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

Pré-monografia apresentada na disciplina Trabalho de Conclusão de Curso I, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

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

Sumário

1 Introdução

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

1.1 Motivação

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.

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

No planejamento estratégico de logística o principal problema esta relacionado a roteirização de veículos (TSUDA, 2007) também conhecido como PRV. O PRV é baseado em definir um conjunto de rotas que será percorrido por veículos obedecendo que cada rota começa e termina no depósito, todo consumidor é visitado somente uma vez e a demanda total de qualquer rota não pode ultrapassar capacidade dos veículos para encontrar a rota menos custosa, é necessário calcular as possíveis combinações de um determinado problema , contudo, dependendo do numero de combinações pode requerer um processamento muito elevado, levando muito tempo para encontrar a solução ótima.

Como exemplo de aplicações podemos citar:

- Entrega postal;
- Entrega em domicilio, 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;

Temos um especialização do PRV é o problema de roteamento de veículos com janela de tempo , o PRVJT, nele deve-se considerar um intervalo de tempo para as entregas serem realizadas. Se aproximando mais do mundo real onde existem janelas de tempo validas para entregas de mercadorias. O PRVJT exige um alto esforço computacional, pois pertence a classe dos problemas NP-difíceis, torna-se importante uma boa escolha do método usado para sua solução que consiga resolvê-lo em tempo polinomial. Desde então, o problema vem sendo muito estudado, principalmente por sua alta complexidade e pela grande variedade de problemas reais a ele associados.

1.2 Objetivos

O trabalho tem objetivo de desenvolver uma solução que resolva o problema de PRVJV utilizando a meta-heurística algoritmos genéticos. O sistema será capaz de calcular uma rota entre vários destinos levando em consideração restrições de tempo e notificando a quantidade de motoristas necessários para realizar todas as entregas até uma data/hora limite estipulada, levando em consideração o tempo de transito entre estes pontos, permitindo que um motorista possa recalcular a sua rota para otimizar o tempo a qualquer momento. Será calculada uma rota minimizando a distancia entre os pontos, o tempo do percurso e se é possível chegar no horário. Não sendo possível cumprir o horário,o caminho dividido para mais entregadores, se o objetivo for impossível, é emitido um aviso sempre que acontecer.

1.2.1 Objetivos Específicos

- Utilizar API do Google Maps para adquirir informações sobre endereços e pontos no mapas.
- Desenvolver um algorítimo genético capas de minimizar a rota entre todos os pontos considerando tempo e distância.
- Introduzir janelas de tempo nas entregas e adaptar algoritmo genético para levar janelas de tempo em consideração.
- Definir dia/hora limite e dividir entrega em mais de um entregador para respeitar essa hora/data limite.
- Identificar quando não é possível calcular a rota respeitando a dia/hora limite.
- Criar um site para a interface de recalculo por entregador e visualização das rotas.

1.3 Método de trabalho

O problema será testando levando em consideração diferentes situações próximas de reais. Simulando endereços para as entregas e horários em um ambiente controlado que sabemos a melhor resposta e sendo colocado em situações impossíveis.

1.4 Organização do trabalho

Este trabalho é dividido em 4 capítulos. O primeiro capitulo faz uma introdução geral do problema, descrever os 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 de dois exemplos de busca heurística, demostrando uma aplicação em um trabalho da literatura e dos algoritmos genéticos, explicando seu funcionamento e aplicação na literatura. O terceiro capitulo é a proposta apresentada para a criação deste trabalho.

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 parte 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 varias iterações ocorrerem, para cada 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 tora. 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. Porem as soluções que suas heurísticas encontram são geralmente de baixa qualidade. Geralmente a mais de 10% da ótima (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 de 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

A base de quase todas as heurísticas de melhoria de rotas é 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, por em este pode eventualmente ser um ótimo global. A

este algoritmo da-se o nome de Hill Climbing. Para maiores informações sobre algoritmos de aprimoramento de rotas, sugere-se ao leitor também o trabalho de Braysy e Gendreau [14]. 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

2.1.4 Metaheurísticas

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 1.

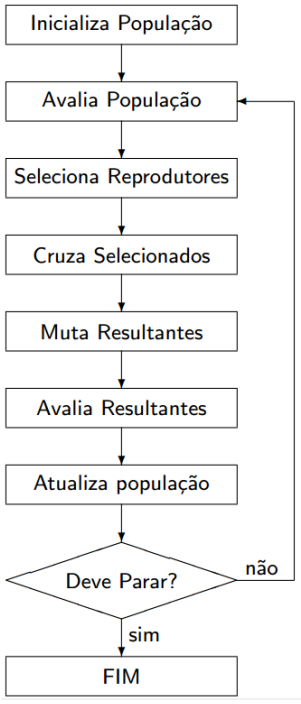

Figura 1 – Estrutura de um AG (LUCAS, 2002)

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 Random: Gera um numero aleatório entre 0 e o tamanho total da população e retorna o indivíduo do índice escolhido.

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 A aleatório entre 0 e 1 e multiplicado pelo valor total dos pesos. Para selecionar o indivíduo é feito um loop nos pesos e seus valores somados até que o valor A seja igual ou menor que zero, o índice do peso que fez a condição acontecer, se o índice do indivíduo selecionado. Desta forma aumentando a possibilidade de selecionar um indivíduo com maior aptidão.

2.2.5 Cruzamento

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

CrossOver Simple: Utiliza dois indivíduos selecionados, 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. Esse tamanho é utilizado para trocar cromossomos entre os dois indivíduos, ou seja, adicionar todos os cromossomos do primeiro indivíduo do índice igual ao primeiro numero, até o índice segundo numeSAro, e repete o processo contrario.

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 cro-

mossomo 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 Atualização

Os indivíduos criados no processo de reprodução e mutação são inseridos na população.

Na forma mais tradicional deste a população mantém um tamanho fixo e os indivíduos são criados em mesmo número que seus antecessores e os substituem por completo.

Existem, porém, algumas alternativas, o número de indivíduos gerados pode ser menor ou o tamanho da população pode sofrer variações e o critério de inserção pode variar, por exemplo, nos casos em que os filhos substituem os pais, ou em que estes só são inseridos se possuírem maior aptidão que o cromossomo que sera substituído, ou o manter sempre o conjunto dos n melhores indivíduos.

2.2.8 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 1.

Tabela 1 – 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)

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

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

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 ao 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.

2.2.9 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, para encontrar uma boa distribuição de viaturas para patrulhamento urbano usado por departamentos de segurando como a policia, guardas municipais ou segurança privada (**??**). 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 2 mostra o grafo da relação de V com W.

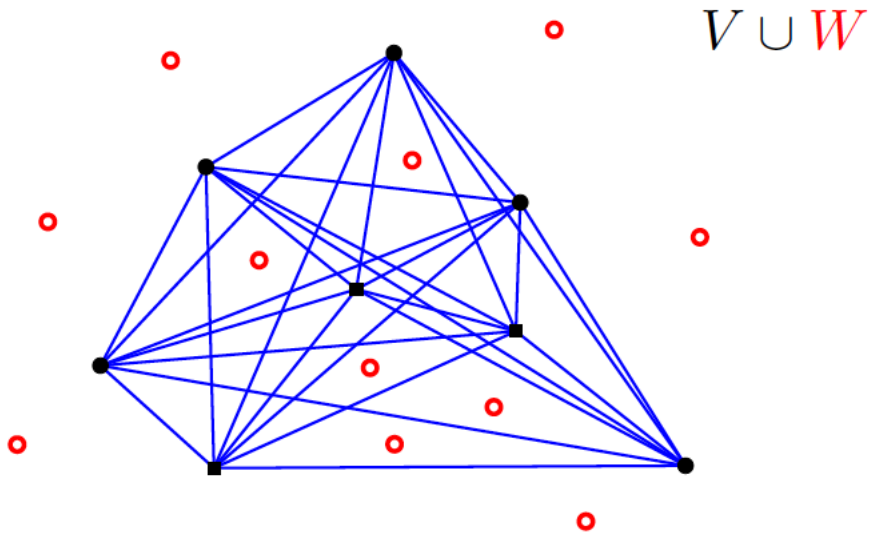

Figura 2 – Exemplo de gráfo não direcionado para V U W. (??)

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 (**??**) é que a utilização de algoritmos genéticos para a resolução de 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.

3 Proposta

4 Metodologia

Um software sera desenvolvido com objetivo de demostrar que serve para a solução do problema proposto, que é definir caminhos viáveis entre diferentes rotas em situações onde é possível encontrar o caminho, e caso não seja possível encontrar um caminho viável dentro das restrições impostas sera comunicado ao usuário a impossibilidade de encontrar uma solução viável.

Utilizando a API do Google Maps como fonte de dados, informações reais de distância, tempo médio e localização são utilizadas para uma simulação mais próxima de uma situação real.

Por se tratar de entregas de pequenos porte, os testes foram criados com coordenadas a nível de cidade, São Paulo é a cidade para os testes, por se tratar de uma cidade com um alto índice de transito segundo o TomTom Traffic Index (TOMTOM INTERNATIONAL BV, 2017), a menor rota pode não ser a melhor escolha para aquele horário do dia.

Todos os entregadores partem de uma única origem que chamamos de deposito, antes de começar as entregas, é calculado uma rota geral, ela é dividida de forma a encontrar rotas possível para cada entrega, e dividir para um próximo entregador caso não seja possível realizar as entregas com a atual quantidade, se o limite definido pelo usuário de entregadores for ultrapassado, um aviso será emitido recomendando que deixe para o próximo dia as entregas mais distantes.

Cada destino tem um período permitido para entrega, e cada entrega demorar no máximo 5 minutos para ser descarregada. Depois que uma entrega é feita, o software recalcula o próximo destino com base no transito atual e o período para entregar.

Se caso não for mais possível entregar no horário por motivos de piora de transito, um alerta será emitido. Depois que o entregador terminar todas suas entregas, o software continua auxiliando os outros entregadores que também não tenham terminado duas entregas, até que todas as entregas sejam concluídas.

O fluxograma a baixo demostra o funcionamento do software.

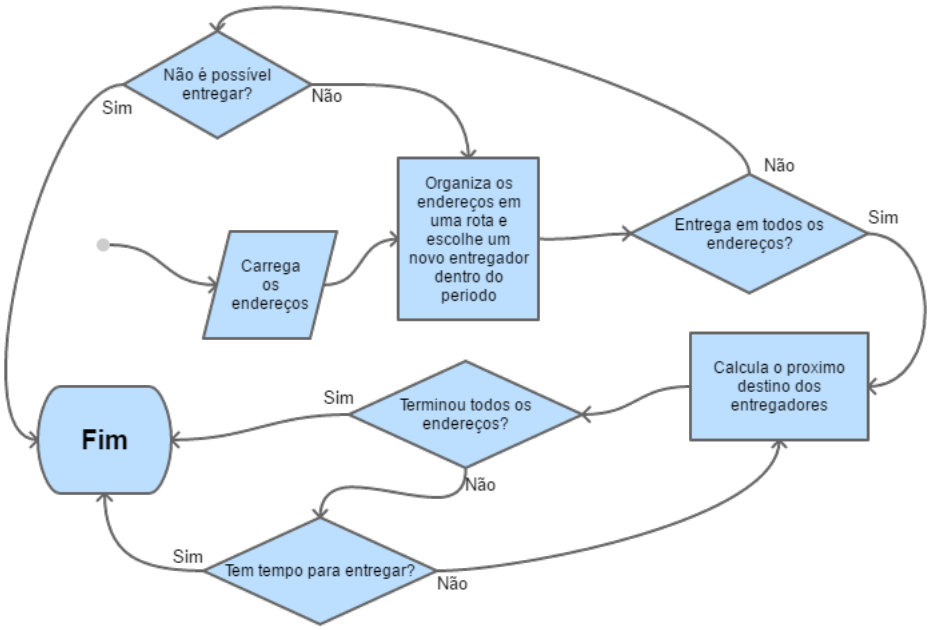

Figura 3 – Fluxograma macro do funcionamento do software.

O GA considera como um individuo um percurso inteiro, ponto do estoque, e todos os destinos.Uma rota aleatória é calculada para cada individuo como população inicial.

A mutação e o cruzamento alterna os pontos de destino, mantendo o estoque sempre fixo. A função de aptidão do GA considera o horário de saída como parâmetros inicial, com isso, utiliza o tempo dado pelo Google Maps entra 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 5 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 3 tipos de cruzamento, sendo Simples,PBX e OBX. Considerando o transito enviado pelo Google Mapas e ignorando, 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. Os parâmetros utilizados são

5 Implementação

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

5.1 O Projeto

O software é separado em três projetos, todos utilizando .Net Core 2.0 com a linguagem C# no Visual Studio 2017 para plataforma Windows ou *nix.

O **PathFinder.Routes** nesse projeto estão os algoritmos de comunicação com o Google Maps e organização de rotas.

O **PathFinder.GeneticAlgorithm** nesse projeto estão as implementações para a utilização do algoritmo genético.

O **PathFinder** projeto principal para inicialização do software e utiliza os dois outros projetos em sua implementação.

5.2 Organização

Os projetos são organizados utilizando o padrão do Visual Studio chamado de Solution.

5.2.1 PathFinder

Projeto de inicialização no modo console, todo progresso do software é exibido em texto.

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 PathFinder.Routes

Projeto de integração com o Google Mapas e classes de apoio do software.

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 completo e o armazém. A definição de cada caminho completo é controlado nessa classe.

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

5.2.3 PathFinder.GeneticAlgorithm

Neste projeto são definidos todas as implementações referentes ao algoritmo genético, pela complexidade do algoritmo ele possui uma estrutura própria de pastas para definições e configurações de injeção de dependência.

5.2.4 Abstraction

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

ISelection: Interface é responsável por abstrair os algoritmos de seleção.

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

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

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

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

IRandom: Interface 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.

5.2.5 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.

5.2.6 Selection

Nesta pasta estão todas as implementações dos algoritmos de seleção.

SelectionRouletteWheel: Implementação de seleção roleta.

5.2.7 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.

CrossoverSimple: Implementação do operador de cruzamento simples.

5.2.8 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.

5.2.9 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.

6 Conclusão

6.1 Testes

6.2 Trabalhos futuros

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 11.

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

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 11.

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 na página 10.

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 11.

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 10.

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 12.

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 13.

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 10.

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 10.

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 11 e 12.

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 2 vezes nas páginas 13 e 17.

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 12.

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

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 2 vezes nas páginas 11 e 12.

TOMTOM INTERNATIONAL BV. 2017. Disponível em: <https://www.tomtom.com/en_gb/trafficindex/list?citySize=LARGE&continent=ALL&country=BR>. Acesso em: 12/11/2017. Citado na página 21.

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 8.