boi Mirim, 4374 - Jardim Angela, São Paulo - SP	09:00	22:00
Extra Hiper	Av. Sen. Teotônio Vilela, 2926 - Jardim Iporanga, São Paulo - SP	09:00	22:00

O calculo da rota dividiu o roteiro para 2 entregadores, de forma que, não foi preciso utilizar todos os disponíveis, um entregador ficou com o único endereço que o primeiro não iria consegui fazer, sendo necessário utilizar pouco tempo dele e com o roteiro quase inteiro utilizando apenas um entregador. Os resultados completos estão nas tabelas abaixo.

Entregador 1							
Endereço Saída	Endereço Chegada	KM	Saída	Percurso	Espera	Entrada	Descarga
Av. das Nações Unid...	R. João Batista de Oli...	8,948	06:00:00	00:20:59	02:39:01	09:00:00	01:00:00
R. João Batista de Oli...	Av. Guido Caloi, 25 - ...	7,585	10:00:00	00:20:12	00:00:00	10:20:12	01:00:00
Av. Guido Caloi, 25 - ...	Avenida Washignton...	10,720	11:20:12	00:19:44	00:00:00	11:39:56	01:00:00
Avenida Washignton...	R. João Cachoeira, 8...	7,300	12:39:56	00:21:13	00:00:00	13:01:09	01:00:00
R. João Cachoeira, 8...	Av. Dr. Ricardo Jafet, ...	7,965	14:01:09	00:18:55	00:00:00	14:20:04	01:00:00
Av. Dr. Ricardo Jafet, ...	R. Nossa Sra. das Me...	4,939	15:20:04	00:13:12	00:00:00	15:33:16	01:00:00
R. Nossa Sra. das Me...	Av. Brigadeiro Luís A...	14,770	16:33:16	00:26:23	00:00:00	16:59:39	01:00:00
Av. Brigadeiro Luís A...	Rua Três Rios, 282 - ...	5,714	17:59:39	00:18:41	00:00:00	18:18:20	01:00:00
Rua Três Rios, 282 - ...	Av. Guarapiranga, 75...	21,956	19:18:20	00:39:16	00:00:00	19:57:36	01:00:00
Av. Guarapiranga, 75...	Estrada Velha do M'...	5,828	20:57:36	00:16:13	00:00:00	21:13:49	01:00:00

Figura 11 – Resultado do Entregador 1

Entregador 2							
Endereço Saída	Endereço Chegada	KM	Saída	Percurso	Espera	Entrada	Descarga
Av. das Nações Unid...	Av. Sen. Teotônio Vil...	13,797	06:00:00	00:30:26	02:29:34	09:00:00	01:00:00

Figura 12 – Resultado do Entregador 2

O teste da Tabela 5 começa o roteiro no McDonald's Jardim Paulista, endereço R. Pamplona, 734 - Jardim Paulista, São Paulo - SP, horário de saída 09:00, horário de volta 23:00, com 3 entregadores disponíveis e um tempo de espera médio em cada ponto 40 minutos.

Tabela 5 – MacDonald's

Nome	Endereço	Aber.	Fech.
McDonald's Augusta	R. Augusta, 1856 - Cerqueira César, São Paulo - SP	08:00	23:00
McDonald's Brigadeiro	Av. Brigadeiro Luís Antônio, 3477/3481 - Jardim Paulista, São Paulo - SP	09:00	19:00
McDonald's José Maria	Av. José Maria Whitaker, 81 - Jardim Paulista, São Paulo - SP	09:00	21:00
McDonald's Nações Unidas	Av. das Nações Unidas, 12555 - Pinheiros, São Paulo - SP	06:00	21:00
McDonald's Eliseu de Almeida	Av. Eliseu de Almeida, 2700 - Jardim Peri Peri, São Paulo - SP	09:00	18:00
McDonald's Vital Brasil	Av. Vital Brasil, 1256 - Butantã, São Paulo - SP	09:00	18:00
McDonald's Henrique Schaumann	Rua Henrique Schaumann, 80/124 - Cerqueira César, São Paulo - SP	09:00	23:00
McDonald's Santo Antônio	Av. Roque Petroni Júnior, 1089 - Chácara Santo Antônio (Zona Sul), São Paulo - SP	09:00	23:00

O calculo da rota dividiu o roteiro para apenas um entregador, de forma que, não foi preciso utilizar todos os 3 disponíveis. Os resultados completos estão na tabela abaixo.

Entregador 1							
Endereço Saída	Endereço Chegada	KM	Saída	Percurso	Espera	Entrada	Descarga
R. Pamplona, 734 - J...	R. Augusta, 1856 - C...	1,693	09:00:00	00:08:37	00:00:00	09:08:37	00:40:00
R. Augusta, 1856 - C...	Av. Brigadeiro Luís A...	2,622	09:48:37	00:09:58	00:00:00	09:58:35	00:40:00
Av. Brigadeiro Luís A...	Av. José Maria Whita...	3,810	10:38:35	00:11:49	00:00:00	10:50:24	00:40:00
Av. José Maria Whita...	Av. das Nações Unid...	10,478	11:30:24	00:21:12	00:00:00	11:51:36	00:40:00
Av. das Nações Unid...	Av. Eliseu de Almeid...	4,902	12:31:36	00:12:14	00:00:00	12:43:50	00:40:00
Av. Eliseu de Almeid...	Av. Vital Brasil, 1256 ...	4,837	13:23:50	00:12:05	00:00:00	13:35:55	00:40:00
Av. Vital Brasil, 1256 ...	Rua Henrique Schau...	5,610	14:15:55	00:21:04	00:00:00	14:36:59	00:40:00
Rua Henrique Schau...	Av. Roque Petroni Jú...	10,426	15:16:59	00:20:00	00:00:00	15:36:59	00:40:00

Figura 13 – Resultado do Entregador 1

Tabela 6 – Uninove

Nome	Endereço	Aber.	Fech.
Uninove Osasco	R. Dante Batiston, 87 - Centro, Osasco - SP	08:00	23:00
Uninove Santo Amaro	R. Amador Bueno - Santo Amaro, São Paulo - SP	08:00	23:00
Uninove São Bernardo do Campo	Av. Dom Jaime de Barros Câmara, 90 - Planalto, São Bernardo do Campo - SP	08:00	23:00
Uninove Vila Prudente	Av. Professor Luiz Ignácio Anhaia Mello, 1363 - Vila Prudente, São Paulo - SP	08:00	23:00
Uninove Vila Maria Baixa	R. Itauna, 74 - Vila Maria Baixa, São Paulo - SP	08:00	23:00
Uninove Barra Funda	Av. Dr. Adolpho Pinto, 109 - Barra Funda, São Paulo - SP	08:00	23:00
Uninove Mauá	R. Álvares Machado, 48 - Vila Bocaina, Mauá - SP	08:00	23:00
Uninove Santo André	R. Princesa Isabel - Vila Guiomar, Santo André - SP	08:00	23:00

O teste da Tabela 6 começa o roteiro no Uninove Vergueiro, endereço Rua Vergueiro, 235/249 - Liberdade, São Paulo - SP, horário de saída 08:00, horário de volta 23:00:00, com 5 entregadores disponíveis e um tempo de espera médio em cada ponto 90 minutos.

O teste da Tabela 7 simula a situação de 1 turista hospedado no Hotel Ibis São Paulo Paulista no endereço Av. Paulista, 2355 - Bela Vista, São Paulo - SP, em seu plano é conheço os endereços, permanecendo 30 minutos em cada, saindo do hotel 09:00 e voltando 23:00.

Tabela 7 – Pontos Turísticos de São Paulo

Nome	Endereço	Aber.	Fech.
Catedral Metropolitana de São Paulo	Praça da Sé - Sé, São Paulo - SP	00:00	23:59
Pte. Estaiada	Av. Jorn. Roberto Marinho, 85 - Cidade Monções, São Paulo - SP	00:00	23:59
Museu Catavento	Pq. Dom Pedro II - Av. Mercúrio, s/n - Brás, São Paulo - SP	09:00	16:00
Aquário de São Paulo	R. Huet Bacelar, 407 - Ipiranga, São Paulo - SP	09:00	17:00
Museu do Ipiranga	Parque da Independência - s/n - Ipiranga, São Paulo - SP	09:00	17:00
Parque Ibirapuera	Av. Pedro Álvares Cabral - Vila Mariana, São Paulo - SP	05:00	23:59
Zoológico De Sao Paulo	Av. Miguel Estefno, 4241 - Vila Santo Estefano, São Paulo - SP	09:00	19:00
Jardim Botânico de São Paulo	Av. Miguel Estefno, 3031 - Vila Água Funda, São Paulo - SP	09:00	17:00
Pateo do Collegio	Pç. Pateo do Collegio, 2 - Centro, São Paulo - SP	09:00	16:30
Parque Estadual Alberto Löfgren	R. do Horto, 931 - Horto Florestal, São Paulo - SP	06:00	18:00
Parque Estadual do Jaraguá	R. Antônio Cardoso Nogueira, 539 - Vila Chica Luisa, São Paulo - SP	07:00	17:00
Autódromo de Interlagos	Av. Sen. Teotônio Vilela, 261 - Interlagos, São Paulo - SP	07:00	17:00
Anhembi Sambadrome	Av. Olavo Fontoura, 1209 - Santana, São Paulo - SP	07:00	17:00

Mesmo com varias tentativas, o calculador de rotas não encontrou um rota onde seria possível com apenas uma pessoa em um dia, visitar todos os endereços do roteiro. Alterando para 2 turistas torna possível realizar todas as visitas, porém como não se trata de entregas, é inviável o conseguir um resultado.

6.3 Comparativo

A implementação se mostrou eficiente na organização do roteiro, identificando padrões não muito visíveis entre todos os destinos. O Google Maps ajuda na definição da rota por entregar o transito médio da rota fazendo com que o caminho escolhido pelo software fique mais próximo de uma situação real. O GA ajuda na escolhas das rotas por exemplo tentar minimizar o tempo e a distância, mesmo podendo demorar para encontrar uma solução ótima se o numero de gerações for muito alto, o tamanho da população ou numero de rotas.

Utilizando um numero baixo de destinos o processo de recalculo de rotas para cada vez que chegar em um destino, se mostra útil para identificar mudanças no transito e ainda chegar no horário proposto. Em casos que é preciso calcular para muitos destinos,é mais recomendado somente utilizar para uma divisão de tarefas ou pré-analise do roteiro, por tornar a analise muito demora, o tamanho da população e numero de gerações precisa ser mais altos para melhor precisão dos resultados.

Para comparativo foi utilizado os diferentes tipos de mutação e cruzamentos, de forma, a identificar uma melhor combinação para ser utilizada no problema. A tabela abaixo estão os resultados encontrados:

Tabela 8 – Comparação dos Cruzamentos e Mutações

Extras.txt	**EM**	**DIVM**	**DM**	**IM**	**IVM**	**SM**
OBX	109560	109560	109560	109560	109560	109560
PBX	109560	109560	109560	109560	109560	109560

MacDonalts.txt	**EM**	**DIVM**	**DM**	**IM**	**IVM**	**SM**
OBX	43320	43320	43320	43320	43320	43320
PBX	43320	43320	43320	43320	43320	43320

Senacs.txt	**EM**	**DIVM**	**DM**	**IM**	**IVM**	**SM**
OBX	311876	311876	311876	311876	311876	311876
PBX	311876	311876	311876	311876	311876	311876

Uninoves.txt	**EM**	**DIVM**	**DM**	**IM**	**IVM**	**SM**
OBX	190656	190656	190656	190656	190656	190656
PBX	190656	190656	190656	190656	190656	190656

Turisticos.txt	**EM**	**DIVM**	**DM**	**IM**	**IVM**	**SM**
OBX	84573	84573	84573	84573	84573	84573
PBX	84573	84573	84573	84573	84573	84573

Como mostrado na tabela 8, a mutação ou cruzamento escolhida não influencia no resultado, apresentando o mesmo valor de aptidão para todas as combinações em nos diferentes roteiros.

6.4 Trabalhos futuros

Devido a média de tempo para se encontrar as rotas, avaliar a possível paralelização da rotina de algoritmos genéticos para tornar o projeto viável para uso em uma aplicação web. Avaliar a aplicação de heurísticas $\lambda - interchange$, *k-node*, *Or-opt* no calculo do *Fitness*. Utilizar uma base de dados maior para os testes, baseada em casos reais de logística. Criar uma aplicação web para utilização em um caso real de entregas.

Referências

ALVARENGA, G. B. Um algoritmo híbrido para os problemas de roteamento de veículos estático e dinâmico com janela de tempo. Universidade Federal de Minas Gerais, 2005. Disponível em: <http://www.bibliotecadigital.ufmg.br/dspace/bitstream/handle/1843/RVMR-6EAKH8/guilherme_bastos.pdf?sequence=1>. Citado na página 12.

ANTES, J.; DERIGS, U. A new parallel tour construction algorithm for the vehicle routing problem with time windows. 12 1997. Citado na página 13.

BAKER, E. K.; SCHAFFER, J. R. Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *American Journal of Mathematical and Management Sciences*, v. 6, n. 3-4, p. 261–300, 1986. Disponível em: <http://dx.doi.org/10.1080/01966324.1986.10737197>. Citado na página 12.

BRÄYSY, O.; DULLAERT, W.; GENDREAU, M. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, v. 10, n. 6, p. 587–611, Dec 2004. ISSN 1572-9397. Disponível em: <https://doi.org/10.1007/s10732-005-5431-6>. Citado na página 17.

BRäYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, v. 39, n. 1, p. 104–118, 2005. Disponível em: <https://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0056>. Citado na página 14.

BRäYSY, O.; HASLE, G.; DULLAERT, W. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research*, Elsevier, v. 159, n. 3, p. 586–605, 12 2004. ISSN 0377-2217. Citado na página 16.

CHABRIER, A. Vehicle routing problem with elementary shortest path based column generation. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 33, n. 10, p. 2972–2990, out. 2006. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2005.02.029>. Citado 2 vezes nas páginas 6 e 11.

CHIANG, W.-C.; RUSSELL, R. A. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, v. 63, n. 1, p. 3–27, Feb 1996. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/BF02601637>. Citado na página 16.

CHRISTOFIDES, N.; BEASLEY, J. E. The period routing problem. *Networks*, Wiley Subscription Services, Inc., A Wiley Company, v. 14, n. 2, p. 237–256, 1984. ISSN 1097-0037. Disponível em: <http://dx.doi.org/10.1002/net.3230140205>. Citado na página 14.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, v. 12, n. 4, p. 568–581, 1964. Disponível em: <https://doi.org/10.1287/opre.12.4.568>. Citado na página 12.

CORDEAU, J.-F. et al. 7. vrp with time windows. In: _____. *The Vehicle Routing Problem*. [s.n.]. p. 157–193. Disponível em: <http://epubs.siam.org/doi/abs/10.1137/1.9780898718515.ch7>. Citado na página 11.

DIAS, M. A. P. *Administração de materiais: uma abordagem logística.* [S.l.: s.n.], 2010. Citado na página 5.

EL-SHERBENY, N. A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science*, v. 22, n. 3, p. 123 – 131, 2010. ISSN 1018-3647. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1018364710000297>. Citado na página 13.

GARCIA, B.-L.; POTVIN, J.-Y.; ROUSSEAU, J.-M. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, v. 21, n. 9, p. 1025 – 1033, 1994. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305054894900736>. Citado na página 16.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, v. 13, n. 5, p. 533 – 549, 1986. ISSN 0305-0548. Applications of Integer Programming. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305054886900481>. Citado na página 16.

HOLLAND, J. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975. Disponível em: <http://books.google.com/books?id=YE5RAAAAMAAJ>. Citado na página 18.

HOMBERGER. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, v. 162, n. 1, p. 220–238, 2005. Disponível em: <https://EconPapers.repec.org/RePEc:eee:ejores:v:162:y:2005:i:1:p:220-238>. Citado na página 17.

JEPSEN, M.; SPOORENDONK, S. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. 01 2006. Citado na página 11.

JUNG, S.; MOON, B.-R. A hybrid genetic algorithm for the vehicle routing problem with time windows. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. (GECCO'02), p. 1309–1316. ISBN 1-55860-878-8. Disponível em: <http://dl.acm.org/citation.cfm?id=2955491.2955733>. Citado na página 17.

KARP, R. M. On the computational complexity of combinatorial problems. 1975. Citado na página 5.

KOHL, N. Exact methods for time constrained routing and related scheduling problems. Richard Petersens Plads, Building 321, 2800 Kgs. Lyngby, p. 234, 1995. Disponível em: <http://www2.imm.dtu.dk/pubdb/p.php?2100>. Citado na página 11.

LANDEGHEM, H. V. A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, v. 36, n. 2, p. 217 – 226, 1988.

ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/0377221788904286>. Citado 2 vezes nas páginas 12 e 13.

LORENA, G. M. R. e L. A. N. Roteamento de veículos dinâmico usando algoritmos genéticos. Citado na página 25.

LUCAS, D. C. Algoritmos genéticos: uma introdução. Universidade Federal do Rio Grande do Sul, 2002. Disponível em: <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Citado 3 vezes nas páginas 18, 19 e 22.

OLIVEIRA, G. C. V. e. G. B. A. Humberto César Brandão de. Uma abordagem evolucionária para o problema de roteamento de veículos com janela de tempo. 2005. Citado na página 25.

OLIVEIRA, H. C. B. de. Algoritmo evolutivo no tratamento do problema de roteamento de veículos com janela de tempo. 2005. Citado na página 5.

OLIVEIRA, W. A. de. Algoritmo genético para o problema de rotas de cobertura multiveículo. 2009. Citado na página 24.

OSMAN, I. H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, v. 41, n. 4, p. 421–451, Dec 1993. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/BF02023004>. Citado na página 14.

POTVIN, J.-Y.; ROBILLARD, C. Clustering for vehicle routing with a competitive neural network. *Neurocomputing*, v. 8, n. 2, p. 125 – 139, 1995. ISSN 0925-2312. Optimization and Combinatorics, Part II. Disponível em: <http://www.sciencedirect.com/science/article/pii/092523129400012H>. Citado na página 14.

POTVIN, J.-Y.; ROUSSEAU, J.-M. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, v. 66, n. 3, p. 331 – 340, 1993. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/0377221793902218>. Citado na página 13.

RIBAS, S. Um algoritmo híbrido para o problema de roteamento de veículos com janelas de tempo. 2011. Citado na página 25.

RIBEIRO, C. C. Metaheuristics and applications. In: _____. *Advanced School on Artificial Intelligence*. [S.l.: s.n.], 1996. Citado na página 16.

RODRIGUES, P. R. A. *Introdução aos sistemas de transporte do Brasil e à logística internacional*. [S.l.: s.n.], 2007. Citado 2 vezes nas páginas 5 e 6.

SCHULZE, J.; FAHLE, T. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research*, v. 86, n. 0, p. 585–607, Jan 1999. ISSN 1572-9338. Disponível em: <https://doi.org/10.1023/A:1018948011707>. Citado na página 15.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, v. 35, n. 2, p. 254–265, 1987. Disponível em: <https://doi.org/10.1287/opre.35.2.254>. Citado 3 vezes nas páginas 12, 13 e 17.

SOUZA, M. J. F. 2011. Disponível em: <http://www.decom.ufop.br/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.htm>. Acesso em: 01/12/2017. Citado na página 16.

TOMTOM INTERNATIONAL BV. 2017. Disponível em: <https://www.tomtom.com/en_gb/trafficindex/list?citySize=LARGEcontinent=ALL&country=BR>. Acesso em: 12/11/2017. Citado na página 27.

TSUDA, D. S. Modelo de roteirização de veículos em uma empresa importadora de produtos japoneses. 2007. Citado na página 5.

YVES ROCHAT, E. Probabilistic diversification and intensification in local search for vehicle routing. 1995. Citado na página 9.