UNIVERSIDADE FEDERAL DO PARANÁ

PAULA FRANCIS BENEVIDES

APLICAÇÃO DE HEURÍSTICAS E METAHEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE EM UM PROBLEMA REAL DE ROTERIZAÇÃO DE VEÍCULOS

CURITIBA

2011

PAULA FRANCIS BENEVIDES

APLICAÇÃO DE HEURÍSTICAS E METAHEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE EM UM PROBLEMA REAL DE ROTERIZAÇÃO DE VEÍCULOS

Dissertação apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática, Departamentos de Construção Civil e de Matemática, Setores de Tecnologia e de Ciências Exatas, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Mestre em Ciências.

Orientadora: Prof. Dra. Deise M. Bertholdi Costa.

Co-orientador: Prof. Dr. Luiz Fernando Nunes

CURITIBA

2011

TERMO DE APROVAÇÃO

PAULA FRANCIS BENEVIDES

APLICAÇÃO DE HEURÍSTICAS E METAHEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE EM UM PROBLEMA REAL DE ROTERIZAÇÃO DE VEÍCULOS

Dissertação aprovada como requisito parcial à obtenção de grau de Mestre em Ciências, Programa de Pós-Graduação em Métodos Numéricos de Engenharia, área de concentração em Programação Matemática, Setor de Tecnologia, Departamento de Construção Civil e Setor de Ciências Exatas, Departamento de Matemática da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientadora: Prof.ª Dr.ª Deise Maria Bertholdi Costa
PPGMNE e Departamento de Expressão Gráfica, UFPR

Co-orientador: Prof. Dr. Luiz Fernando Nunes
Departamento de Matemática, UTFPR

Prof. Dr. Paulo Henrique Siqueira
PPGMNE e Departamento de Expressão Gráfica, UFPR

Prof.ª Dr.ª Angela Olandoski Barboza
Departamento de Matemática, UTFPR

Curitiba, 25 de Novembro de 2011

AGRADECIMENTOS

A realização deste trabalho só foi possível mediante a contribuição de muitas pessoas. A elas, gostaria de deixar algumas palavras de reconhecimento.

Inicialmente gostaria de agradecer ao professor, co-orientador e amigo Luiz Fernando Nunes por confiar em mim e me incentivar a entrar nesse programa de mestrado, sempre me dando força e palavras de incentivo nos momentos em que me sentia para baixo, nunca deixando de acreditar que eu seria capaz, quando eu mesma muitas vezes não acreditava. Agradeço também por disponibilizar seu tempo livre, sanando minhas dúvidas, mesmo as mais simples, sem me criticar. Ainda agradeço principalmente por todo o apoio e ajuda durante essa jornada.

À professora Deise Maria Bertholdi Costa, por apostar em minha capacidade, dando-me a chance de por ela ser orientada. Também, pelo empenho e dedicação em minha orientação.

Ao meu marido Marcelo que sempre me apoiou, participando comigo nesta caminhada com muita paciência e compreensão, tendo sempre uma palavra de carinho e incentivo. Agradeço pelas madrugadas que passou ao meu lado, apenas para me dar força, enquanto eu estudava. Também, pelos finais de semana que fiquei ausente e mesmo assim continuar me apoiando.

À minha filha Gabriela, que apesar da pouca idade soube entender todas as horas de trabalho e de estudo em que tive que me ausentar de seu convívio, sempre me apoiando com um abraço sincero.

À minha mãe Salma (*in memorian*) por tudo que sou e ao meu pai Benevides, pelo apoio e por proporcionar condições que me permitiram concluir o curso da melhor forma possível.

Às minhas irmãs Marta e Márcia, e também ao meu sobrinho Guilherme por estarem juntos a mim nos momentos em que foi preciso.

À professora e colega Angela Olandoski Barboza, que me ajudou com a implementação dos algoritmos e com as dúvidas decorrentes no caminho.

Aos professores Celso Carnieri, Paulo Henrique Siqueira, Luzia Vidal de Souza e Arinei Carlos Lindbeck da Silva, excelentes docentes, por me darem a certeza de ter feito a escolha certa, fazendo despertar em mim a paixão por Métodos Numéricos.

À chefe do Departamento Acadêmico de Matemática da UTFPR, Violeta Maria Estephan por estar sempre disposta ajudar para que eu pudesse desenvolver o meu trabalho da melhor maneira possível, apoiando na participação de eventos e em tudo que fosse necessário.

Ao gerente de setor da distribuidora de produtos por me fornecer a rota de um de seus representantes para que esse trabalho pudesse ser desenvolvido.

À Flavia Konowalenko, irmã de mestrado, pela amizade inestimável que surgiu durante o curso, pelos momentos de alegrias e tristezas, assim como o ombro amigo e apoio nas horas mais difíceis.

À secretária do programa Maristela Bandil, que sempre de bom humor aguentava minhas “chatices” e atendia prontamente minhas solicitações.

Aos colegas de mestrado, em especial, Sandro Rodrigues, Fabio Andre Balbo e Ricardo Petterle pela paciência e dedicação em me ensinar a desvendar áreas de conhecimento inéditas para mim e, com tudo isso, tornar realidade a conclusão deste trabalho.

À minha amiga Crisiane Rezende Vilela de Oliveira, pelo apoio, incentivo e empréstimos de livros e materiais para que eu pudesse estudar e entender as disciplinas que para mim pareciam tão difíceis.

E por fim ao demais familiares e amigos pelo apoio e incentivo.

A todos, meus sinceros e humildes agradecimentos, pois se cheguei até aqui foi graças a vocês.

RESUMO

O transporte, em geral, representa nos dias atuais, o maior percentual de custos do na atividade logística. Por isso, muitas empresas estão repensando seus processos para redução dos mesmos. A otimização da distribuição de produtos é um problema estudado há muito tempo por pesquisadores da área de matemática, pesquisa operacional e da computação. Este tipo de problema é dado como um típico problema de otimização combinatória. O Problema do Caixeiro Viajante (PCV) é um clássico deste tipo de problema. Assim, como o Problema de Roteamento de Veículos (PRV), o qual busca o menor caminho dentre *N* lugares de destino. Na literatura podem ser encontrados trabalhos e abordagens propostos, que utilizam formulações exatas, algoritmos heurísticos e metaheurísticos. O objetivo deste trabalho foi realizar um estudo de caso que envolvesse um número significativo de pontos visitados por algum tipo de veículo, visando analisar e comparar, em termos de desempenho computacional e qualidade das soluções obtidas, as Heurísticas de Construção e Melhoria de Rota e das Metaheurísticas *Ant System, Simulated Annealing* e Algoritmos Genéticos para o PCV. Também foi aplicado o algoritmo 2-opt para melhoria das rotas geradas. As técnicas foram aplicadas tendo em vista que a otimização das visitas e distribuição dos produtos pode reduzir custos e principalmente os atrasos nas entregas. Para implementação foram utilizados dados reais de uma distribuidora de produtos para uma determinada região da cidade de Curitiba (PR), Brasil. Através do aplicativo *online*, Google Earth foram obtidas as coordenadas geográficas dos pontos de visitação, que foram então convertidas para coordenadas cartesianas, para a utilização nos algoritmos. Os resultados obtidos foram comparados com as rotas reais praticadas na época por um dos representantes da referida distribuidora.

Palavras Chave: PCV. PRV. Otimização combinatória.

ABSTRACT

Transport, in general, on average absorbs the highest percentage of costs than any other logistics activity, so many companies are rethinking their processes to reduce them. The optimization of the distribution of products is a problem that is studied for a long time by researchers in mathematics, operational research and computing. This type of problem is given as a typical combinatorial optimization problem. The Traveling Salesman Problem (TSP) is a classic of this, as well as the Vehicle Routing Problem (VRP), where it briefly conceptualizes in finding the shortest path from *N* places of destination. In the literature there are many jobs and proposed approaches, and some of these heuristics and metaheuristics will be studied and analyzed. The objective of this work was to perform a case study involving a significant number of points visited by some kind of vehicle in order to analyze and compare in terms of computational performance and quality of the solutions obtained, the Construction and Improvement Heuristics and Route Ant System metaheuristics, Simulated Annealing and Genetic Algorithms for the TSP, and has also applied the 2-opt algorithm for improving the routes generated, with a view that the optimization of the visits and distribuition product will reduce costs and above all, the delivery delays. We used real data from a distributor of products in a specific region of Curitiba (PR), Brazil. Using Google Earth has picked the geographical coordinates of points of visitation, which were converted to Cartesian coordinates, for application of the algorithms used. The results were compared with the true routes that are being used by a particular representative of that distributor.

Keywords: TSP. VRP. Combinatorial optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

ABREVIATURAS

ACO	Ant Colony Optimization
AG	Algoritmo Genético
CD	Centro de Distribuição
CE	Computação Evolucionária
EE	Estratégias Evolucionistas
LC	Lista de Candidatos
LRC	Lista Restrita de Candidatos
LT	Lista Tabu
MATLAB	MATrix LABoratory
PE	Programação Evolucionista
PRC	Problema do Caixeiro Viajante
PRV	Problema de Roteamento de Veículos
TSP	Traveling Salesman Problem

SUMÁRIO

1. INTRODUÇÃO

1.1 DESCRIÇÃO E RELEVÂNCIA DO PROBLEMA

O problema de roteamento de veículos, devido a sua complexidade, bem como seu papel crítico nos sistemas de distribuição de mercadoria e atendimento de serviços, tem atraído cada vez mais a atenção de pesquisadores de diversas áreas.

A popularidade da internet e a larga utilização do comércio eletrônico fazem com que as empresas e prestadoras de serviço adotem uma logística em tempo real. Com isso os pesquisadores tem se esforçado para desenvolver algoritmos cada vez mais eficientes para serem aplicados a tais problemas.

Nos dias atuais as empresas focam-se na redução de custos, tanto na fabricação de seus produtos como na otimização da logística de seus processos. Para que isso aconteça, alguns conceitos sobre processos devem ser estudados e reavaliados.

O estudo é feito com base no Problema do Caixeiro Viajante (PCV), pois segundo Prestes (2006) o PCV é um problema que tem servido de plataforma de testes para investigação de diversas idéias algorítmicas porque além de ser um problema de larga aplicabilidade no mundo real, é de fácil compreensão e descrição, mas de difícil solução, pois pertence à classe de problemas *NP-Hard,* isto é, não existe algoritmo com limitações polinomiais capaz de resolvê-lo. Por esse motivo, vários métodos têm sido desenvolvidos com o propósito de se resolver instâncias cada vez maiores desse problema em um tempo computacional menor.

Considerando a dificuldade de se resolver problemas de grande escala de forma exata, soluções aproximadas podem ser uma boa alternativa com um tempo computacional aceitável. Ainda que os valores obtidos possam não ser exatos, é possível a obtenção de limites superiores para se avaliar os erros que estão sendo cometidos.

O montante gasto com distribuição justifica a importância deste problema, pois pequenas reduções de custos podem representar uma economia significativa para as empresas (BODIN, GOLDEN e ASSAD, 1983).

Os resultados podem servir de apoio para os gestores encarregados de tomadas de decisão, pois geralmente, uma pequena redução que se consiga em

relação aos custos relacionados a esta atividade, corresponde a uma significativa economia para as empresas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Neste trabalho, são comparadas algumas abordagens heurísticas e metaheurísticas aplicadas ao PCV com um número de pontos que devem ser visitados por algum tipo de veículo. O principal objetivo é oferecer soluções aproximadas e computacionalmente rápidas para o problema proposto, bem como prover uma sugestão de roteiro de visita para o representante de forma a minimizar a distância percorrida na execução da tarefa, esperando-se com essa otimização, um ganho de tempo e minimização dos custos da empresa.

Pode-se descrever sucintamente o problema referente ao estudo de caso em questão da seguinte forma: tem-se um Centro de Distribuição (CD), de onde os representantes saem para o inicio das visitas. Espalhados pela região estão os operadores logísticos (mercados, restaurantes, lanchonetes, etc.) que são os compradores/receptores das caixas de produtos, que farão a distribuição para o cliente final. O problema consiste em visitar todos os clientes programados, assim como transportar as caixas do depósito até os operadores logísticos, utilizando um roteiro de menor custo possível.

1.2.2 Objetivos Específicos

- Estudar os principais conceitos, heurísticas e metaheurísticas existentes para a resolução deste problema.
- Implementar e comparar essas heurísticas e metaheurísticas com os dados reais usando como objetivos, a otimização da distância e tempo e do tempo de resposta.
- Obter uma rota otimizada para ser utilizada pelo representante comercial nas visitas aos seus clientes e entrega dos produtos por meio da aplicação do Problema do Caixeiro Viajante.

1.3 ESTRUTURA DO TRABALHO

A organização dos capítulos permite uma compreensão gradativa e crescente dos assuntos relacionados. Para tanto, o trabalho está organizado da forma que segue.

No Capítulo 2 é feita uma introdução ao Problema do Caixeiro Viajante (PCV), bem como suas variações e formulação matemática.

No Capítulo 3 são apresentadas algumas heurísticas de construção e melhoria de rota aplicadas na solução do PCV.

No Capítulo 4 as Metaheurísticas *Ant System*, *Simulated Annealing*, Algoritmos Genéticos, *GRASP* e Busca Tabu são apresentadas e detalhadas.

No capítulo 5 é caracterizado e definido o problema estudado, como também é detalhada a implementação computacional e seus resultados.

No capítulo 6 são apresentadas as considerações finais sobre o estudo realizado e no capítulo 7 são descritas recomendações para próximas pesquisas sobre o tema.

2. PROBLEMAS DE ROTEIRIZAÇÃO DE VEÍCULOS

Segundo Malaquias (2006), o termo roteirização de veículos, embora não encontrado nos dicionários da língua portuguesa, é a forma que vem sendo utilizada como equivalente ao inglês *routing* para designar o processo para a determinação de um ou mais roteiros ou sequências de paradas a serem cumpridos por veículos de uma frota, objetivando visitar um conjunto de pontos geograficamente dispersos, em locais pré-determinados, que necessitam de atendimento.

Os problemas de roteamento em sua maioria lidam com rotas sobre pontos de demanda e oferta. Esses pontos podem representar cidades, centros de distribuição, depósitos, pontos de coleta, etc. Dentre os tipos de rotas um dos mais importantes é o denominado Hamiltoniano. De acordo com Goldbarg e Luna (2005), seu nome é devido a Willian Rowan Hamilton que, em 1857, propôs um jogo que denominou *Around the World*. O jogo era feito sobre um dodecaedro (Figura1), em que cada vértice estava associado a uma cidade importante na época. O desafio era encontrar uma rota através dos vértices do dodecaedro que iniciasse e terminasse em uma mesma cidade, sem repetir vértices.

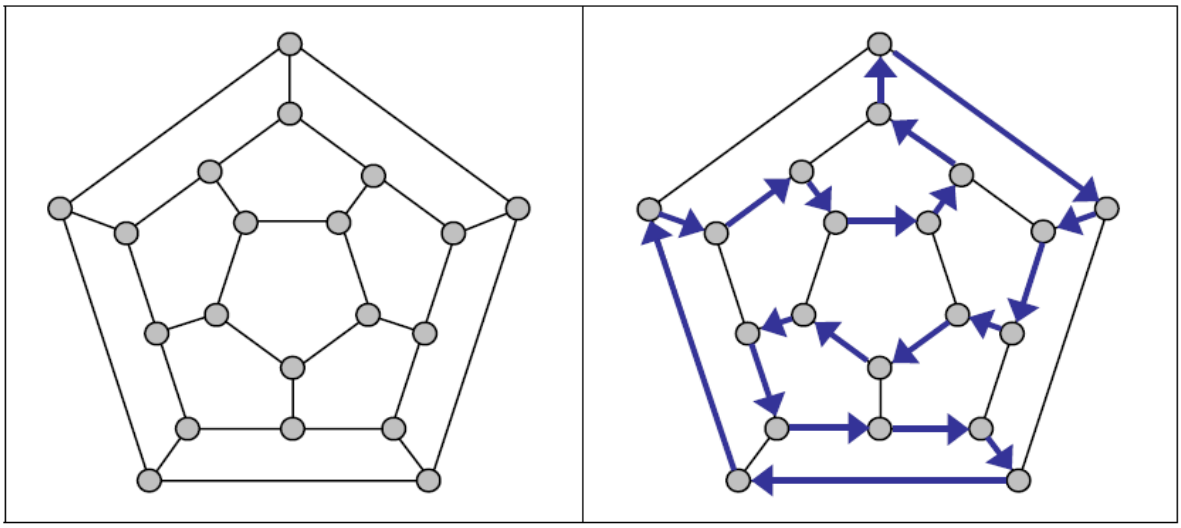


FIGURA 1 - JOGO DE HAMILTON
Fonte: (GANHOTO, 2004)

Os problemas de roteirização de veículos podem ser do tipo de roteirização pura ou combinados de roteirização e programação. Quando a definição dos roteiros abrange não somente os aspectos espaciais ou geográficos, como também os temporais, tais como restrições de horários de atendimento nos pontos a serem

visitados ou precedências entre tarefas, os problemas são denominados roteirização e programação de veículos. No caso de roteirização pura, tais restrições não são consideradas para a definição dos roteiros e das sequências de atendimento. As soluções estão voltadas aos aspectos espaciais da localização dos pontos a serem visitados. Neste trabalho utilizou-se a roteirização pura.

2.1 PROBLEMA DO CAIXEIRO VIAJANTE

De acordo com Zamboni (1997), a primeira vez que o termo “Problema do Caixeiro Viajante” foi utilizado nos meios matemáticos foi entre 1931 e 1932. Entretanto, em 1832, um livro publicado na Alemanha intitulado *“Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein. Von einem alten Commis-Voyageur”* em seu último capítulo, traz a essência do Problema do Caixeiro Viajante.

Zamboni (1997) também cita que o primeiro artigo que propôs uma “Solução de Larga Escala para o Problema do Caixeiro Viajante” foi publicado por Dantzig, Fulkerson e Johnson em 1954, no Jornal da Sociedade de Pesquisa Operacional da América, podendo ser considerado um dos principais eventos da história da otimização combinatória.

O Problema do Caixeiro Viajante (PCV) consiste em estabelecer uma única rota que passe em cada nó de um grafo, uma e apenas uma vez, retornando ao nó inicial no final do percurso. Este roteiro Hamiltoniano deve ser feito de modo que a distância total percorrida seja mínima. A Figura 2 mostra com setas de cor vermelha um roteiro Hamiltoniano.

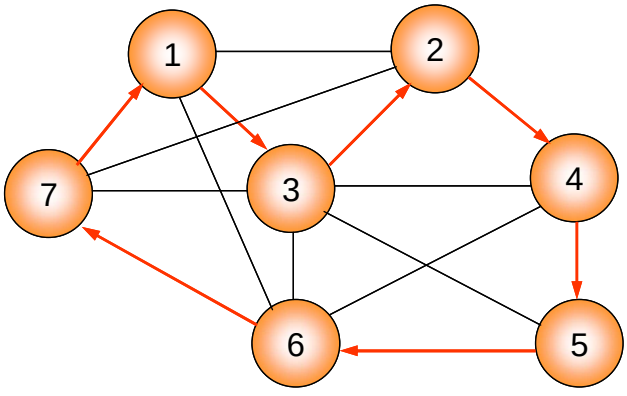


FIGURA 2 - EXEMPLO DO PCV

O conjunto de rotas possíveis é o resultado de todas as combinações possíveis e pode ser calculado por $(n-1)!$, sendo *n* o número de nós.

Este problema pertence à classe de problemas conhecida por *NP-Hard*, isto é, não existem algoritmos com limitação polinomial capazes de resolvê-lo. Assim a quantidade de passos de um algoritmo para solucioná-lo otimamente, não pode ser dada por uma função polinomial. Logo, apenas os problemas de pequeno porte podem ser solucionados de forma ótima. Problemas de grande porte tornam-se inviáveis se forem utilizados métodos exatos, em virtude do esforço computacional que seria exigido para resolvê-los. Muitas abordagens de algoritmos heurísticos, que fornecem soluções factíveis próximas da ótima, têm sido desenvolvidas para resolver os problemas *NP-Hard*. Estes algoritmos vêm apresentando soluções parciais e ótimas para o problema, visto que, devido ao grande número de grandezas que influenciam o processamento computacional, tais como a capacidade de carga, a velocidade, o número de veículos, o tempo e a distância, esses precisariam de uma grande capacidade computacional para apresentar soluções exatas.

O Problema do Caixeiro Viajante possui muitas variações sendo que algumas possuem algoritmos de aproximação que fornecem resultados próximos do ótimo.

Essas variações do PCV podem ser classificas em relação a:

- **Simetria:** É dito simétrico se a distância do ponto "a" ao ponto "b" é igual a distancia do ponto "b" ao ponto "a". Do contrário é dito assimétrico.
- **Completude:** É dito completo se existe um caminho direto entre todos os pontos, do contrário é dito não completo.

Suponha que o grafo G, com peso nas arestas é completo. Dizemos que os pesos de suas arestas satisfazem a desigualdade triangular se $c_{ik} \leq c_{ij} + c_{jk}$ para quaisquer três vértices *i, j, k*. Por exemplo, se os vértices de um grafo são pontos no plano e o custo de se viajar entre dois vértices é a distância euclidiana comum entre eles, então a desigualdade triangular é satisfeita. O PCV restrito ao conjunto de instâncias em que G é completo e o custo de suas arestas satisfaz a desigualdade triangular é conhecido como métrico.

Neste trabalho as restrições utilizadas nas heurísticas e metaheurísticas descritas são tais que o grafo seja completo, simétrico e que satisfaz a desigualdade triangular.

2.1.1 Formulação Matemática para o PCV

Seja o grafo $G(N, A)$ onde *N* representa o conjunto de vértices $(|N| = n)$ e *A* o conjunto de arcos. Seja $C = [c_{ij}]$, uma matriz simétrica com os custos ou distâncias mínimas entre os nós da rede considerando ainda que $c_{ii} = +\infty \quad \forall i \in N$. A matriz $X = [x_{ij}]$ é composta pelas variáveis de decisão do problema, com $x_{ij} = \begin{cases} 1, \text{ se o arco } a_{ij} \in \text{ rota} \\ 0, \text{ se o arco } a_{ij} \notin \text{ rota} \end{cases}$. Desta forma,a formulação de Programação Linear Inteira para o problema, devido a Golden *et al.*, de 1977, (BODIN *et al.,* 1983), pode ser escrita como:

$$\text{Minimizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{a}$$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, ..., n) \tag{b}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, ..., n) \tag{c}$$

$$X = (x_{ij}) \in S \tag{d}$$

$$x_{ij} = 0 \quad \text{ou} \quad x_{ij} = 1 \quad (i, j = 1, 2, ..., n) \tag{e}$$

Os dois primeiros grupos de restrições (b) e (c) garantem que exatamente um arco (i, j) tem origem cada nó *i* da rota e exatamente um arco (i, j) é direcionado para cada nó *j* da rota. A penúltima restrição (d) contém um conjunto S que pode ser qualquer conjunto de restrições que impeça a formação de sub-rotas. Estas restrições são chamadas restrições de quebra de sub-rotas e podem ser, entre outras:

$$S = \left\{ (x_{ij}) : \sum_{i \in Q}\sum_{j \notin Q} x_{ij} \geq 1, \text{ para todo subconjunto próprio não vazio Q de N} \right\} \tag{f}$$

$$S = \left\{ (x_{ij}) : \sum_{i \in R} \sum_{j \in R} x_{ij} \leq |R| - 1, \text{ para todo subconjunto não vazio R de } \{2, 3, ..., n\} \right\} \quad \text{(g)}$$

$$S = \left\{ (x_{ij}) : y_i - y_j + n x_{ij} \leq (n-1), \text{ para } 2 \leq i \neq j \leq n \text{ para alguns números reais } y_i \right\} \quad \text{(h)}$$

onde:

$$y_i = \begin{cases} t & \text{se o nó i é visitado no passo t da rota} \\ 0 & \text{caso contrário} \end{cases} \quad \text{(i)}$$

2.1.2 PROBLEMA DOS MÚLTIPLOS CAIXEIROS VIAJANTES

O problema dos múltiplos caixeiros viajantes é uma generalização do problema do caixeiro viajante onde há uma necessidade em se determinar mais de um roteiro com o menor custo possível. Os veículos da frota devem partir e voltar de um depósito comum. Não há restrições quanto ao número de nós que cada veículo pode visitar, apenas que cada veículo visite pelo menos um ponto uma única vez.

Não são abordadas neste problema restrições operacionais como capacidade dos veículos.

3. HEURÍSTICAS

As chamadas técnicas heurísticas permitem resolver de forma aproximada o Problema do Caixeiro Viajante. Existem três grandes classes de procedimentos para resolver o problema, a citar (BODIN, 1983):

a) Procedimentos de construção de rotas, que constroem rotas ótimas ou quase ótimas;
b) Procedimentos de melhorias de rotas, que efetuam melhorias em rotas já existentes;
c) Procedimentos compostos, que constroem uma rota inicial com o auxilio de um dos procedimentos de construção e utilizam melhorias para obter um resultado mais eficiente.

Mais recentemente, surgiram algumas técnicas conhecidas por Metaheurísticas, dentre as quais se destacam os Algoritmos Genéticos, *Ant System*, *Simulated Annealing*, *GRASP*, Busca Tabu, etc. Estas metaheurísticas serão abordadas no próximo capitulo.

As heurísticas ou algoritmos heurísticos são um conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas de elevado nível de complexidade, em tempo computacional razoável. Embora a exploração seja feita de forma algorítmica, o progresso é obtido pela avaliação puramente empírica do resultado.

Ao se pensar em um problema altamente combinatório, uma opção seria analisar todas as combinações possíveis em busca da melhor. Se o problema possui um universo de dados pequeno, realmente esta é a maneira correta de se buscar a melhor solução. Mas, os problemas reais, normalmente, possuem um número de combinações muito grande, o que torna inviável a análise de todas as combinações, uma vez que o tempo computacional torna-se muito elevado.

3.1 HEURISTICAS DE CONSTRUÇÃO DE ROTA

Heurísticas de construção de rotas para o Problema do Caixeiro Viajante são algoritmos que geram um circuito viável partindo de conjunto inicial de vértices, e modificando esse conjunto a cada iteração utilizando algum critério de escolha. Este processo busca boas soluções a um custo computacional razoável, porém, sem ser

capaz de garantir otimalidade ou até, em vários casos, de estabelecer quão perto de uma dada solução viável está da solução ótima.

Inicialmente as rotas podem ser construídas utilizando as seguintes técnicas de construção de rotas (BODIN *et al*, 1983):

- Procedimento do vizinho mais próximo;
- Inserção do mais próximo;
- Inserção do mais distante;
- Inserção mais rápida

Existem ainda outros procedimentos de construção de rotas que podem ser citados tais como Inserção do Mais Barato, Inserção Arbitrária, Cobertura Convexa, Inserção do Maior Ângulo e outros descritos detalhadamente em Bodin *et al*.(1983).

Após a aplicação dessas técnicas, eventualmente é possível melhorá-las utilizando Heurísticas de Melhoramento de Rotas. Segundo Lin e Kernighan (1973), dentre as técnicas existentes para realizar estas melhorias podem ser citadas as técnicas de troca de arcos mais conhecidas por 2-opt e 3-opt.

3.1.1 Vizinho mais próximo

É a técnica mais intuitiva das heurísticas e trabalha da seguinte forma: o ciclo inicia com um nó v_i atual e, a partir deste, encontra o próximo nó v_k de forma que a distância entre os dois nós seja mínimo. Ao encontrar este nó v_k, o algoritmo insere o mesmo no final do ciclo e repete a operação até que todos os nós pertençam à solução. Finalmente o ciclo é fechado ligando o nó inicial ao nó final. Não é permitido visitar mais de um nó duas vezes ou modificar a escolha, ou seja, após um nó ser inserido em uma determinada posição da rota, ele não sofrerá modificação de posicionamento (BODIN, 1983).

No Algoritmo 1 está representado o pseudocódigo da heurística do Vizinho mais próximo:

P1: Escolha um nó inicial;
P2: Encontre o nó mais próximo do último nó adicionado na rota e adicione-o na rota;
P3: Repita o passo 2 até que a rota contenha todos os nós e retorne ao primeiro.

ALGORITMO 1 - Algoritmo do Vizinho mais próximo

O tempo de computação envolvido ao algoritmo do vizinho mais próximo é da ordem n^2.

3.1.2 Inserção do mais próximo

Segundo Goldbarg (2005) o algoritmo da inserção do mais próximo é uma heurística que possui um processo onde três níveis de decisão são envolvidos: a escolha do vértice a ser inserido na solução; a posição de inserção desse novo vértice; e a decisão de um ciclo inicial.

No Algoritmo 2 está representado o pseudocódigo da heurística da Inserção do mais próximo:

P1: Inicie com um sub-grafo contendo apenas o nó i;
P2: Encontre o nó k tal que c_{ik} seja mínima e forme a rota i - k - i;
P3: Dada a sub-rota, encontre o nó k não pertencente a sub-rota mais próximo de qualquer nó da sub-rota;
P4: Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$. Insira k entre i e j;
P5: Volte ao passo 3 até formar um circuito Hamiltoniano.

ALGORITMO 2 - Algoritmo da Inserção do mais próximo

O tempo de computação envolvido ao algoritmo da Inserção Mais Próxima também é da ordem n^2.

3.1.3 Inserção do mais distante

Esse algoritmo é semelhante ao da heurística Inserção do Mais Próximo, diferente apenas no passo 2, quando então se escolhe a cidade *k* não pertencente ao ciclo, mais distante de qualquer cidade do ciclo. Também, no passo 3, o nó *k* não pertencente a sub-rota deve ser mais distante de qualquer nó e não o mais próximo

No Algoritmo 3 está representado o pseudo-código da heurística da Inserção do mais distante:

P1: Inicie com um sub-grafo contendo apenas o nó i

P2: Encontre o nó k tal que c_{ik} seja máxima e forme a rota i - k - i

P3: Dada a sub-rota, encontre o nó k não pertencente a sub-rota mais distante de qualquer nó da sub-rota

P4: Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$. Insira k entre i e j

P5: Volte ao passo 3 até formar um circuito Hamiltoniano

ALGORITMO 3 - Algoritmo Inserção do mais Distante

O tempo de computação envolvido ao algoritmo da Inserção Mais Distante também é da ordem n^2.

3.1.4 Inserção mais rápida

No Algoritmo 4 está representado o pseudo-código da heurística da Inserção mais Rápida:

P1: Tome um nó inicial para formar um circuito T com 1 nó e 0 arcos

P2: Dado o conjunto T_k, ache o nó z_k não pertencente a T_k mais próximo de um nó y_k em T_k

P3: Seja T_{k+1} a rota com k + 1 nós inserindo z_k imediatamente após y_k

P4: Repita P2 e P3 até a formação do circuito Hamiltoniano

ALGORITMO 4 - Algoritmo Inserção mais Rápida

O tempo de computação envolvido ao algoritmo da Inserção mais rápida também é da ordem n^2.

Exemplos bem detalhados e simples aplicando os Algoritmos (1), (2), (3) e (4) podem ser vistos no Apêndice 1.

3.2 HEURISTICAS DE MELHORIA DE ROTA

As heurísticas de melhoria são baseadas em modificações simples no circuito. Dado um circuito hamiltoniano, essas heurísticas fazem trocas para que seu comprimento seja reduzido, até que seja impossível reduzi-lo mais (um circuito localmente ótimo).

Além de oferecer bons resultados, essas heurísticas são usadas por vários pesquisadores como parte de outras técnicas usadas no PCV.

As heurísticas de melhorias de rota conhecidas por 2-opt e 3-opt, surgiram em meados da década de 60 e consistem em permutar arcos em uma rota inicial factível, buscando encontrar uma rota de menor custo.

Na heurística 2-opt, dois arcos são desligados e substituídos outros dois de modo que a distância total na nova rota formada seja menor que na rota inicial. Analogamente ocorre na heurística 3-opt, onde três arcos são permutados. Estes procedimentos terminam geralmente em um ótimo local, e são considerados métodos eficientes para resolver o Problema do Caixeiro Viajante. Mais tarde foi criada outra heurística ainda mais forte do que a 2-opt e 3-opt. Chama-se heurística k-opt, onde são permutados k arcos ($k \geq 3$). Embora a qualidade das soluções desta última técnica melhore à medida que k aumenta, elas se tornam quase impraticáveis do ponto de vista computacional quando $k \geq 4$. Algumas aplicações práticas destas heurísticas podem ser vistas em Costa (1997) e Zamboni (1997).

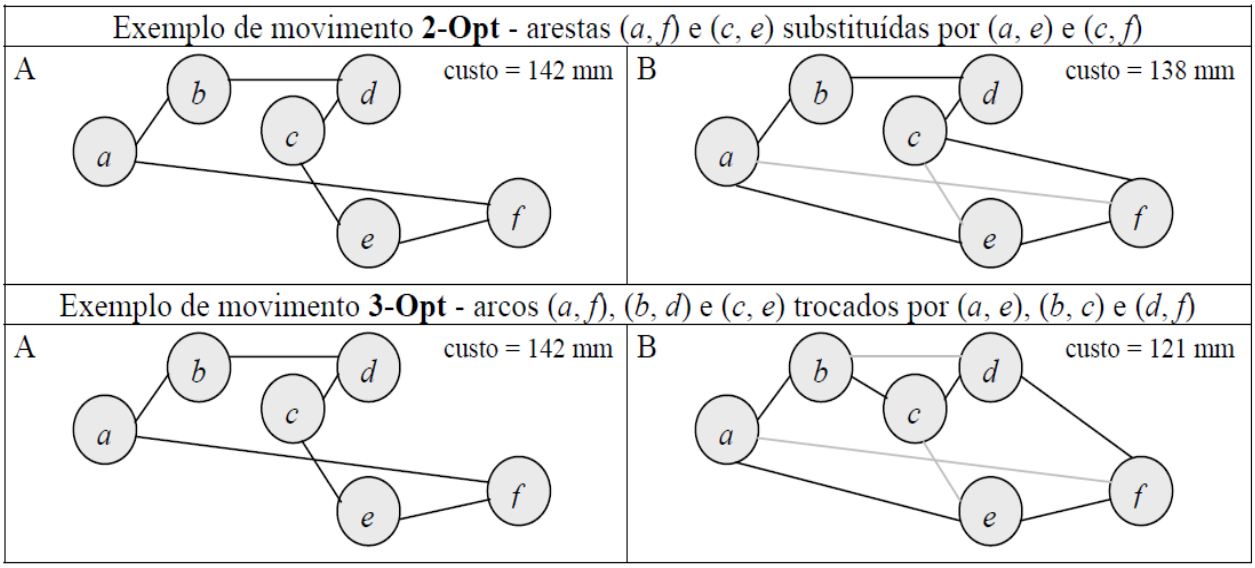


FIGURA 3 - EXEMPLOS DE SUBSTITUIÇÃO 2-OPT E 3-OPT

Fonte: (GANHOTO, 2004)

Os passos do algoritmo para efetuar melhoramentos 2-opt e 3-opt são os seguintes:

P1: Obtenha uma rota inicial factível, utilizando algum dos procedimentos de formação de rotas;

P2: Desligar 2 arcos da rota atual para o caso de 2-opt e 3 arcos para o caso 3-opt, reconectando os nós por meio de arcos diferentes daqueles que foram desconectados e que determinem uma nova rota factível. Se o comprimento da rota nova for menor que o comprimento da rota anterior, troque a rota atual pela rota nova;

P3: Prossiga no passo 2 até que nenhuma melhoria possa ser alcançada.

ALGORITMO 5 - Algoritmo de Melhoria 2-opt e 3-opt

4. METAHEURISTICAS

> "Uma metaheurística é um conjunto de conceitos que podem ser usados para definir os métodos heurísticos que podem ser aplicados a um vasto conjunto de problemas diferentes. Em outras palavras, uma metaheurística pode ser vista como um quadro geral de algoritmos que podem ser aplicados a diferentes problemas de otimização com relativamente poucas modificações para torná-los adaptados a um problema específico."
> (METAHEURISTICS NETWORK, 2011)

Metaheurísticas 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. Conforme Chaves (2003), a principal característica das metaheurísticas é a capacidade que estas possuem de escapar de ótimos locais[1] dando certa flexibilidade às restrições da função objetivo.

Heurísticas baseiam-se na melhoria do movimento a ser executado, buscando um ótimo local, mas são limitadas, fornecendo sempre a mesma solução quando iniciadas em um mesmo ponto inicial. As Metaheurísticas suprem essa deficiência, baseando-se na melhoria da solução, deixando temporariamente um ótimo local para procurar um ótimo global[2], mesmo que isso motive a perda temporária do valor da função objetivo.

A seguir estão descritas algumas das metaheurísticas, aplicáveis ao PCV.

4.1 OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS

Ant Colony Optimization (ACO) é uma abordagem inspirada no uso que as formigas fazem dos feromônios, um sinal químico liberado por um animal que dispara uma resposta natural em outros membros da mesma espécie, como um canal de comunicação. (GLOVER e KOCHENBERGER, 2003).

Basicamente, de início, uma formiga sai em busca por alimento, aleatoriamente, e marca seu trajeto através dos feromônios. Cada colônia tem um cheiro o que distingue as formigas de um ninho e de outro. Os feromônios secretados pelas formigas informam qual o tipo de trabalho que deve ser feito, informa se há perigo por perto, se há necessidade de recrutamento de operárias

[1] **Ótimo Local:** um ponto é de ótimo local quando fornece o melhor valor para a função objetivo em uma vizinhança do mesmo.

[2] **Ótimo Global:** um ponto é de ótimo global quando fornece o melhor valor para a função objetivo, considerando toda a região factivel.

para carregarem alimento rapidamente para a colônia, entre outros comportamentos específicos. Os feromônios podem ainda atrair o sexo oposto para que haja a cópula e fazer com que as operárias andem em fila formando trilhas imensas em direção ao alimento e de volta para o ninho. Este sistema complexo de comunicação torna as formigas aptas a colonizarem diversos ambientes, além de terem um sistema impressionante de defesa da colônia, tornando-as quase invencíveis.

Ao achar alimento ela reforça esse caminho, assim outras formigas ao se depararem com esse trajeto, de acordo com a quantidade de feromônio, têm mais probabilidade de segui-lo, e assim estabelecem um caminho “seguro” para a comida, reforçando ainda mais o caminho com feromônio, ou seja, quanto mais formigas seguem um caminho, mais atrativo ele se torna.

Na Figura (4), exemplifica-se uma demonstração do que ocorre na heurística ACO. No primeiro momento tem-se o ninho das formigas e o caminho livre até seu alimento. Num segundo momento, tem-se um obstáculo entre o ninho e sua comida e observa-se como as formigas ultrapassam a barreira que as impede de chegar ao ninho, sendo que um maior número de formigas opta pelo menor caminho e finalmente, no terceiro momento, pode-se observar que todas as formigas fazem o mesmo trajeto e esse trajeto é o de menor distância entre a comida e seu ninho.

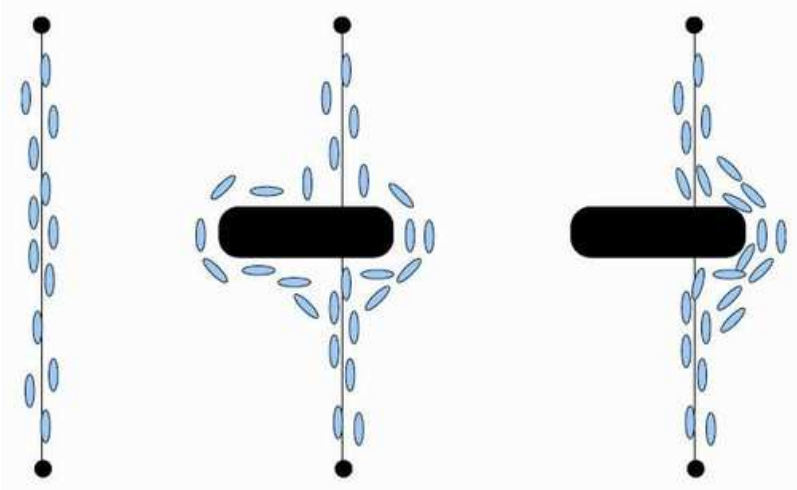


FIGURA 4 - COMPORTAMENTO DAS FORMIGAS MEDIANTE UM OBSTÁCULO

4.1.1 Heurística *Ant System* (AS)

Pesquisadores, baseados no estudo do comportamento de uma colônia de formigas, propuseram o algoritmo *Ant System* (AS). A heurística conhecida por *Ant System* foi primeiramente desenvolvida por Colorni e colaboradores (COLORNI *et*

al., 1991, 1992), e a idéia básica é utilizar um conjunto de agentes que cooperam e se comunicam entre si para resolver problemas de otimização (DORIGO *et al.,* 1996). A técnica foi inspirada no comportamento de uma colônia de formigas, no processo de levar alimento até o formigueiro, considerando os seguintes aspectos:

- As formigas reais são capazes de encontrar o caminho do formigueiro até o local onde encontram o alimento e voltar ou contornar obstáculos com relativa facilidade embora sejam quase cegas;
- Estudos descobriram que esta capacidade é o resultado de uma espécie de comunicação "via química", entre estes insetos, utilizando uma substância chamada de feromônio.

O fenômeno é causado pela presença simultânea de muitas formigas.

O método *Ant System*, ensaia uma reprodução artificial deste mecanismo observado em uma colônia de formigas.

4.1.2 PCV Baseado no Algoritmo AS

Para encontrar uma solução aproximada para o PCV, a idéia é modelar a construção de uma trilha de feromônio produzida pelas formigas. As soluções são formadas pelas formigas que caminham entre as cidades representadas no grafo do problema até completar uma rota.

Considere o problema do caixeiro viajante definido em um grafo $G(N,A)$ com $|N|$ nós e $|A|$ arcos, onde o arco (i, j) conecta os nós n_i e n_j e tem associado um valor real $d_{ij} = d(n_i,n_j)$. O problema consiste em encontrar uma permutação π dos nós que minimizam: $\sum_{i=1}^{|N|} d(n_{\pi(i)},n_{\pi(i+1)})$ onde $\pi(|N|+1) \equiv \pi(1)$, ou seja, procura-se uma seqüência de vértices que forneça o menor circuito Hamiltoniano possível.

Define-se que as formigas são agentes com as seguintes propriedades:

- As formigas "lembram" as cidades já visitadas usando uma lista ordenada, chamada de lista tabu (LT). Observa-se não se tratar aqui do *Tabu Search*, que é outro algoritmo;
- A cada passo, as formigas escolhem, usando uma regra probabilística, uma cidade para se mover, dentre aquelas que não constam na LT;

- Depois que a rota tiver sido completada por uma formiga, ela atribui a substância feromônio τ_{ij} de cada arco (i, j) usado, “limpando” a LT.

A regra probabilística (no caso do PCV) usada como função de seleção é o procedimento de Monte Carlo onde a possibilidade de escolher uma cidade *j* como destino é dada por:

$$\text{Prob(j)} = \frac{\tau_{ij}^{\alpha} . \eta_{ij}^{\beta}}{\sum_{l \notin LT} \tau_{il}^{\alpha} . \eta_{il}^{\beta}} \qquad (1)$$

Onde:

- τ_{ij} é a intensidade de feromônio do arco (i, j);
- $\eta_{ij} = \frac{1}{d_{ij}}$ é chamado de “visibilidade” do nó j a partir do nó i;
- $\rho\,(0 \leq \rho \leq 1)$ é um parâmetro do algoritmo, tal que $(1 - \rho)$ pode ser interpretado como um fator de “evaporação” do feromônio;
- τ é um parâmetro do algoritmo tal que inicialmente as formigas estão livres para se mover quase aleatoriamente;
- α e β são parâmetros que permitem um controle do impacto relativo dos dois critérios (visibilidade e intensidade de feromônio).

Os passos do algoritmo onde Q é uma constante positiva são:

P1: Faça $\eta_{ij} = \frac{1}{d_{ij}}$; $\tau_{ij} = \tau$

P2: Para cada formiga *k* (k=1 até M) faça

LT_k = (começar com uma cidade para a formiga *k*)

Enquanto $|LT_k| < |N|$ faça

Selecione uma cidade *j* para se deslocar

Adicione *j* em ordem na lista LT_k

L_k = distância total do percurso descrito na lista LT_k

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{se o arco (i, j), pertence ao percurso descrito em } LT_k \\ 0 & \text{caso contrario} \end{cases}$$

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$$

P3: $\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij}$

P4: Se não foi atendida a condição de término vá ao passo 2.

ALGORITMO 6 - Algoritmo *Ant System* para o PCV

Diversas variações do *Ant System* têm sido testadas e dentre elas, três se destacam:

- A primeira consiste em um procedimento de construção que começa com um número pequeno de cidades. Outras cidades vão sendo adicionadas, uma a uma, permitindo que as formigas identifiquem boas distribuições de feromônio para cada subproblema, que serão utilizados como base para novos subproblemas;
- A segunda consiste em modificar o nível de feromônio quando uma estagnação é detectada. Um comportamento de estagnação ocorre quando a distribuição do feromônio é tal que todas as formigas seguintes fazem o mesmo percurso. A modificação é obtida, com a troca em um pequeno número de iterações dos parâmetros α e β;
- A terceira variação designa valores específicos de α e β para cada formiga, e os faz evoluir por algoritmos genéticos, com uma função de "*fitness*" inversamente proporcional ao comprimento total da rota. Os parâmetros podem ser codificados por cadeias binárias e os operadores genéticos clássicos podem ser utilizados.

Um exemplo didático de uma iteração do funcionamento deste algoritmo para o PCV encontra-se no Apêndice 2.

4.2 *SIMULATED ANNEALING*

A origem da técnica de otimização conhecida por *Simulated Annealing* vem de 1953, quando foi usada para simular em um computador o processo de "*annealing*" de cristais. A idéia de aplicar este método para resolver problemas de otimização combinatória surgiu bem mais tarde (KIRKPATRICK *et al*., 1983), (ARAGON *et al.,* 1984).

O método surgiu da seguinte observação da mecânica:

O resfriamento gradativo de um material a partir de uma alta temperatura inicial leva o material a estados mínimos de energia. Informalmente esses estados são caracterizados por uma perfeição estrutural do material resfriado que não se obteria caso o resfriamento não tivesse sido gradativo. Sob outras condições menos cuidadosas de resfriamento, o material se cristalizaria com uma energia "localmente mínima", apresentando imperfeições estruturais. A esse processo cuidadoso de resfriamento dá-se o nome de "*annealing*".

A referida simulação a uma temperatura fixa *T* consiste em dar um pequeno deslocamento a um dos átomos, computando a variação ΔE da energia do sistema. Se $\Delta E \leq 0$, o deslocamento é incorporado ao estado do sistema, que é então utilizado no passo seguinte. Caso contrário, isto é, se $\Delta E > 0$, a aceitação ou não do deslocamento passa a ser uma decisão probabilística.

A "intuição" que está por trás da proposta em se utilizar *simulated annealing* como ferramenta de otimização combinatória é a seguinte:

- Identifica-se a função energia do sistema com a função objetivo que se quer otimizar, como por exemplo, minimizar, e, por outro lado, os átomos do sistema são associados às variáveis do problema;
- Para cada temperatura de uma seqüência de temperaturas decrescentes realiza-se a simulação descrita. No final do processo, espera-se que o sistema estacione em um estado de energia globalmente mínima, por analogia com a física do problema.

O fato do método *simulated annealing* permitir a aceitação de configurações intermediárias do problema em que cresce o valor da função objetivo que se deseja

minimizar é crucial. Essa aceitação temporária de soluções “piores” significa que o método admite caminhar “morro acima”, na esperança de encontrar “vales” mais profundos.

Na proposta original de *simulated annealing* a função *r* que promove a redução de temperatura foi dada por $r(k) = \alpha^{k+1} T_o$, para todo $k \geq 0$ e algum $0 < \alpha < 1$.

A probabilidade do algoritmo aceitar *j* como solução factível a partir de uma solução *i* corrente é dada por:

$$\text{Prob \{ j vir depois de i \}} = \begin{cases} 1 & \text{se } \Delta \leq 0 \\ e^{-\frac{\Delta}{T}} & \text{se } \Delta > 0 \end{cases} \qquad (2)$$

onde $\Delta = f(j) - f(i)$.

Quando a temperatura *T* possui um valor relativamente alto, praticamente todo deslocamento é aceito, uma vez que a probabilidade é praticamente igual a 1. À medida que o valor de *T* decresce, a aceitação de soluções com maior valor na função objetivo torna-se cada vez mais improvável.

Algumas decisões e escolhas de parâmetros dependem do problema particular em questão. Por exemplo, a escolha da solução inicial x_0 e a escolha do conjunto $A(x) \subset D$ de soluções dentre as quais uma será selecionada para o teste de aceitação, quando a simulação se encontra no ponto *x*, sendo *D* o conjunto dos valores factíveis, podem variar de problema para problema. Segundo Barbosa (1989), e Mitra *et al.,* (1986), um dos principais resultados relativos às condições sob as quais se pode garantir a convergência para um mínimo local, é selecionar T_0 e $r(k)$ de tal forma que:

$$T_o = \frac{\xi}{\log(1 + k_o)} \qquad (3)$$

$$r(k) = \frac{\xi}{\log(2 + k_o + k)} \qquad (4)$$

para $k \geq 0$, onde k_0 é qualquer parâmetro que satisfaça $1 \leq k_0 \leq \infty$ e ξ é uma constante que depende das características da cadeia de Markov associada ao Processo de “*Simulated Annealing*”.

O maior “mistério” do algoritmo é decidir quando a temperatura *T* será adaptada. A verificação da verdadeira condição de equilíbrio é realmente bastante difícil. Uma solução simples é introduzir outro parâmetro *H*, fixando o número máximo de iterações para ser usado com a temperatura *T*, provavelmente dependendo do número de variáveis do problema.

Os passos do algoritmo contidos em Barbosa (1989) são os seguintes:

Seja $x_0 \in D$ a solução inicial e T_0 a temperatura inicial;
$k = 0$; $x_k = x_0$; $T_k = T_0$
Enquanto $T_k \geq T_{min}$ faça
Inicio
Enquanto a temperatura T_k não atinge o equilíbrio, **faça**
Inicio
selecione aleatoriamente um ponto $x' \in A(x_k) \subset D$
$\Delta = f(x') - f(x)$
se $\Delta \leq 0$ **então**
$x_k = x'$
senão
$x_k = x'$ com probabilidade $e^{-\frac{\Delta}{T_k}}$
fim
$T_{k+1} = r(k)$; $x_{k+1} = x_k$; $k = k + 1$
Fim
Retorne a solução x_k .

ALGORITMO 7 - Algoritmo Simulated Annealing

Um exemplo didático de duas iterações do funcionamento deste algoritmo para o PCV encontra-se no Apêndice 3.

4.3 ALGORITMOS GENÉTICOS

De acordo com Nunes (1998), os Algoritmos Genéticos (AG) constituem um método de otimização inspirado no processo Darwiniano de seleção natural dos seres vivos. Na realidade, os AG fazem parte de uma classe de paradigmas e técnicas computacionais inspiradas na evolução natural, denominada de Computação Evolucionista.

Hans-Paul Schwefel, um dos pioneiros da Computação Evolucionária (CE) considera difícil definir quem teria sido o primeiro a conceber um algoritmo evolucionista.

Além dos algoritmos genéticos, pode-se destacar outros dois paradigmas da CE, todos desenvolvidos independentemente: as Estratégias Evolucionistas (EE) que surgiram a partir do trabalho de Ingo Rechenberg de 1973 e a Programação Evolucionista (PE), introduzida por Lawrence Fogel e colaboradores em 1966.

Segundo Barbosa (1997), fica cada vez menos nítida e também menos úteis as fronteiras entre estes três paradigmas. Neste trabalho utilizou-se apenas os algoritmos genéticos.

4.3.1 Funcionamento dos Algoritmos Genéticos

Os algoritmos genéticos (AG) foram concebidos por John Holland, no final da década de 60, buscando inspiração no que se conhece sobre o processo de evolução natural, conhecimento este iniciado solidamente com a teoria da evolução de Darwin no famoso trabalho *The Origin of Species* (HOLLAND, 1992).

Segundo Grefenstette (1986), um AG é um procedimento iterativo que mantém uma população de estruturas, chamadas de “indivíduos”, que representam as possíveis soluções para um determinado problema. A cada iteração (“geração”), os indivíduos da população passam por uma avaliação que verifica sua capacidade em oferecer uma solução satisfatória para o problema. Esta avaliação é feita conforme uma função que recebe o nome de função de aptidão, ou função de *fitness*. De acordo com esta avaliação, alguns indivíduos são selecionados, de acordo com uma regra probabilística, para passar por um processo de reprodução. Na verdade, aplica-se sobre os indivíduos selecionados os chamados operadores genéticos, gerando uma nova população de possíveis soluções. Pressupõe-se que a população, em média, vai ficando incrementalmente mais apta para solucionar o problema. Após um grande número de gerações, de acordo com um critério de término do algoritmo, o indivíduo mais apto até então é uma possível solução para o problema.

Embora os AG nem sempre possam encontrar a solução ótima para um determinado problema (ótimo global), na maioria das vezes são capazes de encontrar uma solução quase ótima, o que é aceitável quando se considera problemas muito complexos, como problemas de otimização combinatória, onde os métodos convencionais normalmente são inviáveis em razão do esforço computacional que seria necessário para resolvê-los.

Assim sendo, os AG constituem uma classe de ferramentas muito versátil e robusta, pois a busca da solução pode inclusive se dar em conjuntos não-convexos e mesmo disjuntos, com funções objetivo também não convexas e não diferenciáveis, podendo-se ainda trabalhar simultaneamente com variáveis reais, lógicas e inteiras. É importante ressaltar que em virtude de suas características, os AGs evitam atrações irremediáveis para ótimos locais, o que ocorre freqüentemente com alguns algoritmos usuais de programação matemática, permitindo uma melhor exploração do espaço de busca.

Algumas das principais características que diferenciam os AG de outras técnicas de programação matemática são as seguintes:

- Empregam uma população de indivíduos, ou soluções, que pode ter tamanho fixo ou variável, ao contrário da maioria das técnicas que efetuam uma busca “ponto-a-ponto”;
- Não trabalham diretamente com as possíveis soluções do problema, chamadas de fenótipos, mas sobre uma codificação das mesmas chamadas de genótipos;
- Empregam regras de transição probabilísticas ou estocásticas, sendo que a maioria dos algoritmos tradicionais usam regras determinísticas;
- Não exigem maiores informações adicionais sobre a função a otimizar.

Um pseudocódigo que generaliza a maioria dos AG existentes seria o seguinte (BARBOSA, 1997):

```
Inicie a população
    Avalie indivíduos na população
    Repita
        Selecione indivíduos para reprodução
        Aplique operadores de recombinação e mutação
        Avalie indivíduos na população
        Selecione indivíduos para sobreviver
    Até critério de parada satisfeito
Fim
```

ALGORITMO 8 - Algoritmo AG genérico

Os procedimentos associados aos verbos iniciar, avaliar, selecionar e aplicar, utilizados nesse algoritmo são comentados nas seções seguintes.

4.3.2 O Sistema de Representação e Codificação

Considerando que os AG não operam diretamente sobre os elementos do espaço de busca, a primeira etapa para se resolver um dado problema utilizando um AG consiste na codificação/representação destes elementos (GOLDBERG, 1989), (BARBOSA, 1997). Costuma-se chamar de fenótipos aos elementos deste espaço de busca, enquanto que o código que os representa é denominado de genótipo, em analogia com a terminologia encontrada na Genética. Matematicamente, a escolha da codificação para um dado problema é a função ou regra que associa os elementos do espaço de genótipos com aqueles do espaço de busca, os fenótipos. Convêm lembrar que os espaços de busca podem ser formados por elementos das mais diversas naturezas. Pode-se considerar o espaço das matrizes, vetores, as combinações de variáveis lógicas, inteiras e reais, e até programas de computador numa dada linguagem.

Normalmente, a codificação se resume à utilização de cadeias (*strings*) de comprimento *l*, formadas por caracteres de um determinado alfabeto. O caso mais comum é o binário, onde o alfabeto é composto pelos símbolos 0 e 1. Em alguns casos utiliza-se também o código de Gray, onde a diferença entre duas cadeias consecutivas é de apenas 1 bit. Tal construção é sempre possível, haja vista a bi-univocidade entre as cadeias de 0 e 1 e os números que elas representam.

Assim, no sistema binário a cadeia “10010101” poderia representar uma possível solução de um dado problema. Neste caso, tem-se $l = 8$ e o conjunto dos genótipos é formado por todos os números binários de 00000000 a 11111111 contendo, portanto $2^l = 2^8 = 256$ elementos. A codificação é a regra que associa a cada uma destas cadeias de números binários uma solução. De acordo com Nunes (1998) um possível elemento do espaço de busca composto pelas variáveis v_i de diferentes tipos poderia ser codificado como uma cadeia da forma $\underbrace{10010}_{v_1}\underbrace{11}_{v_2}\underbrace{101}\ldots\underbrace{100111001}_{v_n}$ também denominada cromossomo.

Cada uma destas subcadeias é denominada de gene e representa uma das diversas variáveis que compõe o cromossomo.

Existem situações onde mais de um cromossomo é associado a um indivíduo. O Homo sapiens é diplóide, isto é, possui 2 cromossomos. Um exemplo de aplicação de um AG com indivíduo diplóide pode ser visto em Rocha e Ochi (1997). A grande maioria das aplicações de AG's utiliza, entretanto, indivíduos haplóides com um só cromossomo.

Resumindo, o genótipo é composto de um ou mais cromossomos que são compostos por subcadeias de símbolos pertencentes ao alfabeto utilizado. Esta representação/codificação é uma etapa que depende do problema a ser resolvido.

4.3.3 A Função de Aptidão

A função de aptidão deve refletir a qualidade de um elemento em solucionar o problema (GOLDBERG, 1989), (BARBOSA, 1997). A regra que a determina depende do tipo de problema que está sendo considerado, e nos problemas de otimização, minimização ou maximização, ela está diretamente relacionada com a função objetivo. Ao se considerar o Problema do Caixeiro Viajante, por exemplo, a regra da função de aptidão pode ser a expressão que fornece a distância total percorrida pelo viajante.

Considerando que no decorrer das iterações os indivíduos vão se tornando cada vez mais semelhantes, pois a população tende a convergir, pode ser interessante aumentar a pressão de seleção, utilizando como função de aptidão uma composição da função objetivo com alguma função conveniente.

4.3.4 Os Esquemas de Seleção

Segundo Nunes (1998), nesta fase o objetivo principal é selecionar os indivíduos mais aptos para a geração de uma nova população.

São diversos os esquemas de seleção utilizados nos AG's. No esquema de seleção conhecido como seleção proporcional, a probabilidade de um indivíduo ser selecionado para participar do processo de reprodução é proporcional à medida do grau de *fitness* (aptidão) do indivíduo relativamente à população. Neste caso, a probabilidade p_i do indivíduo a_i ser selecionado, poderia ser dada pela fórmula (5):

$$p_i = \frac{f(a_i)}{\sum_{j=1}^{n} f(a_j)} \quad (5)$$

onde *f* é a função de aptidão e *n* o tamanho da população.

Este tipo de esquema de seleção que tem o efeito de aumentar a aptidão média da população, costuma ser chamada de seleção direcional (BARBOSA, 1997; LOPES, 1996; GOLDEBERG, 1989). Na prática, tudo se passa como se o sorteio dos elementos fosse feito através de um jogo de roleta, onde a probabilidade de cada indivíduo ser selecionado é proporcional ao seu *fitness*.

Em algumas situações, pode-se deixar de lado a magnitude do grau de *fitness* de um indivíduo, levando em consideração apenas o seu "*ranking*", ou posição relativa da medida de aptidão (MAYERLE, 1994).

4.3.5 O Processo de Reprodução

Segundo Barbosa (1997), existem basicamente duas classes ou tipos de algoritmos genéticos relativamente à forma com que os novos indivíduos são inseridos na população. O primeiro tipo é chamado de AG generacional, e tem como característica o fato de que toda a população é substituída pelos novos indivíduos criados depois do processo de seleção e aplicação dos operadores genéticos. Pode-se representar este tipo de algoritmo da seguinte forma:

Inicie a população P de alguma forma
  Avalie os indivíduos da população P
  Repita
    Repita
      Selecione indivíduos da população P
      Aplique os operadores genéticos
      Insira os novos indivíduos em P'
    Até que a população P' esteja completa
    Avalie os indivíduos da população P'
    P $\leftarrow$ P'
  Até que um critério de parada esteja satisfeito
Fim

ALGORITMO 9 - Algoritmo Genético generacional

Considerando que neste processo toda a população é substituída pela nova, corre-se o risco de perder bons indivíduos. Para evitar isto, pode-se utilizar um procedimento conhecido como elitismo, que consiste em passar para a geração seguinte uma cópia de alguns dos melhores indivíduos. O outro tipo de AG é conhecido como "*steady-state*", e caracteriza-se por criar apenas um indivíduo de cada vez, sendo que o indivíduo gerado pode ou não ser passado para a geração seguinte. Normalmente ele é transmitido para a próxima geração, se o seu valor de *fitness* for melhor do que o pior valor de *fitness* da população antiga. Pode-se representar estes AG com o seguinte pseudocódigo:

```
Inicie a população P de alguma forma
    Avalie os indivíduos da população P
    Ordene a população de acordo com o seu fitness
    Repita
        Selecione os indivíduos na população P
        Aplique os operadores genéticos
        Selecione um indivíduo f para sobreviver
        Se f é melhor que o pior elemento de P então
            Remova um indivíduo da população
            Insira f em P de acordo com seu "ranking"
    Até que um critério de parada esteja satisfeito
Fim
```

ALGORITMO 10 - Algoritmo AG "*steady-state*"

4.3.6 Os Operadores Genéticos

O objetivo dos operadores genéticos é operar sobre os indivíduos que foram selecionados para reprodução, produzindo um ou mais "descendentes". Os operadores são construídos após definida uma codificação para os elementos do espaço de busca. Dos diversos operadores genéticos propostos por Goldberg (1989) e outros autores, destacam-se basicamente dois tipos: os operadores de recombinação e os operadores de mutação, detalhados a seguir:

4.3.6.1 Operadores de Recombinação

Os operadores de recombinação atuam sobre os genótipos dos indivíduos selecionados, promovendo uma recombinação do material genético dos elementos “pais”, gerando os elementos “filhos”. Este tipo de operador costuma se chamar na literatura de AG’s de operadores de *crossover* em analogia com o termo da genética.

Entre os operadores comumente utilizados, pode-se citar o chamado *crossover* simples ou *crossover* de um ponto, onde mediante um procedimento aleatório, uma posição nos cromossomos é sorteada e o material genético dos cromossomos “pais”, situados à direita deste ponto, é trocado conforme o exemplo a seguir:

p_1 : 11111 1111111 $\qquad$ f_1 : 11111 0000000

p_2 : 00000 0000000 $\qquad$ f_2 : 00000 1111111

Os elementos p_1 e p_2 são os “pais” dos elementos f_1 e f_2.

Já *crossover* de dois pontos, são duas as posições sorteadas para a troca do material genético que está localizado entre eles:

p_1 : 11111 111 1111 $\qquad$ f_1 : 11111 000 1111

p_2 : 00000 000 0000 $\qquad$ f_2 : 00000 111 0000

Se esta exploração local desenvolvida pelo operador *crossover* for muito maior em relação à global, poderá haver uma convergência mais rápida, não necessariamente para um ótimo global, porém explorando insuficientemente o espaço de busca.

Outros tipos de operadores de recombinação podem ser criados de acordo com o tipo de problema e codificação dos fenótipos do problema abordado.

4.3.6.2 Operadores de Mutação

A mutação é possivelmente o operador genético mais simples de ser implementado. Segundo Barboza (2005) seleciona-se uma posição de um cromossomo e muda-se aleatoriamente o valor do gene correspondente para outro alelo possível. Desta maneira, consegue-se inserir novos elementos na população. A probabilidade de se efetuar uma mutação deve ser relativamente baixa, caso contrário o algoritmo se comportará fazendo uma busca aleatória, dificultando a convergência. Estes operadores exploram globalmente o espaço de busca,

possibilitando inclusive, recuperar algum bom material genético que possa ter sido perdido após sucessivas recombinações. Assim sendo, pode-se considerar os operadores de mutação como uma espécie de “apólice de seguro” contra perdas acidentais deste material genético de boa qualidade. Entre os tipos de mutação tem-se:

- **Mutação Simples:** Ao se considerar o alfabeto binário, uma posição do cromossomo é sorteada e o bit correspondente é invertido, isto é, se for 1 ele passa a ser 0 e vice-versa.
- **Mutação por Troca (*Swap*):** Consiste na troca aleatória de posição entre dois genes. Por exemplo, temos o indivíduo {1, 2, 3, 4, 5, 6, 7} e que após a mutação torna-se {1, 2, 6, 4, 5, 3, 7}.

4.3.7 Convergência, Diversidade Populacional e Nichos

Um dos critérios para a convergência de uma população e de um gene em um cromossomo foi definido por K. A. De Jong em 1975. Segundo ele, um gene converge quando 95% da população possui o mesmo gene, enquanto que uma população converge quando todos os seus genes convergem (NUNES, 1998).

O termo diversidade diz respeito à falta de semelhança entre os indivíduos de uma população e sua perda está diretamente ligada à convergência da mesma. Em uma situação ideal, um AG deveria convergir sem perda de diversidade genética. Isto aumentaria as chances de se encontrar o ótimo global através de um equilíbrio entre uma exploração global e local.

Para diminuir a perda da diversidade, alguns AG utilizam a chamada “redução de incesto” que reduz a operação de *crossover* entre elementos muito semelhantes, permitindo a recombinação apenas entre indivíduos cuja distância de *Hamming* seja grande. A distância de *Hamming* permite medir a diversidade entre dois cromossomos, sendo definida como o número de alelos (valores que os genes podem tomar) diferentes para as mesmas posições relativas (LOPES, 1996).

Muitas vezes se utiliza também o conceito de nicho ecológico, que consiste em manter subpopulações estáveis de indivíduos, de uma forma tal que cada subpopulação explore uma região do espaço de busca, sem que haja competição entre elas. Para aumentar a possibilidade de formação destas subpopulações pode-se diminuir a pressão seletiva, utilizando o fator de *crowding* que faz com que os

“cromossomos-filhos” venham a substituir os elementos com os quais tenham a máxima semelhança.

Quando o espaço de busca se torna muito grande, torna-se necessário o uso de uma população também numerosa, fazendo muitas vezes com que os AG’s clássicos (seqüenciais) se tornem menos eficientes. Ainda existe a questão dos parâmetros que deve ser considerada. É muito difícil definir os parâmetros mais adequados para um determinado problema. O tamanho da população, por exemplo, vai depender, dentre outros fatores, do espaço de busca considerado, não sendo possível, a priori, determinar o tamanho ideal de uma população para uma determinada classe de problemas. Grefenstette (1986) *apud* Lopes (1996) sugere que na grande maioria das aplicações uma população de 50 a 200 indivíduos possa parecer adequada. Assim, o método empírico de tentativas e erros, guiado por uma experiência prévia parece ser o caminho mais indicado para o equacionamento de problemas.

4.3.8 Algoritmo Genético para o Problema do Caixeiro Viajante

Existem diversas abordagens para o problema do caixeiro viajante utilizando algoritmos genéticos. Elas diferem entre si não apenas na questão dos parâmetros, mas também na forma de representar as soluções viáveis, de selecionar os indivíduos para reprodução, na maneira de definir os operadores genéticos (WHITLEY *et al*., 1989). Mayerle (1994) desenvolveu um AG cujos testes revelaram resultados bastante satisfatórios, comparativamente a outras técnicas clássicas para resolver o problema. Este algoritmo também foi utilizado com sucesso em Bezerra (1995).

Uma rota solução para o PCV pode ser considerada como sendo o conjunto $X = \{x_1, x_2, ..., x_n\}$ o composto com as *n* cidades pelas quais o caixeiro viajante deve passar, retornando ao ponto de partida no final da jornada. Assumindo que o custo da viagem entre cada par de cidades é conhecido e dado por $w(x_i, x_j)$, $\forall x_i, x_j \in X$, sendo $w(x_i, x_j) > 0$, define-se a estrutura do cromossomo, a função de *fitness* e os operadores de *crossover* e mutação utilizados da forma seguir.

4.3.8.1 Estrutura do Cromossomo

Existem várias maneiras de codificação do espaço de busca para o PCV. Dentre as principais representações estão: a ordinal, por caminho (ou inteiros) e por adjacência.

- **Representação Ordinal:** Nesta forma de codificação, cada solução é representada como uma lista de *n* cidades, onde o i-ésimo elemento da lista é um número entre 1 e (n – i + 1). Existe uma lista ordenada de cidades que serve como referência para construir a representação. A lista *L*, varia de 1 a **n** cidades. O cromossomo é um vetor de posicionamento da cidade na lista. Um exemplo didático desta representação encontra-se no Apêndice 4.
- **Representação por Caminho (ou Inteiros):** O cromossomo é formado pela sequência dos nós na solução.

  Por exemplo:

  O cromossomo c1 = {1, 2, 3, 4, 5, 6, 7} representa a solução r_1 = {1, 2, 3, 4, 5, 6, 7}. Esta representação é talvez a mais natural de uma solução para o PCV e de simples implementação. Neste trabalho utilizaremos esta representação.
- **Representação por Adjacência:** Cada cromossomo representará um circuito hamiltoniano, onde cada vértice possui um sucessor. Assim, cada cromossomo terá a forma $r_i = (s_{i1}, s_{i2}, ...s_{ij}, ..., s_{in})$, onde o gene s_{ij} é o vértice sucessor do vértice x_j no i-ésimo circuito (MAYERLE, 1994).

  Por exemplo:

  Um cromossomo c_1 = {4, 3, 6, 2, 1, 5} apresenta os seguintes arcos em uma rota associado a c1: 1 → 4, 2 → 3, 3 → 6, 4 → 2, 5 → 1, 6 → 5. Assim o cromossomo representa a seguinte solução r_1= {1, 4, 2, 3, 6, 5}. O inconveniente desta notação é que mesmo não havendo repetição, *R* pode não representar um circuito hamiltoniano, como por exemplo; c_2 = {2, 4, 5, 1, 6, 3}, que apresenta como solução r_2 = {1, 2, 4, 1} e {3, 5, 6, 3}.

4.3.8.2 Função de Aptidão

A função de aptidão ou função de *fitness* representa a capacidade de um indivíduo em adaptar-se ao meio ambiente.

Para problemas de minimização, quanto menor for o custo total do circuito, melhor será o *fitness* do cromossomo correspondente e conseqüentemente maiores serão as chances deste indivíduo sobreviver e vir a se reproduzir (MAYERLE, 1994).

4.3.8.3 Processo de Seleção

A principal idéia do processo de seleção é permitir que os indivíduos mais adaptados (melhor *fitness*) tenham maior chance de se reproduzir. Barboza (2005) afirma que a seleção elitista consiste em copiar ou reproduzir os melhores indivíduos da população atual para a próxima geração, garantindo que estes cromossomos não sejam destruídos nas etapas de recombinação e mutação. Sua vantagem é que se no caso ótimo global for descoberto durante o processo de busca, o AG deve convergir para tal solução.

Considerando que a população possui m cromossomos dispostos em ordem crescente dos seus custos, isto é $C_1 \leq C_2 \leq ..... \leq C_m$, onde C_i é o custo associado ao cromossomo r_i, a escolha de um indivíduo para ser submetido aos operadores genéticos é feita considerando uma distribuição de probabilidade proporcional ao índice dos cromossomos. A função de seleção proposta por Mayerle (1994) e modificada por Nunes (1998) segue esta distribuição de probabilidade. A fórmula para encontrar o indivíduo é dada por (6):

$$\text{Select}(R) = \left\{ r_j \in R / j = m + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.\text{Rnd}\,(m^2 + m)}}{2} \right\rceil \right\} \qquad (6)$$

Onde:

- $\text{Rnd} \in [0,1)$ é um número aleatório uniformemente distribuído;
- $\lceil b \rceil$ é o menor inteiro maior do que *b;*
- $R = \{r_1, r_2, ..., r_m\}$ é o conjunto ordenado dos cromossomos, de modo que $C_1 \leq C_2 \leq ..... \leq C_m$.

4.3.8.4 Operador de Cruzamento

De acordo com Potvin (1996), existem diversos operadores de cruzamento desenvolvidos para manipular as representações descritas acima e estão classificados em: os que preservam a posição absoluta das cidades e os que preservam a ordem relativa.

Dentre os operadores que mantém a posição absoluta estão: o PMX (*Partially Mapped Crossover*) proposto por Goldberg e Lingle (1985) *apud* Potvin (1996) e o CX (*Cycle Crossover*), proposto por Oliver (1987) *apud* Potvin (1996). E entre os operadores que mantém a ordem relativa está o OX (*Order Crossover*) desenvolvido por Davis (1985) *apud* Potvin (1996)

- **Operador PMX:** Faz um *crossover* parcialmente mapeado utilizando dois pontos de corte e fazendo o *crossover* normalmente entre os dois pontos. Neste processo troca-se um vértice do pai 1 por um vértice do pai 2 para cada posição da rota entre os pontos, onde cada uma destas trocas define um mapeamento.

Exemplo:

p1 = (6, 5, | 4, 3, 2, | 1, 7)

p2 = (5, 2, | 1, 6, 7, | 3, 4)

O operador troca a cadeia de números entre os dois pontos de corte, e repete as cidades dos pais que não são repetidas:

f1 = (6, 5, | 1, 6, 7, | 1, 7)

f2 = (5, 2, | 4, 3, 2, | 3, 4)

A troca entre os pontos de *crossover* define os seguintes mapeamentos:

1↔ 4, 6 ↔ 3, 7 ↔ 2

Desta forma, as cidades repetidas dos cromossomos originais são substituídas pelas cidades mapeadas:

f1 = (3, 5, | 1, 6, 7, | 1, 2)

f2 = (5, 7, | 4, 3, 2, | 6, 1)

- **Operador OX:** É um operador *crossover* de dois pontos de corte onde estes cruzamentos geram dois filhos que herdam a ordem de visita dos pais.

Exemplo:

p1 = (6, 5, | 4, 3, 2, 1, | 7)

p2 = (5, 2, | 1, 6, 7, 3, | 4) - mantém os segmentos selecionados

f1 = (x, x, | 4, 3, 2, 1, | x)

f2 = (x, x, | 1, 6, 7, 3, | x)

Neste tipo de crossover, o segmento entre os dois pontos de corte é mantido. A seguir, para f1, partindo do ponto do segundo corte dos pais, as cidades do outro pai são copiadas na mesma ordem, omitindo-se as cidades que estão entre os pontos de corte. Ao se atingir o final do vetor continua-se no início do mesmo. Isto quer dizer, que a partir da sequência 4, 5, 2, 1, 6, 7, 3, obtém-se a sequência 5, 6, 7 (removendo-se o 4, 3, 2, 1 já presente) a ser inserida no filho 1 a partir do seu segundo ponto de corte. O mesmo raciocínio é usado para o filho 2.

f1 = (6, 7, | 4, 3, 2, 1, | 5)

f2 = (4, 2, | 1, 6, 7, 3 | 5)

- **Operador CX:** O operador CX possui um procedimento muito diferente dos operadores PMX e OX, visto que não utiliza pontos de corte. Ele executa a recombinação de modo que cada gene das descendências vem das posições correspondentes de qualquer um dos dois pais.

Exemplo:

p1 = (1, 2, 3, 4, 5, 6, 7, 8, 9)

p2 = (4, 1, 2, 8, 7, 6, 9, 3, 5)

A partir do primeiro pai, seleciona-se a primeira cidade e a coloca no filho 1:

f1 = (1, x, x, x, x, x, x, x, x)

O gene de mesma posição em p2 possui valor 4. Procurando-se este valor em p1, verifica-se que ele se encontra no quarto gene, e este valor deve ser levado para a quarta posição de f1:

f1 = (1, x, x, 4, x, x, x, x, x)

A seguir, verifica-se que o gene de mesma posição em p2 possui valor 8, e que este valor encontra-se na oitava posição em p1. Assim, este valor é colocado para a oitava posição de f1:

f1 = (1, x, x, 4, x, x, x, 8, x)

O gene que ocupa a mesma posição em p2 possui valor 3, o qual encontra-se na terceira posição em p1. Transporta-se esse valor para f1.

f1 = (1, x, 3, 4, x, x, x, 8, x)

Com a continuação do processo, coloca-se o 2 no segundo gene de f1:

f1 = (1, 2, 3, 4, x, x, x, 8, x)

Verifica-se que o gene que ocupa a mesma posição em p2 possui valor 1 e encontra-se na primeira posição, mas essa operação já foi realizada. Como não teremos mais escolhas, completamos com as cidades de p2:

f1 = (1, 2, 3, 4, 7, 6, 9, 8, 5)

O segundo filho é obtido através do mesmo processo, começando por p2, e resulta em:

f2 = (4, 1, 2, 8, 5, 6, 7, 3, 9)

Além destes operadores, Silva e Oliveira (2006), fizeram um estudo comparativo utilizando uma versão adaptada do HX (*Heuristic Crossover*), o qual destacou-se sobre os demais operadores, sobretudo quando o número de nós aumenta.

- **Operador HX:** O operador HX utiliza as distâncias entre as cidades, isto é, o tamanho dos arcos. Esta operação de *crossover* pode ser efetuada pelos seguintes passos:

P1: Escolha uma cidade aleatória inicial i de um dos pais.
P2: Calcular as distâncias entre a cidade aleatória i escolhida e suas vizinhas (i -1 e i + 1) em ambos os pais.
P3: Escolher a cidade de menor distância, a qual formará uma sub-rota.
P4: Se a menor distância escolhida formar um ciclo, então, escolha uma nova cidade aleatória que não introduza um ciclo.
P5: Repita os passos 2, 3 e 4 até que todas as cidades sejam incluídas na rota.

ALGORITMO 11 - Algoritmo HX

Um exemplo detalhado para este operador encontra-se no Apêndice 4.

4.3.8.5 O Operador de Mutação

Segundo Malaquias (2006), o algoritmo genético pode convergir muito rapidamente para uma região específica do espaço de busca caso nenhum mecanismo para evitar essa convergência seja implementado. Existe uma tendência de convergência rápida para uma região de mínimos locais ao invés de mínimos

globais. Para que isso não ocorra, pode-se criar uma rotina para explorar outras áreas do espaço de busca por meio de alterações nos genes por meio da mutação.

Os operadores de mutação introduzem uma alteração aleatória no individuo. Assim sendo, pode-se considerar os operadores de mutação como uma espécie de "apólice de seguro" contra perdas acidentais deste material genético de boa qualidade.

A mutação é feita por troca (*Swap*), e consiste na troca aleatória de posição entre dois genes.

4.3.8.6 Descrição do Algoritmo

Com as definições anteriores, e usando os princípios da evolução das espécies, pode-se construir um algoritmo genético, cujos principais passos são os seguintes (MAYERLE, 1994; BEZERRA, 1995):

P1: Construção da população inicial: gere os m cromossomos aleatórios representando os roteiros Hamiltonianos. Calcule os custos de todos os cromossomos gerados, construindo uma lista $R = (r_1, r_2, ..., r_m)$, de forma que $C_1 \leq C_2 \leq ..... \leq C_m$; faça *k* = 0, defina o erro ε e o número máximo de iterações k_{max};

P2: Teste: se $C_n - C_1 \leq \varepsilon$ ou $k \geq k_{max}$, então PARE e apresente o cromossomo r_1;

P3: Seleção Natural: elimine da lista *R* o pior cromossomo (último da lista *R*) e selecione dois cromossomos r_p = *Select* (R) e r_q = *Select* (R) com $r_p \neq r_q$;

P4: Reprodução: faça r_f = *Crossover* (r_p, r_q);

P5: Mutação: se o cromossomo r_f não representa um circuito Hamiltoniano, então proceda a mutação r_f = *Mutate* (r_f);

P6: Calcule F_f = *Fitness* (r_f) e insira o cromossomo r_f na lista *R*, mantendo a ordem crescente dos custos; faça k = k +1, e volte ao P2.

ALGORITMO 12 - Algoritmo Genético para o PCV

4.4 *GRASP*

O *GRASP (Greedy Randomized Adaptive Search Procedures)* é um método cuja introdução ocorreu em 1989, por Feo e Rezende. A motivação do desenvolvimento desse método foram os problemas difíceis de cobertura de

conjuntos. É uma metaheurística constituída por heurísticas construtivas de busca local, cada uma iniciando de uma solução diferente. As soluções iniciais são geradas por algum tipo de construção randômica gulosa ou algum esquema de perturbação.

De acordo com Festa e Resende (2002), *GRASP* é um processo iterativo em que cada iteração consiste em duas fases: construção da solução e aplicação da busca local na solução construída.

4.4.1 Fase da Construção

Nesta fase, produz-se uma solução factível, também de forma iterativa (os elementos da solução são escolhidos um a um). Inicialmente o elemento está em uma lista de candidatos (LC). Através de um fator α (alfa), onde $\alpha \in [0,1]$, é criada uma lista restrita de candidatos (LRC) que possui os melhores elementos de LC. Essa lista pode ser limitada por um número de elementos ou pela qualidade dos elementos que a compõem. Se a lista for limitada a um elemento, a solução encontrada será a única solução e não haverá uma diversificação da solução. Se a lista for ampla, serão geradas várias soluções diferentes produzindo uma maior variação.

Após a definição da LRC, seleciona-se um elemento da mesma para compor a solução. Esta seleção pode ser feita aleatoriamente ou através de um critério guloso. Após a adição do elemento na solução, o processo continua com a atualização de ambas as listas LC e LRC. O processo de construção é finalizado quando a cardinalidade de LC possuir valor zero.

Procedimento Construção(s)

S ← { }

Enquanto solução não completa **faça**:

$LCR = \{c \in C / g(c) \leq s_1 + \alpha(s_2 - s_1)\}$

c= selec_elem_aleat(LRC)

$S = S \cup \{c\}$

Fim enquanto

Fim Construção

$s_1 = \min\{g(t), t \in C\}$

$s_2 = \max\{g(t), t \in C\}$

$\alpha \in (0,1)$

ALGORITMO 13 - Algoritmo de Construção (GRASP)

De acordo com Feo e Resende (1995) o valor de α influencia na qualidade e diversidade da solução gerada na fase de construção.

Para $\alpha = 0$, t = s1 + α (s2 – s1)}→ t = s1 (construção gulosa)

Para $\alpha = 1$, t = s1 + α (s2 – s1)}→ t = s2 (construção aleatória)

4.4.2 Fase da Melhoria

O objetivo nesta fase é encontrar o máximo local na vizinhança da solução obtida na fase anterior. A busca é útil na grande maioria das vezes, dado que o *GRASP* não garante que a solução da primeira fase seja localmente ótima. Assim, testam-se as soluções vizinhas sucessivamente, até que se encontre a melhor. É importante que se escolha uma boa estrutura para representar a vizinhança e que as técnicas de busca nessa estrutura sejam eficientes.

Segundo Freddo e Brito (2008), a busca local pode ser sofisticada, entretanto não se deve esquecer do diferencial do *GRASP* que é amostrar o espaço com gerações rápidas. Quanto melhor for a qualidade da solução gerada na primeira fase, maior será a velocidade para encontrar um ótimo local pela fase de busca local.

O Apêndice 5 mostra um exemplo detalhado de aplicação do algorítmo *GRASP* incluindo essa segunda fase.

Freddo e Brito (2008) também salientam que o *GRASP* não faz uso de históricos no processo de busca. É possível armazenar uma ou várias soluções, como as melhores até um determinado momento. Ainda, o *GRASP* é simples, rápido e pode ser facilmente integrado com outras técnicas de busca.

4.5 BUSCA TABU (*TABU SEARCH*)

De acordo com Fraga (2006), a metaheurística conhecida como Busca Tabu foi proposta de forma independente, por dois autores, Glover em 1986 para problemas de Programação Inteira, sendo que em 1987 fez uma extensão para problemas de Pesquisa Operacional e por Hansen em 1986, e rapidamente tornou-se uma das mais robustas e usadas metaheuristicas para resolução de problemas combinatoriais.

É um procedimento adaptativo auxiliar, que guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca, superando o problema da convergência local em problemas de otimização. A Busca Tabu mantém uma lista de soluções candidatas visitada recentemente (Lista Tabu) e se recusa a revisitar estas candidatas até que um determinado tempo se passe. Quando um máximo/mínimo é encontrado, força-se uma busca para um ponto distante do corrente, pois não é permitida a permanência ou retorno para o máximo/mínimo. Isto faz com que uma área mais abrangente do espaço de soluções seja visitada.

Segundo Schopf *et al* (2004), a Busca Tabu utiliza estruturas flexíveis de memória para armazenar conhecimento sobre os espaços percorridos. Graças ao uso intensivo de memória adaptativa, este método consegue ter boas soluções, além disso, a solução final tem pouca dependência com a inicial pelo fato de os mecanismos que o método implementa fogem de ótimos locais.

Para cada problema tratado por esta metaheurística, é importante definir três elementos:

- **Espaço de Busca:** é o espaço de todas as possíveis soluções que podem ser consideradas durante a busca.
- **Vizinhança:** é um elemento fortemente relacionado ao espaço de busca. A cada iteração da Busca Tabu, as modificações que podem ser aplicadas na solução atual formam um conjunto de soluções vizinhas no espaço de busca, definindo a vizinhança do ponto transformado. Para cada espaço de busca considerado para um problema, diferentes vizinhanças deverão surgir.

- **Tabus:** são elementos que previnem visitas cíclicas no espaço de busca quando se está tentando fugir de um máximo/mínimo local e nenhuma transformação aplicada traz melhorias à solução atual. A idéia chave para resolver este problema é proibir movimentos que revertam o efeito de movimentos recentes. Os tabus são armazenados na Lista Tabu e possuem um número fixo e limitado de entradas.

Fraga (2006) também coloca que o número de iterações que um movimento estará proibido de ser realizado é definido pelo tamanho da lista tabu. Uma lista tabu muito grande pode restringir demais a busca, pois, ao tornar tabu um grande número de movimentos, o procedimento termina de forma prematura, por não existirem movimentos factíveis a realizar. Por outro lado, uma lista tabu muito pequena aumenta as chances do processo entrar em ciclo, pois reduz o número de iterações em que o movimento é proibido de ser realizado. Por isso, a Busca Tabu deve manter um rigoroso controle sobre o tamanho da lista tabu.

Apesar de todos os esforços para manter uma lista tabu de tamanho adequado, é possível que um movimento considerado tabu leve a uma solução não analisada ainda. Neste caso, uma função de aspiração pode tornar possível a aceitação de um movimento pertencente à lista tabu, desde que leve a uma solução desconhecida e com bons atributos. O critério de aspiração mais comum na Busca Tabu é baseado na função de avaliação, ou seja, o movimento tabu é aceito somente se levar a uma solução melhor do que a melhor solução existente até o momento.

Um exemplo do funcionamento do Algoritmo Busca Tabu encontra-se no Apêndice 6.

Os critérios de parada mais comuns à Busca Tabu são:

- **Número de iterações sem melhora:** o processo pára após certo número de iterações sem melhora;
- **Tempo de processamento:** o processo pára após um tempo de processamento pré-determinado;
- **Limite a atingir:** o processo pára quando um limite pré-definido é alcançado

Brazil, Leite e Mendes (2006) ilustram o pseudocódigo do algoritmo da Busca Tabu da seguinte maneira:

```
So ← Solução Inicial
S  ← So
S* ← So
listaTabu ← { }
Enquanto critério de parada não satisfeito faça
   encontrar melhor solução S' ∈ viz(S) e ∉ listaTabu
   se custo (S') < custo (S*) então
      S* ← S'
   senão
      se custo (S') > custo (S) então
         listaTabu ← listaTabu U {S}
      fim se
   fim se
   S ← S'
fim enquanto
fim BuscaTabu
```

ALGORITMO 14 - Algoritmo Busca Tabu

5. ESTUDO DE CASO

Para esse trabalho uma distribuidora de produtos situada em Curitiba - PR foi escolhida como estudo de caso. Para proteger a fonte e seus clientes, não se fará referências a ela e tampouco será relatada a sua localização e área de atuação.

O objetivo da utilização deste tipo de empresa será a análise de suas rotas de visitação e distribuição para, se possível, otimizá-la.

5.1 CARACTERÍSTICAS DO PROBLEMA

Deseja-se resolver o problema de roteirização no trabalho de um representante comercial ao visitar seus clientes e também na entrega dos produtos.

Existe uma distribuidora de produtos e vários pontos de visita/entrega, que são os clientes. Estes são divididos por setores, os quais são distribuídos aos representantes. Cada representante recebe uma planilha de visitação, na qual constam os clientes que devem ser visitados em determinados dias da semana. Nessas planilhas não são consideradas janelas de tempo, isto é, os clientes podem ser visitados a qualquer hora do dia, sem restrições. Os representantes trabalham com veículos de propriedade da empresa.

Para este estudo utilizou-se a rota de um determinado representante e considerou-se apenas a primeira semana, pois seus clientes são distribuídos em duas semanas.

O presente trabalho consiste em avaliar os roteiros de visita do representante aos seus clientes, sugerindo rotas alternativas obtidas através dos métodos citados, visando a minimização das distâncias totais percorridas.

5.1.1 Coleta de dados

Utilizou-se como base de comparação o roteiro atual de visitação, onde nele encontram-se todas as informações como ordem de visitação e endereços. Estas informações são fundamentais para a montagem do cenário, pois nele estão os "pontos" a serem visitados. Contudo, para que esses dados possam ser utilizados, se faz necessária uma codificação dos mesmos, pois somente os endereços dos clientes não são parâmetros suficientes para se poder efetuar a otimização, são necessárias também suas distâncias relativas a todos os outros pontos.

5.1.2 Mapeamento dos dados

De posse da rota dos representantes com todos os endereços e ordem de visitação, o mapeamento se deu em duas etapas. Na primeira etapa todos os endereços foram plotados no aplicativo *online* "*Google Earth*", no qual foram obtidas as coordenadas geográficas dos pontos de visitação. A utilização dessas coordenadas nos fornece um maior grau de precisão e com isso resultados mais significativos para a pesquisa.

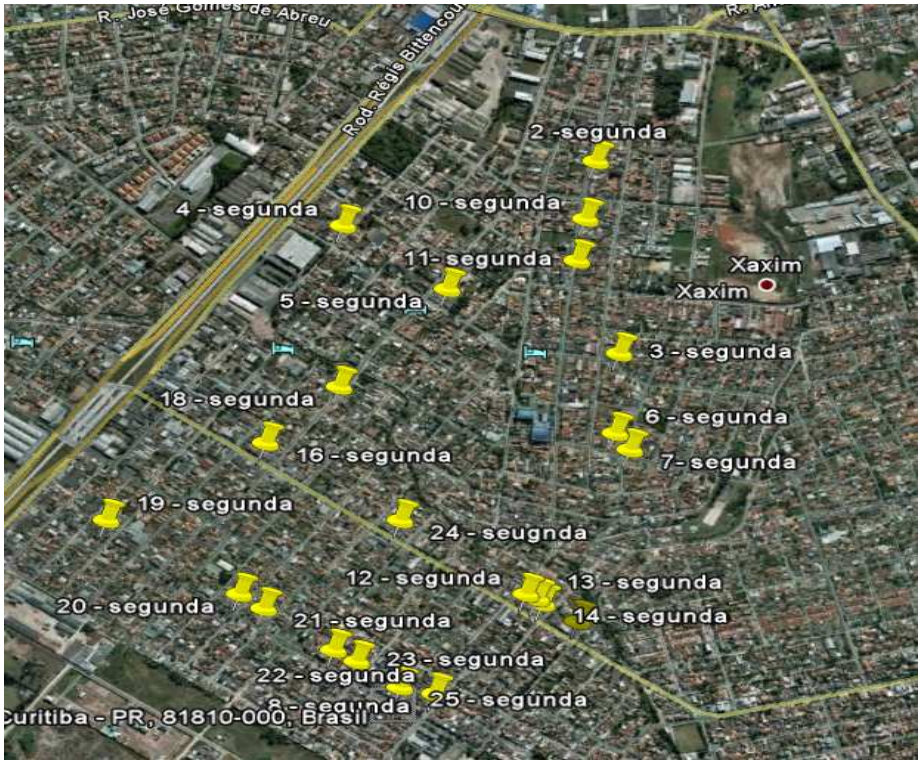


FIGURA 5 - MAPEAMENTO GEOGRÁFICO

fonte (http://maps.google.com, 2011)

De posse de todos os pontos mapeados geograficamente, parte-se para a segunda etapa, que é a conversão destes pontos em um plano cartesiano. Para que isso seja feito, deve-se converter as coordenadas geográficas que são dadas em latitudes e longitudes, referência para localização global, em coordenadas cartesianas, que se referem às distâncias métricas a partir de determinada referência.

Para fazer esta conversão foi utilizada uma planilha no Excel, no qual insere-se a latitude e longitude (transformadas em latitude decimal e longitude decimal),

para que possam ser transformadas em coordenadas cartesianas X e Y. As fórmulas (8) e (9) para essa conversão são:

$$X = (\text{Raio da Terra}) * \text{sen}\left(\frac{(90 + \text{latitude decimal}) * \pi}{180}\right) * cos\left(\frac{(360 - \text{longitude decimal}) * \pi}{180}\right) \quad (7)$$

$$Y = (\text{Raio da Terra}) * \text{sen}\left(\frac{(90 + \text{latitude decimal}) * \pi}{180}\right) * \text{sen}\left(\frac{(360 - \text{longitude decimal}) * \pi}{180}\right) \quad (8)$$

Com todas as coordenadas codificadas, foi possível a determinação das distâncias entre todos os pontos de visitação.

5.1.3 Distâncias

Utilizou-se as distâncias euclidianas, obtidas pela raiz quadrada da soma dos quadrados das diferenças de valores para cada coordenada cartesiana, sendo dada pela fórmula (9):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (9)$$

Onde $P(x_1, y_1)$ e $Q(x_2, y_2)$.

Para o propósito da análise e validação dos dados neste estudo de caso, as distâncias serão consideradas geograficamente sobre uma superfície plana.

5.1.4 Veículos

Os veículos disponíveis para a locomoção dos representantes constituem uma frota homogênea, mas utilizou-se para o estudo apenas um veículo e a rota de um representante.

5.1.5 Rota Atual

Após a coleta dos dados e todo refinamento construiu-se a rota atual praticada pelo representante em questão para as suas visitas. Esta rota é composta de 128 (cento e vinte oito) pontos distribuídos em cinco dias da semana conforme a Tabela 1:

TABELA 1 - DISTRIBUIÇÃO DE CLIENTES POR DIA DA SEMANA

Dia da Semana	Segunda	Terça	Quarta	Quinta	Sexta
Lugares a serem visitados	24	26	27	19	32
Distância Euclidiana Total (em metros)	15.808,84	25.198,19	25.185,89	36.156,98	20.691,49

FONTE: O autor (2011)

O objetivo deste trabalho é minimizar as distâncias totais percorridas e, para que isto aconteça, os pontos relativos às visitas do referido representante para a aplicação dos algoritmos propostos foram utilizados.

5.2 IMPLEMENTAÇÃO COMPUTACIONAL

Conforme já relatado, este trabalho busca a elaboração de uma análise experimental de abordagens heurísticas e metaheurísticas aplicadas ao Problema do Caixeiro Viajante Simétrico.

Neste seção serão apresentados os experimentos realizados. As heurísticas descritas no capítulo 3, foram implementadas na linguagem Visual Basic 6.0 e os testes foram realizados em um notebook com sistema operacional Windows XP, processador Intel(R) Core(TM) 2 Duo T5550, 1,83 GHz e 2,99 GB de RAM. Para a implementação das metaheurísticas *Ant System*, *Simulated Annealing* e Algoritmo Genético, descritas no capítulo 4 utilizou-se o software MATLAB 5.3 e o mesmo computador já citado.

O objetivo dos experimentos é encontrar resultados que minimizem o custo das rotas obtidas quando comparadas com a solução inicial do problema.

A redução dos custos das rotas já validam as metodologias propostas. No entanto, para que se possa comparar os custos das rotas atuais, bem como das soluções obtidas, em relação às soluções ótimas, utilizou-se um software baseado em algoritmos exatos conhecido como LINGO 12.0. A justificativa da escolha desse software se dá pela razão de permitir comparar um método exato com as heurísticas e metaheurísticas utilizadas, sendo útil para validar o presente estudo. Os valores gerados pelo LINGO encontram-se no APÊNDICE 11 e nas Tabelas 2 e 4.

5.2.1 Software LINGO

De acordo com Junior e Souza (2004) o LINGO é um software de modelagem e resolução de problemas lineares e não lineares de otimização.

Para este estudo utilizou-se a versão do LINGO 12.0, cujo aplicativo faz uso do algoritmo de *Branch and Bound* para a resolução de problemas de Programação Linear Inteira. É um algoritmo enumerativo, cuja estrutura de resolução baseia-se na construção de uma árvore onde os nós representam os problemas candidatos e os ramos representam as novas restrições que devem ser consideradas.

5.2.2 Heurísticas de Construção e Melhoria de Rotas

As interfaces gráficas obtidas do aplicativo desenvolvido, para resolver problemas com as heurísticas de construção de rotas bem como as melhorias 2-opt podem ser vistas nas Figuras 6, 7, 8, 9 E 10.

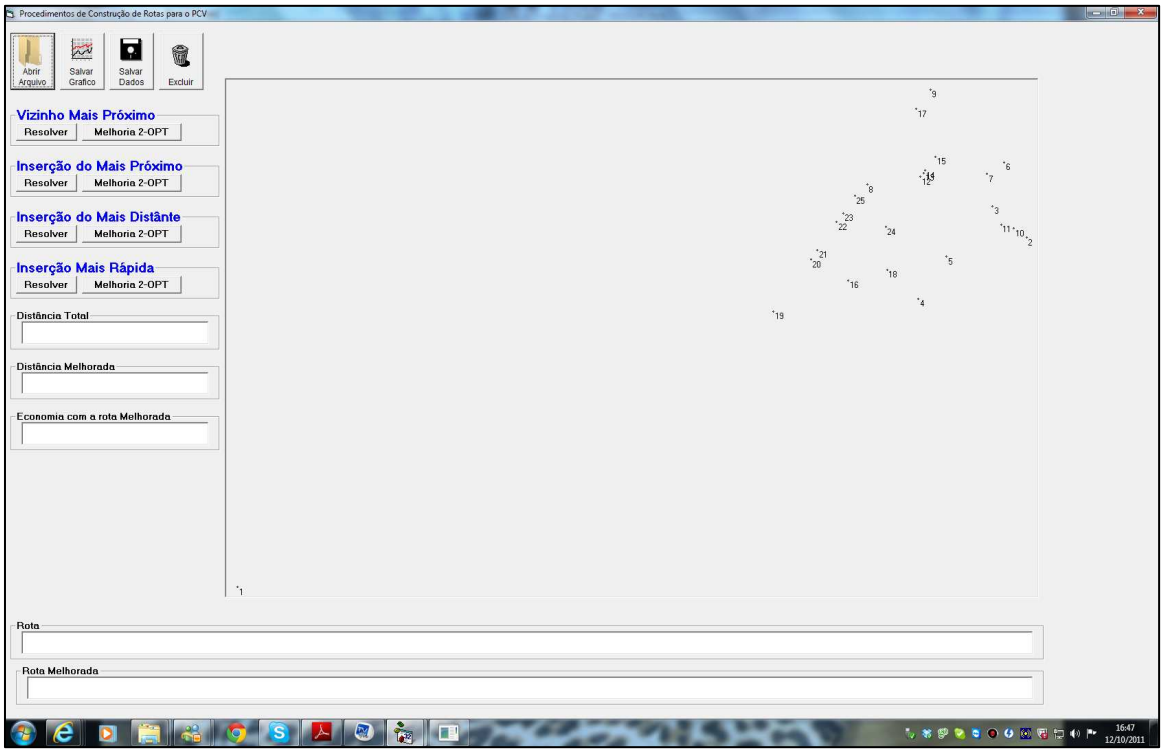


FIGURA 6 - TELA DO PROGRAMA COM OS PONTOS DE UMA DAS ROTAS

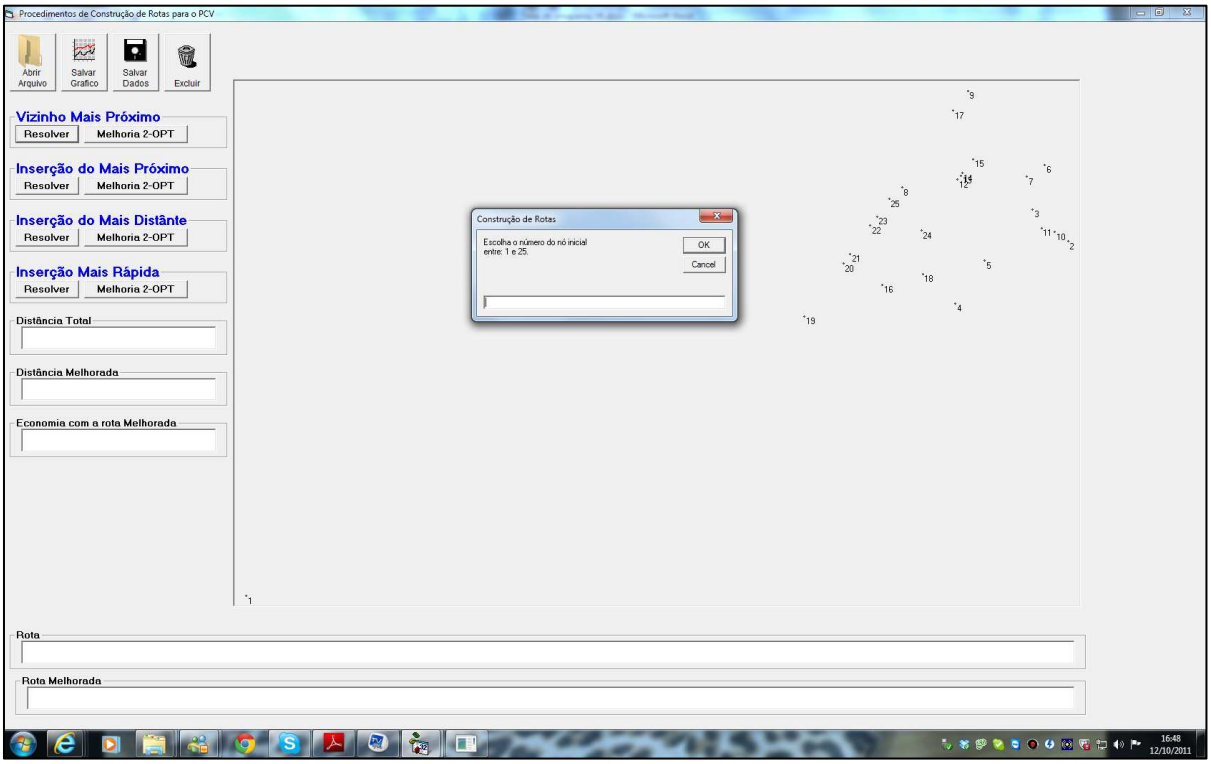


FIGURA 7 - TELA DO PROGRAMA APÓS ESCOLHIDA A HEURÍSTICA A SER IMPLEMENTADA

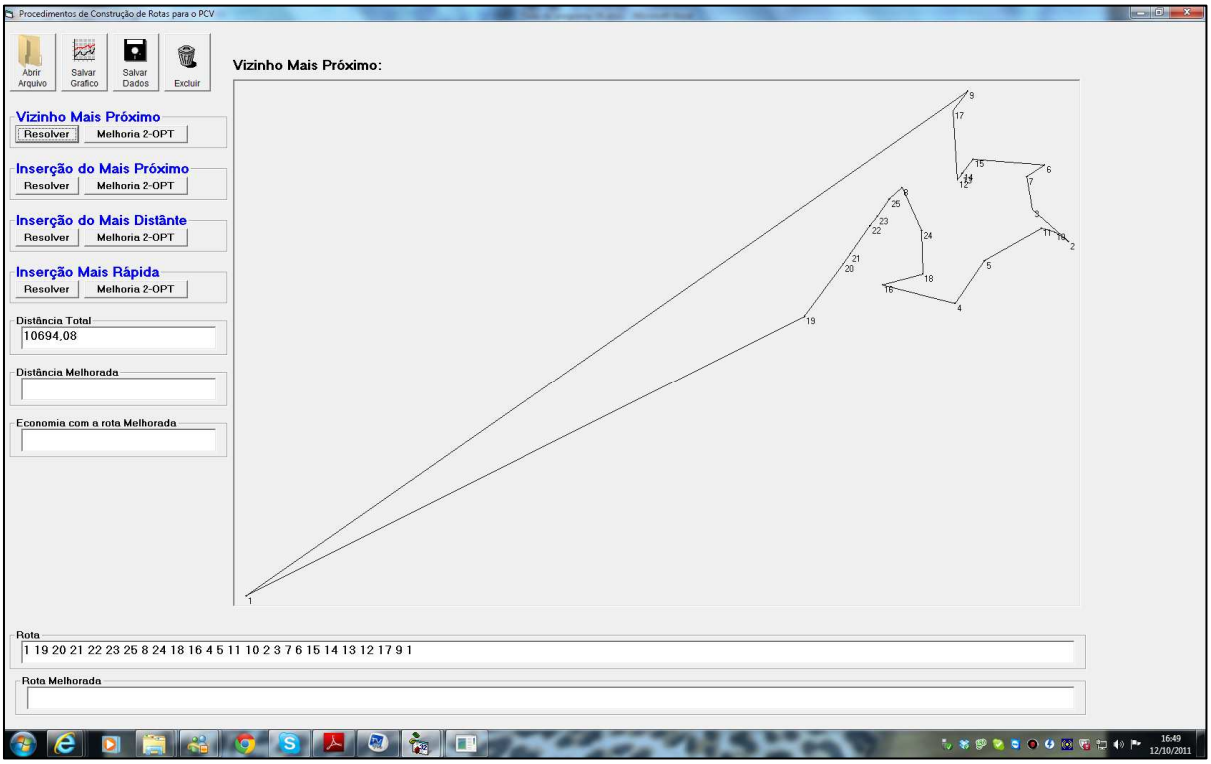


FIGURA 8 - TELA DO PROGRAMA APÓS SELECIONADO O PONTO DE INICIO DA ROTA

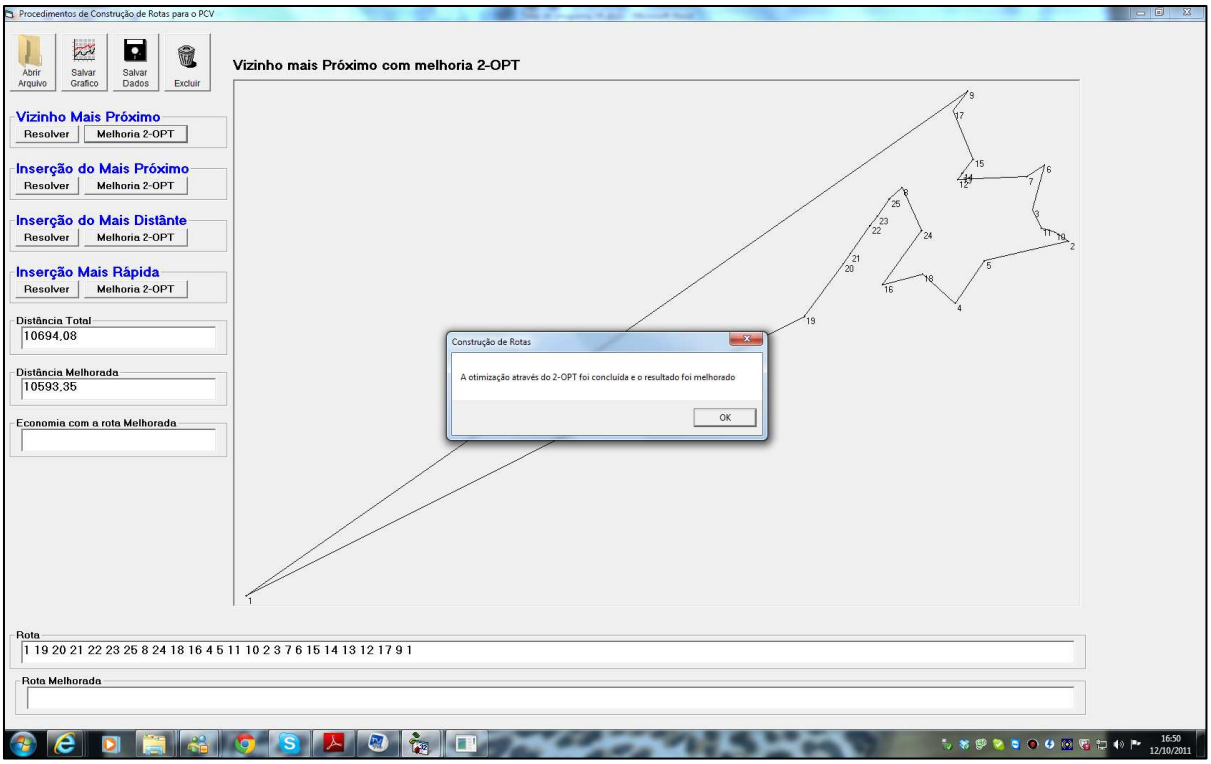


FIGURA 9 - TELA DO PROGRAMA AO SELECIONAR O RECURSO DE MELHORIA 2-OPT

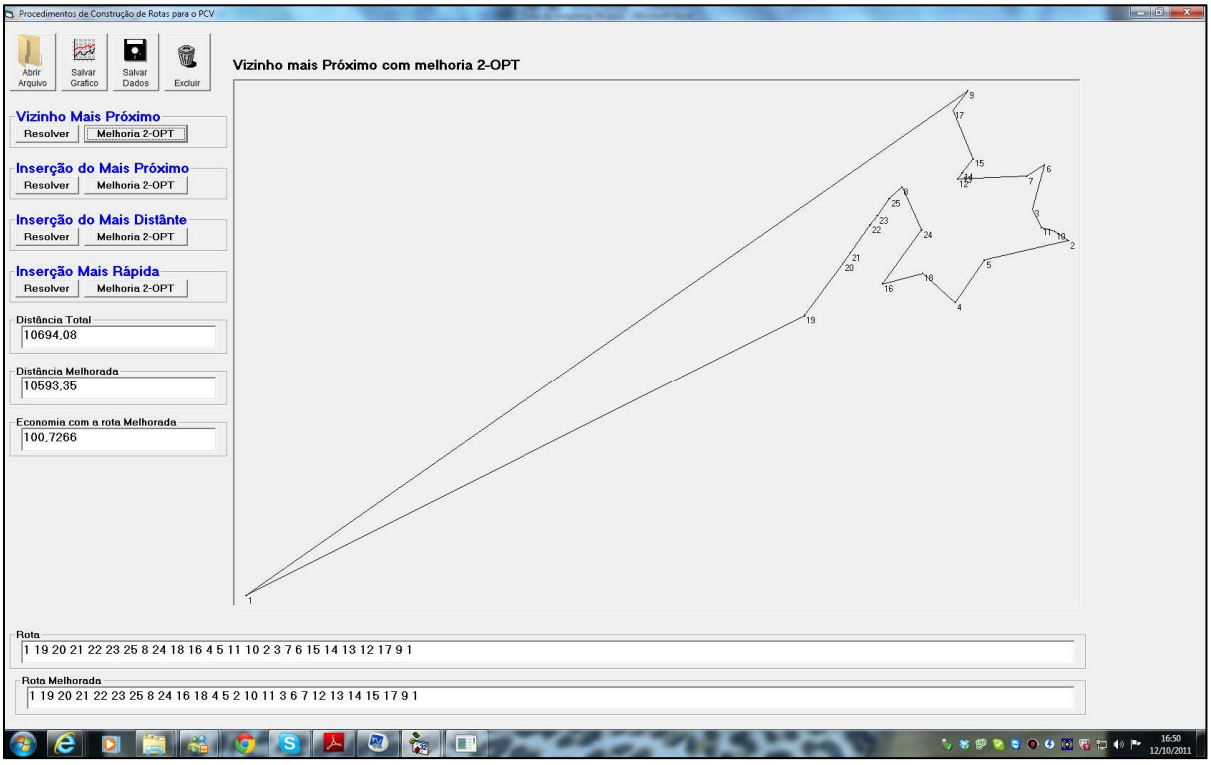


FIGURA 10 - TELA DO PROGRAMA COM A ROTA APÓS O 2-OPT

O detalhamento dos resultados e figuras após todas as implementações das heurísticas de construção e melhoria de rotas encontra-se no APÊNDICE 7.

5.2.3 *Ant System*

Para a implementação da metaheurística *Ant System*, usou-se os seguintes parâmetros:

- Número máximo de formigas: M = 10000;
- Expoente do fator de feromônio: $\alpha = 0{,}5$;
- Expoente de fator de visibilidade: $\beta = 0{,}5$;
- Taxa de feromônio: $\tau = 1{,}0$;
- Taxa de evaporação do feromônio: $\rho = 1$;
- Constante positiva usada para o acréscimo de feromônio: Q= 0,001;

O detalhamento dos resultados e figuras da metaheuristica *Ant System*, juntamente com a melhoria 2-opt encontra-se no APÊNDICE 8.

5.2.4 *Simulated Annealing*

Para a implementação da metaheurística *Simulated Annealing* utilizou-se os seguintes parâmetros:

- Número máximo de iterações do algoritmo: M = 1000;
- Número de perturbações por iteração: P =1000;
- Fator de redução de temperatura: alfa = 0.95;
- Valor empírico entre 0 e 1: $\xi = 0{,}03$

O detalhamento dos resultados e figuras da metaheuristica *Simulated Annealing,* juntamente com a melhoria 2-opt encontra-se no APÊNDICE 9

5.2.5 Algoritmo Genético

Para a implementação do Algoritmo Genético usou-se os seguintes parâmetros:

- Número de cromossomos: m = 50;
- Número máximo de iterações: itermax = 10000;
- Diferença máxima entre o melhor e pior *fitness*: $\varepsilon = 0{,}00001$;
- Probabilidade de mutação: $\theta = 0{,}01$;

- Para a representação dos cromossomos utilizou-se a representação por caminho.
- No processo de seleção foi utilizada a expressão:

$$\text{Select}(R) = \left\{ r_j \in R / j = m + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.\text{Rnd}(m^2 + m)}}{2} \right\rceil \right\}$$

cujos parâmetros estão detalhados na seção 4.4.4.

- O operador de cruzamento utilizado foi o HX (*Heuristic Crossover*), pois segundo o estudo comparativo de Silva e Oliveira (2006), este operador destaca-se sobre os demais operadores, sobretudo quando o número de nós é grande.
- A mutação é feita por troca *(Swap),*que consiste na troca aleatória de posição entre dois genes.

O detalhamento dos resultados, figuras e gráficos do Algoritmo Genético, juntamente com a melhoria 2-opt encontram-se no APÊNDICE 10

5.2.6 RESULTADOS

Na Tabela 2 tem-se os dados das rotas atuais praticadas pelo representante, os valores ótimos que foram obtidos pelo Lingo 12.0 e os resultados computacionais obtidos com as Heurísticas de Construção de Rotas, Melhorias de Rotas e pelas Metaheurísticas *Ant System, Simulated Annealing* e Algoritmo Genético.

Os dados que constam na tabela são as distâncias euclidianas, em metros, das rotas geradas para os pontos a serem visitados. Nesta tabela destacou-se em vermelho os melhores valores obtidos em cada rota.

TABELA 2 - RESULTADOS GERADOS (EM METROS)

Dias da semana	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**
	25 pontos	26 pontos	28 pontos	20 pontos	33 pontos
VALOR ÓTIMO	9811,77	16451,17	18598,65	21309,23	18206,61
Rota atual	15808,84	25198,19	25185,87	36156,98	20691,49
Vizinho mais próximo	10694,08	16641,36	19173,03	23779,66	20137,86
Vizinho mais próximo com 2-opt	10593,35	**16451,17**	18940,70	23320,39	19793,07
Inserção do mais próximo	10483,08	17199,55	19362,35	23506,59	19294,12
Inserção do mais próximo com 2-opt	9910,51	16986,96	19274,71	23506,59	18542,89
Inserção do mais distante	10054,84	17331,30	18789,62	**21309,23**	18344,68
Inserção do mais distante com 2-opt	9879,76	17322,14	**18598,65**	**21309,23**	**18301,71**
Inserção do mais rápido	11438,71	17532,03	20656,51	23733,36	19857,29
Inserção do mais rápido com 2-opt	10979,42	17367,07	19478,81	23501,20	18512,03
Algoritmo Genético	9842,52	**16451,17**	18728,42	**21309,23**	18314,63
Algoritmo Genético com 2-opt	**9811,77**	**16451,17**	18697,54	**21309,23**	18314,63
S.A.	9879,76	16600,13	18691,17	22348,60	18808,38
S.A com 2-opt	9879,76	16600,13	18691,17	22348,60	18328,65
Ant System	10444,86	19897,31	20859,11	23510,10	20798,98
Ant System com 2-opt	9879,76	16861,14	18653,77	**21309,23**	18448,51

FONTE: o autor (2011)

Na tabela 3, tem-se a porcentagem de melhora em relação a rota atual dos dados gerados pelas heurísticas e metaheuristicas. Nela tem-se uma visão mais objetiva de quanto a técnica aplicada pode reduzir a distância percorrida pelo representante. Novamente foram destacadas em vermelho os casos com maiores reduções das distâncias.

TABELA 3 - PERCENTUAL DE MELHORA EM RELAÇÃO A ROTA ATUAL

Dias da semana	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**
	25 pontos	26 pontos	28 pontos	20 pontos	33 pontos
Vizinho mais próximo	32,35%	33,96%	23,87%	34,23%	2,68%
Vizinho mais próximo com 2-opt	32,99%	**34,71%**	24,80%	35,50%	4,34%
Inserção do mais próximo	33,69%	31,74%	23,12%	34,99%	6,75%
Inserção do mais próximo com 2-opt	37,31%	32,59%	23,47%	34,99%	10,38%
Inserção do mais distante	36,40%	31,22%	25,40%	**41,06%**	11,34%
Inserção do mais distante com 2-opt	37,50%	31,26%	**26,15%**	**41,06%**	**11,55%**
Inserção do mais rápido	27,64%	30,42%	17,98%	34,36%	4,03%
Inserção do mais rápido com 2-opt	30,55%	31,08%	22,66%	35,00%	10,53%
Algoritmo Genético	37,74%	**34,71%**	25,64%	**41,06%**	11,49%
Algoritmo Genético com 2-opt	**37,93%**	**34,71%**	25,76%	**41,06%**	11,49%
S.A.	37,50%	34,12%	25,79%	38,19%	9,10%
S.A com 2-opt	37,50%	34,12%	25,79%	38,19%	11,42%
Ant System	33,93%	21,04%	17,18%	34,98%	-0,52%
Ant System com 2-opt	37,50%	33,09%	25,94%	**41,06%**	10,84%

FONTE: o autor (2011)

Na tabela 4, apresenta-se a porcentagem em que cada heurística e metaheurística implementada, bem como a rota praticada atualmente pelo representante encontra-se próximo do valor ótimo para a rota, gerado pelo LINGO 12.0. Observa-se em alguns casos os procedimentos heurísticos ou metaheurísticos atingiram o valor ótimo.

TABELA 4 - PERCENTUAL EM RELAÇÃO AO VALOR ÓTIMO

Dias da semana	**Segunda**	**Terça**	**Quarta**	**Quinta**	**Sexta**
	25 pontos	26 pontos	28 pontos	20 pontos	33 pontos
Rota atual	62,07%	65,29%	73,85%	58,94%	87,99%
Vizinho mais próximo	91,75%	98,86%	97,00%	89,61%	90,41%
Vizinho mais próximo com 2-opt	92,62%	100,00%	98,19%	91,38%	91,98%
Inserção do mais próximo	93,60%	95,65%	96,06%	90,65%	94,36%
Inserção do mais próximo com 2-opt	99,00%	96,85%	96,49%	90,65%	98,19%
Inserção do mais distante	97,58%	94,92%	98,98%	**100,00%**	99,25%
Inserção do mais distante com 2-opt	99,31%	94,97%	**100,00%**	**100,00%**	99,48%
Inserção do mais rápido	85,78%	93,83%	90,04%	89,79%	91,69%
Inserção do mais rápido com 2-opt	89,37%	94,73%	95,48%	90,67%	98,35%
Algoritmo Genético	99,69%	**100,00%**	99,31%	**100,00%**	99,41%
Algoritmo Genético com 2-opt	**100,00%**	**100,00%**	99,47%	**100,00%**	99,41%
S.A.	99,31%	99,10%	99,51%	95,35%	96,80%
S.A com 2-opt	99,31%	99,10%	99,51%	95,35%	99,33%
Ant System	93,94%	82,68%	89,16%	90,64%	87,54%
Ant System com 2-opt	99,31%	97,57%	99,70%	**100,00%**	98,69%

FONTE: o autor (2011)

Na tabela 5, apresenta-se o tempo computacional de cada Heurística e Metaheurística implementada.

TABELA 5 - TEMPO COMPUTACIONAL

Dias da semana	**Segunda** 25 pontos	**Terça** 26 pontos	**Quarta** 28 pontos	**Quinta** 20 pontos	**Sexta** 33 pontos
Rota atual (Lingo 12.0)	00:00:13	01:33:05	00:02:55	00:00:02	00:00:56
Vizinho mais próximo	00:00:13	00:00:02	00:00:01	00:00:01	00:00:02
Vizinho mais próximo com 2-opt	00:00:14	00:00:03	00:00:03	00:00:02	00:00:03
Inserção do mais próximo	00:00:03	00:00:02	00:00:01	00:00:02	00:00:02
Inserção do mais próximo com 2-opt	00:00:04	00:00:03	00:00:04	00:00:03	00:00:03
Inserção do mais distante	00:00:07	00:00:02	00:00:02	00:00:02	00:00:01
Inserção do mais distante com 2-opt	00:00:09	00:00:03	00:00:03	00:00:03	00:00:03
Inserção do mais rápido	00:00:04	00:00:02	00:00:02	00:00:01	00:00:01
Inserção do mais rápido com 2-opt	00:00:05	00:00:03	00:00:03	00:00:02	00:00:02
Algoritmo Genético	00:00:03	00:00:02	00:00:02	00:00:02	00:00:07
Algoritmo Genético com 2-opt	00:00:05	00:00:04	00:00:04	00:00:04	00:00:09
S.A.	00:00:22	00:00:23	00:00:24	00:00:22	00:00:24
S.A com 2-opt	00:00:24	00:00:25	00:00:26	00:00:24	00:00:26
Ant System	00:00:10	00:00:10	00:00:12	00:00:07	00:00:16
Ant System com 2-opt	00:00:11	00:00:12	00:00:14	00:00:09	00:00:18

6. CONCLUSÕES

A empresa que forneceu os dados para este trabalho possui uma carteira de clientes agrupados por setores, onde cada um desses setores é atendido diariamente por um representante. Os próprios representantes são responsáveis pela construção de suas rotas de visita, assim como, a rota de entrega dos produtos. Estes levam em em conta apenas o conhecimento prévio da localização e experiência adquirida no atendimento diário dos clientes do setor, sem critérios de otimização matemático.

Assim é proposta a utilização de programas computacionais para contruir automaticamente as rotas de visitação, para com isso gerar uma economia de recursos e tempo para a empresa e os representantes. Vale acrescentar que os veículos utilizados são de propriedade da distribuidora.

O objetivo deste trabalho foi o estudo e aplicação das Heurísticas de Construção e Melhoria de Rota e das Metaheurísticas *Ant System, Simulated Annealing* e Algoritmos Genéticos para a construção de rotas do representante da distribuidora de produtos, comparando os resultados obtidos com as rotas praticadas atualmente pelo representante.

A aplicação dos referidos algoritmos mostra como é possível a minimização de custos, pois há uma redução significativa na quilometragem total percorrida.

Os resultados computacionais demonstram que em todos os dias da semana houve melhoria comparando-se com a rota já praticada pelo representante.

Em relação a comparação de desempenho entre as Heurísticas, a Inserção do Mais Distante com melhoria 2-opt foi a que apresentou melhores resultados juntamente com a metaheurística Algoritmo Genético com 2-opt. Em todas as heurísticas e metaheurísticas implementadas o tempo computacional foi baixo.

O uso dos Algoritmos Genéticos na busca de uma solução para o PCV permite encontrar rotas que, mesmo não tendo garantia de oferecer a melhor solução, são de boa qualidade. Considerando os resultados obtidos e levando em conta a dificuldade de se resolver o PCV para um número elevado de pontos, percebe-se uma vantagem no uso dos Algoritmos Genéticos na busca de uma solução de boa qualidade para este problema específico. Na maioria dos casos o desempenho do Algoritmo Genético foi melhor, e quando isso não ocorreu, a diferença não foi tão significativa em comparação com o *Simulated Annealling*.

Com estes resultados, concluí-se que a aplicação destas metaheurísticas é viável para melhorar os roteiros já utilizados pela empresa. Os tempos computacionais são reduzidos, o que permite rápidas atualizações dos roteiros quando são alterados os pontos dos mesmos. Ainda, em quase todos os casos, a solução ótima foi atingida ao comparar o melhor resultado encontrado com os resultados gerados pelo LINGO 12.0. Quando isso não ocorreu, o valor atingido pelas metaheuristicas foi muito próximo do mesmo.

7. SUGESTÕES PARA TRABALHOS FUTUROS

Objetivando um aprimoramento dos resultados obtidos, são dadas algumas sugestões para trabalhos futuros:

a) Utilizar as distâncias reais, considerando os sentidos das ruas;

b) Utilizar janelas de tempo, se o representante necessita visitar o cliente em determinados horários do dia;

c) Testar outros parâmetros para as metaheuristicas, utilizando Algoritmos Genéticos para sua calibração;

d) Implementar outras metaheuristicas, tais como GRASP, Busca Tabu, Nuvem de Partículas, entre outras.

REFERÊNCIAS

ARAGON, C.R.; JOHNSON, D.S.; McGEOCH L.A.; SCHEVON C., Optimization by simulated annealing: an experimental evaluation; Workshop on Statistical Physics in Engineering and Biology, 1984.

BARBOSA, V.C.; Redes Neurais e Simulated Annealing como Ferramentas para Otimização Combinatória; Investigación Operativa, Vol.1, N.2, 1989.

BARBOSA, H.J.C.; Introdução aos Algoritmos Genéticos; Mini Curso – XX CNMAC, Gramado(RS), 1997.

BARBOZA, A.O.; Simulação e Técnicas da Computação Evolucionária Aplicadas a Problemas de Programação Linear Inteira Mista. Tese de Doutorado, UTFPR, 2005.

BEZERRA, O.B.; Localização de postos de coleta para apoio ao escoamento de produtos extrativistas - Um estudo de caso aplicado ao babaçu. Dissertação de Mestrado, UFSC, 1995

BODIN, L.; GOLDEN, B.; ASSAD, A.; BALL, M.; Routing and Scheduling of Vehicles and Crews – The State of the Art; Computers Ops Res; Vol. 10, Number 2; Pergamon Press, 1983.

BRAZIL, J.C.; LEITE, M.S.; MENDES, R.B.S.; Uma Metaheurística Grasp Busca Tabu para o Problema do Caixeiro Viajante. II Workshop de Computação Científica da UENF - IIWCC, 2006

CHAVES, A.A.; Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios. Universidade Federal de Ouro Preto Instituto de Ciências Exatas e Biológicas Departamento de Computação. Ouro Preto, MG. 2003.

COSTA, D.M.B., Aplicação de Algumas Técnicas da Pesquisa Operacional na Otimização dos Serviços Postais; Dissertação de Mestrado, UFPR, 1997

COLORNI, A.; DORIGO, M.; MANIEZZO, V.; Distributed Optmization by Ant Colonies; Proceedings of ECAL-91-European Conference on Artificial Life, Paris, France; F.Varella & P.Bourgine, pp.134-142, 1991.

COLORNI, A .; DORIGO, M.; MANIEZZO, V.; An Investigation of some properties of an ant algorithm; Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, Belgium, R.Männer, B Manderick. p.509-520, 1992.

COLORNI, A.; DORIGO, M.; MAFFIOLI, F.; MANIEZZO, V.; RIGHINI, G.; TRUBIAN, M.; Heuristics from Nature for Hard Combinatorial Optimization Problems; International Transactions in Operational Research, Vol.3, N.1, Pergamon, 1996.

DORIGO,M.; GAMBARDELLA, L. M.; Ant colonies for the traveling salesman problem. Belgium; Université Libre de Bruxelles, 1996

DORIGO, M.; MANIEZZO, V.; COLORNI, A.; Ant System: Optimization by a colony of cooperating agents. IEEE Trans. On Systems, Man and Cybernetics-Part B: Cybernetics, 26, v.1, p.29-41, 1996.

FEO, T.A.; RESENDE, M.G.C; A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8: pp 67-71, 1989.

FEO, T.A.; RESENDE, M.G.C; Greedy randomized adaptive search procedures. Journal of Global Optimization 6, pp 109-133, 1995

FESTA, P.; RESENDE M.G.C.; GRASP: An Annotated Bibliography. Em RIBEIRO, C.C. e HASEN P., editores, Essays and Surveys on Metaheuristics, Kluwer Academic Publishers, 2002.

FRAGA, M.C.P.;Uma metodologia híbrida Colônia de Formigas - Busca Tabu Reconexão por Caminhos para resolução do Problema de Roteamento de Veículos com Janelas de Tempo; Dissertação de Mestrado, CEFET-MG, 2006.

FREDDO, A.R.; BRITO, R.C.; Implementação da Metaheuristica GRASP para o problema do Caixeiro Viajante Simetrico. 2008. Disponivel em: <http://www.inf.ufpr.br/aurora/disciplinas/topicosia2/downloads/trabalhos/GraspTSP.pdf>. Acesso em 02/10/2011.

GANHOTO, M.A. Abordagens para problemas de roteamento; Dissertação de Mestrado, Universidade Estadual de Campinas, 2004.

GLOVER, F., KOCHENBERGER, G.A. Handbook of Metaheuristics. Kluwer Academic Publishers. Boston, 2003.

GOLDBERG, D.E.; Genetic Algorithms in Search, Optimization e Machine Learning; Addison-Wesley Publishing Company, Inc., 1989
GOLDBARG, M. C.; LUNA, H. P. L. Otimização Combinatória e Programação Linear – Modelos e Algoritmos. Editora Campus. Rio de Janeiro, 2005.

GREFENSTETTE, J.J.; Optimization of control parameters for genetic algorithms; IEEE Transactions on Systems, Man and Cybernetics, v.16, n.1, p.122-128, 1986.

JUNIOR, A. de C. G; SOUZA, M.J.F; Lingo - Parte 1: Manual de Referência. Departamento de Computação. Universidade Federal de Ouro Preto. Rio de Janeiro, 2004. Disponivel em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/OtimizacaoCombinatoria/lingo_p.pdf>. Acesso em 27/10/2011.

HOLLAND, J.H.; Genetic algorithms. Scientific American, v. 267, n.1, p.44-50, 1992.

KIRKPATRICK, S.; GELATT Jr, C.D.; VECCHI, M.P.; Optimization by Simulated Annealing; Science, Vol 220, N.4598, 1983.

LIN, S.; KERNIGHAN, B.W.; An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research, Vol.21, N.2, p 498 – 516, 1973. Disponível em: <http://www.ie.metu.edu.tr/~ie505/CourseMaterial/Lin%20and%20Kernighan%20(1973).pdf>. Acesso em 28/11/2011.

LOPES, H.S.; Algoritmos Genéticos; Trabalho de circulação interna do CEFET - Centro Federal de Educação Tecnológica do Paraná, 1996.

MALAQUIAS, N.G.L.; Uso dos Algoritmos Genéticos para a Otimização de Rotas de Distribuição; Dissertação de Mestrado. Universidade Federal de Uberlândia, 2006

MAYERLE, S.F.; Um Algoritmo Genético para o Problema do Caixeiro Viajante. Artigo de Circulação Interna do Departamento de Engenharia de Produção e Sistemas da UFSC, 1994.

METAHEURISTICS NETWORK. Metaheuristic. Disponível em: <http://www.metaheuristics.net/index.php?main=1&sub=11>. Acesso em 14 de março de 2011

MITRA, D.; ROMEO, F.; SANGIOVANNI-VINCENTELLI, A.; Convergence and finite-time behavior of simulated annealing; Advances in Applied Probability, N.18, pp.747-771, 1986.

MOURA, A.M.P.; Abordagens Heurísticas para o Planejamento de Rotas e Carregamento de Veículos. Tese de Doutorado. Faculdade de Engenharia da Universidade do Porto. Porto, 2005.

NUNES, L.F.; Algoritmos Genéticos Aplicados na Abordagem de um PRoblema Real de Roterização de Veiículos. Dissertação de Mestrado. UFPR, 1998.

POTVIN, J. Y. Genetic algorithms for the traveling salesman problem. Annals of Operations Research 6, p.339 - 370, 1996. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=F65927180919D4CDA044AFF23E372615?doi=10.1.1.93.2179&rep=rep1&type=pdf>. Acesso em 03/05/2011.

PRESTES, A.N.; Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante. Dissertação de Mestrado. UFRGN, 2006.

PUREZA, V.M.M.; Problemas de Roteamento de Veículos Via Metaheuristica Tabu. Dissertação de Mestrado. Universidade Estadual de Campinas, 1990

ROCHA, M.L.; OCHI, L.S. Uma meta-heurística baseada em algoritmos genéticos não convencionais para o problema de roteamento de veículos multi-depósitos;. Resumes Extendidos de Primeira ELIO - Optima 97 - Conception, Chile, Ed. Dr.ª Lorena Pravenas, 1997.

SCHOPF, E.C.; SCHEPKE, C.; SILVA, M.L.; SILVA, P.F.; Avaliação de Heurísticas de Melhoramento e da Metaheuristica Busca Tabu para Soluções de PRV. Santo

Ângelo: VII Fórum de Tecnologias e XIV Simpósio Regional de Informática, 2004. Disponivel em: http://www.inf.ufrgs.br/~cschepke/graduacao/AvaliacaoDeHeuristicasDeMelhoramentoETabu.pdf. Acesso em 09/10/2011.

SILVA, A. F., OLIVEIRA, A.C. Algoritmos genéticos: alguns experimentos com os operadores de cruzamento (“Crossover”) para o problema do caixeiro viajante assimétrico. XXVI ENEGEP – Fortaleza, 2006. Disponível em: http://www.abepro.org.br/biblioteca/ENEGEP2006_TR460314_7093.pdf. Acesso em 13/04/2011.

SILVA, E.O.A.; BAHIENSE, C.B.L.; CASTRO,M.C.S.; Uma abordagem Paralela Baseada em Colônia de Fromigas para o Problema do Caixeiro Viajante. Caderno do IME: Série Informática: Vol.18, pp18-29, 2005.

ZAMBONI, L.V.S.; Técnicas de Roteirização de Veículos Aplicadas ao Transporte Escolar; Dissertação de Mestrado, UFPR, 1997.

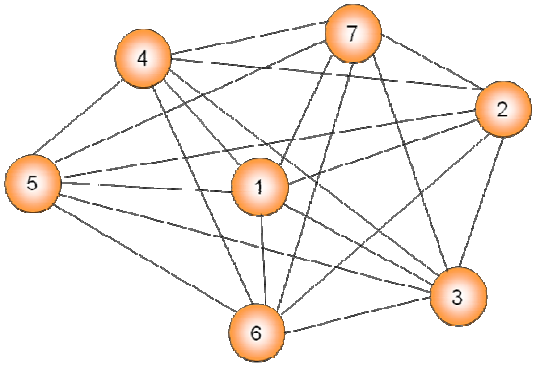
WHITLEY, D.; STARKWEATHER, T.; FUQUAY, D.; Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator; Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, 1989.

APÊNDICES

APÊNDICE 1 - HEURÍSTICAS DE CONSTRUÇÃO DE ROTA

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

Seja:

L: Comprimento da rota

H: Seqüência de nós

VIZINHO MAIS PRÓXIMO:

Nesta heurística, parte-se da cidade de origem e adiciona-se a cada passo a cidade *k* ainda não visitada cuja distância a última cidade visitada é a menor possível. O procedimento de construção termina quando todas as cidades forem visitadas, situação na qual é feita a ligação entre a última cidade visitada e a cidade de origem.

Para o exemplo considerado, tem-se:

- Escolha um nó inicial k qualquer
- A cada iteração escolha o nó mais próximo
  - k = 1

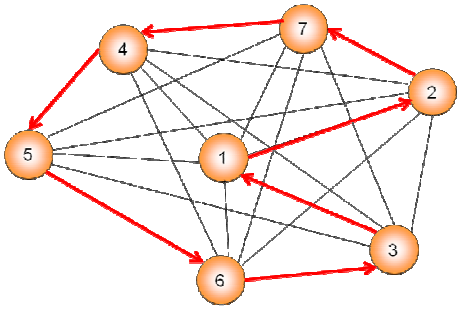
- 1ª Iteração: L = 63,12686
  H = {1, 2}
- 2ª Iteração: L = = 63,12686 + 61,6198 = 124,74666
  H = {1, 2, 7}
- 3ª Iteração: L = 124,74666 + 63,81222 = 188,55888
  H = {1, 2, 7, 4}
- 4ª Iteração: L = 188,55888 + 62,17717 = 250,73605
  H = {1, 2, 7, 4, 5}
- 5ª Iteração: L = 250,73605 + 84,64632 = 335,38237
  H = {1, 2, 7, 4, 5, 6}
- 6ª Iteração: L = 335,38237 + 76,65507 = 412,03744
  H = {1, 2, 7, 4, 5, 6, 3}
- 7ª Iteração: L = 412,03744 + 70,03571 = 482,07315
  H = {1, 2, 7, 4, 5, 6, 3, 1}

Ao final de todos os passos, obtém-se a solução H = {1, 2, 7, 4, 5, 6, 3, 1}. Para esta solução, a distância total percorrida é:

$L = d_{12} + d_{27} + d_{74} + d_{45} + d_{56} + d_{63} + d_{31}$

L = 63,13 + 61,62 + 63,81 + 62,18 + 84,65 + 76,66 + 70,04

L = 482,07



Complexidade da Heurística do Vizinho Mais Próximo:

Iterações	# operações	Observações
1	n - 1	Há n - 1 ligações para serem analisadas
2	n - 2	Há n - 2 ligações para serem analisadas
...	...	...
n - 1	1	Há apenas uma cidade ainda não visitada
Total	1 + 2 + ... + n - 1 = n(n - 1)/2 operações	

A soma anterior é uma Progressão Aritmética cujo primeiro elemento é 1, o último elemento n – 1, a razão é igual a 1 e o número de termos é n – 1. A soma de termos dessa PA é:

$$S = \left(\frac{a_1 + a_{n.elem.}}{2}\right) \cdot n.elem. = \left(\frac{1+(n-1)}{2}\right)(n-1) = \frac{n(n-1)}{2}$$

INSERÇÃO DO MAIS PRÓXIMO:

- Inicie com um sub-grafo contento apenas o nó i
  - i = 1
- Encontre um nó k tal que c_{ik} seja mínima e forme a rota i - k - i
  - i = 1
  - k = 2
  - c_{12} = 63,12686
- Encontre o nó k não pertencente à sub-rota, mais próxima de qualquer nó da sub-rota.
  - Sub-rota inicial:

    H: {1, 2, 1}
  - Nó mais próximo do ciclo:

    k = 7
  - Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

    Insira k entre i e j.

    $c_{17} + c_{72} - c_{12} = c_{27} + c_{71} - c_{21}$

    Insira o nó 7 entre 1 e 2 ou entre 2 e 1

    H: {1, 2, 7, 1}
- Volte ao passo anterior até formar um circuito Hamiltoniano!
  - H: {1, 2, 7, 1}
    - Nó mais próximo do ciclo:

      k = 4
  - Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

    Insira k entre i e j.
    - $c_{14} + c_{42} - c_{12}$ = 70,83784 + 95,7549 - 63,12686 = 103,4659
    - $c_{24} + c_{47} - c_{27}$ = 95,7549 + 63,81222 - 61,6198 = 97,9473
    - $c_{74} + c_{41} - c_{71}$ = 63,81222 + 70,83784 - 87,28115 = 47,3689

Insira o nó 4 entre 7 e 1

H: {1, 2, 7, 4, 1}

- H: {1, 2, 7, 4, 1}
  - Nó mais próximo do ciclo:

$$k = 5$$

- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.
  - $c_{15} + c_{52} - c_{12} = 76{,}02631 + 131{,}244 - 63{,}12686 = 144{,}1434$
  - $c_{25} + c_{57} - c_{27} = 131{,}244 + 121{,}6963 - 61{,}6198 = 191{,}3206$
  - $c_{75} + c_{54} - c_{74} = 121{,}6963 + 62{,}17717 - 63{,}81222 = 120{,}0613$
  - $c_{45} + c_{51} - c_{41} = 62{,}17717 + 76{,}02631 - 70{,}83784 = 67{,}3656$

  Insira o nó 5 entre 4 e 1

  H: {1, 2, 7, 4, 5, 1}

- H: {1, 2, 7, 4, 5, 1}
  - Nó mais próximo do ciclo:

$$k = 3$$

- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.
  - $c_{13} + c_{32} - c_{12} = 70{,}03571 + 89{,}05055 - 63{,}12686 = 95{,}9594$
  - $c_{23} + c_{37} - c_{27} = 89{,}05055 + 141{,}9894 - 61{,}6198 = 169{,}4202$
  - $c_{73} + c_{34} - c_{74} = 141{,}9894 + 140{,}7302 - 63{,}81222 = 218{,}9075$
  - $c_{43} + c_{35} - c_{45} = 140{,}7302 + 135{,}0741 - 62{,}17717 = 213{,}6271$
  - $c_{53} + c_{31} - c_{51} = 135{,}0741 + 70{,}03571 - 76{,}02631 = 129{,}0834$

  Insira o nó 3 entre 1 e 2

  H: {1, 3, 2, 7, 4, 5, 1}

- H: {1, 3, 2, 7, 4, 5, 1}
  - Nó mais próximo do ciclo:

$$k = 6$$

- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.
  - $c_{16} + c_{63} - c_{13} = 71{,}7844 + 130{,}0846 - 70{,}03571 = 78{,}4038$
  - $c_{36} + c_{62} - c_{32} = 76{,}65507 + 130{,}0846 - 89{,}05055 = 117{,}6891$
  - $c_{26} + c_{67} - c_{27} = 130{,}0846 + 157{,}8765 - 61{,}6198 = 226{,}3413$
  - $c_{76} + c_{64} - c_{74} = 157{,}8765 + 123{,}3572 - 63{,}81222 = 217{,}4215$

- $c_{46} + c_{65} - c_{45} = 123{,}3572 + 84{,}64632 - 62{,}17717 = 145{,}8264$
- $c_{56} + c_{61} - c_{51} = 84{,}64632 + 71{,}7844 - 76{,}02631 = 80{,}4044$

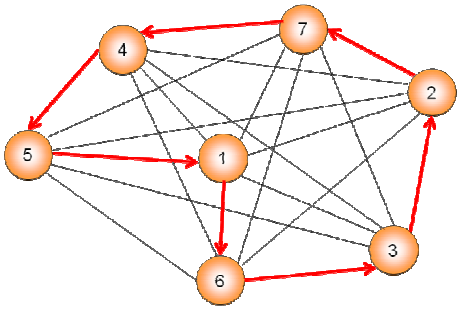
Insira o nó 6 entre 1 e 3

Ao final de todos os passos, teremos produzido a solução H: {1,6,3,2,7,4,5,1}. Para esta solução, a distância total percorrida é:

$L = d_{16} + d_{63} + d_{32} + d_{27} + d_{74} + d_{45} + d_{51}$

$L = 71{,}7844 + 76{,}6551 + 89{,}0505 + 61{,}6198 + 63{,}8122 + 62{,}1772 + 76{,}0263$

$L = 501{,}1255$



INSERÇÃO DO MAIS DISTANTE:

- Inicie com um sub-grafo contento apenas o nó i
  - i = 1
- Encontre um nó k tal que c_{ik} seja máxima e forme a rota i - k - i
  - i = 1
  - k = 7
  - $c_{17} = 87{,}28115$
- Encontre o nó k não pertencente à sub-rota, mais distante de qualquer nó da sub-rota.
  - Sub-rota inicial:

    H: {1, 7, 1}
  - Nó mais distante do ciclo:

    k = 6
    - Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$ .Insira k entre i e j.
      - $c_{16} + c_{67} - c_{17} = c_{76} + c_{61} - c_{71}$

      Insira o nó 6 entre 1 e 7 ou entre 7 e 1

      H: {1, 6, 7, 1}
- Volte ao passo anterior até formar um circuito Hamiltoniano!

- H: {1, 6, 7, 1}
  - Nó mais distante do ciclo:

    k = 3
- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.
  - $c_{13} + c_{36} - c_{16}$ = 70,0357 + 76,6551 - 71,7844 = 74,9064
  - $c_{63} + c_{37} - c_{67}$ = 76,6551 + 141,9894 - 157,8765 = 60,7680
  - $c_{73} + c_{31} - c_{71}$ = 141,9894 + 70,0357 - 87,28115 = 124,7440

  Insira o nó 3 entre 6 e 7

  H: {1, 6, 3, 7, 1}
- H: {1, 6, 3, 7, 1}
  - Nó mais distante do ciclo: k = 4
- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$ .Insira k entre i e j.
  - $c_{14} + c_{46} - c_{16}$ = 70,8378 + 62,1772 - 71,7844 = 122,4106
  - $c_{64} + c_{43} - c_{63}$ = 123,3572 + 140,7302 - 76,6551 = 187,4324
  - $c_{34} + c_{47} - c_{37}$ = 140,7302 + 63,8122 - 141,9894 = 62,5530
  - $c_{74} + c_{41} - c_{71}$ = 63,8122 + 70,8378 - 87,2812 = 47,3689

  Insira o nó entre 7 e 1

  H: {1, 6, 3, 7, 4, 1}
- H: {1, 6, 3, 7, 4, 1}
  - Nó mais distante do ciclo:

    k = 5
- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.
  - $c_{15} + c_{56} - c_{16}$ = 76,0263 + 84,6463 - 71,7844 = 88,8882
  - $c_{65} + c_{53} - c_{63}$ = 84,6463 + 135,0741 - 76,6551 = 143,0653
  - $c_{35} + c_{57} - c_{37}$ = 135,0741 + 121,6963 - 141,9894 = 114,7810
  - $c_{75} + c_{54} - c_{74}$ = 121,6963 + 62,1772 - 63,8122 = 120,0613
  - $c_{45} + c_{51} - c_{41}$ = 62,1772 + 76,0263 - 70,8378 = 67,3656

  Insira o nó 5 entre 4 e 1

  H: {1, 6, 3, 7, 4, 5, 1}
- H: {1, 6, 3, 7, 4, 5, 1}

- Nó mais distante do ciclo:

  k = 2

- Encontre o arco (i, j) na sub-rota que minimiza $c_{ik} + c_{kj} - c_{ij}$.

  Insira k entre i e j.

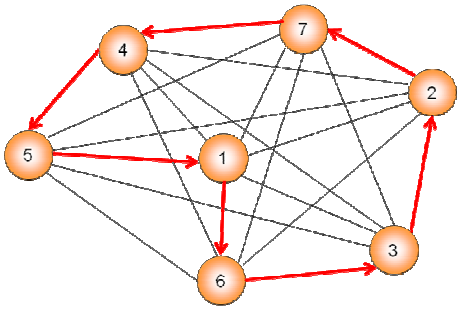
  - $c_{12} + c_{26} - c_{16} = 63,1269 + 130,0846 - 71,7844 = 121,4270$
  - $c_{62} + c_{23} - c_{63} = 130,0846 + 89,0505 - 76,6551 = 142,4801$
  - $c_{32} + c_{27} - c_{37} = 89,0505 + 61,6198 - 141,9894 = 61,6198$
  - $c_{72} + c_{24} - c_{74} = 61,6198 + 95,7549 - 63,8122 = 93,5625$
  - $c_{42} + c_{25} - c_{45} = 95,7549 + 131,2440 - 62,1772 = 164,8218$
  - $c_{52} + c_{21} - c_{51} = 131,2440 + 63,1269 - 76,0263 = 118,3446$

  Insira o nó 2 entre 3 e 7

  H: {1, 6, 3, 2, 7, 4, 5, 1}

L = 71,7844 + 76,6551 + 89,0505 + 61,6198 + 63,8122 + 62,1772 + 76,0263

L = 501,1255



INSERÇÃO DO MAIS RÁPIDO:

- Inicie com um nó inicial para formar um circuito T com 1 nó e 0 arcos
  - $T_1 = \{ 1 \}$
- Dado o conjunto T_k, ache o nó z_k não pertencente a T_k mais próximo de um nó y_k em T_k
  - $z_1 = 2$ $y_1 = 1$ $d_{12} = 63,1269$
- Seja T_{k+1} a rota com k + 1 nós obtida inserindo z_k imediatamente em seguida a y_k
  - $T_2 = \{1, 2\}$
  - $d_{13} = 70,0357$

    $d_{27} = 61,6198$
  - $z_2 = 7$ $y_2 = 2$

- Repetir os passos acima até formar o circuito Hamiltoniano

$T_3 = \{1, 2, 7\}$

$d_{14} = 70,8378$

$d_{23} = 89,0505$

$d_{74} = 63,8122$

$z_3 = 4 \qquad y_3 = 7$

$T_3 = \{1, 2, 7, 4\}$

$d_{13} = 70,0357$

$d_{23} = 89,0505$

$d_{75} = 121,6963$

$d_{45} = 62,1772$

$z_4 = 5 \qquad y_4 = 4$

$T_4 = \{1, 2, 7, 4, 5\}$

$d_{13} = 70,0357$

$d_{23} = 89,0505$

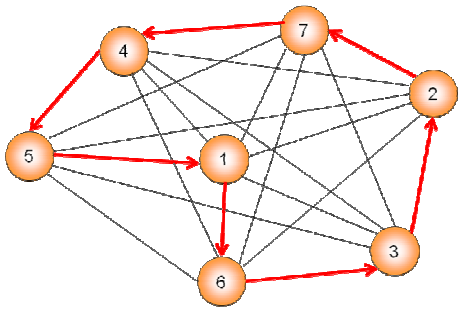
$d_{73} = 141,9894$

$d_{46} = 123,3572$

$d_{56} = 84,6463$

$z_5 = 3 \qquad y_4 = 1$

$T_4 = \{1, 3, 2, 7, 4, 5\}$



$d_{16} = 71,7844$

$d_{36} = 76,6551$

$d_{26} = 130,0846$

$d_{76} = 1157,8765$

$d_{46} = 123,3572$

$d_{56} = 84,6463$

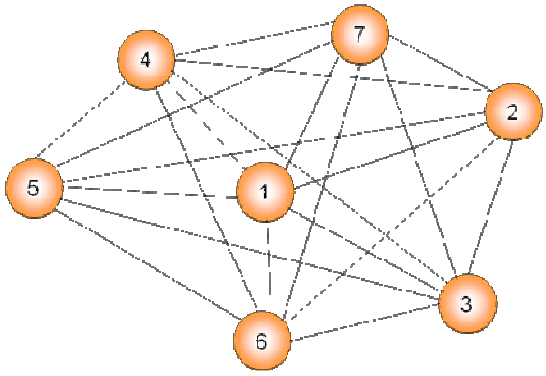
$z_5 = 6 \qquad y_4 = 1$

$T_4 = \{1, 6, 3, 2, 7, 4, 5, 1\}$ $L = 501,1255$

APÊNDICE 2 - ALGORITMO *ANT SYSTEM*

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

ALGORITMO ANT SYSTEM - PCV

Inicialização:
Para cada aresta (i, j) do grafo, estabelece-se um nível inicial de feromônio
Coloque cada formiga k em uma cidade aleatória
Loop
  Para *t = 1* até o número de iterações **faça**
    Para *k = 1* até *m* (número de formigas) **faça**
      Enquanto a formiga *k* não construir a viagem S_k
      Selecione a próxima cidade pela regra da probabilidade
      Fim
    Após cada transição da formiga *k*, aplique a regra de atualização local, motivada pela evaporação do feromônio.
    Fim
  Para cada solução calcule a distância L_K da viagem S_k descoberta pela formiga *k*
    Se $L_K < L^*$ **então**
      Para cada aresta *(i, j)*, atualize o feromônio $\tau_{ij}(t), \forall (i,j) \in L_k$, de acordo como processo de deposição e evaporação do feromônio.
      $S^* = S_k$
      $L^* = L_k$
    fim
  fim
fim
Retornar para a melhor solução S^*

- Cada formiga irá construir uma solução movendo-se de uma cidade para outra. No inicio, cada formiga é colocada em uma cidade diferente (ou colocada aleatoriamente)
- Começando de uma cidade *i*, a formiga move-se escolhendo probabilisticamente a cidade vizinha *j* (entre os vizinhos que ainda não foram escolhidos)

- **Informação heurística do PCV:**
  - Associada a aresta (*i, j*) existe um valor heurístico η_{ij} dado por $\eta_{ij} = \frac{1}{d_{ij}}$, que representa a atratividade da formiga visitar a cidade *i* depois de visitar a cidade *j.*
  - O valor de η_{ij} é inversamente proporcional a distância d_{ij} entre as cidades *i* e *j*.

η_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	∞	0,01584	0,01428	0,01412	0,01315	0,01393	0,01146
2	0,01584	∞	0,01123	0,01044	0,00762	0,00769	0,01623
3	0,01428	0,01123	∞	0,00711	0,00740	0,01305	0,00704
4	0,01412	0,01044	0,00711	∞	0,01608	0,00811	0,01567
5	0,01315	0,00762	0,00740	0,01608	∞	0,01181	0,00822
6	0,01393	0,00769	0,01305	0,00811	0,01181	∞	0,00633
7	0,01146	0,01623	0,00704	0,01567	0,00822	0,00633	∞

- Probabilidade de Transição:
  - A probabilidade da formiga *k,* que esta na cidade *i*, de escolher a cidade *j* é dada pela regra:

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \notin LT} \tau_{il}^{\alpha}\eta_{il}^{\beta}}$$

onde:

- τ_{ij} é a quantidade de feromônio existente no arco (*i, j*). Inicialmente adota-se o mesmo valor para todos os arcos da rede;
- η_{ij} é função heurística que representa a atratividade do arco (*i, j),* No caso do PCV, adota-se $\eta_{ij} = \frac{1}{d_{ij}}$;
- α e β são parâmetros para determinar a influência do feromônio e da informação heurística.
- LT (Lista Tabu), que é uma lista ordenada das cidades já visitadas.

Inicio:

α e β = 0,5 (parâmetros)
$\rho = 1$ (taxa de evaporação)
t = 1 (número de iterações)
m = 3 (número de formigas)
n = 7 (número de nós)

- Para cada aresta (i, j) atribua um valor inicial $\tau_{ij}(t)$ para a intensidade da trilha.
- $\tau_{ij}(t)$ = quantidade de feromônio na aresta (i, j) na iteração t.
- $\Delta\tau_{ij}(t) = 0$ (variação da quantidade de feromônio na aresta (i, j) na iteração t)
- Temos $\tau_{ij}(t)$ como a matriz dos feromônios na iteração inicial;

$$\tau_{ij}(0) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Obs.: A escolha da próxima cidade é feita de acordo com a probabilidade de transição, mas para o desenvolvimento do exemplo, foram escolhidas as cidades com maior probabilidade. Na prática isso nem sempre acontece.

$$p_{12}^{k} = \frac{(\tau_{12})^{0,5}.(\eta_{12})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5} + (\tau_{13})^{0,5}.(\eta_{13})^{0,5} + (\tau_{14})^{0,5}.(\eta_{14})^{0,5} + (\tau_{15})^{0,5}.(\eta_{15})^{0,5} + (\tau_{16})^{0,5}.(\eta_{16})^{0,5} + (\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,01584^{\ 0,5}}{1^{0,5}.0,01584^{\ 0,5} + 1^{0,5}.0,01428^{\ 0,5} + 1^{0,5}.0,01412^{\ 0,5} + 1^{0,5}.0,01315^{\ 0,5} + 1^{0,5}.0,01393^{\ 0,5} + 1^{0,5}.0,01146^{\ 0,5}} = 0,179$$

$$p_{13}^{k} = \frac{(\tau_{13})^{0,5}.(\eta_{13})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5} + (\tau_{13})^{0,5}.(\eta_{13})^{0,5} + (\tau_{14})^{0,5}.(\eta_{14})^{0,5} + (\tau_{15})^{0,5}.(\eta_{15})^{0,5} + (\tau_{16})^{0,5}.(\eta_{16})^{0,5} + (\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,01428^{\ 0,5}}{1^{0,5}.0,01584^{\ 0,5} + 1^{0,5}.0,01428^{\ 0,5} + 1^{0,5}.0,01412^{\ 0,5} + 1^{0,5}.0,01315^{\ 0,5} + 1^{0,5}.0,01393^{\ 0,5} + 1^{0,5}.0,01146^{\ 0,5}} = 0,170$$

$$p_{14}^{k} = \frac{(\tau_{14})^{0,5}.(\eta_{14})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5} + (\tau_{13})^{0,5}.(\eta_{13})^{0,5} + (\tau_{14})^{0,5}.(\eta_{14})^{0,5} + (\tau_{15})^{0,5}.(\eta_{15})^{0,5} + (\tau_{16})^{0,5}.(\eta_{16})^{0,5} + (\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,01412^{\ 0,5}}{1^{0,5}.0,01584^{\ 0,5} + 1^{0,5}.0,01428^{\ 0,5} + 1^{0,5}.0,01412^{\ 0,5} + 1^{0,5}.0,01315^{\ 0,5} + 1^{0,5}.0,01393^{\ 0,5} + 1^{0,5}.0,01146^{\ 0,5}} = 0,169$$

$$p_{15}^{k} = \frac{(\tau_{15})^{0,5}.(\eta_{15})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5}+(\tau_{13})^{0,5}.(\eta_{13})^{0,5}+(\tau_{14})^{0,5}.(\eta_{14})^{0,5}+(\tau_{15})^{0,5}.(\eta_{15})^{0,5}+(\tau_{16})^{0,5}.(\eta_{16})^{0,5}+(\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0132^{0,5}}{1^{0,5}.0,01584^{0,5}+1^{0,5}.0,01428^{0,5}+1^{0,5}.0,01412^{0,5}+1^{0,5}.0,01315^{0,5}+1^{0,5}.0,01393^{0,5}+1^{0,5}.0,01146^{0,5}} = 0,163$$

$$p_{16}^{k} = \frac{(\tau_{16})^{0,5}.(\eta_{16})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5}+(\tau_{13})^{0,5}.(\eta_{13})^{0,5}+(\tau_{14})^{0,5}.(\eta_{14})^{0,5}+(\tau_{15})^{0,5}.(\eta_{15})^{0,5}+(\tau_{16})^{0,5}.(\eta_{16})^{0,5}+(\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0139^{0,5}}{1^{0,5}.0,01584^{0,5}+1^{0,5}.0,01428^{0,5}+1^{0,5}.0,01412^{0,5}+1^{0,5}.0,01315^{0,5}+1^{0,5}.0,01393^{0,5}+1^{0,5}.0,01146^{0,5}} = 0,168$$

$$p_{17}^{k} = \frac{(\tau_{17})^{0,5}.(\eta_{17})^{0,5}}{(\tau_{12})^{0,5}.(\eta_{12})^{0,5}+(\tau_{13})^{0,5}.(\eta_{13})^{0,5}+(\tau_{14})^{0,5}.(\eta_{14})^{0,5}+(\tau_{15})^{0,5}.(\eta_{15})^{0,5}+(\tau_{16})^{0,5}.(\eta_{16})^{0,5}+(\tau_{17})^{0,5}.(\eta_{17})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0115^{0,5}}{1^{0,5}.0,01584^{0,5}+1^{0,5}.0,01428^{0,5}+1^{0,5}.0,01412^{0,5}+1^{0,5}.0,01315^{0,5}+1^{0,5}.0,01393^{0,5}+1^{0,5}.0,01146^{0,5}} = 0,152$$

A maior probabilidade foi de 1 para 2, portanto não considerou-se os pontos 1 e 2 para as próximas probabilidades:

$$p_{23}^{k} = \frac{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}}{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}+(\tau_{24})^{0,5}.(\eta_{24})^{0,5}+(\tau_{25})^{0,5}.(\eta_{25})^{0,5}+(\tau_{26})^{0,5}.(\eta_{26})^{0,5}+(\tau_{27})^{0,5}.(\eta_{27})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0112^{0,5}}{1^{0,5}.0,0112^{0,5}+1^{0,5}.0,0104^{0,5}+1^{0,5}.0,076^{0,5}+1^{0,5}.0,077^{0,5}+1^{0,5}.0,0162^{0,5}} = 0,208$$

$$p_{24}^{k} = \frac{(\tau_{24})^{0,5}.(\eta_{24})^{0,5}}{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}+(\tau_{24})^{0,5}.(\eta_{24})^{0,5}+(\tau_{25})^{0,5}.(\eta_{25})^{0,5}+(\tau_{26})^{0,5}.(\eta_{26})^{0,5}+(\tau_{27})^{0,5}.(\eta_{27})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0104^{0,5}}{1^{0,5}.0,0112^{0,5}+1^{0,5}.0,0104^{0,5}+1^{0,5}.0,076^{0,5}+1^{0,5}.0,077^{0,5}+1^{0,5}.0,0162^{0,5}} = 0,200$$

$$p_{25}^{k} = \frac{(\tau_{25})^{0,5}.(\eta_{25})^{0,5}}{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}+(\tau_{24})^{0,5}.(\eta_{24})^{0,5}+(\tau_{25})^{0,5}.(\eta_{25})^{0,5}+(\tau_{26})^{0,5}.(\eta_{26})^{0,5}+(\tau_{27})^{0,5}.(\eta_{27})^{0,5}} =$$

$$= \frac{1^{0,5}.0,076^{0,5}}{1^{0,5}.0,0112^{0,5}+1^{0,5}.0,0104^{0,5}+1^{0,5}.0,076^{0,5}+1^{0,5}.0,077^{0,5}+1^{0,5}.0,0162^{0,5}} = 0,171$$

$$p_{26}^{k} = \frac{(\tau_{23})^{0,5}.(\eta_{26})^{0,5}}{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}+(\tau_{24})^{0,5}.(\eta_{24})^{0,5}+(\tau_{25})^{0,5}.(\eta_{25})^{0,5}+(\tau_{26})^{0,5}.(\eta_{26})^{0,5}+(\tau_{27})^{0,5}.(\eta_{27})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0077^{0,5}}{1^{0,5}.0,0112^{0,5}+1^{0,5}.0,0104^{0,5}+1^{0,5}.0,076^{0,5}+1^{0,5}.0,077^{0,5}+1^{0,5}.0,0162^{0,5}} = 0,172$$

$$p_{27}^{k} = \frac{(\tau_{27})^{0,5}.(\eta_{27})^{0,5}}{(\tau_{23})^{0,5}.(\eta_{23})^{0,5}+(\tau_{24})^{0,5}.(\eta_{24})^{0,5}+(\tau_{25})^{0,5}.(\eta_{25})^{0,5}+(\tau_{26})^{0,5}.(\eta_{26})^{0,5}+(\tau_{27})^{0,5}.(\eta_{27})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0162^{0,5}}{1^{0,5}.0,0112^{0,5}+1^{0,5}.0,0104^{0,5}+1^{0,5}.0,076^{0,5}+1^{0,5}.0,077^{0,5}+1^{0,5}.0,0162^{0,5}} = 0,250$$

Agora, a maior probabilidade foi de 2 para 7, portanto, não considerou-se para as próximas probabilidades os pontos 1, 2 e 7:

$$p_{73}^{k} = \frac{(\tau_{73})^{0,5}.(\eta_{73})^{0,5}}{(\tau_{73})^{0,5}.(\eta_{73})^{0,5}+(\tau_{74})^{0,5}.(\eta_{74})^{0,5}+(\tau_{75})^{0,5}.(\eta_{75})^{0,5}+(\tau_{76})^{0,5}.(\eta_{76})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0070^{0,5}}{1^{0,5}.0,0070^{0,5}+1^{0,5}.0,0157^{0,5}+1^{0,5}.0,0082^{0,5}+1^{0,5}.0,0063^{0,5}} = 0,221$$

$$p_{74}^{k} = \frac{(\tau_{74})^{0,5}.(\eta_{74})^{0,5}}{(\tau_{73})^{0,5}.(\eta_{73})^{0,5} + (\tau_{74})^{0,5}.(\eta_{74})^{0,5} + (\tau_{75})^{0,5}.(\eta_{75})^{0,5} + (\tau_{76})^{0,5}.(\eta_{76})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0157^{0,5}}{1^{0,5}.0,0070^{0,5} + 1^{0,5}.0,0157^{0,5} + 1^{0,5}.0,0082^{0,5} + 1^{0,5}.0,0063^{0,5}} = 0,330$$

$$p_{75}^{k} = \frac{(\tau_{75})^{0,5}.(\eta_{75})^{0,5}}{(\tau_{73})^{0,5}.(\eta_{73})^{0,5} + (\tau_{74})^{0,5}.(\eta_{74})^{0,5} + (\tau_{75})^{0,5}.(\eta_{75})^{0,5} + (\tau_{76})^{0,5}.(\eta_{76})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0082^{0,5}}{1^{0,5}.0,0070^{0,5} + 1^{0,5}.0,0157^{0,5} + 1^{0,5}.0,0082^{0,5} + 1^{0,5}.0,0063^{0,5}} = 0,239$$

$$p_{76}^{k} = \frac{(\tau_{76})^{0,5}.(\eta_{76})^{0,5}}{(\tau_{73})^{0,5}.(\eta_{73})^{0,5} + (\tau_{74})^{0,5}.(\eta_{74})^{0,5} + (\tau_{75})^{0,5}.(\eta_{75})^{0,5} + (\tau_{76})^{0,5}.(\eta_{76})^{0,5}} =$$

$$= \frac{1^{0,5}.0,0063^{0,5}}{1^{0,5}.0,0070^{0,5} + 1^{0,5}.0,0157^{0,5} + 1^{0,5}.0,0082^{0,5} + 1^{0,5}.0,0063^{0,5}} = 0,210$$

A maior probabilidade foi de 7 para 4, portanto, não considerou-se para as próximas probabilidades os pontos 1, 2 , 7 e 4:

$$p_{43}^{k} = \frac{(\tau_{43})^{0,5}.(\eta_{43})^{0,5}}{(\tau_{43})^{0,5}.(\eta_{43})^{0,5} + (\tau_{45})^{0,5}.(\eta_{45})^{0,5} + (\tau_{46})^{0,5}.(\eta_{46})^{0,5}} = \frac{1^{0,5}.0,0071^{0,5}}{1^{0,5}.0,0071^{0,5} + 1^{0,5}.0,0161^{0,5} + 1^{0,5}.0,0081^{0,5}} = 0,2799$$

$$p_{45}^{k} = \frac{(\tau_{45})^{0,5}.(\eta_{45})^{0,5}}{(\tau_{43})^{0,5}.(\eta_{43})^{0,5} + (\tau_{45})^{0,5}.(\eta_{45})^{0,5} + (\tau_{46})^{0,5}.(\eta_{46})^{0,5}} = \frac{1^{0,5}.0,0161^{0,5}}{1^{0,5}.0,0071^{0,5} + 1^{0,5}.0,0161^{0,5} + 1^{0,5}.0,0081^{0,5}} = 0,421$$

$$p_{46}^{k} = \frac{(\tau_{46})^{0,5}.(\eta_{46})^{0,5}}{(\tau_{43})^{0,5}.(\eta_{43})^{0,5} + (\tau_{45})^{0,5}.(\eta_{45})^{0,5} + (\tau_{46})^{0,5}.(\eta_{46})^{0,5}} = \frac{1^{0,5}.0,0081^{0,5}}{1^{0,5}.0,0071^{0,5} + 1^{0,5}.0,0161^{0,5} + 1^{0,5}.0,0081^{0,5}} = 0,299$$

A maior probabilidade foi de 4 para 5, portanto, não considerou-se para as próximas probabilidades os pontos 1, 2, 7, 4 e 5:

$$p_{53}^{k} = \frac{(\tau_{53})^{0,5}.(\eta_{53})^{0,5}}{(\tau_{53})^{0,5}.(\eta_{53})^{0,5} + (\tau_{56})^{0,5}.(\eta_{56})^{0,5}} == \frac{1^{0,5}.0,074^{0,5}}{1^{0,5}.0,074^{0,5} + 1^{0,5}.0,0118^{0,5}} = 0,442$$

$$p_{56}^{k} = \frac{(\tau_{56})^{0,5}.(\eta_{56})^{0,5}}{(\tau_{53})^{0,5}.(\eta_{53})^{0,5} + (\tau_{56})^{0,5}.(\eta_{56})^{0,5}} == \frac{1^{0,5}.0,0118^{0,5}}{1^{0,5}.0,074^{0,5} + 1^{0,5}.0,0118^{0,5}} = 0,558$$

A maior probabilidade foi de 5 para 6, sobrando apenas o ponto 3, que terá 100% de probabilidade de escolha, ficando assim 1, 2, 7, 4, 5, 6, 3.

Todo esse desenvolvimento esta resumido nas tabelas abaixo, o que será feito de forma análoga para as próximas formigas.

	pij	1	2	3	4	5	6	7	∑
k = 1	**1**	0,000	**0,179**	0,170	0,169	0,163	0,168	0,152	1,000
1ª cidade: 1	**2**	0,000	0,000	0,208	0,200	0,171	0,172	**0,250**	1,000
2ª cidade: 2	**3**	**1,000**	0,000	0,000	0,000	0,000	0,000	0,000	1,000
3ª cidade: 7	**4**	0,000	0,000	0,2799	0,000	**0,421**	0,299	0,000	1,000
4ª cidade: 4	**5**	0,000	0,000	0,442	0,000	0,000	**0,558**	0,000	1,000
5ª cidade: 5	**6**	0,000	0,000	**1,000**	0,000	0,000	0,000	0,000	1,000
6ª cidade: 6	**7**	0,000	0,000	0,221	**0,330**	0,239	0,210	0,000	1,000
7ª cidade: 3									
Fechamento da rota: 1									

Rota: 1 - 2 - 7 - 4 - 5 - 6 - 3 - 1
L_1 = 482,073150

- Atualização do feromônio, sendo $\Delta\tau_{ij}^{k}$ a quantidade de feromônio que a formiga *k* deposita sobre a aresta *(i, j)*, e *Q* é uma constante positiva, que nesse caso utilizou-se *Q* = 10.

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{se o arco (i, j), pertence ao percurso descrito em } LT_k \\ 0 & \text{caso contrario} \end{cases}$$

$$\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \frac{10}{482{,}70315} = 0{,}021$$

$\Delta\tau_{ij}$	1	2	3	4	5	6	7
1	∞	1,021	1,000	1,000	1,000	1,000	1,000
2	1,000	∞	1,000	1,000	1,000	1,000	1,021
3	1,021	1,000	∞	1,000	1,000	1,000	1,000
4	1,000	1,000	1,000	∞	1,021	1,000	1,000
5	1,000	1,000	1,000	1,000	∞	1,021	1,000
6	1,000	1,000	1,021	1,000	1,000	∞	1,000
7	1,000	1,000	1,000	1,021	1,000	1,000	∞

	pij	1	2	3	4	5	6	7	∑
k = 2	**1**	0,000	0,000	**0,503**	0,000	0,000	0,497	0,000	1,000
1ª cidade: 2	**2**	0,197	0,000	0,166	0,160	0,137	0,137	**0,202**	1,000
2ª cidade: 7	**3**	0,000	0,000	0,000	0,000	0,000	**1,000**	0,000	1,000
3ª cidade: 4	**4**	0,282	0,000	0,200	0,000	**0,304**	0,214	0,000	1,000
4ª cidade: 5	**5**	**0,369**	0,000	0,277	0,000	0,000	0,354	0,000	1,000
5ª cidade: 1	**6**	0,000	**1,000**	0,000	0,000	0,000	0,000	0,000	1,000
6ª cidade: 3	**7**	0,219	0,000	0,172	**0,259**	0,186	0,163	0,000	1,000
7ª cidade: 6									
Fechamento da rota: 2									

Rota: 2 - 7 - 4 - 5 - 1 - 3 - 6 - 2
L_2 = 540,410880

- Atualização do feromônio: $\Delta\tau_{ij}^{k} = \frac{10}{540,410880} = 0,0185$

$\Delta\tau_{ij}$	1	2	3	4	5	6	7
1	∞	1,021	1,019	1,000	1,000	1,000	1,000
2	1,000	∞	1,000	1,000	1,000	1,000	1,039
3	1,021	1,000	∞	1,000	1,000	1,019	1,000
4	1,000	1,000	1,000	∞	1,039	1,000	1,000
5	1,019	1,000	1,000	1,000	∞	1,021	1,000
6	1,000	1,019	1,021	1,000	1,000	∞	1,000
7	1,000	1,000	1,000	1,039	1,000	1,000	∞

k = 3
1ª cidade: 5
2ª cidade: 4
3ª cidade: 7
4ª cidade: 2
5ª cidade: 1
6ª cidade: 3
7ª cidade: 6
Fechamento da rota: 5

pij	**1**	**2**	**3**	**4**	**5**	**6**	**7**	∑
1	0,000	0,000	**0,505**	0,000	0,000	0,495	0,000	1,000
2	**0,394**	0,000	0,332	0,000	0,000	0,274	0,000	1,000
3	0,000	0,000	0,000	0,000	0,000	**1,000**	0,000	1,000
4	0,228	0,196	0,162	0,000	**0,000**	0,173	**0,240**	1,000
5	0,188	0,142	0,140	**0,206**	0,000	0,178	0,147	1,000
6	0,000	0,000	**0,000**	0,000	**1,000**	0,000	0,000	1,000
7	0,269	**0,320**	0,211	0,000	0,000	0,200	0,000	1,000

Rota: 5 - 4 - 7 - 2 - 1 - 3 - 6 - 5
L_3 = 482,073150

- Atualização do feromônio: $\Delta\tau_{ij}^{k} = \frac{10}{482,073150} = 0,020743$

$\Delta\tau_{ij}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	∞	1,021	1,039	1,000	1,000	1,000	1,000
2	1,021	∞	1,000	1,000	1,000	1,000	1,039
3	1,021	1,000	∞	1,000	1,000	1,039	1,000
4	1,000	1,000	1,000	∞	1,039	1,000	1,021
5	1,019	1,000	1,000	1,021	∞	1,021	1,000
6	1,000	1,019	1,021	1,000	1,021	∞	1,000
7	1,000	1,021	1,000	1,039	1,000	1,000	∞

A rota é composta pelo caminho que contém maior feromônio:
Rota: 1 - 3 - 6 - 5 - 4 - 7 - 2 - **1**

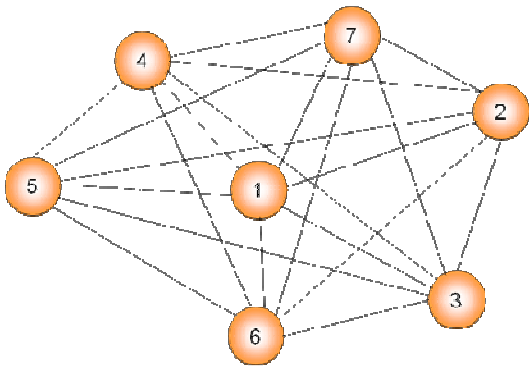
O processo continua até que se atinja o critério de parada escolhido. Entre os critérios sugeridos pela literatura, tem-se:

- Número máximo de iterações;
- Estagnação ou convergência;
- Situação na qual todas as formigas seguem sempre o mesmo percurso
- A estagnação é causada pelo excessivo crescimento de feromônio nas arestas de uma viagem sub-ótima;
- Apesar da natureza estocástica do algoritmo, uma forte concentração de feromônio nas arestas força a formiga a fazer sempre o mesmo percurso;
- As formigas artificiais possuem movimentação discreta, sendo que seus movimentos consistem em origens e destinos discretos.

APÊNDICE 3 - ALGORITMO SIMULATED ANNEALING

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

ALGORITMO SA - PCV

Inicialização (S_0, M, P, L, α)
$S = S_0$
T_0 = Temp_Inic()
$T = T_0$
Rnd(0, 1)
j = 1 (Número da iteração)
Repita
i = 1 (Número de perturbações efetuadas)
Repita
S_i = PERTURBA(S_i – 1)
$\Delta f_i = f(S_i) - f(S_i - 1)$
Se $(\Delta f_i \leq 0)$ ou $e^{-\Delta f_i / T} > Rnd$ **então**
$S = S_i$
nsucc = nsucc + 1
fim
i = i + 1
Até (nsucc ≥ L) ou (i ≥ P)
$T = \alpha . T$
j = j + 1
Até (nsucc = 0) ou (j ≥ M)

n = número de cidades
M = número máximo de iterações
P = número máximo de perturbações
L = limite de sucessos por iteração
(perturbações aceitas em cada iteração)
α = fator de redução de temperatura
S_0 **=** custo da configuração inicial
S = melhor solução (configuração obtida)
T_0 **=** temperatura inicial

$$T_0 = \frac{-\Delta E^+}{ln(\xi_0)}$$

ξ_0 = valor empírico em entre 0 e 8
Rnd = randômico

ΔE^+ = média aritmética, para um número randômico de perturbações, dos incrementos da função objetivo.

Para este exemplo usaremos:

M = 2 (número máximo de iterações)
P = 3 (número Maximo de perturbações por iteração)
L = 3 (limite de sucessos por iteração)
α = 0,9
ξ_0 = 0,8
$T = T_0$
Rnd = 0,4
S_0 = { 1, 6, 3, 2, 7, 4, 5, 1}
$f(S_0)$ = 71,7844 + 76,6551 + 89,0505 + 61,6198 + 63,8122 + 62,1772 + 76,0263 = 501,1255

Usando como soluções aleatórias:

S_1 = { 1, 3, 6, 2, 7, 4, 5, 1} $\rightarrow$ $f(S_1)$ = 540,4109
S_2 = { 2, 3, 6, 1, 7, 4, 5, 2} $\rightarrow$ $f(S_2)$ = 582,0046
S_3 = { 3, 1, 6, 2, 7, 4, 5, 3} $\rightarrow$ $f(S_3)$ = 594,5879

ΔE^+ é a média das diferenças nas perturbações da função objetivo

$$\left.\begin{matrix} S_0 = 501{,}1255 \\ S_1 = 540{,}4109 \end{matrix}\right\rangle 39{,}2854$$

$$\left.\begin{matrix} S_1 = 540{,}4109 \\ S_2 = 582{,}0046 \end{matrix}\right\rangle 41{,}5932 \qquad \Delta E^+ = \frac{39{,}2854 + 41{,}5932 + 12{,}5833}{3} = 31{,}154$$

$$\left.\begin{matrix} S_2 = 583{,}0046 \\ S_3 = 594{,}5879 \end{matrix}\right\rangle 12{,}5833$$

$$T_0 = \frac{-\Delta E^+}{ln(\xi_0)} = \frac{-31{,}154}{-0{,}22} = 141{,}609$$

j = 1 (**1ª Iteração**)

i = 1 (número de perturbações efetuadas)

S_i = perturba (S_{i+1})
S_1 = perturba (S_0) (trocando o 6 e 4)
S_1 = {1, 3, 4, 2, 7, 6, 5, 1}
$f(S_1)$ = 70,0357 + 140,7302 + 95,7549 + 61,6198 + 157,8765 + 84,6463 + 76,0263
$f(S_1)$ = 686,6898

$$\Delta f_i = f(S_i) - f(S_0) = 686{,}6898 - 501{,}1255 = 185{,}5643$$

$$e^{-\frac{\Delta f_i}{T}} = e^{-\frac{185{,}5643}{141{,}609}} = 0{,}26971 < rnd(não\_aceita)$$

S_i = perturba (S_{i+1})
S_1 = perturba (S_0) (trocando o 2 e 7)
S_1 = { 1, 3, 6, 7, 2, 4, 5, 1} $\rightarrow$ $f(S_1)$ = 600,1455

$\Delta f_i = f(S_1) - f(S_0) = 600{,}1455 - 501{,}1255 = 99{,}0200$

$e^{-\frac{\Delta f_i}{T}} = e^{-\frac{99{,}0200}{141{,}609}} = 0{,}49696 > \text{rnd(aceita)}$

$S = S_1$
$n^{o}\text{sucesso} = n^{o}\text{sucesso} + 1$
$n^{o}\text{sucesso} = 1$

i = 2

$S_2 = \text{perturba}(S_1)$ (trocando o 3 e 5)
$S_2 = \{1, 5, 6, 7, 2, 4, 3, 1\} \rightarrow f(S_2) = 686{,}6898$
$\Delta f_i = f(S_2) - f(S_1) = 686{,}6898 - 600{,}1455 = 86{,}5443$

$e^{-\frac{\Delta f_i}{T}} = e^{-\frac{86{,}5443}{141{,}609}} = 0{,}54273 > \text{rnd(aceita)}$

$S = S_2$
$n^{o}\text{sucesso} = n^{o}\text{sucesso} + 1$
$n^{o}\text{sucesso} = 2$

i = 3

$S_3 = \text{perturba}(S_2)$ (trocando o 1 e 6)
$S_3 = \{6, 5, 1, 7, 2, 4, 3, 6\} \rightarrow f(S_3) = 622{,}7138$
$\Delta f_i = f(S_3) - f(S_2) = 622{,}7138 - 686{,}6898 = -63{,}9760$
$\Delta f_3 < 0\text{(aceita)}$

$S = S_3$
$n^{o}\text{sucesso} = n^{o}\text{sucesso} + 1$
$n^{o}\text{sucesso} = 3$

$T_1 = \alpha.T_0$
$T_1 = 0{,}9.141{,}609$
$T_1 = 127{,}4481$

j = 2 (2ª Iteração)

i = 1

$S_1 = \text{perturba}(S_0)$ (trocando o 2 e 7)
$S_1 = \{6, 5, 1, 2, 7, 4, 3, 6\} \rightarrow f(S_1) = 566{,}6168$
$\Delta f_i = f(S_1) - f(S_0) = 566{,}6168 - 622{,}7138 = -56{,}0970$
$\Delta f_1 < 0\text{(aceita)}$

$S = S_1$
$n^{o}\text{sucesso} = n^{o}\text{sucesso} + 1$
$n^{o}\text{sucesso} = 1$

i = 2

S_2 = perturba (S_1) (trocando o 3 e 5)
S_2 = { 6, 3, 1, 2, 7, 4, 5, 6} $\rightarrow$ f(S_2) = 482,0731
$\Delta f_i = f(S_2) - f(S_1) = 482{,}0731 - 566{,}6168 = 44{,}6762$
$\Delta f_2 < 0$(aceita)

$S = S_1$
nºsucesso = nºsucesso + 1
nºsucesso = 2

i = 3

S_3 = perturba (S_2) (trocando o 1 e 2)
S_3 = { 6, 3, 2, 1, 7, 4, 5, 6} $\rightarrow$ f(S_3) = 526,7493
$\Delta f_i = f(S_3) - f(S_2) = 526{,}7493 - 482{,}0731 = -84{,}5437$
$e^{-\frac{\Delta f_i}{T}} = e^{-\frac{44{,}6762}{127{,}4481}} = 0{,}7043 >$ rnd(aceita)

$S = S_1$
nºsucesso = nºsucesso + 1
nºsucesso = 3

$T_2 = \alpha . T_1$
$T_2 = 0{,}9.127{,}4481$
$T_2 = 114{,}70329$

O processo continua até que o se atinja o critério de parada escolhido. Entre os critérios sugeridos pela literatura, tem-se:

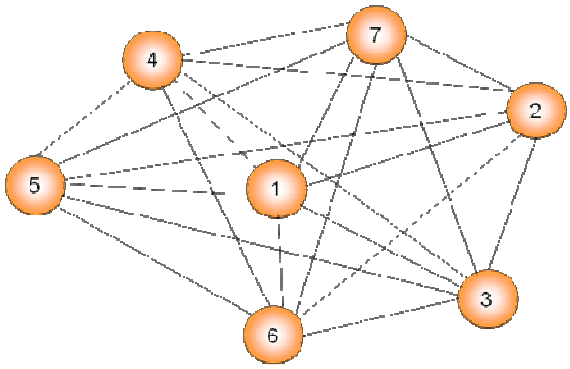
- Número máximo de iterações é atingido;
- Quando o sistema alcançou a temperatura final;
- Quando o sistema está congelado (nem movimentos para cima e nem para baixo são aceitos);

APÊNDICE 4 - ALGORITMO ALGORITMO GENÉTICO

(Representação Ordinal)

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

ALGORITMO GENÉTICO – PCV

Algoritmo Genético

Gerar uma população inicial;
Avaliar o *fitness* dos indivíduos da população;
Enquanto condição não satisfeita **faça**
Selecionar um conjunto de pais na população;
Cruzar os pais de modo que se reproduzam;
Avaliar o *fitness* dos filhos gerados;
Substituir os filhos julgados inadequados;
Fim enquanto
Solução ← Melhor indivíduo

Representação Ordinal:

Existe uma lista ordenada de cidades que serve como referência para construir a representação. A lista ***L***, varia de 1 a *n* cidades. O cromossomo é um vetor de posicionamento da cidade na lista. Para o exemplo:

L = {1, 2, 3, 4, 5, 6, 7}

Seja o cromossomo C_1 = {7, 1, 5, 4, 2, 2, 1}. O primeiro elemento do cromossomo C_1 é 7. Ele corresponde ao sétimo elemento da lista *L* que é o 7 (Rota:7).Retira-se o 7 de L e tem-se então L = {1, 2, 3, 4, 5, 6 }. O próximo elemento do cromossomo C_1 é 1. Ele corresponde ao primeiro elemento da nova lista *L* que é o 1 (Rota: {7, 1}). Retira-se o 1 de L. Tem-se então L = {1, 3, 4, 5, 6}. O próximo cromossomo C_1 é 5, que corresponde ao quinto elemento da lista *L* atualizada que é o 6 (Rota: {7, 1, 6}). Atualiza-se L= {2, 3, 4, 5}. O próximo elemento do cromossomo é 4, correspondendo ao quarto elemento na lista *L* atualizada (sem o 7, 1 e 6) que é o 5 (Rota:{7, 1, 6, 5}). Seguindo esse raciocínio, chega-se a rota final R_1 = {7, 1, 6, 5, 3, 4, 2, **7**} - repete-se o primeiro elemento para fechar a rota.

PASSO 1: Gerar uma lista R, com **m** cromossomos viávies de **p** elementos

População Inicial:

C_1 = {7, 1, 5, 4, 2, 2, 1} $\rightarrow$ r_1 = {7, 1, 6, 5, 3, 4, 2, **7**}
C_2 = {5, 2, 1, 3, 3, 1, 1} $\rightarrow$ r_2 = {5, 2, 1, 6, 7, 3, 4, **5**}
C_3 = {4, 6, 2, 2, 3, 1, 1} $\rightarrow$ r_3 = {4, 7, 2, 3, 6, 1, 5, **4**}
C_4 = {3, 4, 5, 4, 2, 2, 1} $\rightarrow$ r_4 = {3, 5, 7, 6, 2, 4, 1, **3**}
C_5 = {6, 5, 4, 3, 2, 1, 1} $\rightarrow$ r_5 = {6, 5, 4, 3, 2, 1, 7, **6**}

PASSO 2: Avaliar o *fitness* dos indivíduos da população

r_1 = {7,1,6,5,3,4,2,**7**} = 87,28 + 71,78 + 84,66 + 135,07 + 140,73 + 95,75 + 61,62

r_2 = {5,2,1,6,7,3,4,**5**} = 131,24 + 63,13 + 71,78 + 157,88 + 141,99 + 140,73 + 62,18

r_3 = {4,7,2,3,6,1,5,**4**} = 63,81 + 61,62 + 89,05 + 76,66 + 71,78 + 76,03 + 62,18

r_4 = {3,5,7,6,2,4,1,**3**} = 135,07 + 121,70 + 157,88 + 130,08 + 95,75 + 70,84 + 70,04

r_5 = {6,5,4,3,2,1,7,**6**} = 84,65 + 62,18 + 140,73 + 89,05 + 63,13 + 87,28 + 157,88

	Fitness	
r1	676,89	C_1
r2	768,93	C_2
r3	501,13	C_3
r4	781,36	C_4
r5	684,89	C_5
∑	3413,2	

Ordenar a lista dos cromossomos

$C_3 < C_1 < C_5 < C_2 < C_4 \rightarrow R(r_3, r_1, r_5, r_2, r_5)$

SELEÇÃO: Selecionar os pais.

PROCESSO DE SELEÇÃO NATURAL:

Quanto menor for o índice de um cromossomo, maior é a probabilidade do mesmo ser selecionado. De acordo com esta distribuição, a função de seleção é a seguinte [Mayerle, 1994]:

$$\text{Select}(R) = \left\{ r_j \in R \,/\, j = m + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.\text{Rnd}\,(m^2 + m)}}{2} \right\rceil \right\}$$

onde $\text{Rnd} \in [\,0,1\,)$ é um número aleatório uniformemente distribuído, $\lceil b \rceil$ é o menor inteiro maior do que b e $R = \{r_1, r_2, ..., r_m\}$ é o conjunto ordenado dos cromossomos, de modo que $C_1 \le C_2 \le ..... \le C_m$ e $C_i = [\text{Fitness}(r_i)]$.

$Rnd = 0{,}3$

$$j_1 = 5 + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.0{,}3.(5^2 + 5)}}{2} \right\rceil$$

$$j_1 = 6 - \left\lceil \frac{-1 + \sqrt{37}}{2} \right\rceil$$

$$j_1 = 6 - \lceil 2{,}5414 \rceil$$

$$j_1 = 6 - 3$$

$$j_1 = 3$$

$$Rj_1 = select(R)$$

$Rnd = 0{,}1$

$$j_2 = 5 + 1 - \left\lceil \frac{-1 + \sqrt{1 + 4.0{,}1.(5^2 + 5)}}{2} \right\rceil$$

$$j_2 = 6 - \left\lceil \frac{-1 + \sqrt{13}}{2} \right\rceil$$

$$j_2 = 6 - \lceil 1{,}3027 \rceil$$

$$j_2 = 6 - 2$$

$$j_2 = 4$$

$$Rj_2 = select(R)$$

Como $j_1 \neq j_2$, selecionamos r_3 e r_4 de R

Pai 1 → r_3(6,5,4,3,2,1,7,**6**)

Pai 2 → r_4(5,2,1,6,7,3,4,**5**)

Operador HX (Heuristic Crossover):

O operador HX utiliza a distancias entre as cidades, isto é, o tamanho dos arcos. O mesmo pode ser descrito da seguinte forma:

1) Escolha uma cidade aleatória inicial ***i*** de um dos pais.

2) Calcular as distâncias entre a cidade aleatória ***i*** escolhida e suas vizinhas (*i -1 e* i + 1) em ambos os pais.

3) Escolher a cidade de menor distância, a qual formará uma sub-rota.

4) Se a menor distância escolhida formar um ciclo, então, escolha uma nova cidade aleatória que não introduza um ciclo.

5) Repita os passos 2, 3 e 4 até que todas as cidades sejam incluídas na rota.

Pai 1 → r_3 = {6,5,4,3,2,1,7,**6**} **Pai 2** → r_4 = {5,2,1,6,7,3,4,**5**}

Vamos começar com a cidade aleatória ***i = 1***

Pai 1	**Pai 2**
d_{12} = 2 - 1 = 63,13	d_{21} = 2 - 1 = 63,13
d_{17} = 1 - 7 = 87,28	d_{16} = 1 - 6 = 71,78

Como a distância de 1 até 2 é a menor, vamos escolher a cidade 2, formando a sub-rota {1, 2}. Agora, calcula-se a distância da cidadea 2 até suas vizinhas nos pais.

Pai 1	**Pai 2**
d_{32} = 3 - 2 = 89,05	d_{52} = 5 - 2 = 131,24
d_{21} = 2 - 1 = não iremos usar pois fecha um ciclo	d_{21} = 2 - 1 = não iremos usar pois fecha um ciclo

Como a distância de 3 até 2 é a menor, vamos escolher a cidade 3, formando agora a sub-rota {1,2, 3}

Pai 1	**Pai 2**
d_{43} = 4 - 3 = 140,73	d_{73} = 7 - 3 = 141,99
d_{32} = 3 - 2 = não iremos usar pois fecha um ciclo	d_{34} = 3 - 4 = 140,73

Como a distância de 4 até 3 é a menor, vamos escolher a cidade 4, formando agora a sub-rota {1,2,3,4}

Pai 1	**Pai 2**
d_{54} = 5 - 4 = 62,18	d_{34} = 3 - 4 = não iremos usar pois fecha um ciclo
d_{43} = 4 - 3 = não iremos usar pois fecha um ciclo	d_{45} = 4 – 5 = 62,18

Como a distância de 5 até 4 é a menor, vamos escolher a cidade 5, formando agora a sub-rota {1,2,3,4,5}

Pai 1	**Pai 2**
d_{65} = 6 - 5 = 84,65	d_{52} = 5 - 2 = não iremos usar pois fecha um ciclo
d_{54} = 5 - 4 = não iremos usar pois fecha um ciclo	d_{45} = 4 - 5 = não iremos usar pois fecha um ciclo

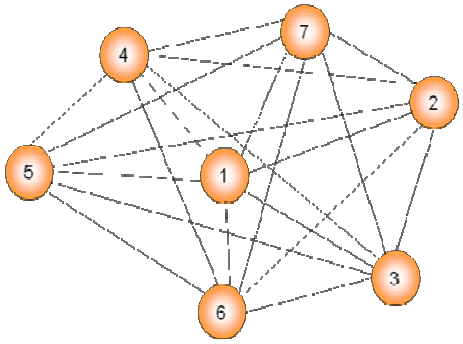
Como a distância de 6 até 5 é a menor, vamos escolher a cidade 6, formando agora a sub-rota {1,2,3,4,5,6}.

Como fica faltando somente a cidade 7, fechamos a rota como sendo {1,2,3,4,5,6,7,1}, repetindo a primeira cidade, cuja o *fitness* vale 684,89.

APÊNDICE 5 - ALGORITMO GRASP

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

ALGORITMO GRASP - PCV

- Considerar um depósito inicial: D
- 1ª FASE: Construção
  Repita
  - Escolha os candidatos da lista LRC tal que: $g(c) \leq s_1 + \alpha(s_2 - s_1)$
    $s_1 = \min\{g(t), t \in c\}$
    $s_2 = \max\{g(t), t \in c\}$
    $\alpha \in (0,1)$
  - Escolha, aleatoriamente um dos candidatos (c1) da lista LRC e montar a rota inicial:

  D – c1 – D;

  Até que todos os pontos tenham sido designados
- Fim da 1ª fase
- 2ª FASE: Melhoria
  - Selecione dois pontos da rota
    - Efetue todas as trocas possíveis
    - Calcule o custo da nova rota
    - Se o custo da nova rota for menor do que o custo da rota anterior, então troque.
  - Parar quando não houver mais melhoria na FO.

1ª Fase: Construção

Iremos considerar como depósito inicial o ponto **1.**

Pontos		Custo
1	2	63,127
1	3	70,036
1	4	70,838
1	5	76,026
1	6	71,784
1	7	87,281

Escolher a LRC tal que: $g(c) \leq s_1 + \alpha(s_2 - s_1)$

$s_1 = \min\{g(t), t \in c\}$

$s_2 = \max\{g(t), t \in c\}$

$\alpha \in (0,1)$

Com isso temos que: $s_1 = 63{,}127$ $s_2 = 87{,}281$ $\alpha = 0{,}9$

$g(c) \leq 63{,}127 + 0{,}9.(87{,}281 - 63{,}127)$

$g(c) \leq 84{,}866$

LRC = { 2, 3, 4, 5, 6}

Escolhe-se aleatoriamente um elemento desta LRC

Escolhemos o ponto **2**

Com isso temos: Rota Inicial: {1, 2, 1}

Custo = 63,127 + 63,127 =126,254

Inserir o próximo elemento na rota tal que a distância seja mínima.

Investigar todos os pontos – Inserção do Mais Próximo

Pontos			Custo
1	2	3	152,177
1	2	4	158,882
1	2	5	194,371
1	2	6	193,211
1	2	7	124,747

$s_1 = 124{,}747 \qquad s_2 = 194{,}371 \qquad \alpha = 0{,}9$

$g(c) \leq 124{,}747 + 0{,}9.(194{,}371 - 124{,}747)$

$g(c) \leq 187{,}408$

LRC = { 3, 4, 7}

Escolhe-se aleatoriamente um elemento desta LRC

Escolhemos o ponto **3**

Com isso temos: Rota : {1, 2, 3 1}

Custo = 63,127 + 89,05055 + 70,03571 =222,213

Pontos				Custo
1	2	3	4	292,908
1	2	3	5	287,251
1	2	3	6	228,832
1	2	3	7	294,167

$s_1 = 228{,}832 \quad s_2 = 294{,}167 \qquad \alpha = 0{,}9$

$g(c) \leq 228{,}832 + 0{,}9.(294{,}167 - 228{,}832)$

$g(c) \leq 287{,}633$

LRC = { 5,6}

Escolhe-se aleatoriamente um elemento desta LRC

Escolhemos o ponto **5**

Com isso temos: Rota : {1, 2, 3, 5, 1}

Custo = 63,127 + 89,05055 + 135,0741 + 76,026 =363,278

Pontos					Custo
1	2	3	5	4	349,429
1	2	3	5	6	371,898
1	2	3	5	7	408,948

$s_1 = 349{,}429 \qquad s_2 = 408{,}948 \qquad \alpha = 0{,}9$

$g(c) \leq 349{,}429 + 0{,}9.(408{,}948 - 349{,}429)$

$g(c) \leq 403$

Escolhe-se aleatoriamente um elemento desta LRC

Escolhemos o ponto **4**

Com isso temos: Rota : {1, 2, 3, 5, 4, 1}

Custo = 63,127 + 89,05055 + 135,0741 + 62,17717 + 70,838 = 420,266

Pontos						Custo
1	2	3	5	4	6	472,786
1	2	3	5	4	7	413,241

Rota :{1, 2, 3, 5, 4, 7, 1}

Custo: 500,522

Inserção do último ponto:

Rota: { 1, 2, 3, 5, 4, 7, 6, 1}

Custo: 642, 902

2ª Fase: Melhoria

Trocar 2 – 3, 2 – 4, 2 – 5, 2 – 6, 2 - 7

Rota								Custo
1	**2**	**3**	**5**	**4**	**7**	**6**	**1**	**642,902**
1	**3**	**2**	5	4	7	6	1	645,981
1	**4**	3	5	**2**	7	6	1	769,167
1	**5**	3	**2**	4	7	6	1	689,379
1	**6**	3	5	4	7	**2**	1	534,250
1	**7**	3	5	4	**2**	6	1	724,146

Como houve uma melhoria na função objetivo, efetua-se a troca onde ocorreu essa melhoria.

Rota: { 1, 6, 3, 5, 4, 7, 2, 1}

Custo: 534,25

Trocar 3 – 2, 3 – 4, 3 – 5, 3 – 6, 3 – 7.

Rota								Custo
1	**6**	**3**	**5**	**4**	**7**	**2**	**1**	**534,250**
1	**3**	**6**	5	4	7	2	1	482,073
1	6	**4**	5	**3**	7	2	1	659,129
1	6	**5**	**3**	4	7	2	1	620,794
1	6	**2**	5	4	7	**3**	1	671,128
1	6	**7**	5	4	3	2	1	706,442

Como houve uma melhoria na função objetivo, efetua-se a troca onde ocorreu essa melhoria.

Rota: { 1, 3, 6, 5, 4, 7, 2, 1}

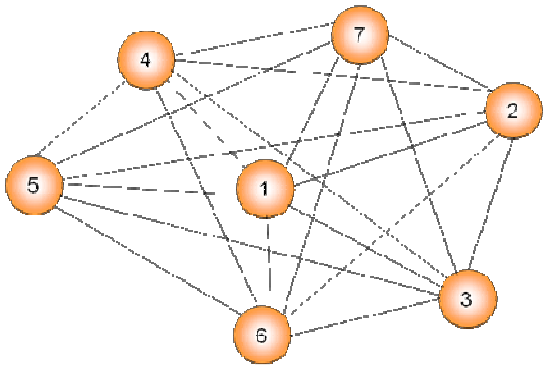
Custo: 482,073

Continuar analisando as possíveis trocas e só efetuar a troca quando houver melhoria na F.O

APÊNDICE 6 - ALGORITMO BUSCA TABU

Exemplo do Problema do Caixeiro Viajante para 7 pontos aleatórios.

	X	Y
1	-5	-1
2	43	40
3	46	-49
4	-52	52
5	-81	-3
6	-28	-69
7	2	86



Matriz Simétrica das distâncias (D_{ij}) **:**

	1	2	3	4	5	6	7
1	0	63,12686	70,03571	70,83784	76,02631	71,7844	87,28115
2	63,12686	0	89,05055	95,7549	131,244	130,0846	61,6198
3	70,03571	89,05055	0	140,7302	135,0741	76,65507	141,9894
4	70,83784	95,7549	140,7302	0	62,17717	123,3572	63,81222
5	76,02631	131,244	135,0741	62,17717	0	84,64632	121,6963
6	71,7844	130,0846	76,65507	123,3572	84,64632	0	157,8765
7	87,28115	61,6198	141,9894	63,81222	121,6963	157,8765	0

BUSCA TABU

- Gere uma solução inicial através de uma heurística rápida qualquer
- Partindo da solução inicial, são executados ao acaso 5 movimentos,
- O par que possui o menor valor de troca é selecionado.
- O par selecionado é incluído na lista tabu, com valor 3, indicando que permanecerá durante 3 iterações sem ser movimentado.
- O menor valor de troca é selecionado e a lista tabu atualizada
- Se o movimento produz um valor de troca que esta na lista tabu, ele é então executado (Critério de Aspiração)
- O processo continua até que seja atingido um critério de parada.

O número de candidatos possiveis é: $C_7^2 = \frac{7!}{2!5!} = 21$

Solução Inicial:

(1,7,3,4,2,5,6,1)
F = 753,430

C	F
2,3	676,891
2,7	634,847
4,2	632,684
5,4	786,485
6,7	725,146

	1	2	3	4	5	6	7
1							
2				3			
3							
4							
5							
6							
7							

Solução atual:

(1, 7, 3, 2, 4, 5, 6, 1)

F = 632,684

C	F	
2,4	753,430	T
2,7	576,587	
3,4	627,404	
4,5	706,884	
6,7	560,468	

	1	2	3	4	5	6	7
1							
2				2			
3							
4							
5							
6							3
7							

Solução atual:

(1, 6, 3, 2, 4, 5, 7, 1)

F = 560,468

C	F	
2,4	725,146	T
2,7	610,789	
4,7	553,821	
5,4	582,005	
7,6	632,684	T

	1	2	3	4	5	6	7
1							
2				1			
3							
4							3
5							
6							2
7							

Obs.: Se alguma das trocas que estivesse na lista tabu produzisse um resultado melhor do que todos os resultados obtidos anteriormente, essa troca seria realizada – Critério de aspiração

APÊNDICE 7 - HEURISTICAS DE CONTRUÇÃO E MELHORIA DE ROTA

VIZINHO MAIS PRÓXIMO

SEGUNDA - FEIRA:

Distância total = 10694,08

Rota:
1 - 19 - 20 - 21 - 22 - 23 - 25 - 8 - 24 - 18 - 16 - 4 - 5 - 11 - 10 - 2 - 3 - 7 - 6 - 15 - 14 - 13 - 12 - 17 - 9 - 1

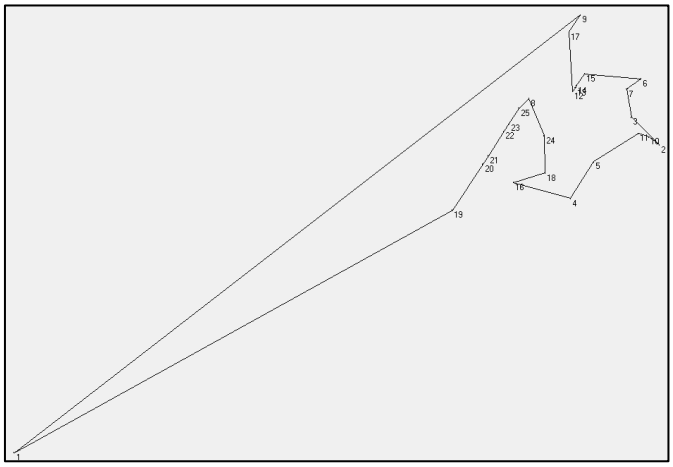


FIGURA 11 - VIZINHO MAIS PRÓXIMO (SEGUNDA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 10593,35

Economia = 100,7266

Rota com melhoria 2-OPT:
1 - 19 - 20 - 21 - 22 - 23 - 25 - 8 - 24 - 16 - 18 - 4 - 5 - 2 - 10 - 11 - 3 - 6 - 7 - 12 - 13 - 14 - 15 - 17 - 9 - 1

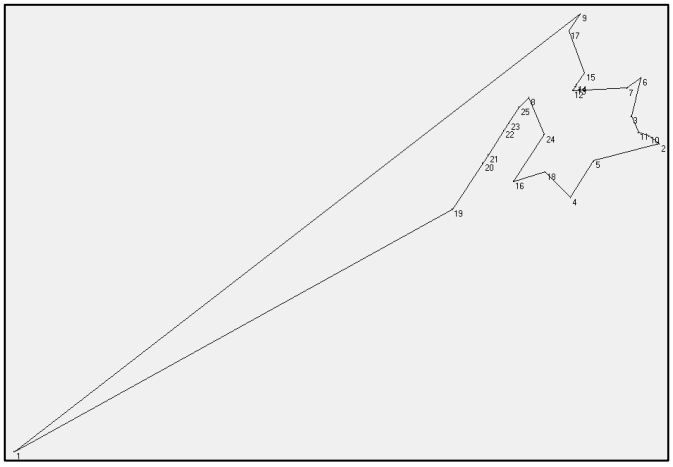


FIGURA 12 - VIZINHO MAIS PRÓXIMO COM 2-OPT (SEGUNDA FEIRA)

TERÇA FEIRA:

Distância total = 16641,36

Rota:
1 - 7 - 2 - 6 - 8 - 4 - 5 - 17 - 14 - 13 - 24 - 18 - 3 - 15 - 16 - 11 - 12 - 19 - 26 - 25 - 20 - 10 - 22 - 21 - 9 - 23 - 1

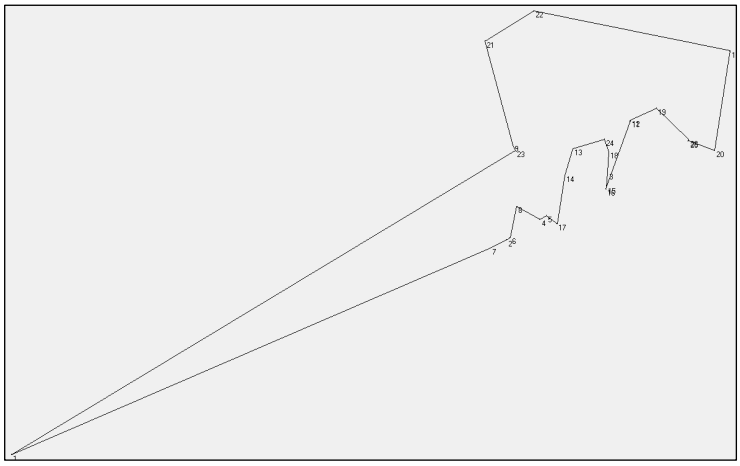


FIGURA 13 - VIZINHO MAIS PRÓXIMO (TERÇA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 16451,17

Economia = 190,1934

Rota com melhoria 2-OPT
1 - 7 - 2 - 6 - 8 - 4 - 5 - 17 - 14 - 13 - 16 - 15 - 3 - 18 - 24 - 11 - 12 - 19 - 26 - 25 - 20 - 10 - 22 - 21 - 9 - 23 - 1

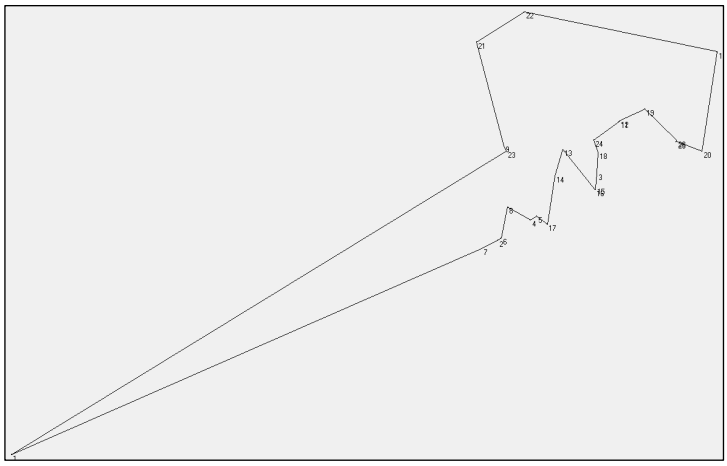


FIGURA 14 - VIZINHO MAIS PRÓXIMO COM 2-OPT (TERÇA FEIRA)

QUARTA FEIRA:

Distância total = 19173,03

Rota:
1 - 15 - 14 - 11 - 12 - 13 - 10 - 4 - 3 - 2 - 5 - 18 - 24 - 27 - 25 - 28 - 23 - 20 - 21 - 26 - 19 -16 - 17 - 22 - 8 - 9 - 7 - 6 - 1

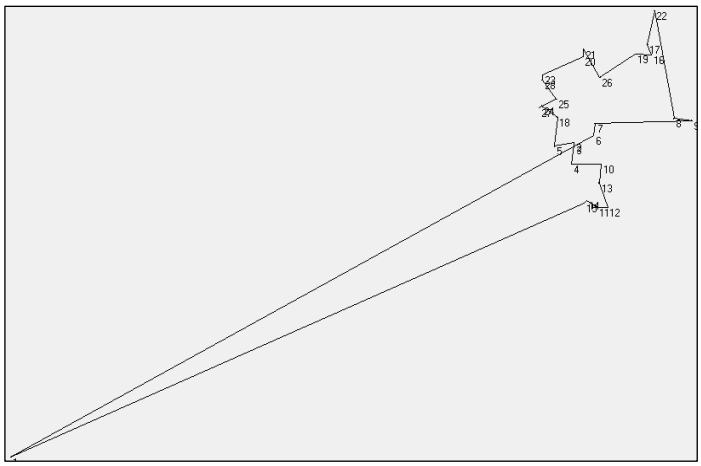


FIGURA 15 - VIZINHO MAIS PRÓXIMO (QUARTA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 18940,7

Economia = 232,3301

Rota com melhoria 2-OPT
1 - 15 - 14 - 11 - 12 - 13 - 10 - 4 - 3 - 2 - 5 - 18 - 27 - 24 - 25 - 28 - 23 - 21 - 20 - 26 - 19 - 22 - 17 - 16 - 9 - 8 - 7 - 6 - 1

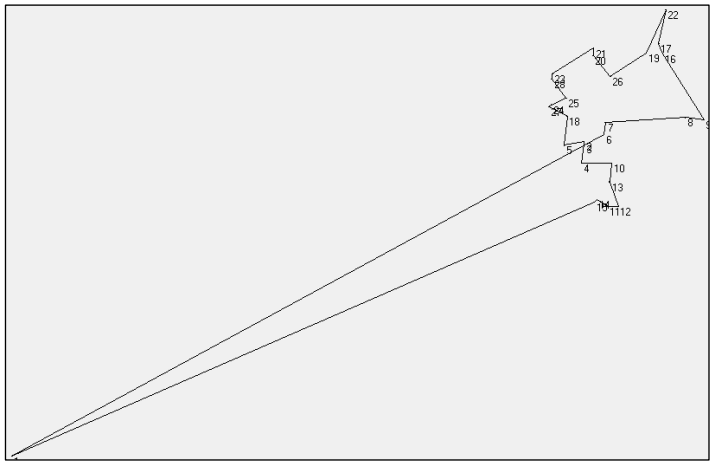


FIGURA 16 - VIZINHO MAIS PRÓXIMO COM 2-OPT (QUARTA FEIRA)

QUINTA FEIRA:

Distância total = 23779,66

Rota
1 - 16 - 18 - 12 - 17 - 7 - 5 - 20 - 6 - 15 - 8 - 9 - 11 - 13 - 3 - 4 - 2 - 10 - 14 - 19 - 1

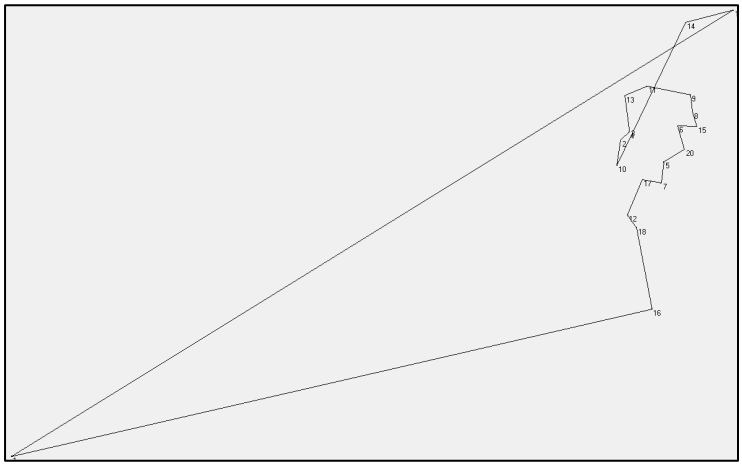


FIGURA 17 - VIZINHO MAIS PRÓXIMO (QUINTA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 23320,39

Economia = 459,2715

Rota com melhoria 2-OPT
1 - 16 - 18 - 12 - 17 - 7 - 5 - 10 - 2 - 4 - 3 - 6 - 20 - 15 - 8 - 9 - 13 - 11 - 14 - 19 - 1

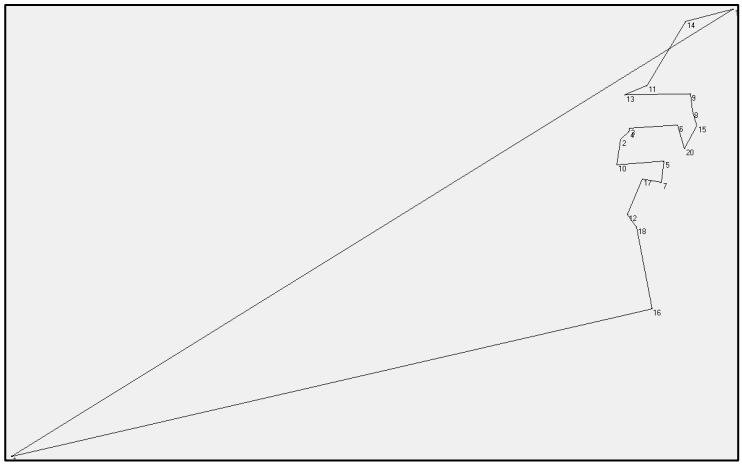


FIGURA 18 - VIZINHO MAIS PRÓXIMO COM 2-OPT (QUINTA FEIRA)

SEXTA FEIRA:

Distância total = 20137,86

Rota
1 - 33 - 3 - 2 - 4 - 5 - 7 - 8 - 9 - 6 - 29 - 28 - 27 - 26 - 25 - 11 - 10 - 12 - 13 - 14 - 30 - 15 -16 - 17 - 18 - 19 - 20 - 21 - 24 - 22 - 32 - 31 - 23 - 1

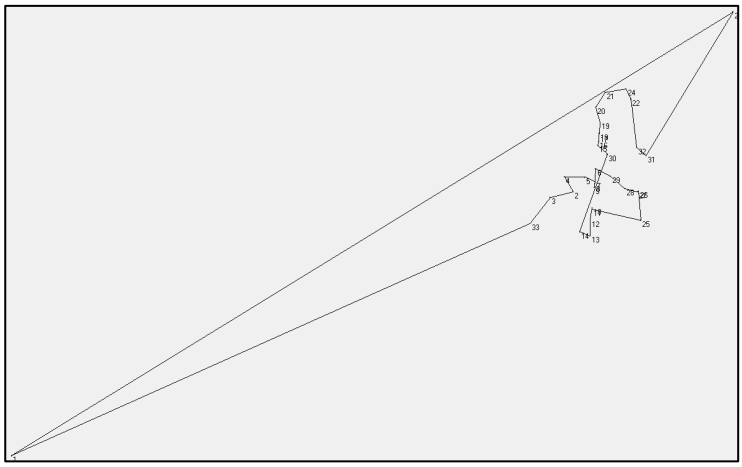


FIGURA 19 - VIZINHO MAIS PRÓXIMO (SEXTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 19793,07

Economia = 344,7891

Rota com melhoria 2-OPT
1 - 33 - 3 - 14 - 13 - 12 - 11 - 10 - 25 - 26 - 27 - 28 - 29 - 7 - 8 - 9 - 2 - 4 - 5 - 6 - 30 - 15 16 - 17 - 18 - 19 - 20 - 21 - 24 - 22 - 32 - 31 - 23 - 1

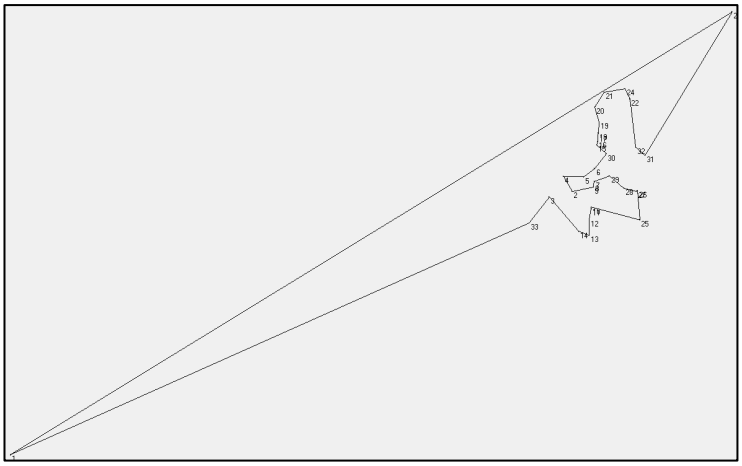


FIGURA 20 - VIZINHO MAIS PRÓXIMO COM 2-OPT (SEXTA FEIRA)

INSERÇÃO DO MAIS PRÓXIMO

SEGUNDA FEIRA:

Distância total = 10483,08

Rota:
1 - 19 - 20 - 21 - 22 - 23 - 25 - 8 - 17 - 9 - 15 - 6 - 7 - 3 - 10 - 2 - 11 - 5 - 14 - 13 - 12 - 24 - 18 - 4 - 16 - 1

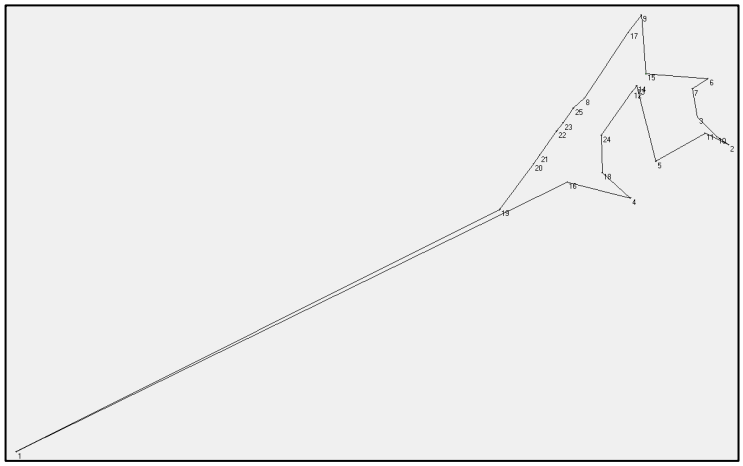


FIGURA 21 - INSERÇÃO DO MAIS PRÓXIMO (SEGUNDA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 9910,512

Economia = 572,5684

Rota com melhoria 2-OPT:
1 - 19 - 20 - 21 - 22 - 23 - 25 - 8 - 12 - 13 - 14 - 15 - 17 - 9 - 6 - 7 - 3 - 11 - 10 - 2 - 5 - 4 - 18 - 24 - 16 - 1

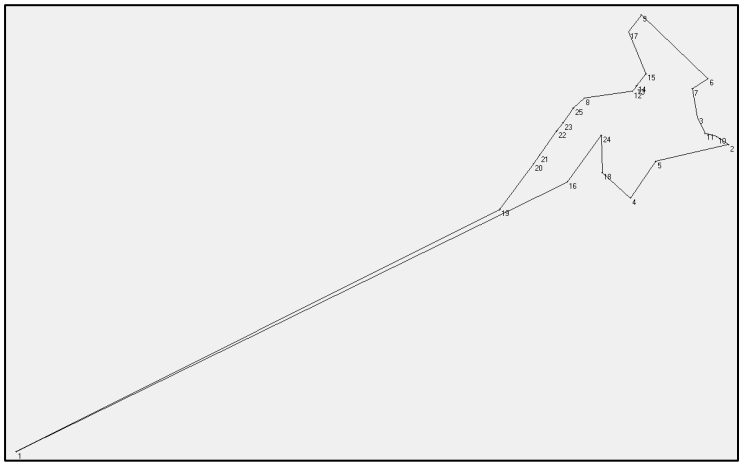


FIGURA 22 - INSERÇÃO DO MAIS PRÓXIMO COM 2-OPT (SEGUNDA FEIRA)

TERÇA FEIRA:

Distância total = 17199,55

Rota
1 - 7 - 2 - 6 - 4 - 17 - 5 - 14 - 16 - 15 - 3 - 18 - 20 - 25 - 26 - 10 - 19 - 11 - 12 - 24 - 13 - 22 - 21 - 9 - 23 - 8 -1

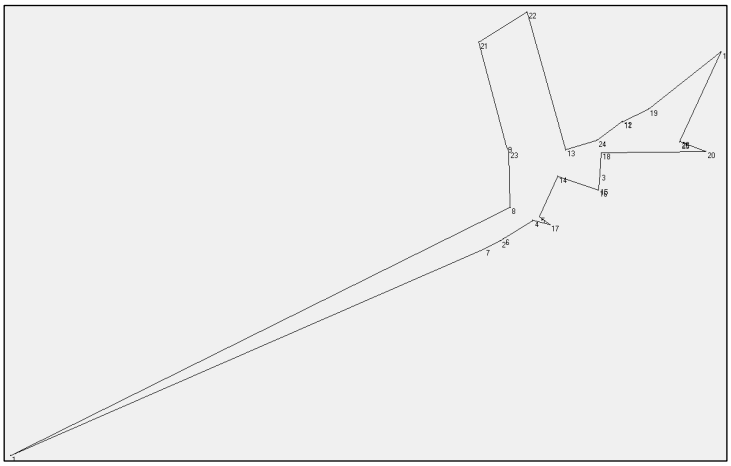


FIGURA 23 - INSERÇÃO DO MAIS PRÓXIMO (TERÇA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 16986,96

Economia = 212,5957

Rota com melhoria 2-OPT
1 - 7 - 2 - 6 - 4 - 5 - 17 - 14 - 16 - 15 - 3 - 18 - 26 - 25 - 20 - 10 - 19 - 12 - 11 - 24 - 13 - 22 - 21 - 9 - 23 - 8 - 1

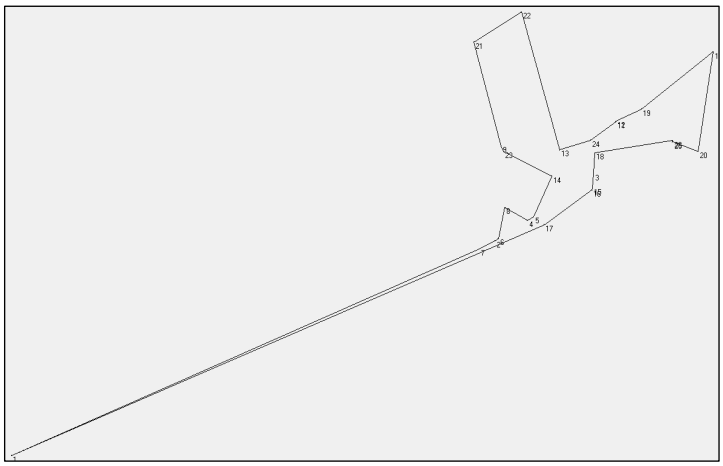


FIGURA 24 - INSERÇÃO DO MAIS PRÓXIMO COM 2-OPT (TERÇA FEIRA)

QUARTA FEIRA:

Distância total = 19362,35

Rota
1 - 15 - 14 - 11 - 12 - 13 - 10 - 6 - 7 - 25 - 26 - 8 - 9 - 16 - 17 - 22 - 19 - 20 - 21 - 23 - 28 -24 - 27 - 18 - 2 - 3 - 5 - 4 -1

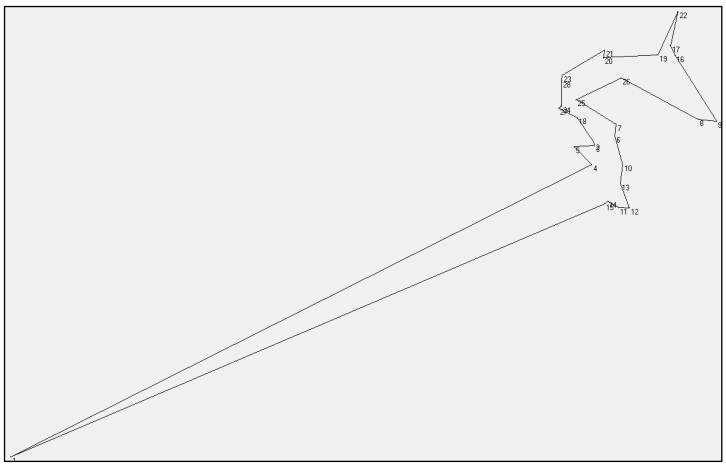


FIGURA 25 - INSERÇÃO DO MAIS PRÓXIMO (QUARTA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 19274,71

Economia = 87,63867

Rota com melhoria 2-OPT
1 - 15 - 14 - 11 - 12 - 13 - 10 - 6 - 7 - 25 - 26 - 8 - 9 - 16 - 17 - 22 - 19 - 21 - 20 - 23 - 28 24 - 27 - 18 - 5 - 2 - 3 - 4 - 1

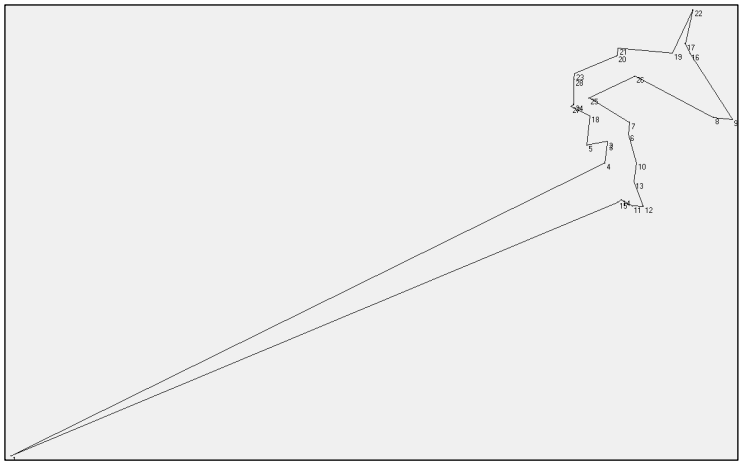


FIGURA 26 - INSERÇÃO DO MAIS PRÓXIMO COM 2-OPT (QUARTA FEIRA)

QUINTA FEIRA:

Distância total = 23506,59

Rota
1 - 16 - 18 - 12 - 7 - 20 - 15 - 8 - 9 - 6 - 5 - 17 - 10 - 4 - 3 - 11 - 19 - 14 - 13 - 2 - 1

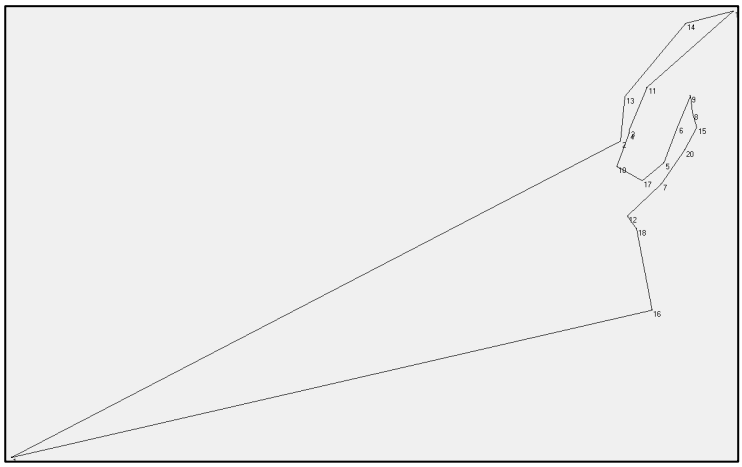


FIGURA 27 - INSERÇÃO DO MAIS PRÓXIMO (QUINTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 23506,59

Economia = 0

Rota com melhoria 2-OPT
1 - 16 - 18 - 12 - 7 - 20 - 15 - 8 - 9 - 6 - 5 - 17 - 10 - 4 - 3 - 11 - 19 - 14 - 13 - 2 - 1

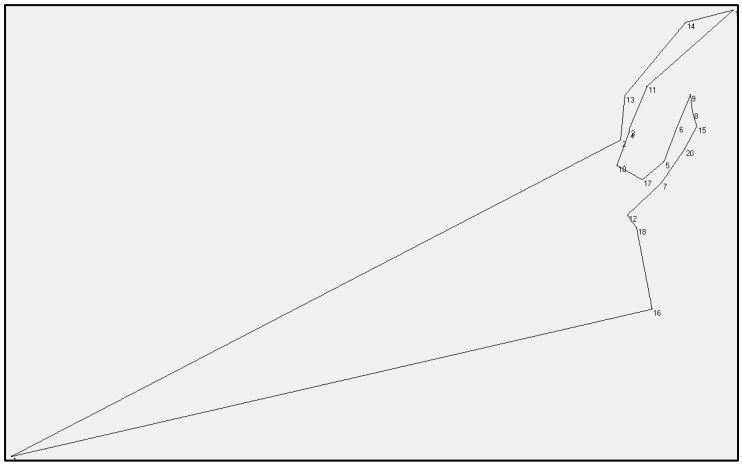


FIGURA 28 - INSERÇÃO DO MAIS PRÓXIMO COM 2-OPT (QUINTA FEIRA)

SEXTA FEIRA:

Distância total = 19294,12

Rota
1 - 33 - 2 - 14 - 13 - 12 - 11 - 10 - 9 - 8 - 7 - 28 - 27 - 26 - 25 - 29 - 30 - 31 - 32 - 19 - 22 -23 - 24 - 21 - 20 - 18 - 17 - 16 - 15 - 6 - 5 - 4 - 3 - 1

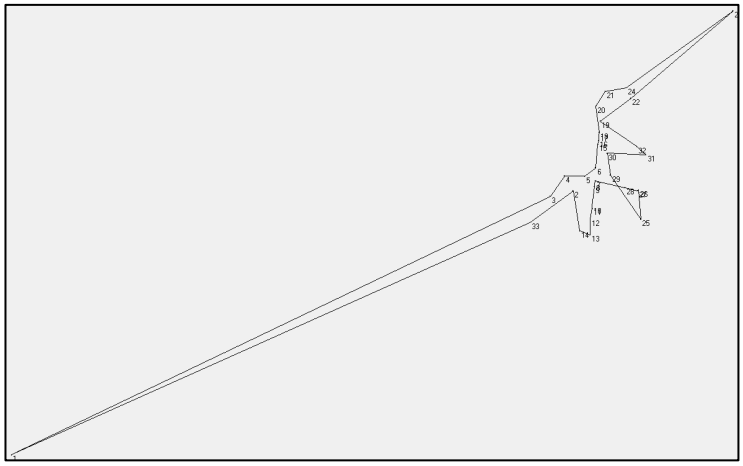


FIGURA 29 - INSERÇÃO DO MAIS PRÓXIMO (SEXTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 18542,89

Economia = 751,2305

Rota com melhoria 2-OPT
1 - 33 - 14 - 13 - 12 - 11 - 10 - 9 - 8 - 7 - 25 - 26 - 27 - 28 - 29 - 30 - 32 - 31 - 23 - 22 - 24 - 21 - 20 - 19 - 18 - 17 - 16 - 15 - 6 - 5 - 2 - 4 - 3 - 1

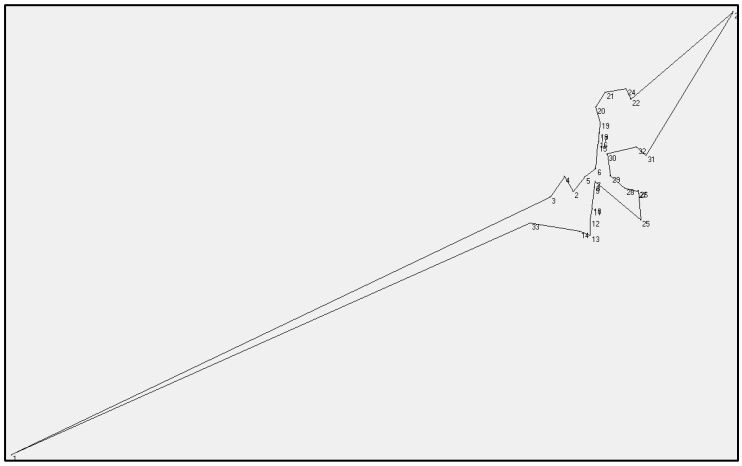


FIGURA 30 - INSERÇÃO DO MAIS PRÓXIMO COM 2-OPT (SEXTA FEIRA)

INSERÇÃO DO MAIS DISTANTE:

SEGUNDA FEIRA:

Distância total = 10054,84

Rota:
1 - 19 - 16 - 24 - 18 - 4 - 5 - 2 - 10 - 11 - 3 - 7 - 6 - 12 - 13 - 14 - 15 - 9 - 17 - 8 - 25 - 23 22 - 21 - 20 - 1

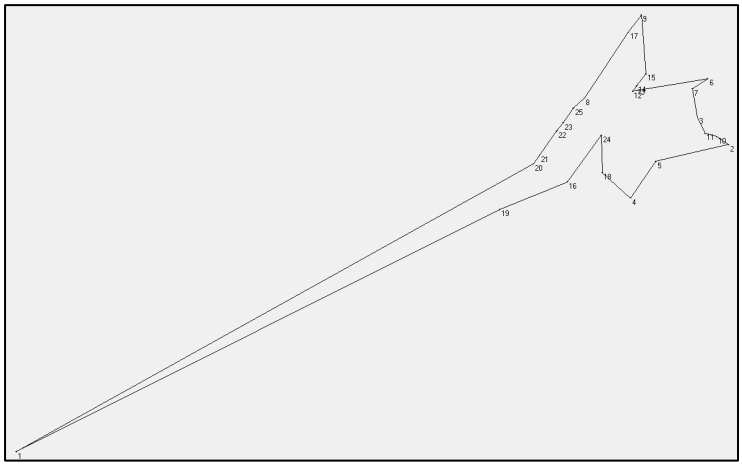


FIGURA 31 - INSERÇÃO DO MAIS DISTÂNTE (SEGUNDA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 9879,759

Economia = 175,0811

Rota com melhoria 2-OPT:
 1 - 19 - 16 - 24 - 18 - 4 - 5 - 2 - 10 - 11 - 3 - 7 - 6 - 9 - 17 - 15 14 - 13 - 12 - 8 - 25 - 23 - 22 - 21 - 20 - 1

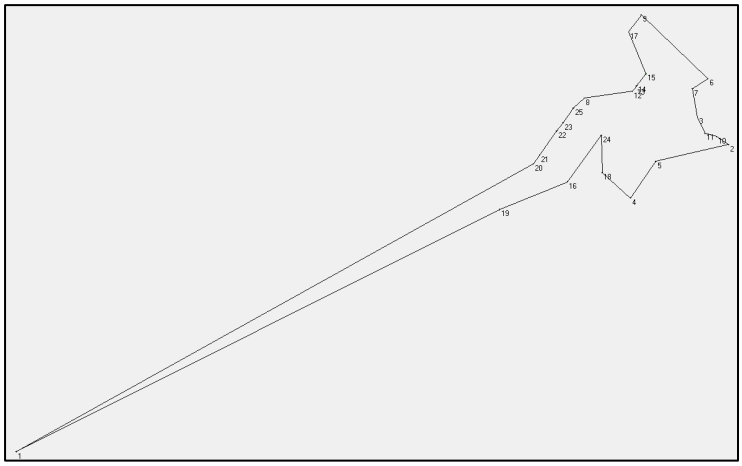


FIGURA 32 - INSERÇÃO DO MAIS DISTANTE COM 2-OPT (SEGUNDA FEIRA)

TERÇA FEIRA:

Distância total = 17331,3

Rota:
1 - 21 - 22 - 11 - 12 - 19 - 10 - 20 - 26 - 25 - 18 - 24 - 13 - 3 - 15 - 16 - 17 - 4 - 5 - 14 - 23 - 9 - 8 - 6 - 2 - 7 - 1

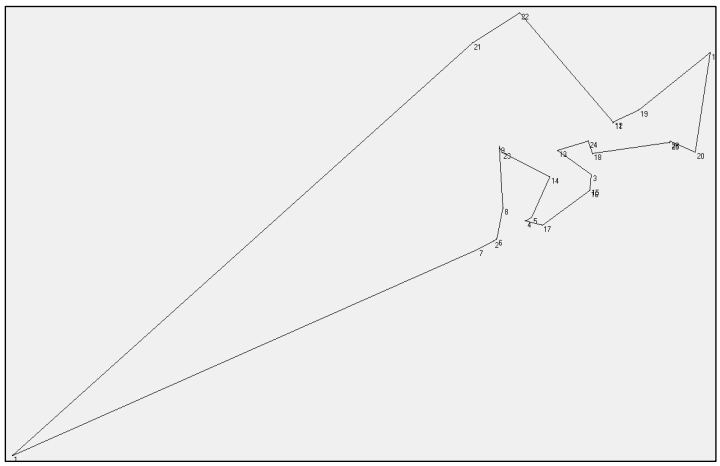


FIGURA 33 - INSERÇÃO DO MAIS DISTANTE (TERÇA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 17322,14

Economia = 9,158203

Rota com melhoria 2-OPT
1 - 21 - 22 - 11 - 12 - 19 - 10 - 20 - 25 - 26 - 18 - 24 - 13 - 3 - 15 - 16 - 17 - 4 - 5 - 14 - 9 23 - 8 - 6 - 2 - 7 - 1

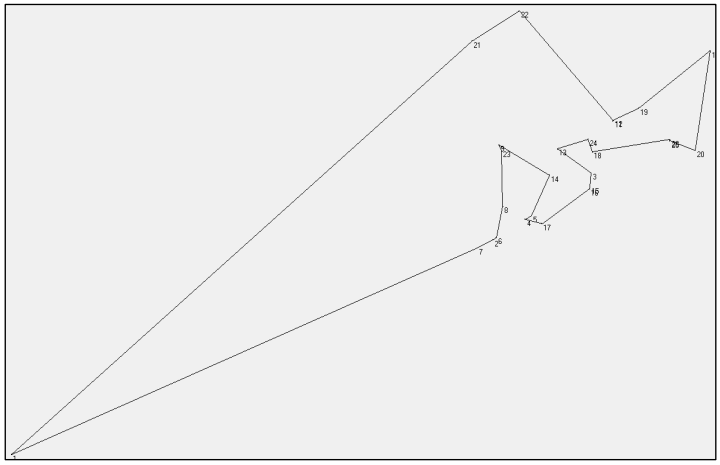


FIGURA 34 - INSERÇÃO DO MAIS DISTANTE COM 2-OPT (TERÇA FEIRA)

QUARTA FEIRA:

Distância total = 18789,62

Rota
1 - 15 - 14 - 11 - 12 - 13 - 10 - 4 - 3 - 2 - 6 - 7 - 8 - 9 - 22 - 17 - 16 - 19 - 21 - 20 - 26 - 23 - 28 - 25 - 24 - 27 - 18 - 5 - 1

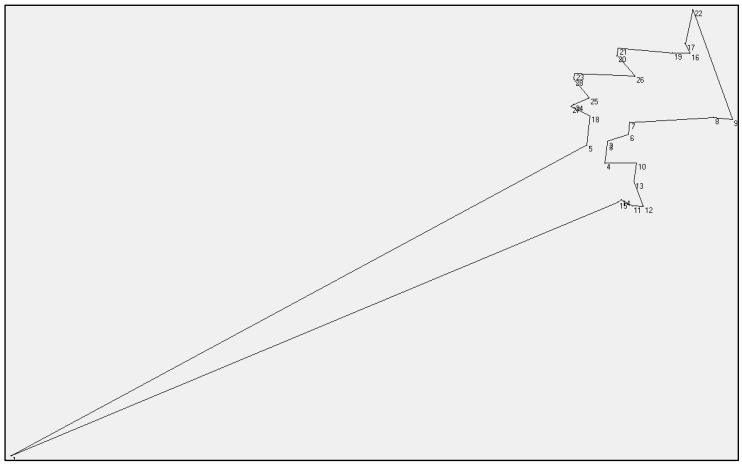


FIGURA 35 - INSERÇÃO DO MAIS DISTANTE (QUARTA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 18598,65

Economia = 190,9727

Rota com melhoria 2-OPT
1 - 15 - 14 - 11 - 12 - 13 - 10 - 4 - 3 - 2 - 6 - 7 - 8 - 9 - 16 - 17 - 22 - 19 - 26 - 20 - 21 - 23 - 28 - 25 - 24 - 27 - 18 - 5 - 1

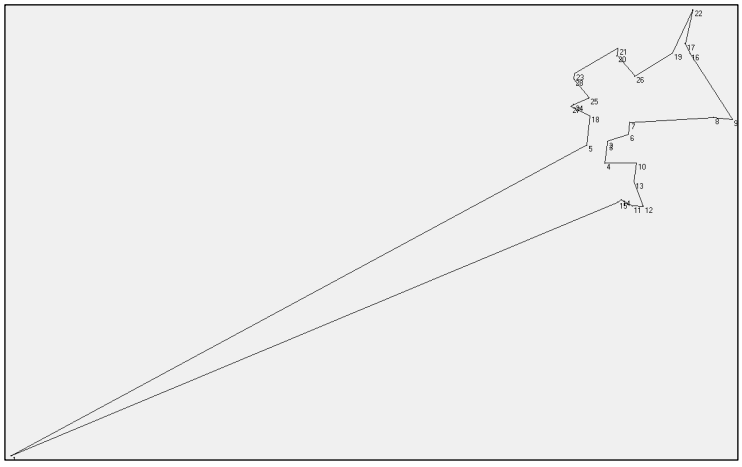


FIGURA 36 - INSERÇÃO DO MAIS DISTANTE COM 2-OPT (QUARTA FEIRA)

QUINTA FEIRA:

Distância total = 21309,23

Rota
1 - 16 - 18 - 12 - 17 - 7 - 5 - 20 - 6 - 15 - 8 - 9 - 19 - 14 - 11 - 13 - 3 - 4 - 2 - 10 - 1

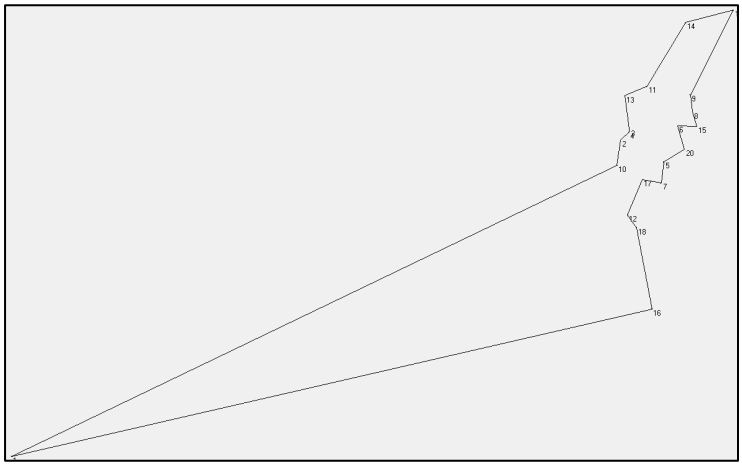


FIGURA 37 - INSERÇÃO DO MAIS DISTANTE (QUINTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 21309,23

Economia = 0

Rota com melhoria 2-OPT
1 - 16 - 18 - 12 - 17 - 7 - 5 - 20 - 6 - 15 - 8 - 9 - 19 - 14 - 11 - 13 - 3 - 4 - 2 - 10 - 1

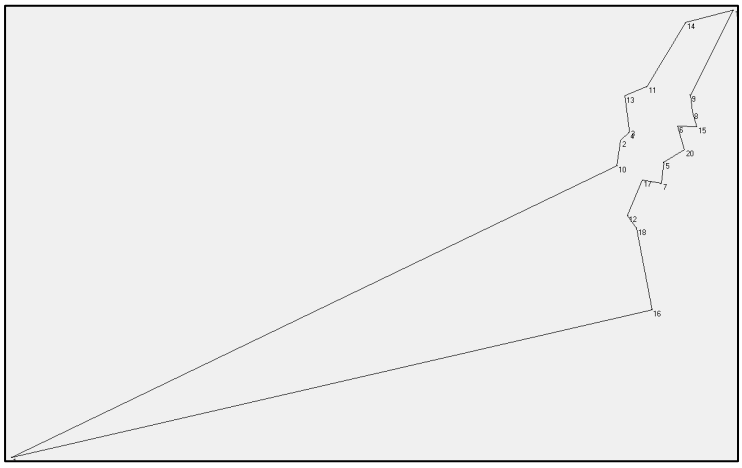


FIGURA 38 - INSERÇÃO DO MAIS DISTANTE COM 2-OPT (QUINTA FEIRA)

SEXTA FEIRA:
Distância total = 18344,68

Rota
1 - 14 - 13 - 12 - 11 - 10 - 25 - 27 - 26 - 28 - 31 - 32 - 22 - 23 - 24 - 21 - 20 - 19 - 18 - 17 -16 - 15 - 30 - 29 - 6 - 7 - 8 - 9 - 5 - 2 - 4 - 3 - 33 - 1

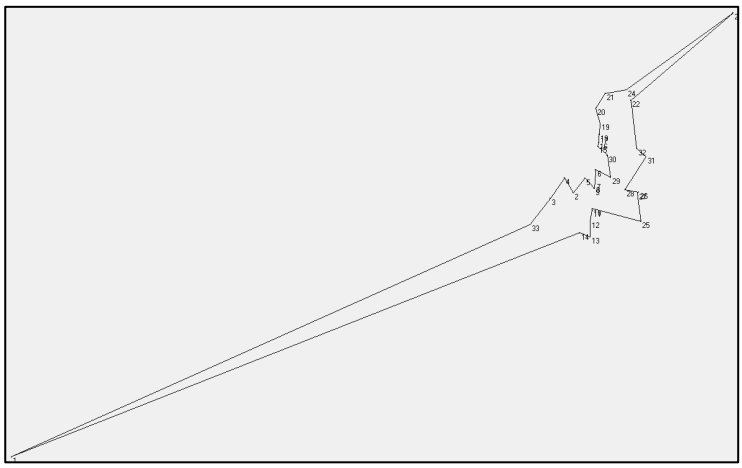


FIGURA 39 - INSERÇÃO DO MAIS DISTANTE (SEXTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 18301,71

Economia = 42,9668

Rota com melhoria 2-OPT
1 - 14 - 13 - 12 - 11 - 10 - 25 - 26 - 27 - 28 - 31 - 32 - 23 - 22 - 24 - 21 - 20 - 19 - 18 - 17 - 16 - 15 - 30 - 29 - 6 - 7 - 8 - 9 - 5 - 2 - 4 - 3 - 33 - 1

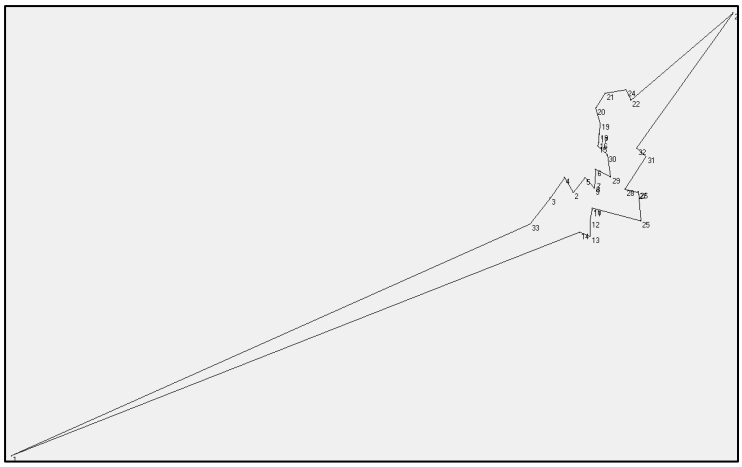


FIGURA 40 - INSERÇÃO DO MAIS DISTANTE COM 2-OPT (SEXTA FEIRA)

INSERÇÃO MAIS RÁPIDA

SEGUNDA FEIRA:

Distância total = 11438,71

Rota:
1 - 19 - 20 - 16 - 18 - 4 - 5 - 21 - 22 - 23 - 24 - 25 - 8 - 12 - 13 - 14 - 15 - 17 - 9 - 7 - 3 - 11 - 10 - 2 - 6 - 1

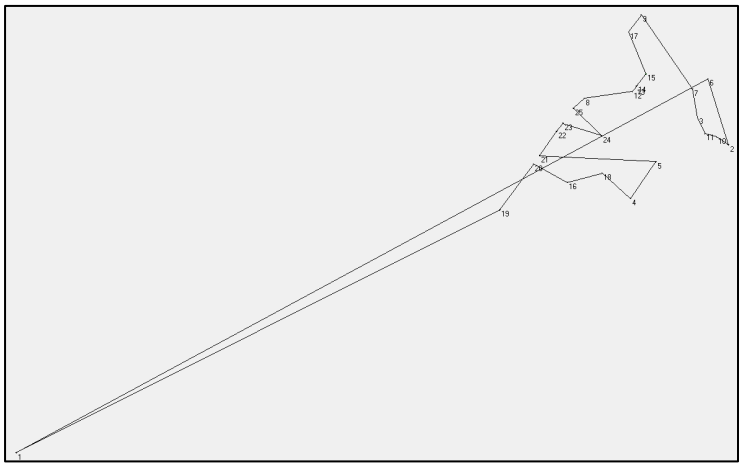


FIGURA 41 - INSERÇÃO MAIS RÁPIDA (SEGUNDA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 10979,42

Economia = 459,29

Rota com melhoria 2-OPT:
1 - 19 - 20 - 21 - 16 - 18 - 4 - 5 - 24 - 22 - 23 - 25 - 8 - 12 - 13 - 14 - 15 - 17 - 9 - 7 - 3 - 11 - 10 - 2 - 6 - 1

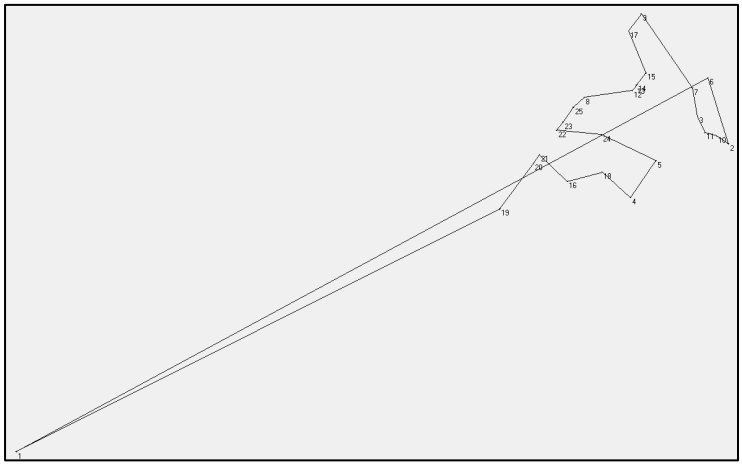


FIGURA 42 - INSERÇÃO MAIS RÁPIDA COM 2-OPT (SEGUNDA FEIRA)

TERÇA FEIRA:

Distância total = 17532,03

Rota:
1 - 7 - 2 - 6 - 8 - 4 - 5 - 14 - 23 - 9 - 21 - 22 - 13 - 24 - 11 - 12 - 19 - 10 - 26 - 25 - 20 - 18 - 3 - 15 - 16 - 17 - 1

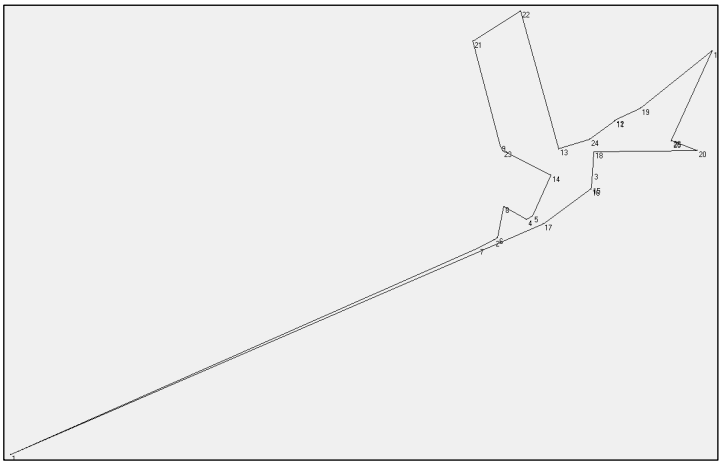


FIGURA 43 - INSERÇÃO MAIS RÁPIDA (TERÇA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 17367,07

Economia = 164,957

Rota com melhoria 2-OPT
1 - 7 - 2 - 6 - 8 - 4 - 5 - 14 - 23 - 9 - 21 - 22 - 13 - 24 - 11 - 12 - 19 - 10 - 20 - 25 - 26 - 18 - 3 - 15 - 16 - 17 – 1

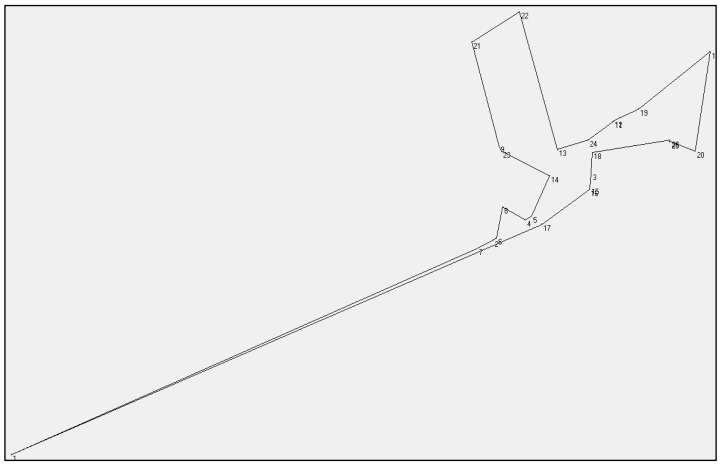


FIGURA 44 - INSERÇÃO MAIS RÁPIDA COM 2-OPT (TERÇA FEIRA)

QUARTA FEIRA:

Distância total = 20656,51

Rota
1 - 15 - 14 - 13 - 10 - 4 - 3 - 5 - 2 - 18 - 24 - 25 - 28 - 23 - 20 - 26 - 19 - 16 - 17 - 22 - 21 - 27 - 6 - 7 - 8 - 9 - 11 - 12 - 1

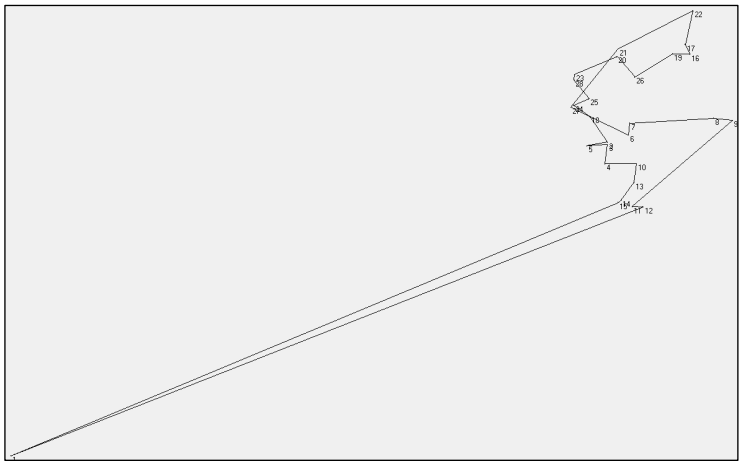


FIGURA 45 - INSERÇÃO MAIS RÁPIDA (QUARTA FEIRA)

Dado da Rota com melhoria 2-OPT

Distância total = 19478,81

Economia = 1177,699

Rota com melhoria 2-OPT
1 - 4 - 3 - 2 - 5 - 18 - 27 - 24 - 28 - 23 - 20 - 21 - 22 - 17 - 16 - 19 - 26 - 25 - 7 - 6 - 8 9 - 10 - 13 - 14 - 15 - 11 - 12 - 1

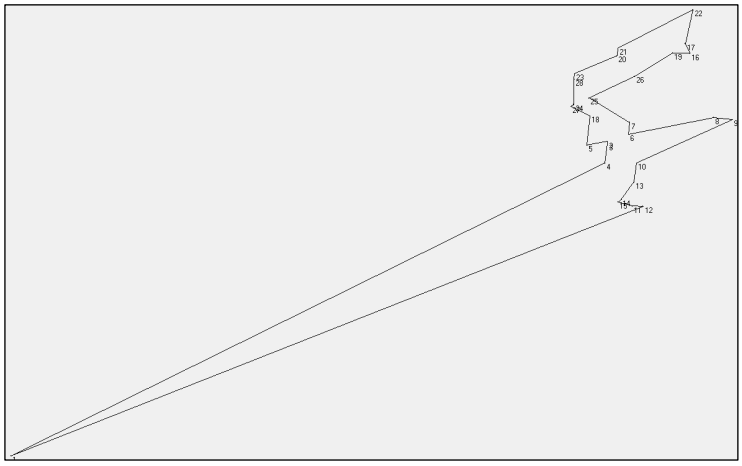


FIGURA 46 - INSERÇÃO MAIS RÁPIDA COM 2-OPT (QUARTA FEIRA)

QUINTA FEIRA:

Distância total = 23733,36

Rota
1 - 16 - 18 - 12 - 17 - 10 - 2 - 4 - 3 - 13 - 11 - 14 - 19 - 7 - 5 - 20 - 6 - 15 - 8 - 9 - 1

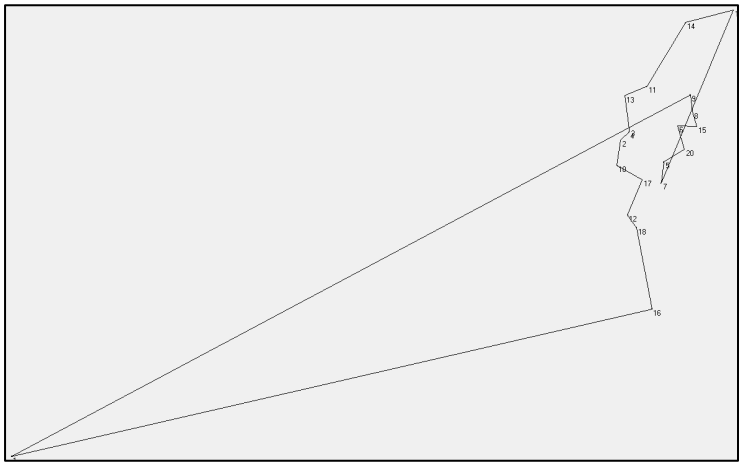


FIGURA 47 - INSERÇÃO MAIS RÁPIDA (QUINTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 23501,2

Economia = 232,1621

Rota com melhoria 2-OPT
1 - 16 - 18 - 12 - 17 - 10 - 2 - 4 - 3 - 13 - 11 - 14 - 19 - 15 - 20 - 7 - 5 - 6 - 8 - 9 - 1

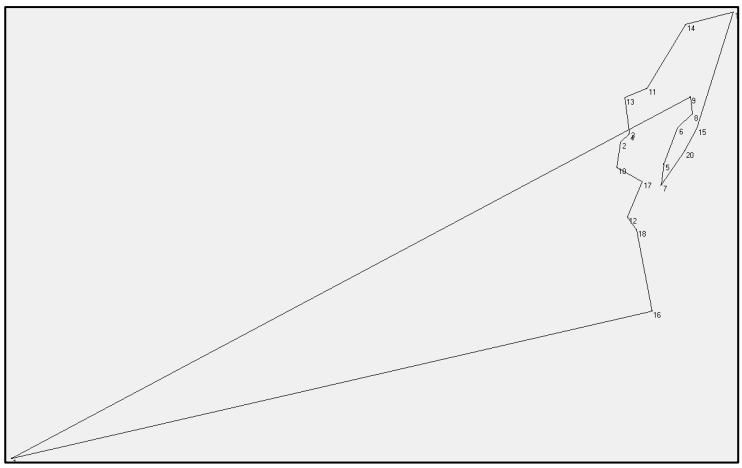


FIGURA 48 - INSERÇÃO MAIS RÁPIDA COM 2-OPT (QUINTA FEIRA)

SEXTA FEIRA:

Distância total = 19857,29

Rota
1 - 33 - 3 - 2 - 5 - 7 - 29 - 28 - 27 - 25 - 26 - 6 - 30 - 32 - 31 - 15 - 16 - 17 - 18 - 19 - 20 - 21 - 24 - 23 - 22 - 8 - 9 - 10 - 11 - 12 - 13 - 14 - 4 - 1

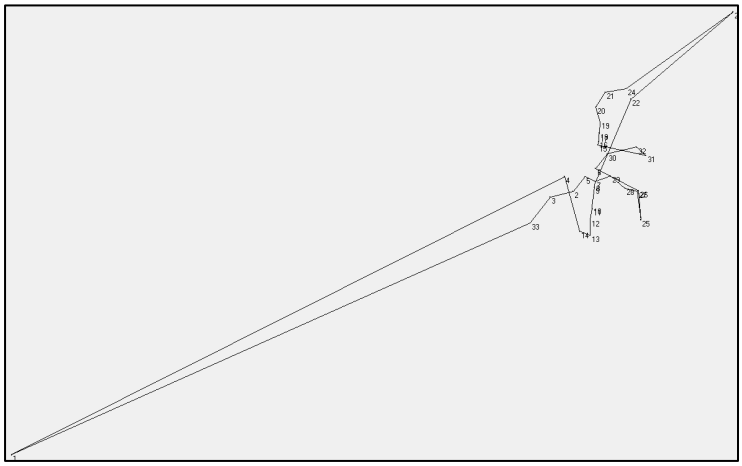


FIGURA 49 - INSERÇÃO MAIS RÁPIDA (SEXTA FEIRA)

Dado da Rota com melhoria 2-OPT
Distância total = 18512,03

Economia = 1345,264

Rota com melhoria 2-OPT
1 - 33 - 14 - 13 - 12 - 11 - 10 - 9 - 8 - 7 - 29 - 28 - 25 - 27 - 26 - 31 - 32 - 23 - 22 - 24 - 21 - 20 - 19 - 18 - 17 - 16 - 15 - 30 - 6 - 5 - 2 - 3 - 4 - 1

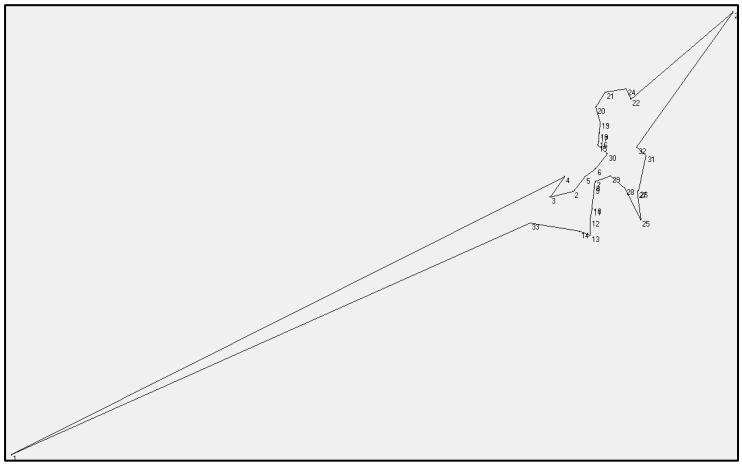


FIGURA 50 - INSERÇÃO MAIS RÁPIDA COM 2-OPT (SEXTA FEIRA)

APÊNDICE 8 - IMPLEMENTAÇÃO DO *ANT SYSTEM*

Segunda Feira:

Roteiro:

1 – 19 – 20 – 21 – 22 – 23 – 25 – 8 – 24 – 18 – 16 – 4 – 5 – 2 – 10 – 11 – 3 – 7 – 6 – 9 – 17 – 15 – 14 – 13 – 12

Custo: 10444,86

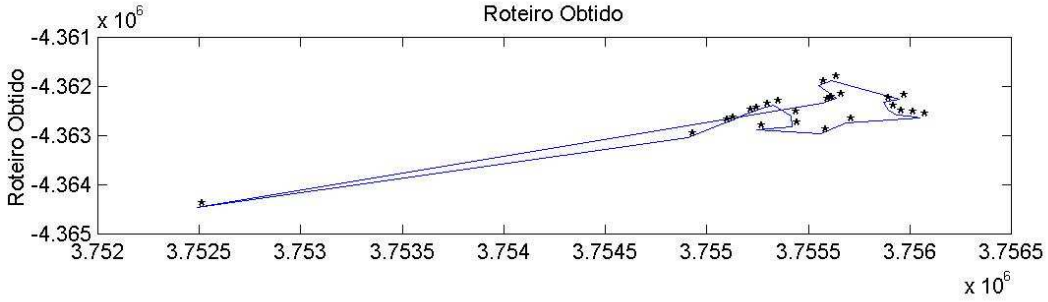


FIGURA 51 - ANT SYSTEM (SEGUNDA FEIRA)

Roteiro com melhoria 2-OPT:

24 – 16 – 19 – 1 – 20 – 21 – 22 – 23 – 25 – 8 – 12 – 13 – 14 – 15 – 17 – 9 – 6 – 7 – 3 – 11 – 10 – 2 – 5 – 4 – 18

Custo: 9879,76

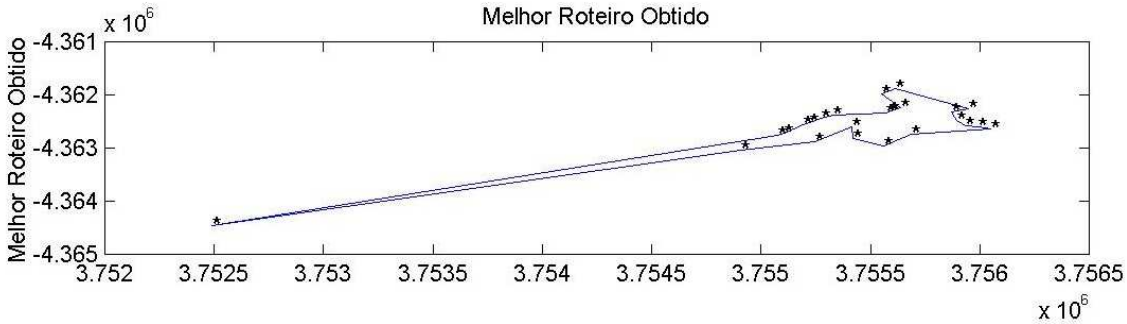


FIGURA 52 - ANT SYSTEM COM 2-OPT (SEGUNDA FEIRA)

Terça Feira:

Roteiro:

1 – 21 – 22 – 10 – 19 – 12 – 11 – 24 – 18 – 3 – 15 – 16 – 14 – 13 – 9 – 23 – 8 – 4 – 5 – 17 – 2 – 6 – 7 – 20 – 26 – 25

Custo: 19897,31

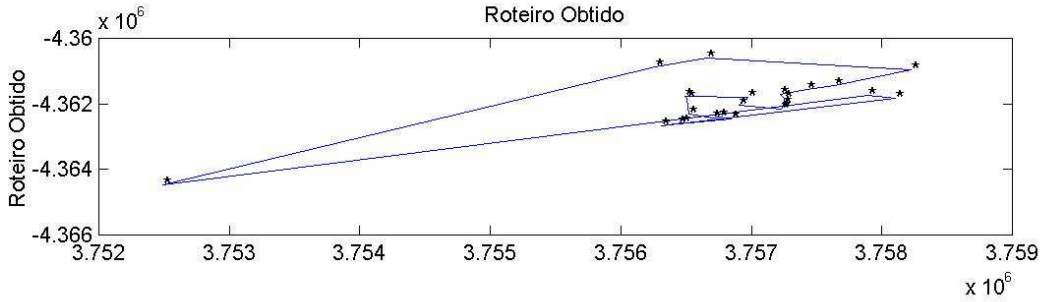


FIGURA 53 - ANT SYTEM (TERÇA FEIRA)

Roteiro com melhoria 2-OPT:

9 – 21 – 22 – 11 – 12 – 19 – 10 – 20 – 26 – 25 – 16 – 15 – 3 – 18 – 24 – 13 – 14 – 17 – 5 – 4 – 6 – 2 – 7 – 1 – 8 – 23

Custo: 16861,14

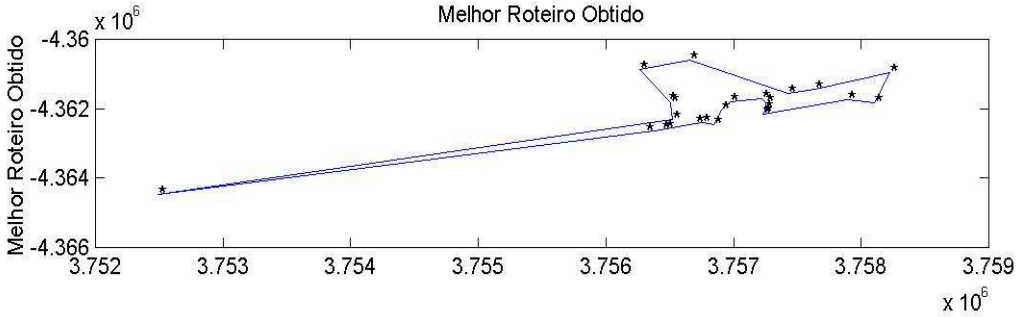


FIGURA 54 - ANT SYSTEM COM 2-OPT (TERÇA FEIRA)

Quarta – Feira:

Roteiro:

1 – 22 – 17 – 16 – 19 – 26 – 20 – 21 – 23 – 28 – 25 – 24 – 27 – 18 – 5 – 2 – 3 – 4 – 10 – 13 – 12 – 11 – 14 – 15 – 9 – 8 – 7 – 6

Custo: 20859,11

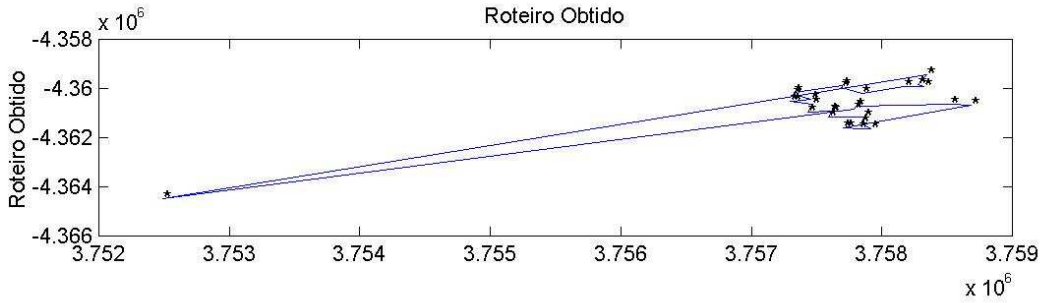


FIGURA 55 - ANT SYSTEM (QUARTA FEIRA)

Roteiro com melhoria 2-OPT:

6 – 10 – 13 – 12 – 11 – 14 – 15 – 1 – 4 – 3 – 2 – 5 – 18 – 27 – 24 – 25 – 28 – 23 – 21 – 20 – 26 – 19 – 22 – 17 – 16 – 9 – 8 – 7

Custo: 18653,77

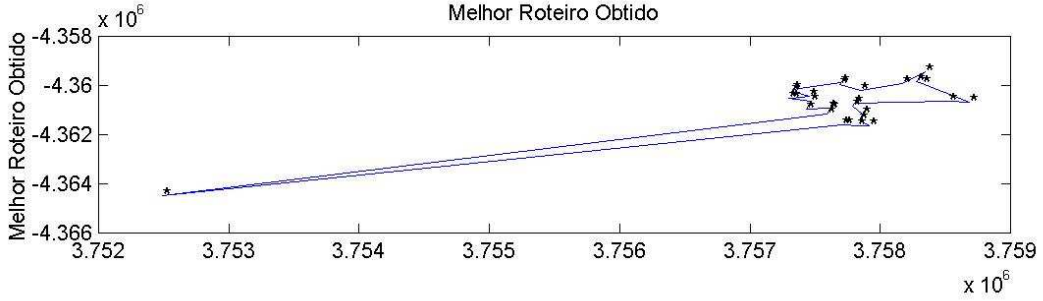


FIGURA 56 - ANT SYSTEM COM 2-OPT (QUARTA FEIRA)

Quinta – Feira:

Roteiro:

1 – 16 – 18 – 12 – 7 – 17 – 10 – 2 – 4 – 3 – 13 – 11 – 9 – 8 – 15 – 6 – 20 – 5 – 19 – 14

Custo: 23510,10

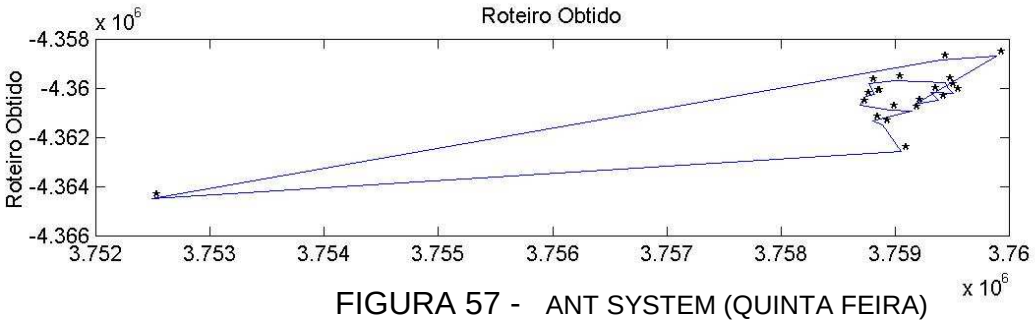


FIGURA 57 - ANT SYSTEM (QUINTA FEIRA)

Roteiro com melhoria 2-OPT:

2 – 4 – 3 – 13 – 11 – 14 – 19 – 9 – 8 – 15 – 6 – 20 – 5 – 7 – 17 – 12 – 18 – 16 – 1 – 10

Custo: 21309,23

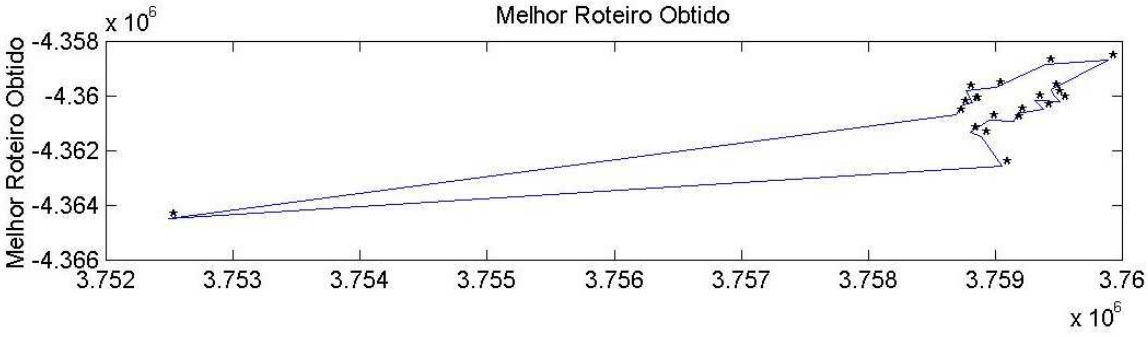


FIGURA 58 - ANT SYSTEM COM 2-OPT (QUINTA FEIRA)

Sexta – Feira:

Roteiro:

1 – 23 – 24 – 22 – 21 – 20 – 19 – 18 – 17 – 16 – 15 – 30 – 6 – 5 – 7 – 8 – 9 – 2 – 4 – 3 – 33 – 14 – 13 – 12 – 11 – 10 – 29 – 28 – 27 – 26 – 25 – 31 – 32

Custo: 20798,98

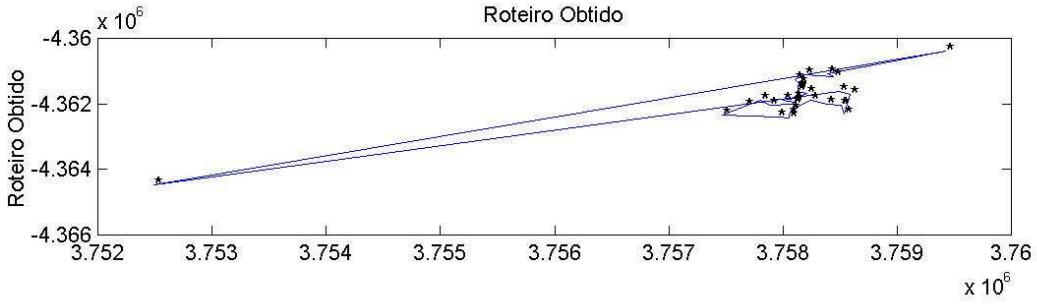


FIGURA 59 - ANT SYSTEM (SEXTA FEIRA)

Roteiro com melhoria 2-OPT:

7 – 5 – 2 – 4 – 3 – 33 – 1 – 14 – 13 – 12 – 11 – 10 – 25 – 27 – 26 – 31 – 32 – 23 – 22 – 24 – 21 – 20 – 19 – 18 – 17 – 16 – 15 – 30 – 6 – 29 – 28 – 9 – 8

Custo: 18448,51

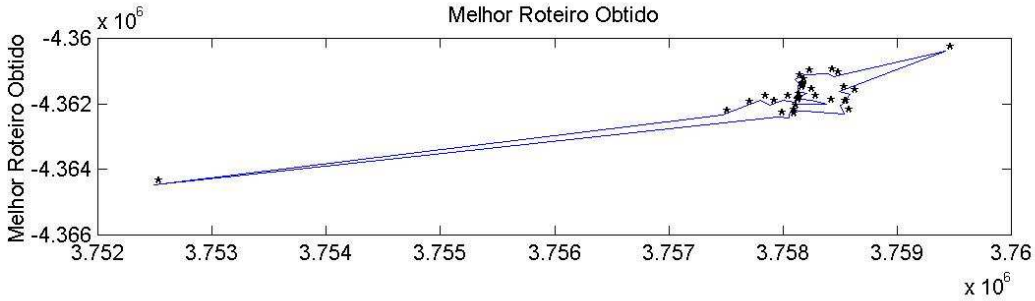


FIGURA 60 - ANT SYSTEM COM 2-OPT (SEXTA FEIRA)

APÊNDICE 9 - IMPLEMENTAÇÃO DO SIMULATED ANNEALING

Segunda feira:

Roteiro:

15 – 14 – 13 – 12 – 8 – 25 – 23 – 22 – 21 – 20 – 1 – 19 – 16 – 24 – 18 – 4 – 5 – 2 – 10 – 11 – 3 – 7 – 6 – 9 – 17

Custo: 9879,76

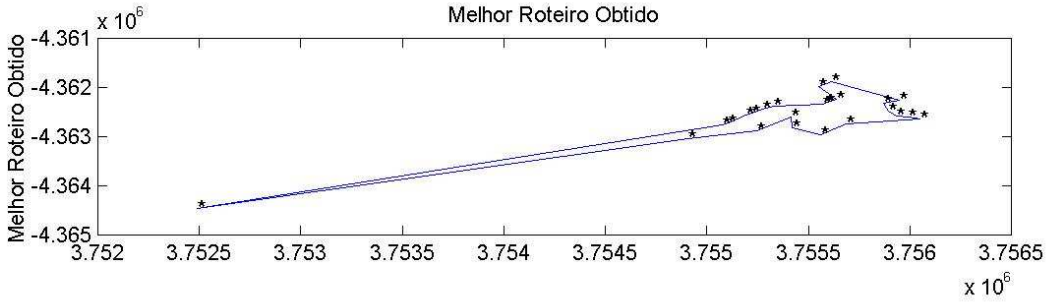


FIGURA 61 - SIMULATED ANNEALING (SEGUNDA FEIRA)

Roteiro com melhoria 2-OPT:

15 – 14 – 13 – 12 – 8 – 25 – 23 – 22 – 21 – 20 – 1 – 19 – 16 – 24 – 18 – 4 – 5 – 2 – 10 – 11 – 3 – 7 – 6 – 9 – 17

Custo: 9879,76

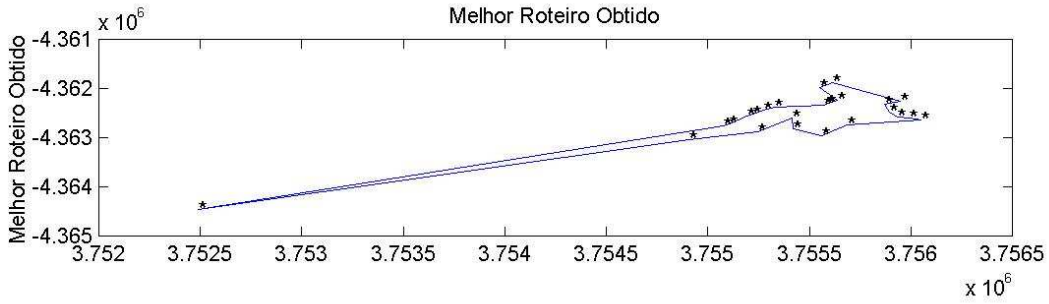


FIGURA 62 - SIMULATED ANNEALING COM 2-OPT (SEGUNDA FEIRA)

Terça feira:

Roteiro:

10 – 20 – 25 – 26 – 19 – 12 – 11 – 24 – 13 – 18 – 3 – 15 – 16 – 14 – 17 – 5 – 4 – 6 – 2 – 7 – 1 – 8 – 23 – 9 – 21 – 22

Custo: 16600,13

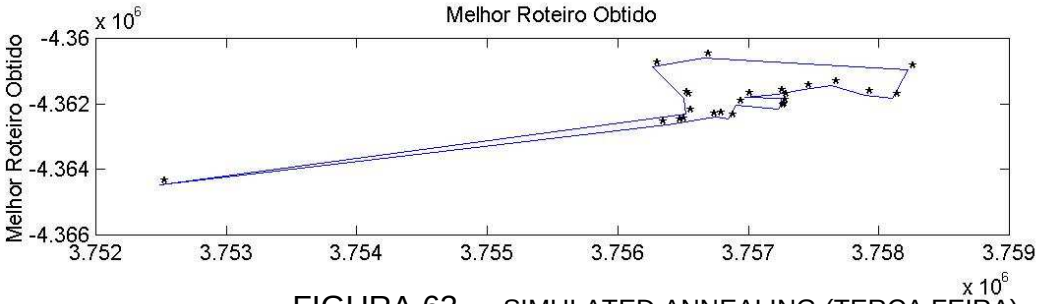


FIGURA 63 - SIMULATED ANNEALING (TERÇA FEIRA)

Roteiro com melhoria 2-OPT:

10 – 20 – 25 – 26 – 19 – 12 – 11 – 24 – 13 – 18 – 3 – 15 – 16 – 14 – 17 – 5 – 4 – 6 – 2 – 7 – 1 – 8 – 23 – 9 – 21 – 22

Custo: 16600,13

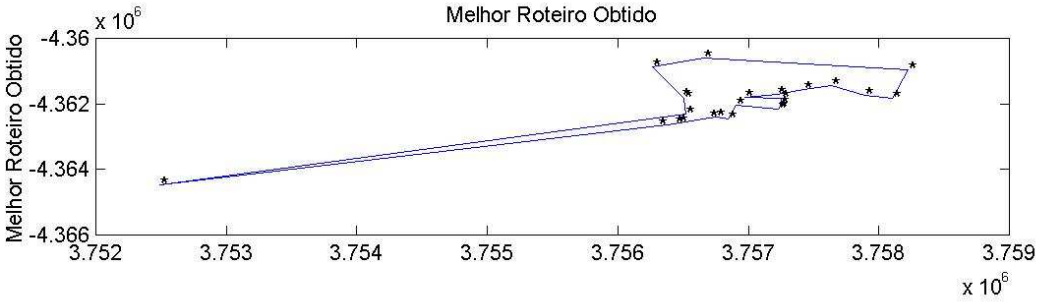


FIGURA 64 - SIMULATED ANNEALING COM 2-OPT (TERÇA FEIRA)

Quarta feira:

Roteiro:

8 – 10 – 13 – 12 – 11 – 14 – 15 – 1 – 4 – 5 – 3 – 2 – 6 – 7 – 18 – 27 – 24 – 25 – 28 – 23 – 21 – 20 – 26 – 19 – 22 – 17 – 16 – 9

Custo: 18691,17

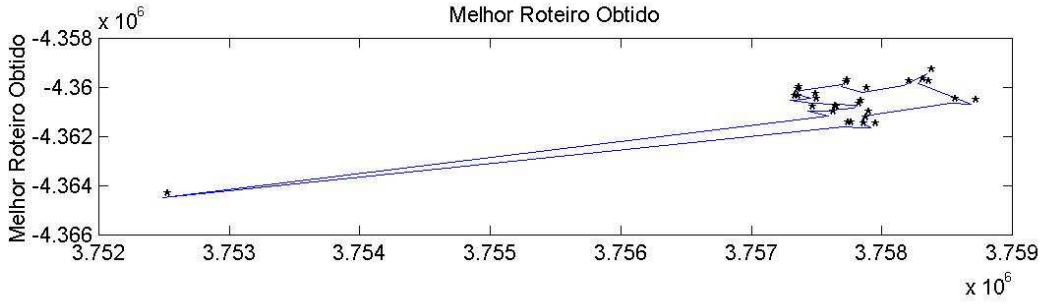


FIGURA 65 - SIMULATED ANNEALING (QUARTA FEIRA)

Roteiro com melhoria 2-OPT:

8 – 10 – 13 – 12 – 11 – 14 – 15 – 1 – 4 – 5 – 3 – 2 – 6 – 7 – 18 – 27 – 24 – 25 – 28 – 23 – 21 – 20 – 26 – 19 – 22 – 17 – 16 – 9

Custo: 18691,17

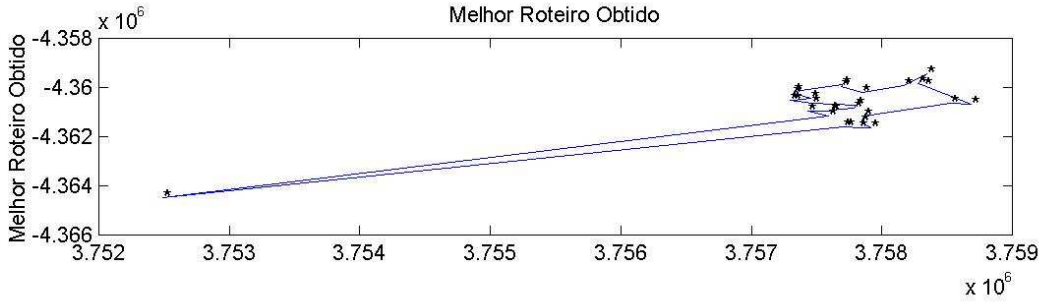


FIGURA 66 - SIMULATED ANNEALING COM 2-OPT (QUARTA FEIRA)

Quinta feira:

Roteiro:

18 – 12 – 7 – 20 – 5 – 17 – 10 – 2 – 4 – 3 – 6 – 15 – 8 – 9 – 19 – 14 – 11 – 13 – 1 – 16

Custo: 22348,60

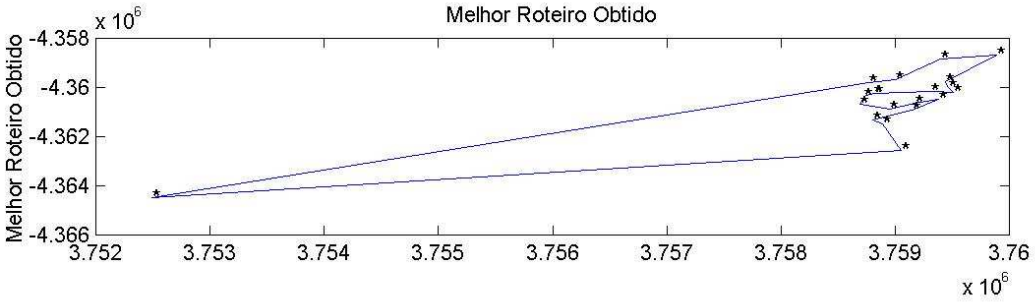


FIGURA 67 - SIMULATED ANNEALING (QUINTA FEIRA)

Roteiro com melhoria 2-OPT:

18 – 12 – 7 – 20 – 5 – 17 – 10 – 2 – 4 – 3 – 6 – 15 – 8 – 9 – 19 – 14 – 11 – 13 – 1 – 16

Custo: 22348,60

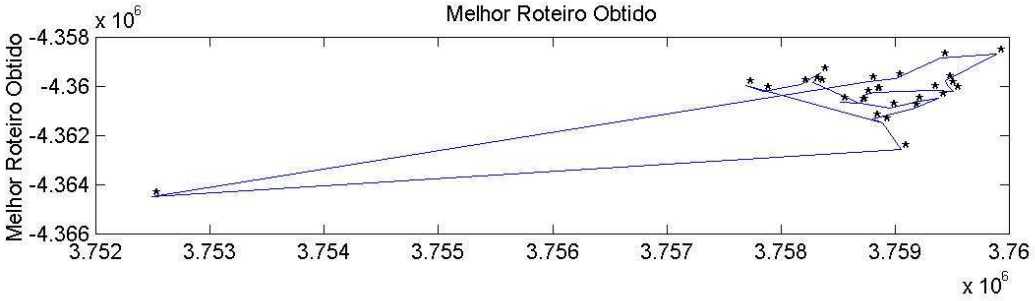


FIGURA 68 - SIMULATED ANNEALING COM 2-OPT (QUINTA FEIRA)

Sexta feira:

Roteiro:

27 – 25 – 28 – 29 – 6 – 5 – 7 – 8 – 9 – 2 – 4 – 3 – 33 – 1 – 14 – 13 – 12 – 11 – 10 – 31 – 32 – 23 – 22 – 24 – 21 – 20 – 19 – 18 – 17 – 16 – 15 – 30 – 26

Custo: 18808,38

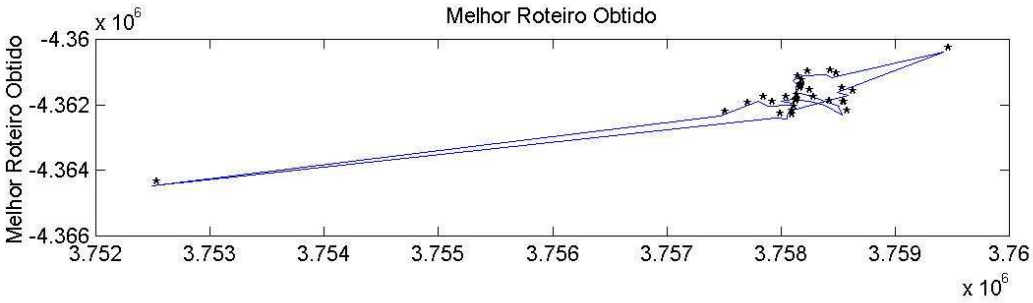


FIGURA 69 - SIMULATED ANNEALING (SEXTA FEIRA)

Roteiro com melhoria 2-OPT:

21 – 20 – 19 – 18 – 17 – 16 – 15 – 30 – 6 – 5 – 7 – 8 – 9 – 2 – 4 – 3 – 33 – 1 – 14 – 13 – 12 – 11 – 10 – 25 – 26 – 27 – 28 – 29 – 32 – 31 – 23 – 22 – 24

Custo: 18328,65

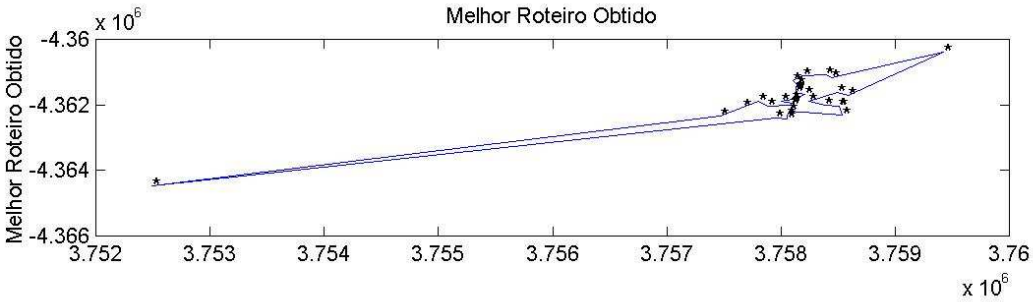


FIGURA 70 - SIMULATED ANNEALING COM 2-OPT (SEXTA FEIRA)

APÊNDICE 10 - IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO

Segunda feira:

Solução:

9 – 17 – 15 – 14 – 13 – 12 – 24 – 8 – 25 – 23 – 22 – 21 – 20 – 19 – 1 – 16 – 18 – 4 – 5 – 2 – 10 – 11 – 3 – 7 – 6

Custo: 9842,52

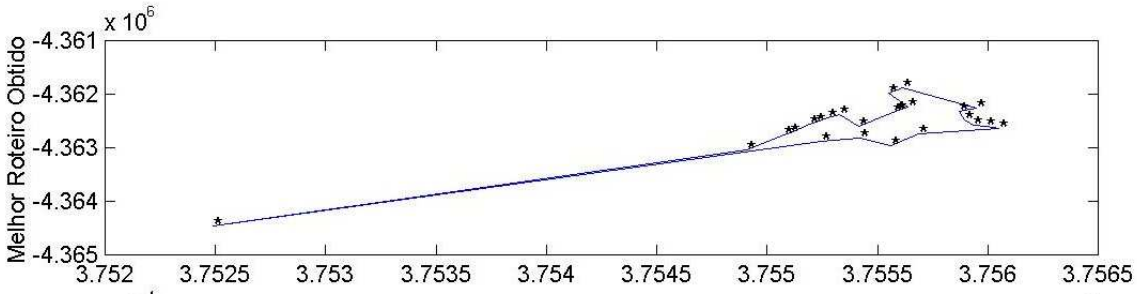


FIGURA 71 - ALGORITMO GENÉTICO - MELHOR ROTEIRO OBTIDO (SEGUNDA FEIRA)

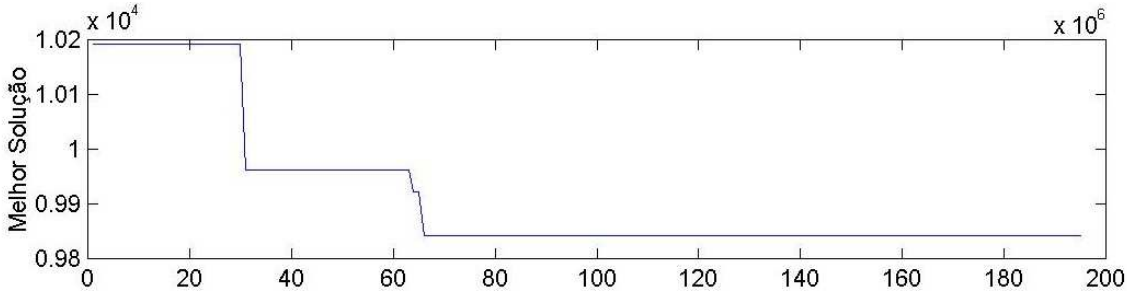


FIGURA 72 - ALGORITMO GENÉTICO - MELHOR SOLUÇÃO (SEGUNDA FEIRA)

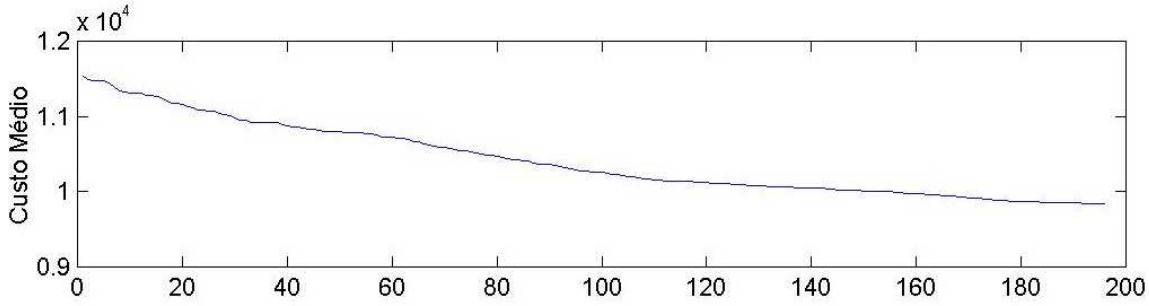


FIGURA 73 - ALGORITMO GENÉTICO – CUSTO MÉDIO (SEGUNDA FEIRA)

Roteiro com melhoria 2opt:

9 – 17 – 15 – 14 – 13 – 12 – 24 – 8 – 25 – 23 – 22 – 21 – 20 – 1 – 19 – 16 – 18 – 4 – 5 – 2 –10 – 11 – 3 – 7 – 6

Custo: 9811,77

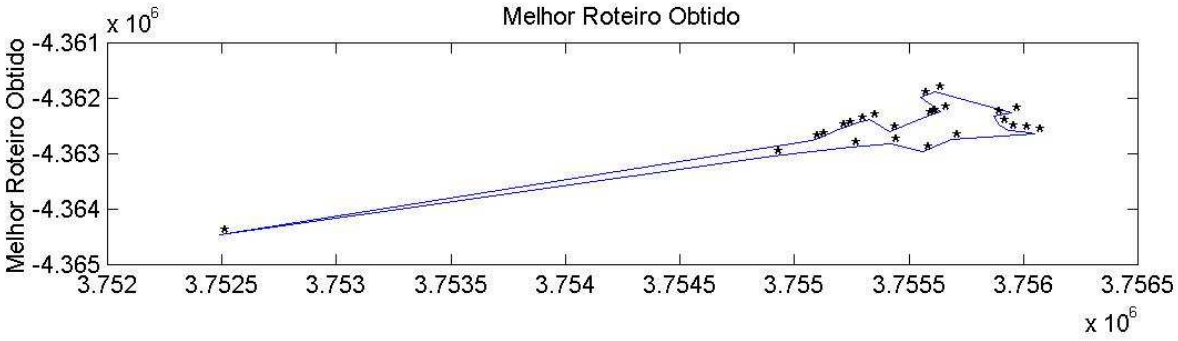


FIGURA 74 - ALGORITMO GENÉTICO COM 2-OPT (SEGUNDA FEIRA)

Terça feira:

Solução:

18 – 24 – 11 – 12 – 19 – 26 – 25 – 20 – 10 – 22 – 21 – 9 – 23 – 1 – 7 – 2 – 6 – 8 – 4 – 5 – 17 – 14 – 13 – 16 – 15 – 3

Custo: 16451,17

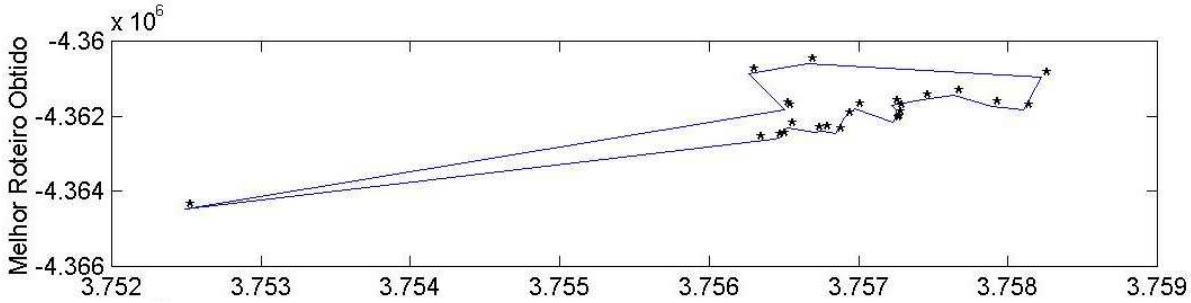


FIGURA 75 - ALGORITMO GENÉTICO - MELHOR ROTEIRO OBTIDO (TERÇA FEIRA)

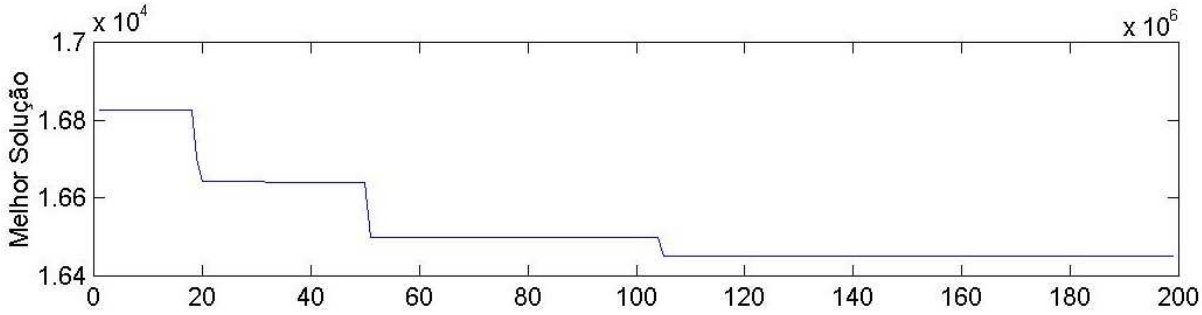


FIGURA 76 - ALGORITMO GENÉTICO - MELHOR SOLUÇÃO (TERÇA FEIRA)

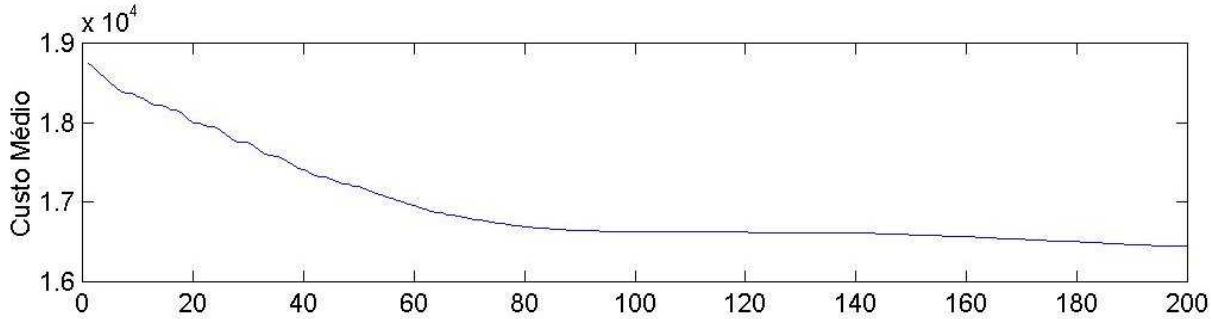


FIGURA 77 - ALGORITMO GENÉTICO – CUSTO MÉDIO (TERÇA FEIRA)

Roteiro com melhoria 2opt:

18 – 24 – 11 – 12 – 19 – 26 – 25 – 20 – 10 – 22 – 21 – 9 – 23 – 1 – 7 – 2 – 6 – 8 – 4 – 5 – 17 – 14 – 13 – 16 – 15 – 3

Custo: 16451,17

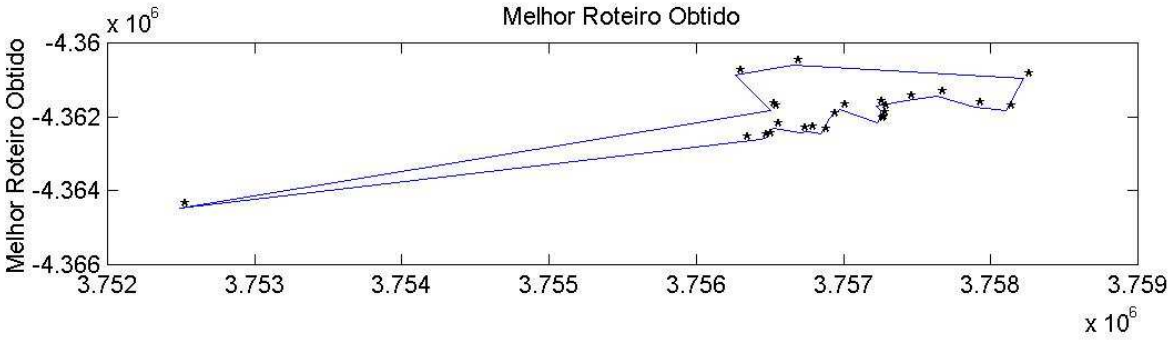


FIGURA 78 - ALGORITMO GENÉTICO COM 2-OPT (TERÇA FEIRA)

Quarta feira

Solução:

23 – 28 – 25 – 24 – 27 – 18 – 1 – 15 – 14 – 11 – 12 – 13 – 10 – 4 – 5 – 3 – 2 – 6 – 7 – 8 – 9 – 22 – 17 – 16 – 19 – 26 – 21 – 20

Custo: 18798,64

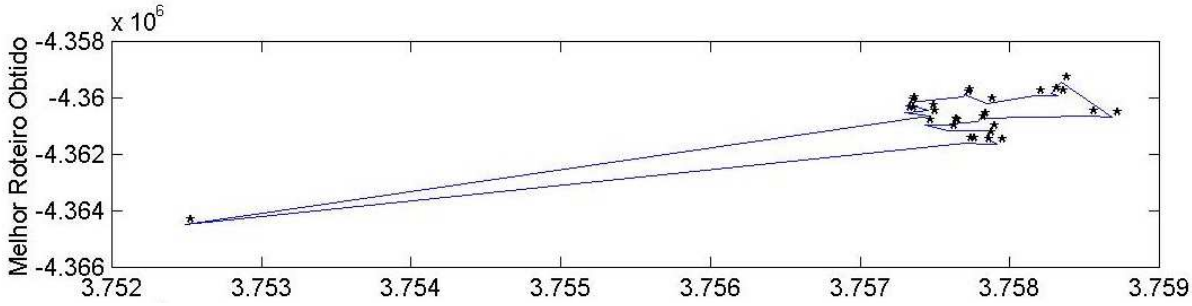


FIGURA 79 - ALGORITMO GENÉTICO - MELHOR ROTEIRO OBTIDO (QUARTA FEIRA)

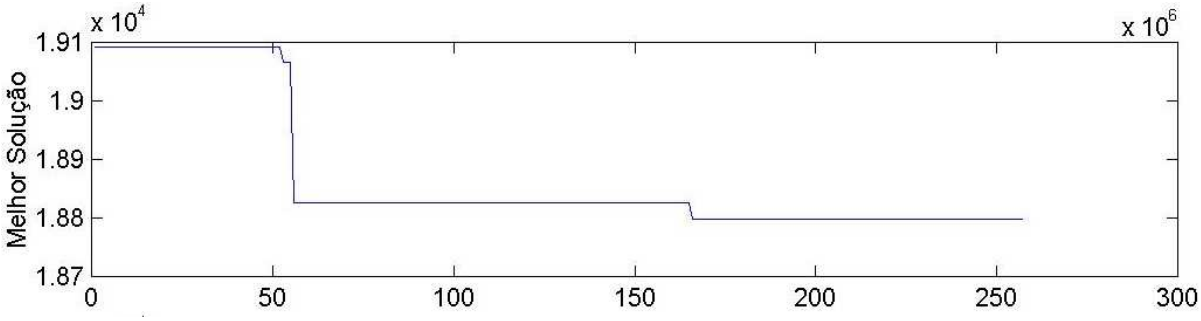


FIGURA 80 - ALGORITMO GENÉTICO - MELHOR SOLUÇÃO (QUARTA FEIRA)

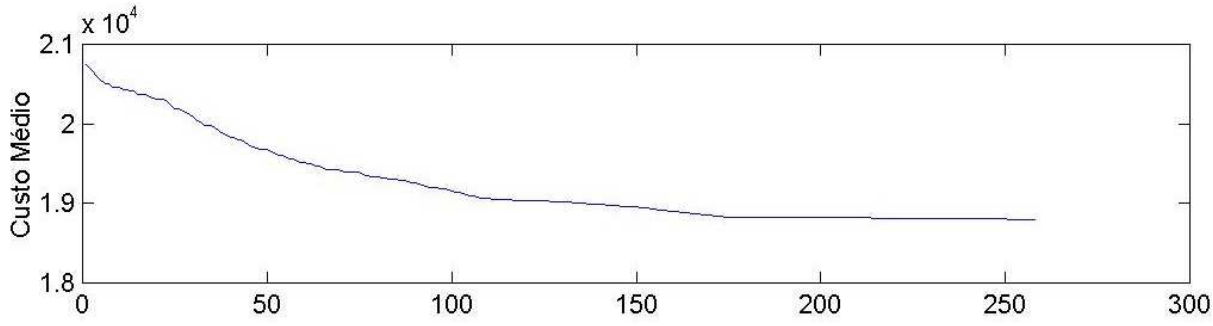


FIGURA 81 - ALGORITMO GENÉTICO – CUSTO MÉDIO (QUARTA FEIRA)

Roteiro com melhoria 2opt:

23 – 28 – 25 – 24 – 27 – 18 – 1 – 15 – 14 – 11 – 12 – 13 – 10 – 4 – 5 – 3 – 2 – 6 – 7 – 8 – 9 – 16 – 17 – 22 – 19 – 26 – 20 – 21

Custo: 18697,54

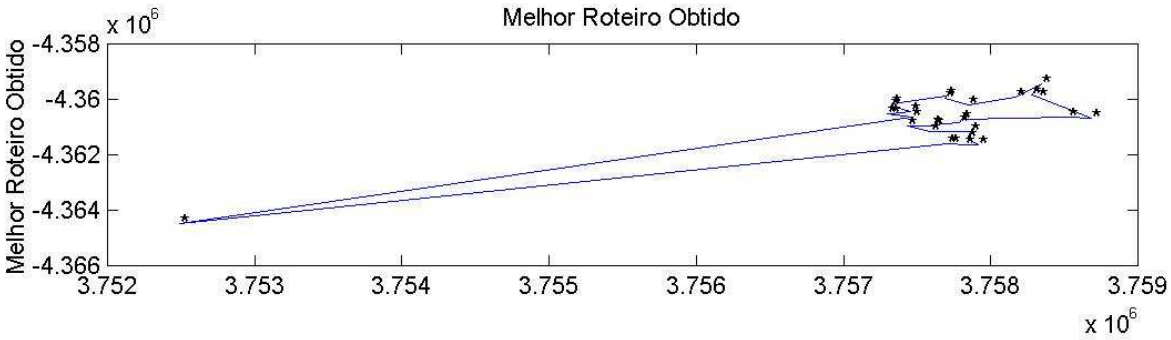


FIGURA 82 - ALGORITMO GENÉTICO COM 2-OPT (QUARTA FEIRA)

Quinta feira

Solução:

7 – 17 – 12 – 18 – 16 – 1 – 10 – 2 – 4 – 3 – 13 – 11 – 14 – 19 – 9 – 8 – 15 – 6 – 20 – 5

Custo: 21309,23

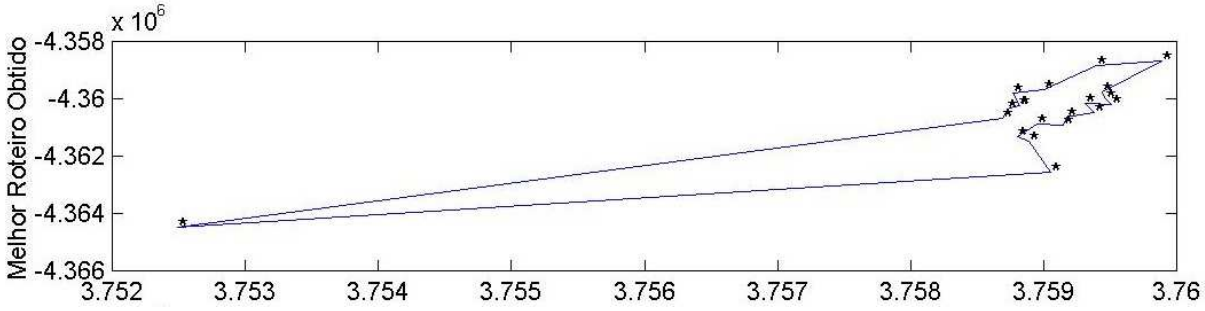


FIGURA 83 - ALGORITMO GENÉTICO - MELHOR ROTEIRO OBTIDO (QUINTA FEIRA)

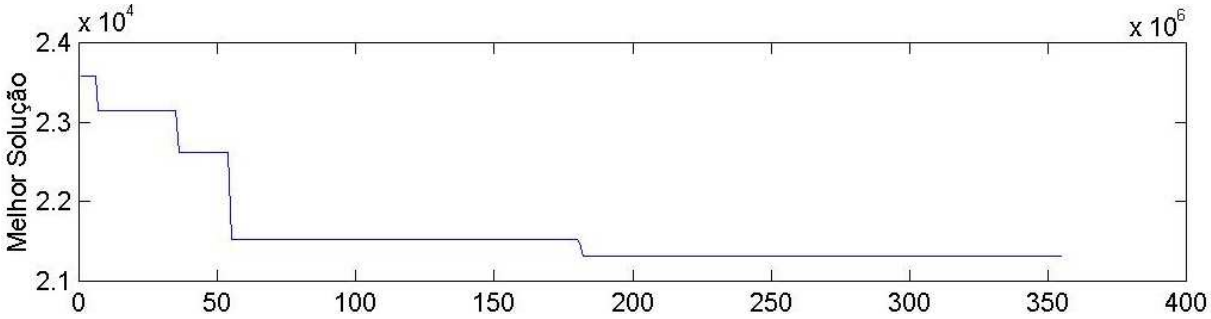


FIGURA 84 - ALGORITMO GENÉTICO - MELHOR ROTEIRO SOLUÇÃO (QUINTA FEIRA)

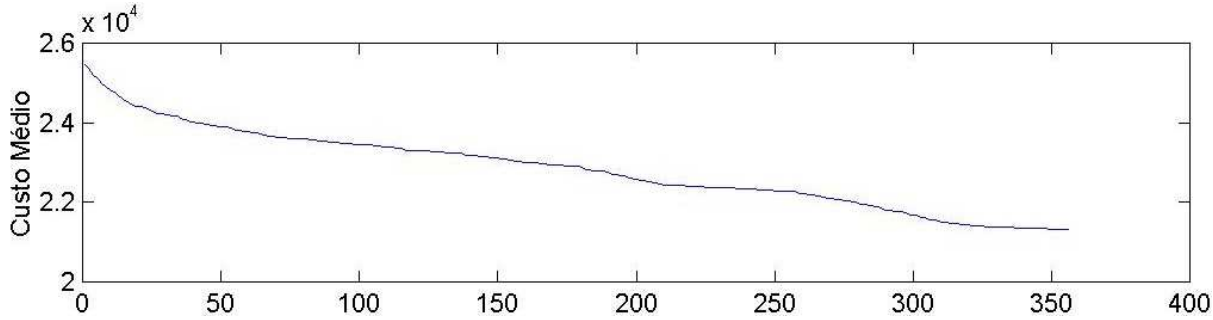


FIGURA 85 - ALGORITMO GENÉTICO – CUSTO MÉDIO (QUINTA FEIRA)

Roteiro com melhoria 2opt:

7 – 17 – 12 – 18 – 16 – 1 – 10 – 2 – 4 – 3 – 13 – 11 – 14 – 19 – 9 – 8 – 15 – 6 – 20 – 5

Custo: 21309,23

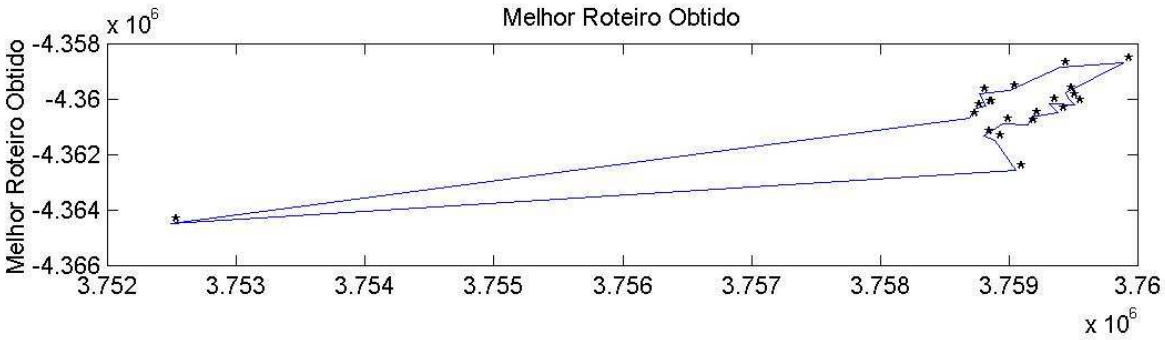


FIGURA 86 - ALGORITMO GENÉTICO COM 2-OPT (QUINTA FEIRA)

Sexta feira

Solução:

28 – 27 – 26 – 25 – 10 – 11 – 12 – 14 – 13 – 1 – 33 – 3 – 4 – 2 – 5 – 9 – 8 – 7 – 6 – 30 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 24 – 22 – 23 – 32 – 31 – 29

Custo: 18409,83

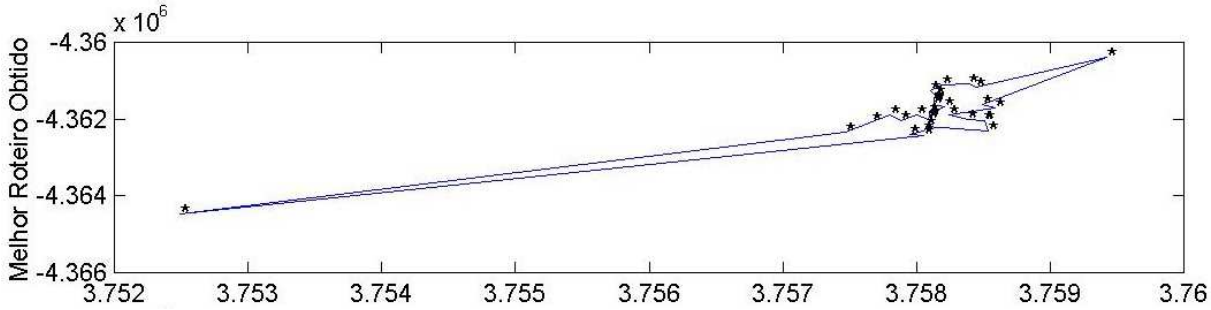


FIGURA 87 - ALGORITMO GENÉTICO - MELHOR ROTEIRO OBTIDO (SEXTA FEIRA)

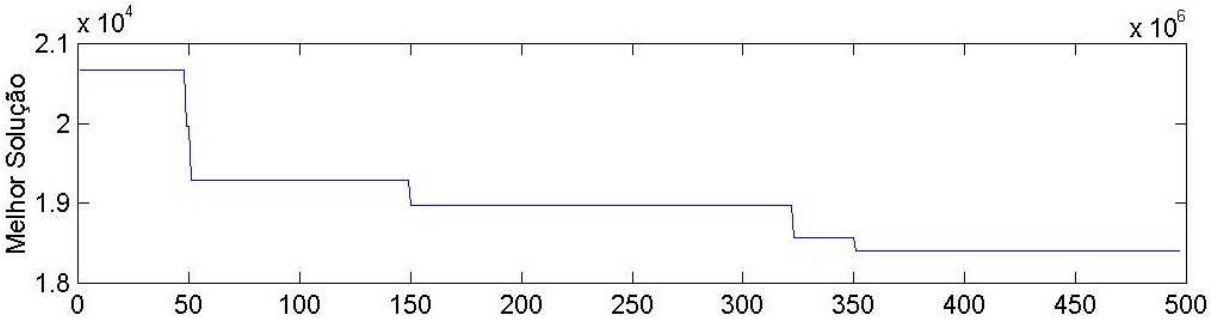


FIGURA 88 - ALGORITMO GENÉTICO - MELHOR SOLUÇÃO (SEXTA FEIRA)

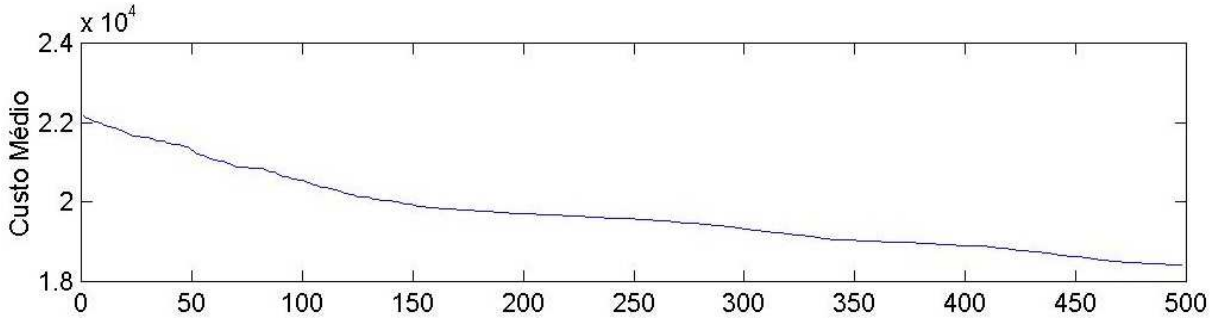


FIGURA 89 - ALGORITMO GENÉTICO – CUSTO MÉDIO (SEXTA FEIRA)

Roteiro com melhoria 2opt:

28 – 27 – 26 – 25 – 10 – 11 – 12 – 13 – 14 – 1 – 33 – 3 – 4 – 2 – 5 – 9 – 8 – 7 – 6 – 30 – 15 – 16 – 17 –18 – 19 – 20 – 21 – 24 – 22 – 23 – 31 – 32 – 29

Custo: 18314,63

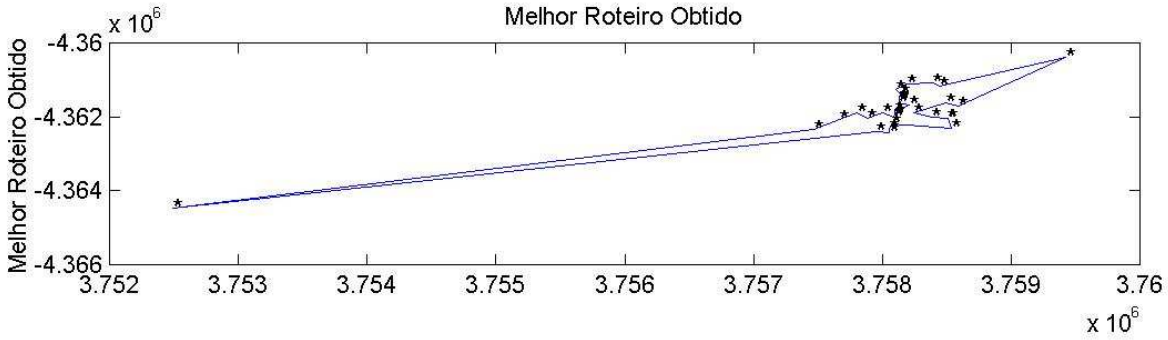


FIGURA 90 - ALGORITMO GENÉTICO COM 2-OPT (SEXTA FEIRA)

APÊNDICE 11 - RESULTADOS OBTIDOS PELO APLICATIVO LINGO

O resultado gerado pelo Lingo 12.0 para cada dia da semana esta exibido a seguir.

Segunda Feira

```
  Global optimal solution found.
  Objective value:                              9811.772
  Objective bound:                              9811.772
  Infeasibilities:                              0.000000
  Extended solver steps:                            1045
  Total solver iterations:                         74078
  Tempo computacional: 00:00:13


Variable           Value        Reduced Cost
A(1, 20)       1.000000            3095.811
A(2, 5)        1.000000            375.1422
A(3, 11)       1.000000            103.6843
A(4, 18)       1.000000            205.3941
A(5, 4)        1.000000            251.5364
A(6, 7)        1.000000            99.75977
A(7, 3)        1.000000            166.7357
A(8, 24)       1.000000            234.6727
A(9, 6)        1.000000            503.7760
A(10, 2)       1.000000            73.98103
A(11, 10)      1.000000            61.49437
A(12, 13)      1.000000            25.45933
A(13, 14)      1.000000            14.21446
A(14, 15)      1.000000            84.30628
A(15, 17)      1.000000            261.6015
A(16, 19)      1.000000            374.8333
A(17, 9)       1.000000            119.0946
A(18, 16)      1.000000            182.6431
A(19, 1)       1.000000            2810.276
A(20, 21)      1.000000            58.19041
A(21, 22)      1.000000            167.7107
A(22, 23)      1.000000            54.43608
A(23, 25)      1.000000            104.2499
A(24, 12)      1.000000            303.0862
A(25, 8)       1.000000            79.68336
```

Rota Gerada:

1 – 20 -21 – 22 – 23 – 25 – 8 – 24 – 12 -13 – 14 – 15 – 17 – 9 – 6 – 7 – 3 – 11 – 10 – 2 – 5 – 4 – 18 – 16 – 19 - 1

Terça feira

```
  Global optimal solution found.
  Objective value:                              16451.17
  Objective bound:                              16451.17
  Infeasibilities:                              0.000000
  Extended solver steps:                           79182
  Total solver iterations:                      28748575
  Tempo computacional: 01:33:05
```

Variable	Value	Reduced Cost
A(1, 23)	1.000000	4810.805
A(2, 7)	1.000000	149.6154
A(3, 15)	1.000000	122.1875
A(4, 8)	1.000000	215.8230
A(5, 4)	1.000000	59.93245
A(6, 2)	1.000000	32.98011
A(7, 1)	1.000000	4221.707
A(8, 6)	1.000000	274.8578
A(9, 21)	1.000000	926.4776
A(10, 20)	1.000000	873.5856
A(11, 24)	1.000000	259.7516
A(12, 11)	1.000000	5.565975
A(13, 14)	1.000000	237.5423
A(14, 17)	1.000000	424.9831
A(15, 16)	1.000000	18.26808
A(16, 13)	1.000000	435.8469
A(17, 5)	1.000000	112.0689
A(18, 3)	1.000000	182.3307
A(19, 12)	1.000000	233.1410
A(20, 25)	1.000000	224.8930
A(21, 22)	1.000000	468.5779
A(22, 10)	1.000000	1604.116
A(23, 9)	1.000000	52.75759
A(24, 18)	1.000000	117.0989
A(25, 26)	1.000000	11.17280
A(26, 19)	1.000000	375.0821

Rota Gerada:

1 – 23 – 9 – 21 – 22 – 10 – 20 – 25 – 26 – 19 – 12 – 11 – 24 – 18 – 3 – 15 – 16 – 13 – 14 – 17 – 5 – 4 – 8 – 6 – 2 – 7 - 1

Quarta Feira

```
  Global optimal solution found.
  Objective value:                              18598.65
  Objective bound:                              18598.65
  Infeasibilities:                              0.000000
  Extended solver steps:                           14696
  Total solver iterations:                        931230
  Tempo computacional: 00:02:55
```

Variable	Value	Reduced Cost
A(1, 5)	1.000000	6056.847
A(2, 3)	1.000000	26.61109
A(3, 4)	1.000000	217.3771
A(4, 10)	1.000000	271.3556
A(5, 18)	1.000000	329.4948
A(6, 2)	1.000000	198.9168
A(7, 6)	1.000000	132.1877
A(8, 7)	1.000000	721.4310
A(9, 8)	1.000000	163.7052
A(10, 13)	1.000000	212.6607
A(11, 14)	1.000000	119.8551
A(12, 11)	1.000000	92.79147
A(13, 12)	1.000000	285.0769
A(14, 15)	1.000000	36.32565
A(15, 1)	1.000000	5945.710
A(16, 9)	1.000000	827.7338
A(17, 16)	1.000000	116.9151
A(18, 27)	1.000000	196.1178
A(19, 22)	1.000000	516.4283
A(20, 26)	1.000000	277.9692
A(21, 20)	1.000000	85.43673
A(22, 17)	1.000000	386.2470
A(23, 21)	1.000000	470.3246
A(24, 25)	1.000000	150.8288
A(25, 28)	1.000000	247.9484
A(26, 19)	1.000000	416.0316
A(27, 24)	1.000000	34.32448
A(28, 23)	1.000000	61.99537

Rota Gerada:
1 – 5 – 18 – 27 – 24 – 25 – 28 – 23 – 21 – 20 – 26 – 19 – 22 – 17 – 16 – 9 – 8 – 7 – 6 – 2 – 3 – 4 – 10 – 13 – 12 – 11 – 14 – 15 - 1

Quinta Feira

```
  Global optimal solution found.
  Objective value:                              21309.23
  Objective bound:                              21309.23
  Infeasibilities:                              0.000000
  Extended solver steps:                               2
  Total solver iterations:                          6346
  Tempo computacional: 00:00:02
```

Variable	Value	Reduced Cost
A(1, 16)	1.000000	6834.315
A(2, 10)	1.000000	332.2805
A(3, 4)	1.000000	38.46385
A(4, 2)	1.000000	133.4236
A(5, 20)	1.000000	269.6560
A(6, 15)	1.000000	197.1894
A(7, 5)	1.000000	273.8428
A(8, 9)	1.000000	225.3518
A(9, 19)	1.000000	1179.620
A(10, 1)	1.000000	7261.095
A(11, 13)	1.000000	257.9115
A(12, 17)	1.000000	481.9133
A(13, 3)	1.000000	435.3779
A(14, 11)	1.000000	915.9590
A(15, 8)	1.000000	194.6653
A(16, 18)	1.000000	1064.010
A(17, 7)	1.000000	199.5256
A(18, 12)	1.000000	190.6223
A(19, 14)	1.000000	514.0474
A(20, 6)	1.000000	309.9625

Rota Gerada:
1 – 16 – 18 – 17 – 7 – 5 – 20 – 6 – 15 – 8 – 9 – 19 – 14 – 11 – 13 – 3 – 4 – 2 – 10 - 1

Sexta Feira

```
  Global optimal solution found.
  Objective value:                              18206.61
  Objective bound:                              18206.61
  Infeasibilities:                              0.000000
  Extended solver steps:                            5327
  Total solver iterations:                        244954
  Tempo computacional: 00:00:56
```

Variable	Value	Reduced Cost
A(1, 33)	1.000000	5420.806
A(2, 5)	1.000000	174.4043
A(3, 4)	1.000000	233.0879
A(4, 2)	1.000000	159.5807
A(5, 6)	1.000000	128.3080
A(6, 30)	1.000000	173.5853
A(7, 8)	1.000000	28.74121
A(8, 9)	1.000000	25.15852
A(9, 10)	1.000000	185.3446
A(10, 11)	1.000000	15.41128
A(11, 12)	1.000000	100.3255
A(12, 13)	1.000000	143.3898
A(13, 14)	1.000000	107.2032
A(14, 1)	1.000000	5836.747
A(15, 16)	1.000000	35.22870
A(16, 17)	1.000000	58.85258
A(17, 18)	1.000000	18.14911
A(18, 19)	1.000000	103.0077
A(19, 20)	1.000000	142.7504
A(20, 21)	1.000000	162.7578
A(21, 24)	1.000000	203.6996
A(22, 23)	1.000000	1269.848
A(23, 32)	1.000000	1552.803
A(24, 22)	1.000000	104.8910
A(25, 28)	1.000000	326.0458
A(26, 27)	1.000000	11.91597
A(27, 25)	1.000000	263.0690
A(28, 29)	1.000000	182.1955
A(29, 7)	1.000000	157.5500
A(30, 15)	1.000000	122.5230
A(31, 26)	1.000000	331.8879
A(32, 31)	1.000000	117.0557
A(33, 3)	1.000000	310.2844

Rota Gerada:
1 – 33 – 3 – 4 – 2 – 5 – 6 – 30 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 24 – 22 – 23 – 32 – 31 – 26 – 27 – 25 – 28 – 29 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 - 1